

CS447: Natural Language Processing

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

Lecture 09:

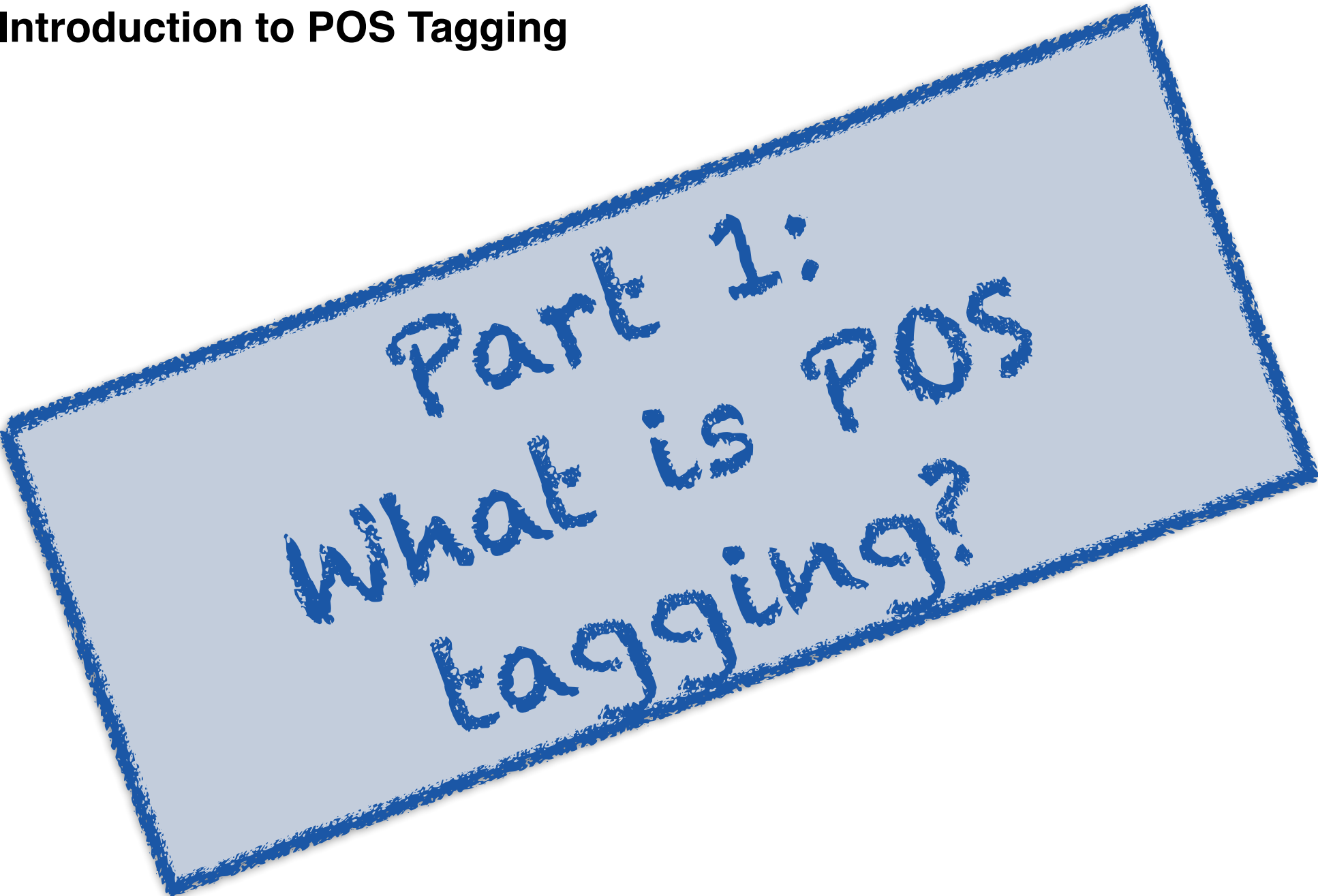
Part-of-Speech Tagging

Julia Hockenmaier

juliahmr@illinois.edu

3324 Siebel Center

Lecture 09: Introduction to POS Tagging



Part 1:
What is POS
tagging?



