Fundamental NLP Tools

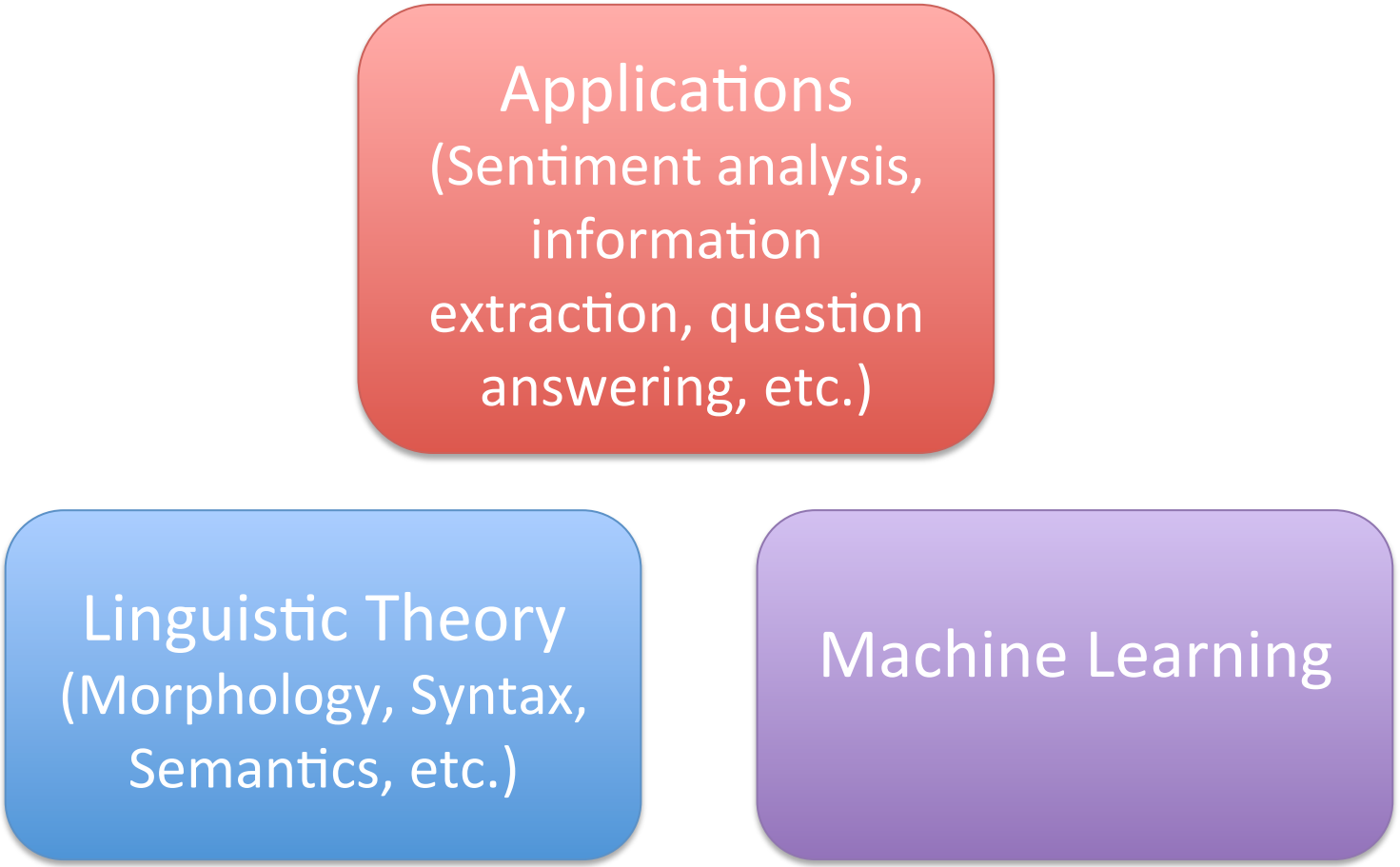
Mihai Surdeanu

Spring 2015

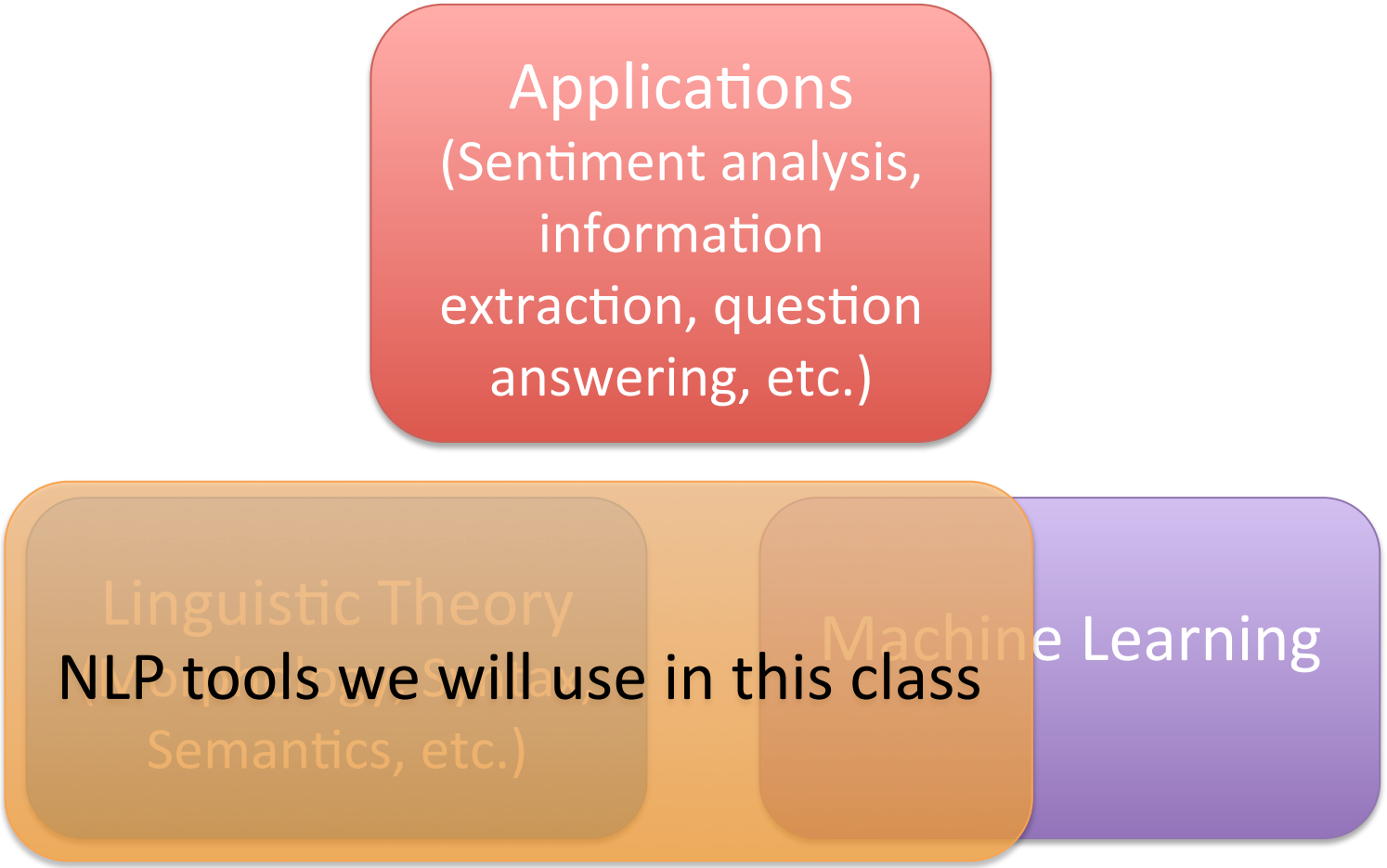
Warning

- Impossible to summarize a NLP course in one lecture…
- You really should take LING 439/539 or equivalent.