What are parts of speech?

Nouns, Pronouns, Proper Nouns,
Verbs, Auxiliaries,
Adjectives, Adverbs
Prepositions, Conjunctions,
Determiners, Particles
Numerals, Symbols,
Interjections, etc.

See e.g. <https://universaldependencies.org/u/pos/>

POS Tagging

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:

Determine the POS tag for all tokens in a sentence.

Due to ambiguity (and unknown words), we cannot rely on a dictionary to look up the correct POS tags.

These examples from Dekang Lin

Why POS Tagging?

POS tagging is one of the first steps in the NLP pipeline (right after tokenization, segmentation).

POS tagging is traditionally viewed as a prerequisite for further analysis:

- Syntactic Parsing:

What words are in the sentence?

- Information extraction:

Finding names, dates, relations, etc.

NB: Although many neural models don't use POS tagging, it is still important to understand what makes POS tagging difficult (or easy), and how the basic models and algorithms work.



Creating a POS Tagger

To handle ambiguity and coverage,
POS taggers rely on learned models.

For a **new language** (or domain)

Step 0: Define a POS tag set

Step 1: Annotate a corpus with these tags

For a **well-studied language** (and domain):

Step 1: Obtain a POS-tagged corpus

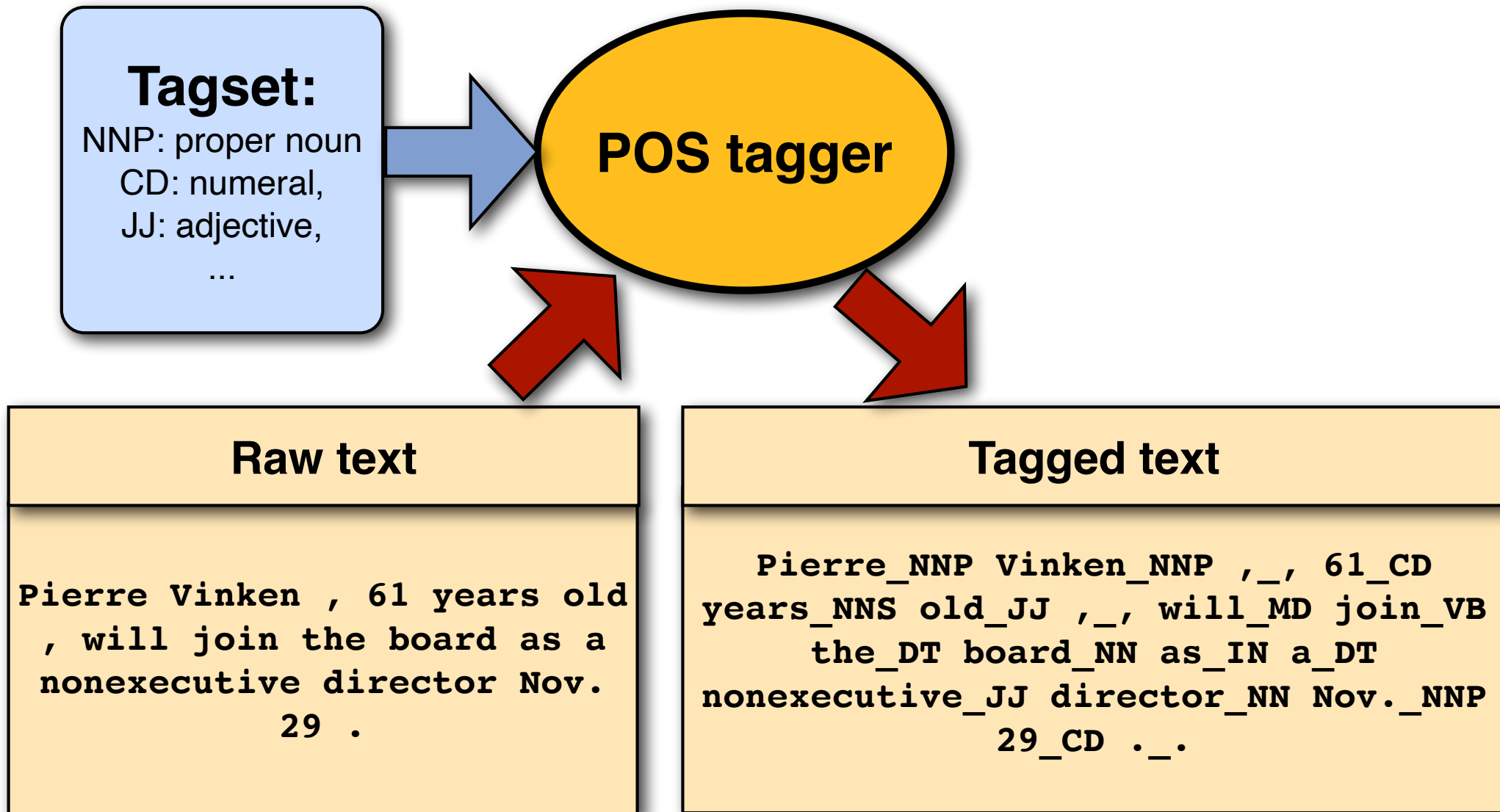
For any language.....:

Step 2: Choose a POS tagging model (e.g. an HMM)

Step 3: Train your model on your training corpus

Step 4: Evaluate your model on your test corpus

POS Tagging



Defining a Tag Set

We have to define an **inventory of labels for the word classes** (i.e. the tag set)

- Most taggers rely on models that have to be trained on **annotated (tagged) corpora**.
- **Evaluation** also requires annotated corpora.
- Since human annotation is expensive/time-consuming, the tag sets used in a few existing labeled corpora become the **de facto standard**.
- Tag sets need to capture **semantically or syntactically important distinctions** that can easily be made by trained human annotators.

Defining a Tag Set

Tag sets 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*

Adjective Regular Feminine Plural Dative....Superlative

[Hajic 2006, VMC tutorial]



How Much Ambiguity is There?

Common POS ambiguities in English:

Noun—Verb: *table*

Adjective—Verb: *laughing, known,*

Noun—Adjective: *normal*

A word is ambiguous if has more than one POS

Unless we have a dictionary that gives all POS tags for each word, we only know the POS tags with which a word appears in our corpus.

Since many words appear only once (or a few times) in any given corpus, we may not know all of their POS tags.

Most **word types** appear with only one POS tag....

Brown corpus with 87-tag set: 3.3% of word types are ambiguous,

Brown corpus with 45-tag set: 18.5% of word types are ambiguous

... but a large fraction of **word tokens** are ambiguous

Original Brown corpus: 40% of tokens are ambiguous

Evaluation Metric: Test Accuracy

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

State of the art on Penn Treebank: around 97%

⇒ **How many *sentences* can you tag correctly?**

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 tag sets 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.

Also: POS tagging accuracy on English text from other domains can be significantly lower.

Qualitative evaluation

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

		Correct Tags						
Predicted Tags		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	—

% of errors caused by mistagging VBN as JJ

See what errors are causing problems:

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

Today's Class

Part 1: What is POS tagging?