A simplified view of NLP



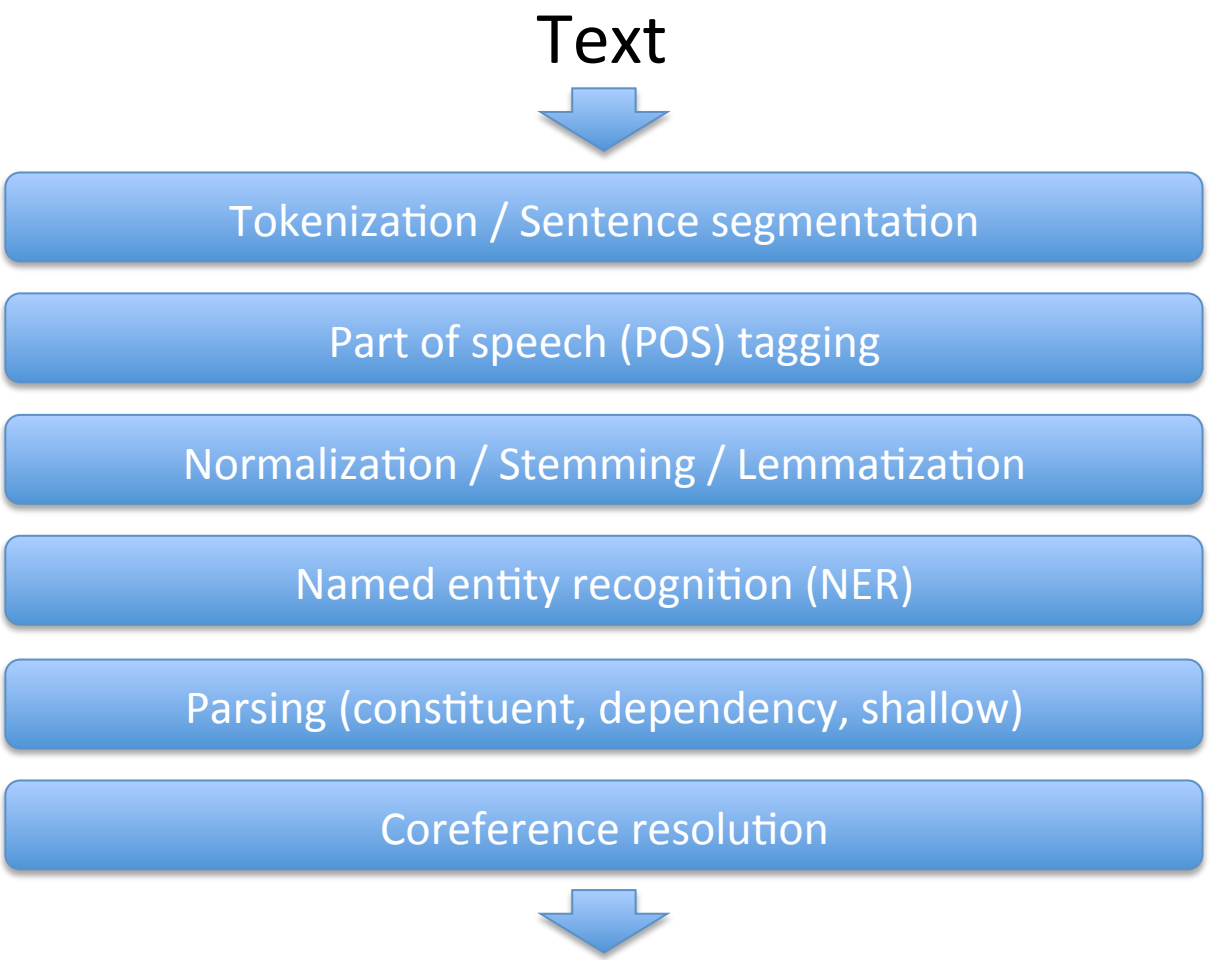
A simplified view of NLP



This lecture

- We will review the most typical NLP tools
- How to use them (with NLTK)
- We will cover implementation details in the next lecture

Typical NLP pipeline



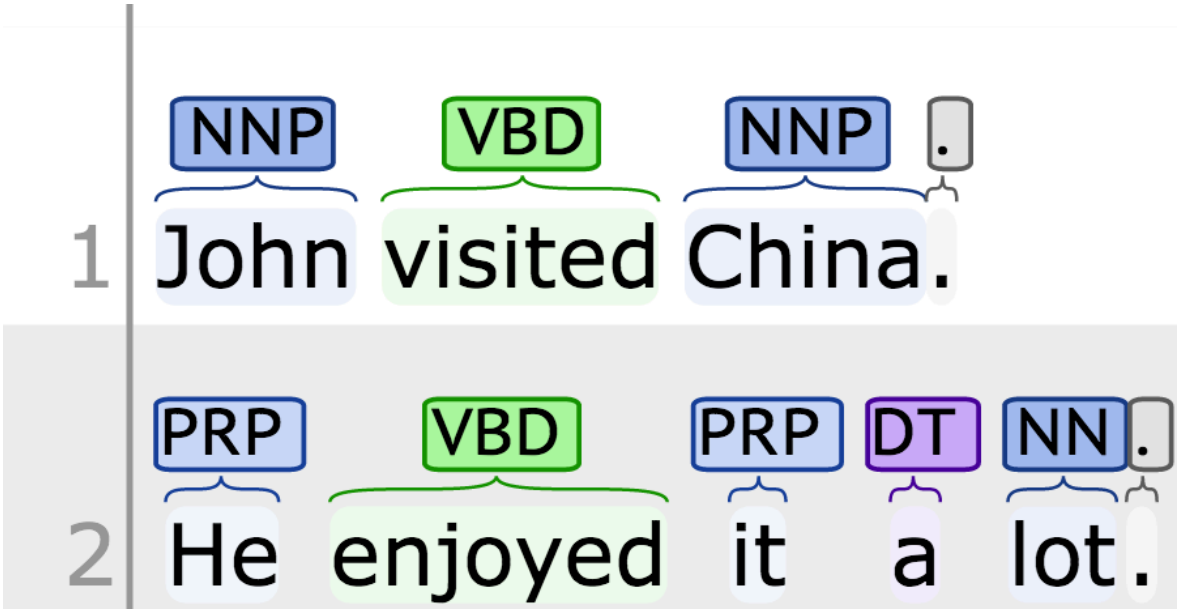
A run-through example

"John visited China. He enjoyed it a lot."

A run-through example:

After tokenization, sentence segmentation, and POS tagging

"John visited China. He enjoyed it a lot."