Part 2: English Parts of Speech

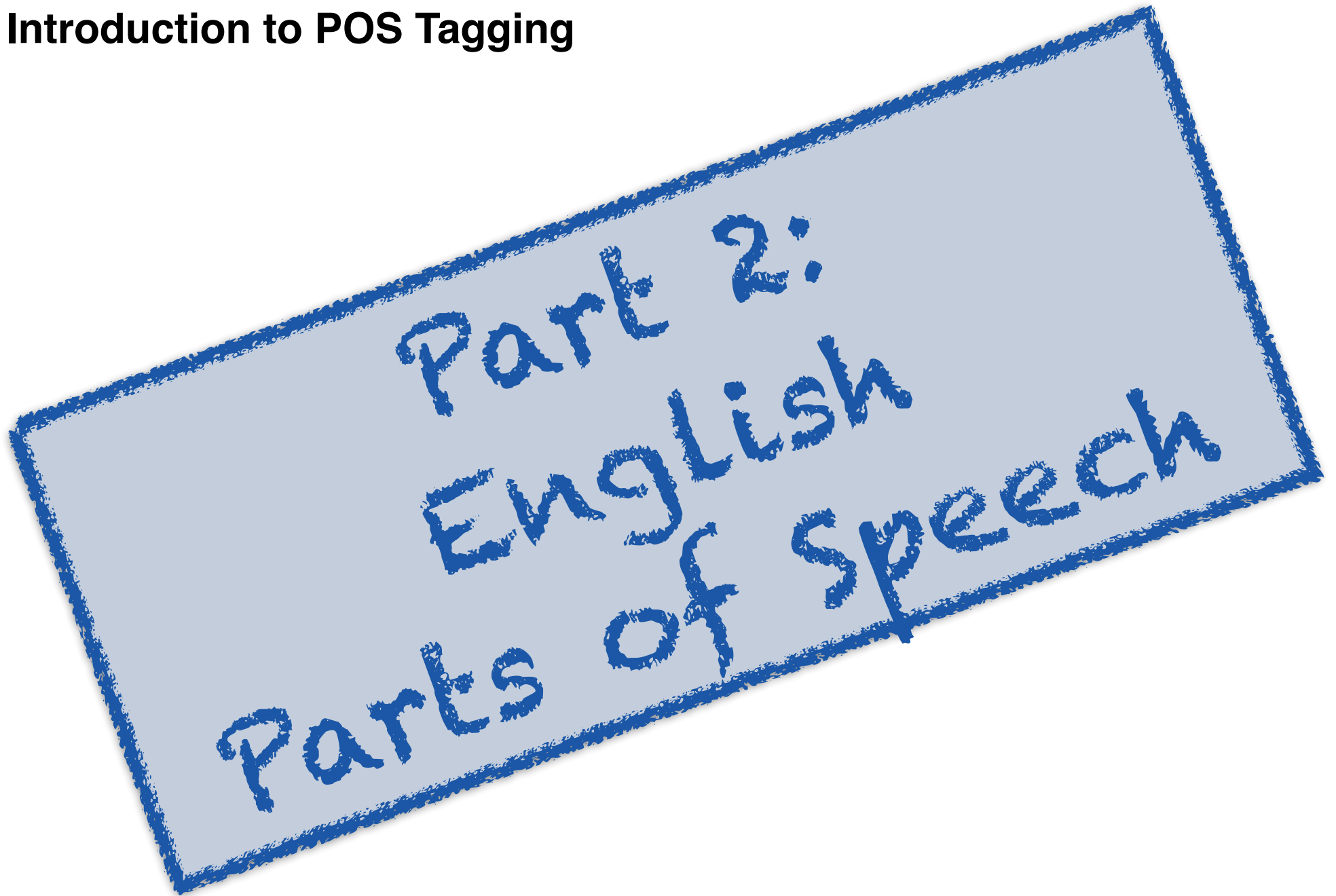
Part 3: Hidden Markov Models (Definition)

Friday's class: The Viterbi algorithm

Reading: Chapter 8



Lecture 09: Introduction to POS Tagging



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, 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

VCN: 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** (modifying a noun): *the blue book*

...**predicative use** (as arguments 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, Monday*)
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 describe **relations** between entities or between entities and events.

They occur **before noun phrases** to form 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

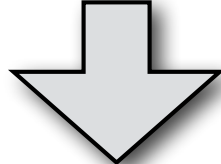
Lecture 09: Introduction to POS Tagging

Part 3:
Hidden Markov
Models (HMMs)
for POS Tagging

Statistical POS tagging

She promised to back the bill

$\mathbf{w} =$ $w^{(1)}$ $w^{(2)}$ $w^{(3)}$ $w^{(4)}$ $w^{(5)}$ $w^{(6)}$



$\mathbf{t} =$ $t^{(1)}$ $t^{(2)}$ $t^{(3)}$ $t^{(4)}$ $t^{(5)}$ $t^{(6)}$

PRP VBD TO VB DT NN

What is the most likely sequence of tags $\mathbf{t} = t^{(1)} \dots t^{(N)}$ for the given sequence of words $\mathbf{w} = w^{(1)} \dots w^{(N)}$?

$$\mathbf{t}^* = \operatorname{argmax}_{\mathbf{t}} P(\mathbf{t} \mid \mathbf{w})$$

POS tagging with generative models

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

$P(\mathbf{t}, \mathbf{w})$: the joint distribution of the labels we want to predict (\mathbf{t}) and the observed data (\mathbf{w}).

We decompose $P(\mathbf{t}, \mathbf{w})$ into $P(\mathbf{t})$ and $P(\mathbf{w} | \mathbf{t})$ since these distributions are easier to estimate.

Models based on joint distributions of labels and observed data are called **generative models**: think of $P(\mathbf{t})P(\mathbf{w} | \mathbf{t})$ as a stochastic process that first generates the labels, and then generates the data we see, based on these labels.

Hidden Markov Models (HMMs)

HMMs are the most commonly used generative models for POS tagging (and other tasks, e.g. in speech recognition)

HMMs make specific **independence assumptions** in $P(\mathbf{t})$ and $P(\mathbf{w} | \mathbf{t})$:

1) $P(\mathbf{t})$ is an n -gram (typically **bigram** or **trigram**) model over tags:

$$P_{\text{bigram}}(\mathbf{t}) = \prod_i P(t^{(i)} | t^{(i-1)})$$

$$P_{\text{trigram}}(\mathbf{t}) = \prod_i P(t^{(i)} | t^{(i-1)}, t^{(i-2)})$$

$P(t^{(i)} | t^{(i-1)})$ and $P(t^{(i)} | t^{(i-1)}, t^{(i-2)})$ are called **transition probabilities**

2) In $P(\mathbf{w} | \mathbf{t})$, each $w^{(i)}$ depends only on [is generated by/conditioned on] $t^{(i)}$:

$$P(\mathbf{w} | \mathbf{t}) = \prod_i P(w^{(i)} | t^{(i)})$$

$P(w^{(i)} | t^{(i)})$ are called **emission probabilities**

These probabilities don't depend on the position in the sentence (i) , but are defined over word and tag types.

With subscripts i, j, k , to index word/tag types, they become $P(t_i | t_j)$, $P(t_i | t_j, t_k)$, $P(w_i | t_j)$