A run-through example:

After lemmatization/normalization

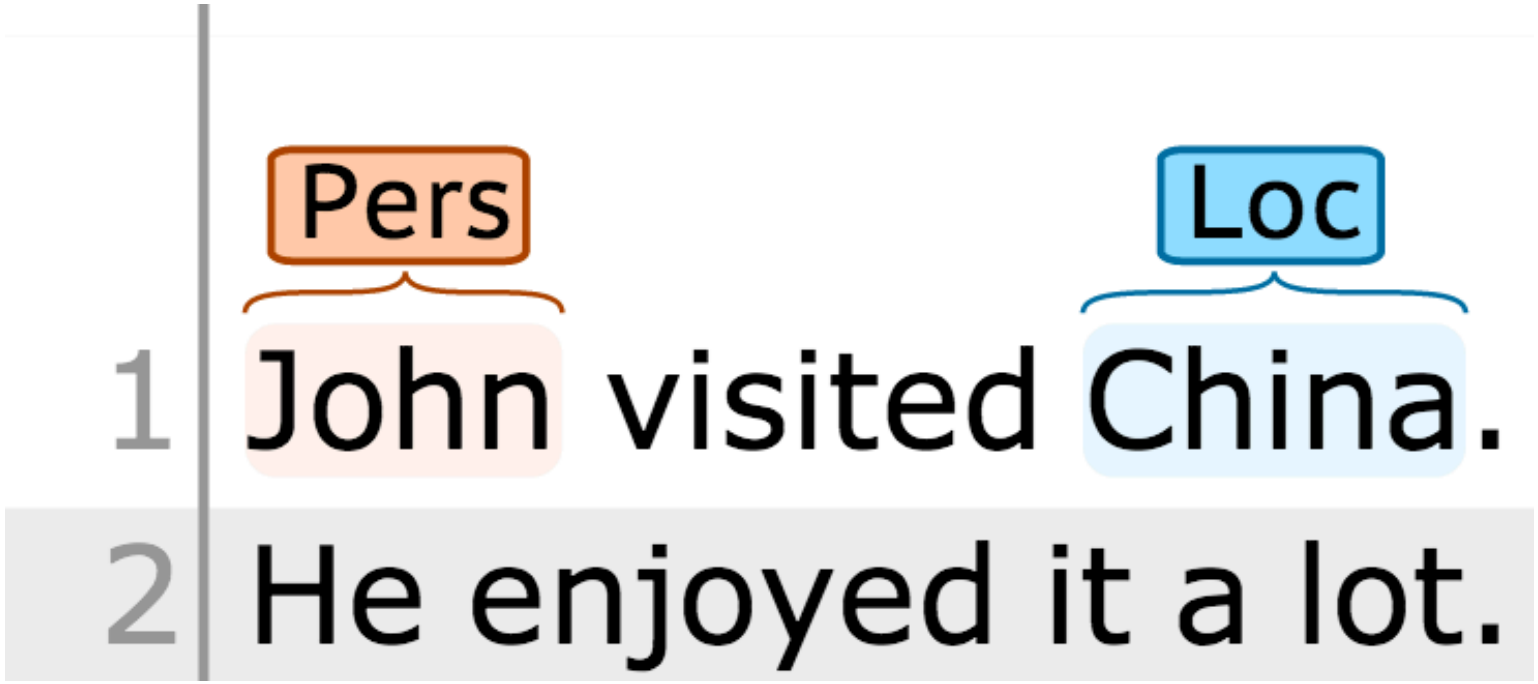
"John visited China. He enjoyed it a lot."

John visit China . He enjoy it a lot .

A run-through example:

After NER

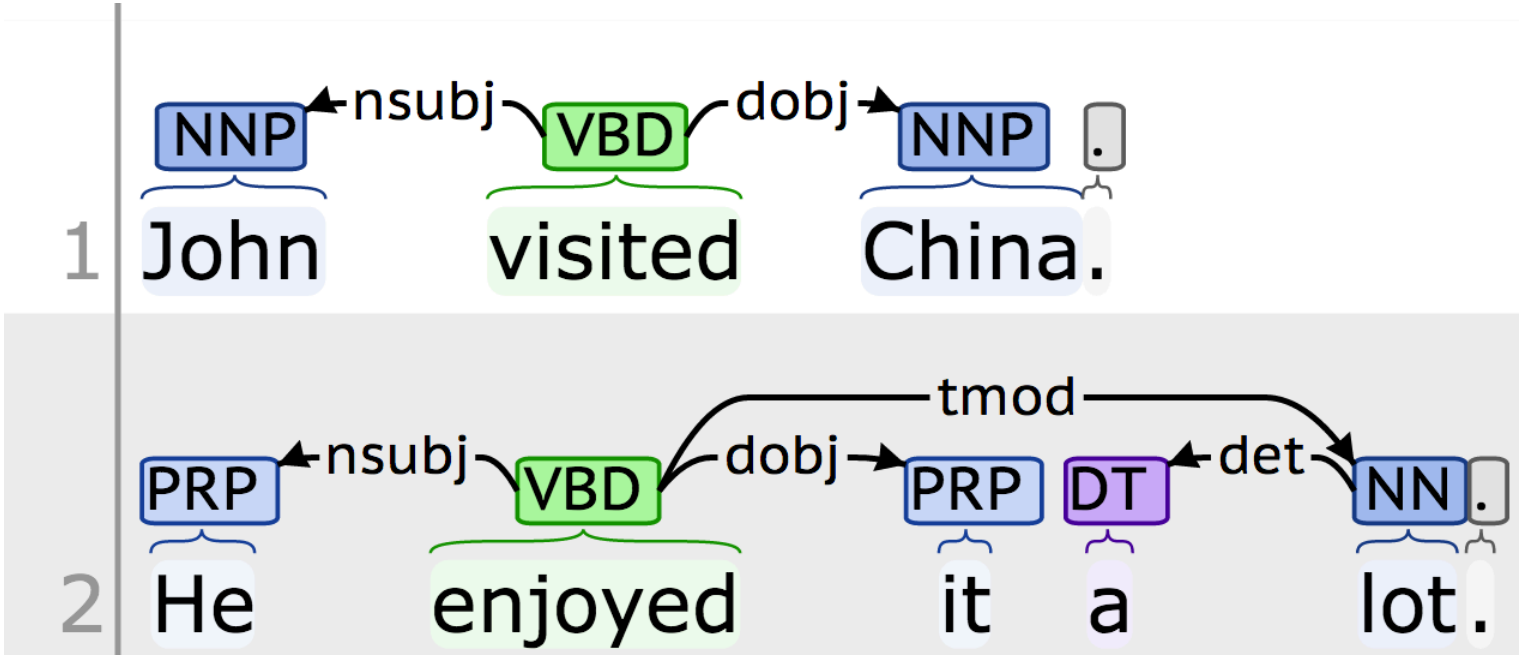
"John visited China. He enjoyed it a lot."