Notation: t_i/w_i vs $t^{(i)}/w^{(i)}$

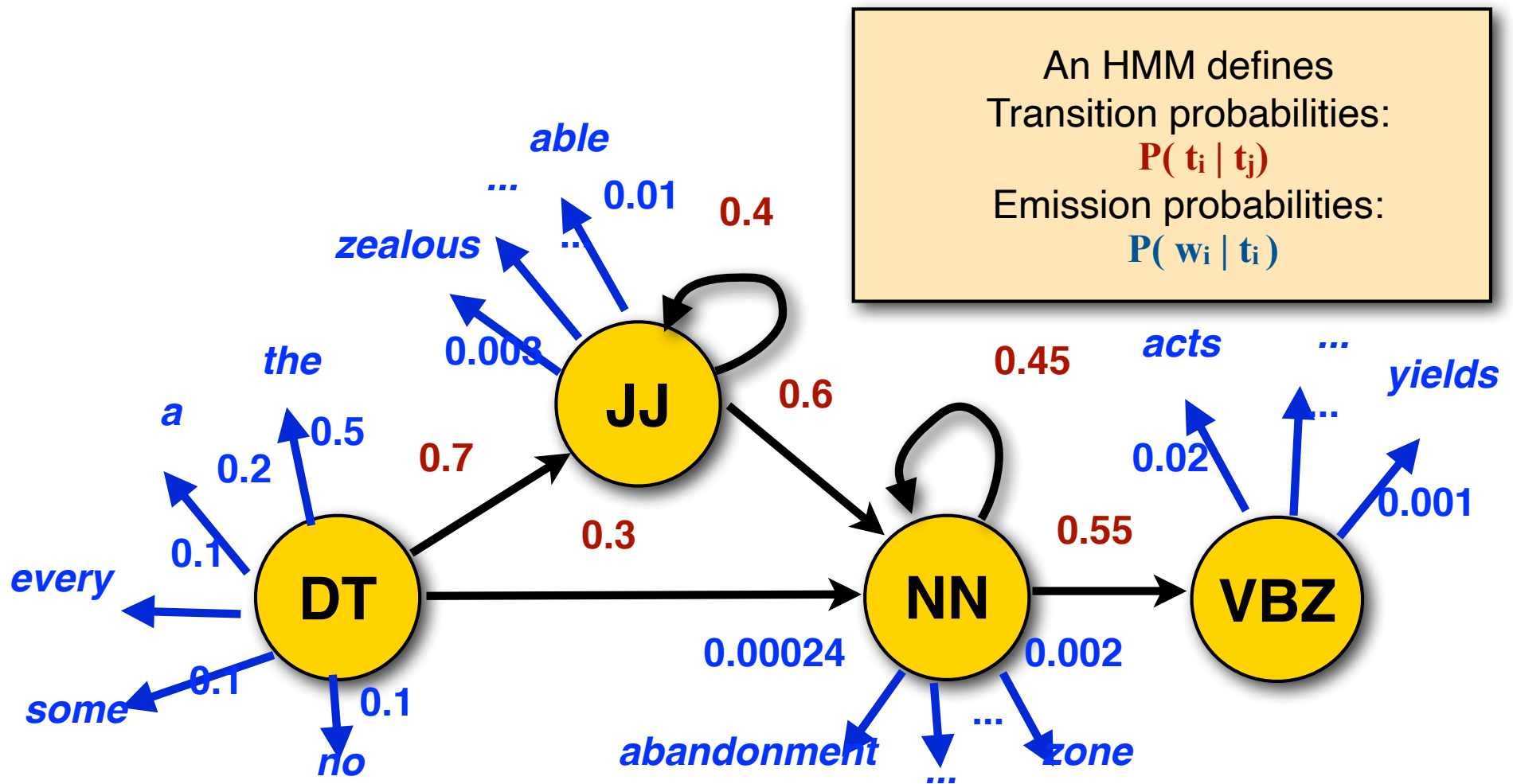
To make the distinction between the i -th word/tag in the vocabulary/tag set and the i -th word/tag in the sentence clear:

Use **superscript notation** $w^{(i)}$ for the **i -th token** in the **sentence/sequence**

and **subscript notation** w_i for the **i -th type** in the **inventory** (tagset/vocabulary)



HMMs as probabilistic automata

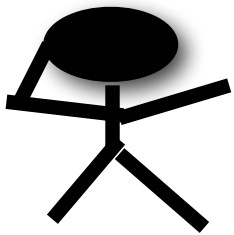


How would the
automaton for a trigram
HMM with transition probabilities
 $P(t_i \mid t_j t_k)$ look like?

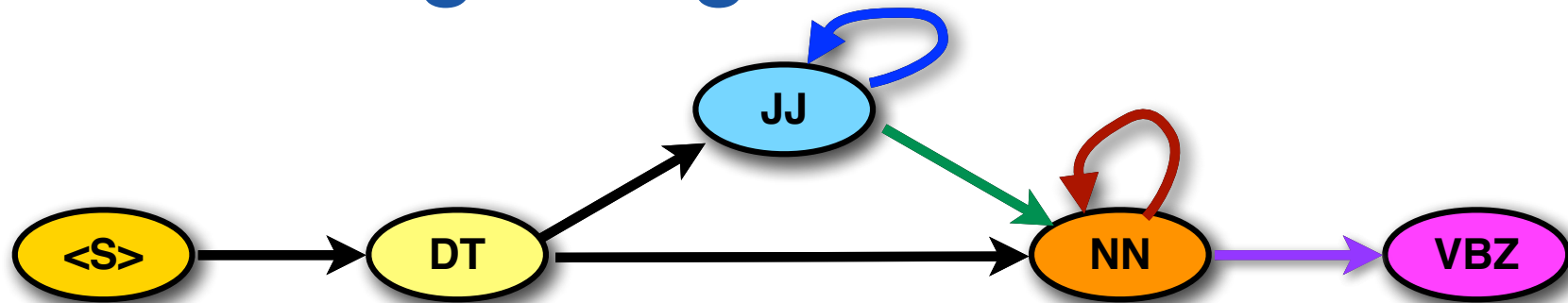
What about unigrams
or n-grams?