A run-through example:

After dependency parsing

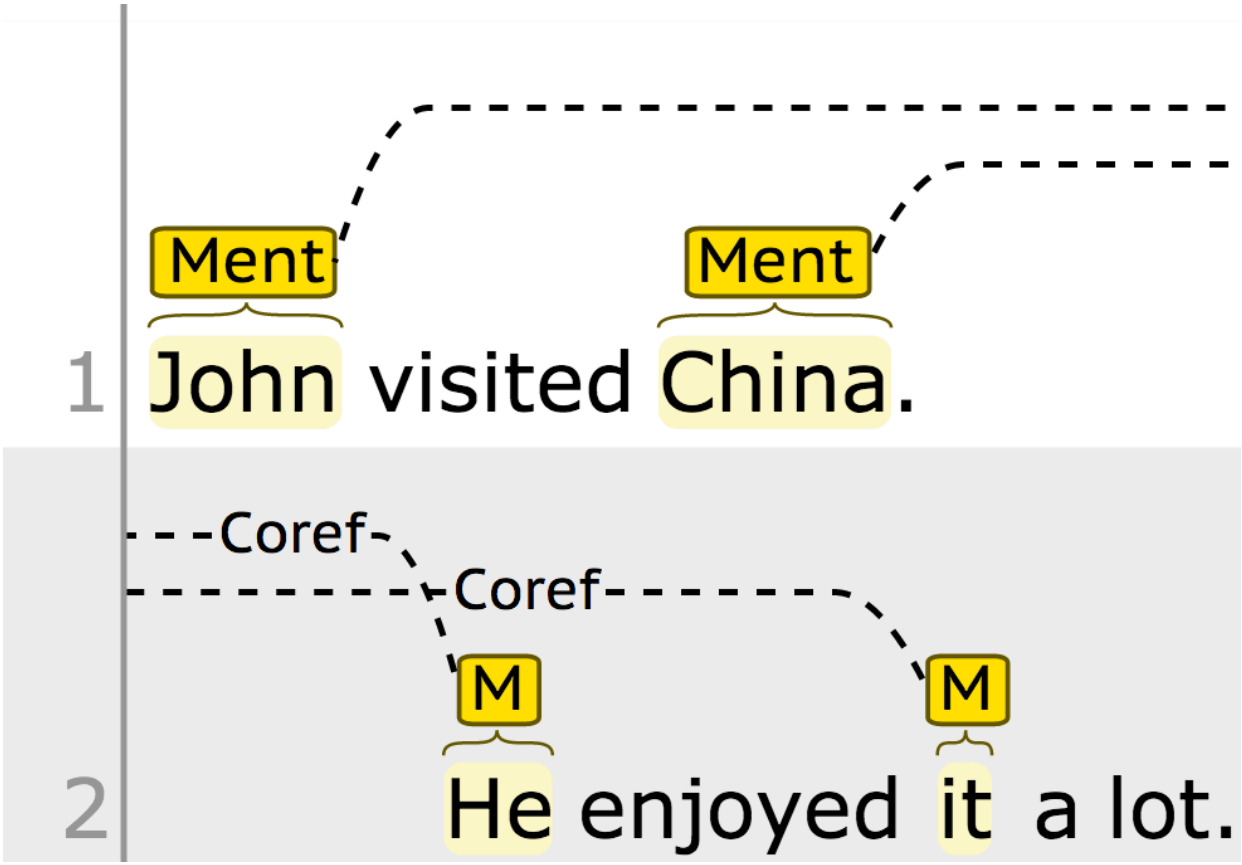
"John visited China. He enjoyed it a lot."



A run-through example:

After coreference resolution

"John visited China. He enjoyed it a lot."