???

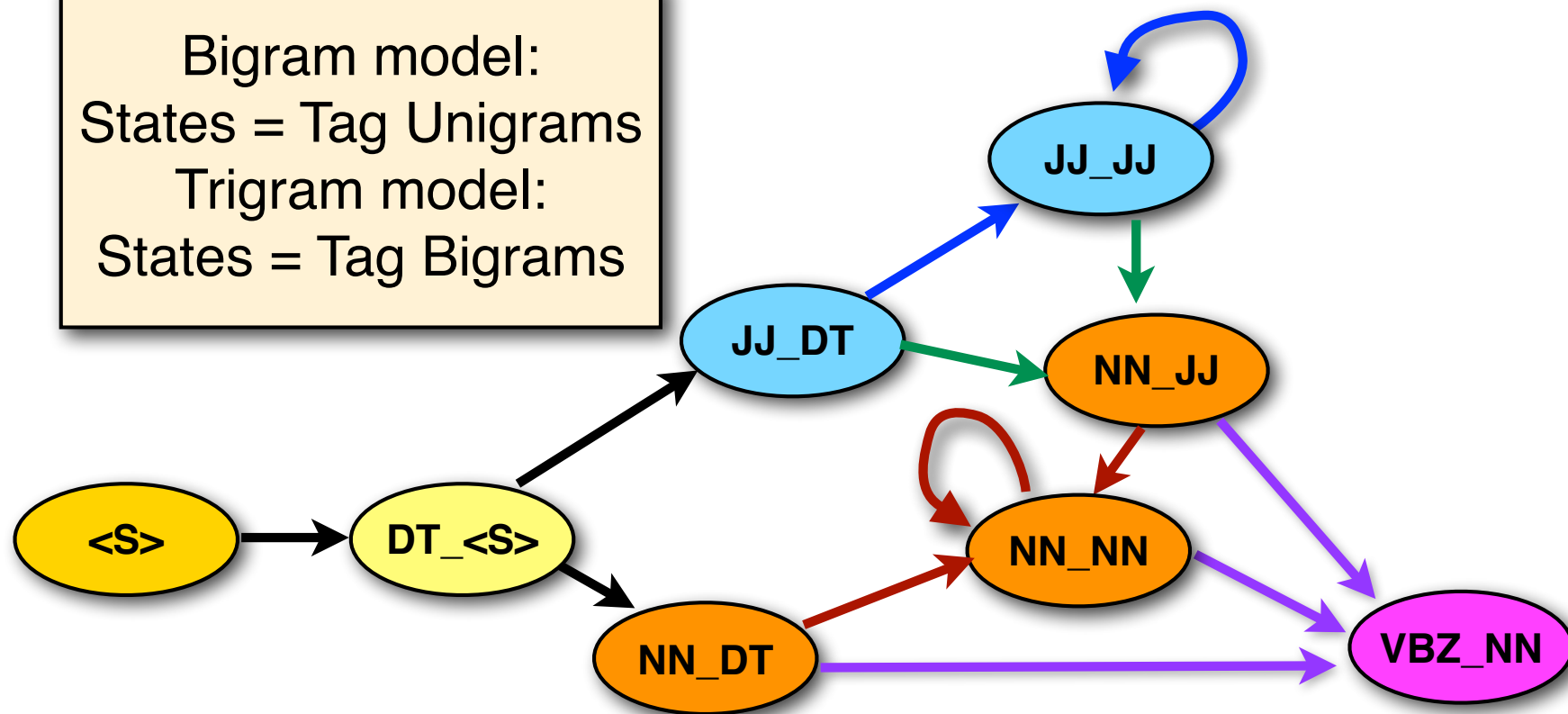
???