Build your own visualizations

- The previous example was built using Stanford's CoreNLP
  - http://nlp.stanford.edu:8080/corenlp/
- Our group's visualizer
  - http://river.sista.arizona.edu:8080/bigmech/demo/enterRules
- Visualize constituent trees
  - http://www.ark.cs.cmu.edu/parseviz/

My favorite NLP packages

- NLTK (python)
  - http://nltk.org/
  - Has everything except coreference resolution
  - May not have state-of-the-art implementations
- Stanford's CoreNLP (java)
  - http://nlp.stanford.edu/software/corenlp.shtml
  - Includes everything
  - State-of-the-art implementations
  - Has Python wrappers
  - Some components may not be the fastest or most memory efficient

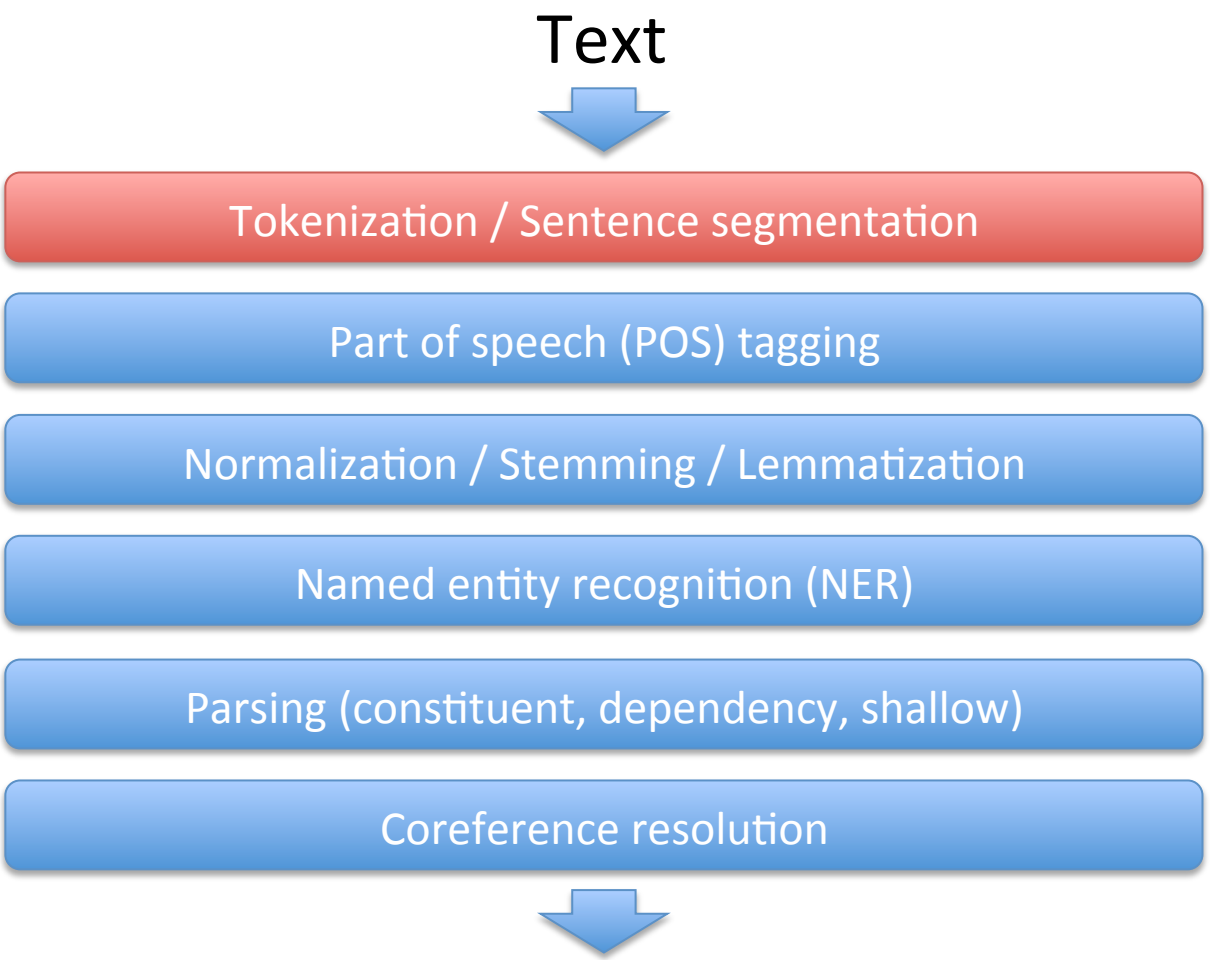
My favorite NLP packages

- OpenNLP (java)
  - http://opennlp.apache.org/
  - Includes everything
  - Very stable, but older algorithms…
- TweetNLP (java)
  - http://www.ark.cs.cmu.edu/TweetNLP/
  - Best tools for working with tweets
  - Only tokenization and POS tagging

My favorite NLP packages

- My group's 'processors':
  - https://github.com/sistanlp/processors
  - Started as a new API for CoreNLP
  - Now it is bigger: includes all CoreNLP and more
    - Dependency parser
    - Discourse parser
    - Framework for IE development
  - In Scala, but easy to use from Java

Typical NLP pipeline



Tokenization

- Segmenting running text into words and sentences.
- First impulse: “Just segment around space characters.” **Not so easy!**

> Mr. Sherwood said reaction to Sea Container’s proposal has been “very positive.” In New York Stock Exchange composite trading yesterday, Sea Containers closed At $62.625, up 62.5 cents.

- Segmenting on white spaces produces tokens such as: “cents.”, “said,”, “positive.””

Tokenization issues

- Many weird words
  - C++, C#, M*A*S*H
  - Emoticons: :) ;-) (see first project)
- Contractions
  - I'll, isn't, dog's, etc.
  - You want to split these in separate words because they impact grammar.
    - If not split, where's the verb in "I'm right."?

Tokenization issues
Hyphenation

- Some are clearly single words
  - e-mail, co-operate, A-1-plus
- Arguable cases, usually single words
  - non-lawyer, pro-Arab, so-called
- Hyphenation often used to indicate the correct grouping of words
  - the once-quiet study of superconductivity
  - a final “take-it-or-leave-it” offer
  - the 26-year-old

Tokenization issues
Whitespace not indicating a word break

- Multi-part names
  - "New York" is different from "York"
- Phrasal verbs
  - Make up, work out
- Phone numbers
  - USA: (202) 555-2230
  - France: 33 1 34 43 32 26
  - Sri Lanka: (94-1) 866854

What is a sentence?

- First impulse: “something ending with a ‘.’, ‘?’, or ‘!’”
- But some periods are used in abbreviations, or to indicate both abbreviations and end of sentence.
  - “Mr**.** Smith lives in the U**.**S**.**A**.** He is an American citizen**.**”

What is a sentence?

- When does a period indicate end of sentence, and when is it part of an abbreviation?
- Two solutions:
  - Heuristics
  - Machine learning model

Sentence segmentation
A simple algorithm

- Place putative sentence boundaries after all occurrences of . ? !
- Move the boundary after quotation marks, if any.
- Disqualify a period boundary if:
  - It is preceded by a known abbreviation that does not typically occur sentence finally, and is commonly followed by a proper name (e.g., *Mr.*)
  - It is preceded by a known abbreviation and not followed by an uppercase word.
- Disqualify a boundary with ? or ! If:
  - It is followed by a lowercase letter (or part of known name).
- Accept all other candidates as sentence boundaries.

Sentence segmentation
A better approach

- Treat it as a binary classification problem: punctuation is/is not sentence final
- A variety of features can be used
  - Case of preceding/following words
  - Length of preceding/following words
  - Actual words preceding/following punctuation
- Performance
  - About 90% of periods are sentence boundaries
  - Classifiers achieve an accuracy over 99%

Tokenization issues in other languages

- Word segmentation in Chinese

提供1小时电话响应，第二个工作日上门响应的快捷服务，为您节省时间成本。
服务遍全国，提供有限上门维修服务，为您节省维修成本。*

  - No white spaces used
  - How would you tokenize “thetabledownthere”? The longest match heuristic fails on the first word.
- Compound nouns in German
  - Lebensversicherungsgesellschaftsangestellter – life insurance company employee
  - It appears in English as well: “hard disk” vs. “harddisk”