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** $Q = \{q_1, \dots, q_N\}$
with $Q_0 \subseteq Q$ a set of **initial states**
~~and $Q_F \subseteq Q$ a set of **final (accepting) states**~~ (Erratum: for POS tagging, no accepting are states required)
- an **output vocabulary** of M items $V = \{v_1, \dots, v_m\}$
- an $N \times N$ **state transition probability matrix** A
with a_{ij} the probability of moving from q_i to q_j .
($\sum_{j=1}^N a_{ij} = 1 \ \forall i; \ 0 \leq a_{ij} \leq 1 \ \forall i, j$)
- an $N \times M$ **symbol emission probability matrix** B
with b_{ij} the probability of emitting symbol v_j in state q_i
($\sum_{j=1}^M b_{ij} = 1 \ \forall i; \ 0 \leq b_{ij} \leq 1 \ \forall i, j$)
- an **initial state distribution vector** $\pi = \langle \pi_1, \dots, \pi_N \rangle$
with π_i the probability of being in state q_i at time $t = 1$.
($\sum_{i=1}^N \pi_i = 1 \ 0 \leq \pi_i \leq 1 \ \forall i$)

An example HMM

Transition Matrix A

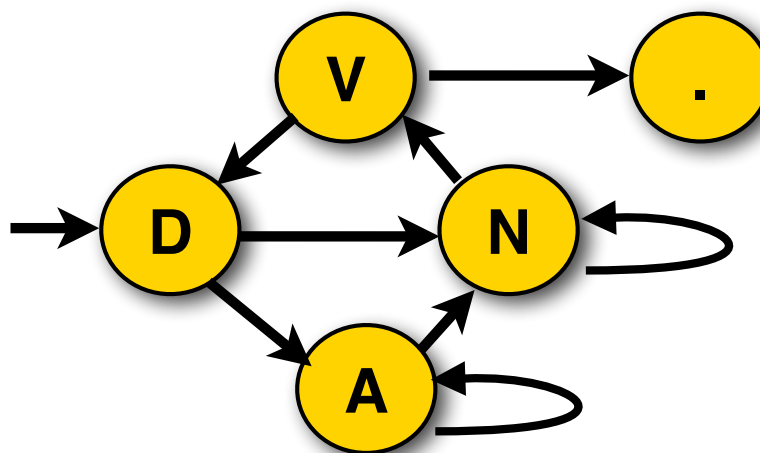
	D	N	V	A	.
D		0.8		0.2	
N		0.7	0.3		
V	0.6				0.4
A		0.8		0.2	
.					

Initial state vector π

	D	N	V	A	.
π	1				

Emission Matrix B

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



Building an HMM tagger

To build an HMM tagger, we have to:

Train the model, i.e. estimate its parameters
(the transition and emission probabilities)

Easy case: We have a corpus labeled with POS tags (supervised learning)

Harder case: We have a corpus, but it's just raw text without tags (unsupervised learning). In that case it really helps to have a dictionary of which POS tags each word can have

Define and implement a **tagging algorithm**
that finds the best tag sequence \mathbf{t}^*
for each input sentence \mathbf{w} :

$$\mathbf{t}^* = \operatorname{argmax}_{\mathbf{t}} P(\mathbf{t})P(\mathbf{w} \mid \mathbf{t})$$

[next lecture]

Learning an HMM from *labeled* data

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

```
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 ._.
```

Learning the transition probabilities:

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

Learning the emission probabilities:

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

Learning an HMM from *unlabeled* data

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

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

We can't count anymore.

We have to *guess* how often we'd *expect* to see $t_i t_j$ *etc.* in our data set. Call this expected count $\langle C(\dots) \rangle$

- Our estimate for the transition probabilities:

$$\hat{P}(t_j | t_i) = \frac{\langle C(t_i t_j) \rangle}{\langle C(t_i) \rangle}$$

- Our estimate for the emission probabilities:

$$\hat{P}(w_j | t_i) = \frac{\langle C(w_j - t_i) \rangle}{\langle C(t_i) \rangle}$$

These expected counts can be obtained via dynamic programming (the Forward-Backward algorithm)

Finding the best tag sequence

The **number of possible tag sequences** is **exponential** in the length of the input sentence:

Each word can have up to T tags.

Given a sentence with N words...

...there are up to T^N possible tag sequences.

We **cannot enumerate** all T^N possible tag sequences.

But we can exploit the **independence assumptions in the HMM** to define an efficient algorithm that returns the tag sequence with the highest probability.

[Viterbi algorithm; next lecture]