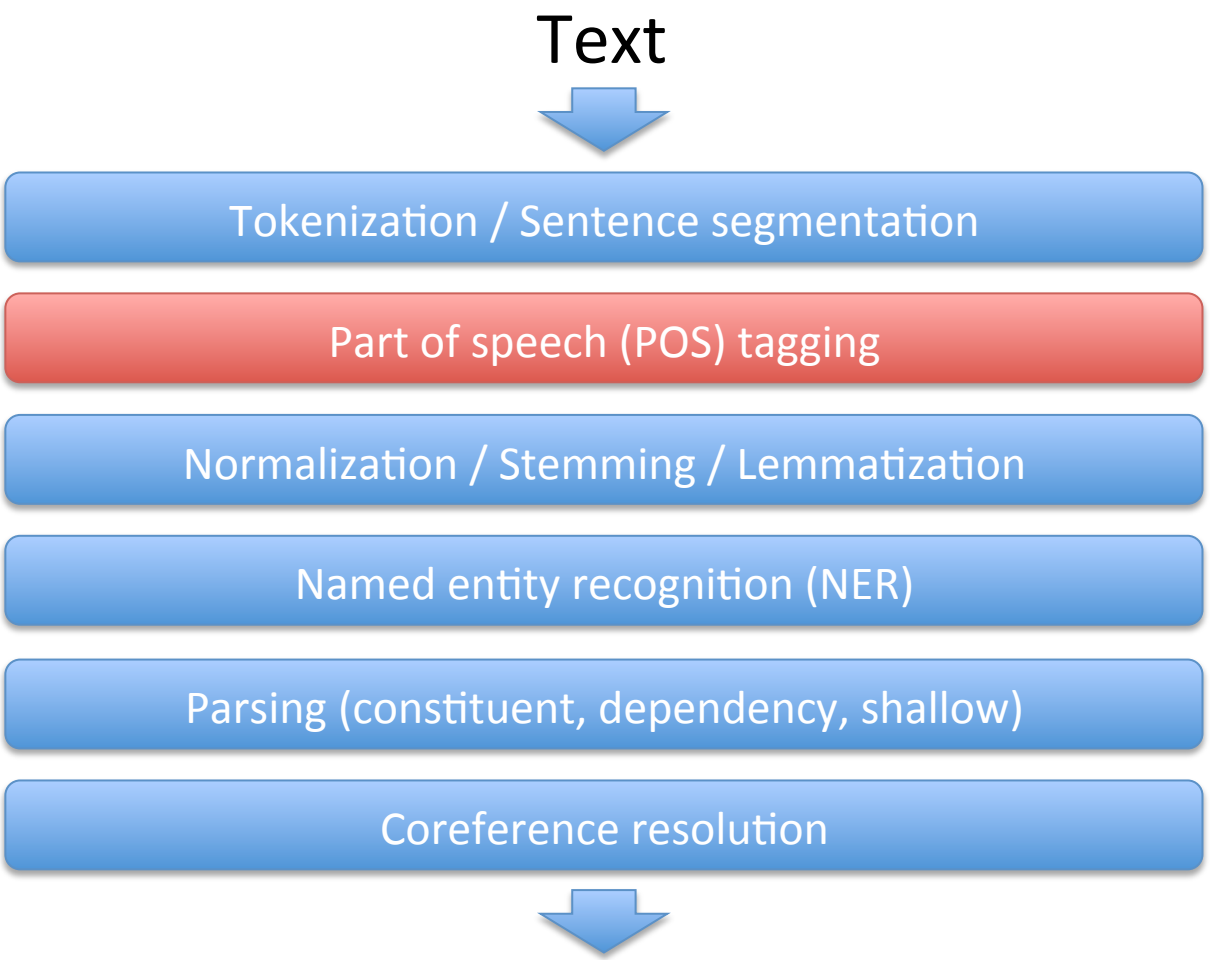
Implementations

- Tokenization is typically implemented with language-specific grammars or regular expressions compiled to finite state automata
  - JFlex: http://jflex.de/
  - Antlr: http://antlr.org/
- Sentence segmentation
  - Heuristics following tokenization (CoreNLP)
  - Classifier preceding tokenization (OpenNLP, NLTK)

NLTK example

```
>>> from nltk.tokenize import word_tokenize, sent_tokenize
>>> s = '''Good muffins cost $3.88\nin New York.  Please buy me two of them.\n
\nThanks.'''
>>> sent_tokenize(s)
['Good muffins cost $3.88\nin New York.', 'Please buy me two of them.', 'Thanks.']
>>> [word_tokenize(t) for t in sent_tokenize(s)]
[['Good', 'muffins', 'cost', '$', '3.88', 'in', 'New', 'York', '.'], ['Please', 'buy', 'me', 'two',
'of', 'them', '.'], ['Thanks', '.']]
```

Typical NLP pipeline



POS Tagging

- Process of performing automatic grammatical tagging for word categories
- *Part-of-speech tags* aka *word classes*, *morphological classes*, or *lexical tags*
- Implementation similar to other information extraction algorithms, so we will discuss it later in the course

(Mostly) English word classes

- Two class types
  - Closed: relatively fixed membership
  - Open
- Open classes
  - Nouns
  - Verbs
  - Adjectives
  - Adverbs

Open classes
Nouns

- Theoretically: people, places, things, etc.
- But nouns are defined syntactically and morphologically
  - Ability to occur with determiners (*a goat*), to take possessives (*IBM's revenue*), to have a plural (*goats*)
  - This allows other things to be labeled as nouns, e.g., verb nominalizations
- Grouped into proper nouns and common nouns
- Common nouns divided into
  - Count nouns: *one goat, two goats*
  - Mass nouns: *snow, communism*

Open classes
Verbs, adjectives, adverbs

- Verbs
  - Actions, processed
  - Some languages don't have the distinction between verbs and nouns: Tongan, Riau Indonesian
- Adjectives
  - Properties, qualities (e.g., color, age, value)
  - Some languages do not have adjectives: in Korean adjectives act as a subclass of verbs
- Adverbs
  - "Semantic hodge podge" (Jurafsky and Martin, 2008)
  - Modifiers of verbs, other adverbs and entire verb phrases
  - Directional or locative (*home, here, downhill*), degree (*extremely, very, somewhat*), manner (*slowly, delicately*), temporal (*yesterday, Monday*)

Closed classes

- Prepositions
- Determiners
- Pronouns
- Conjunctions
- Auxiliary verbs
- Particles
- Numerals

Closed classes
Prepositions and particles

- Prepositions
  - Occur before noun phrases
  - Semantically: spatial/temporal relations (*on it, before then, on time, beside herself*), other relations (*written by Shakespeare*)
  - English has fewer than 100 prepositions
- Particles
  - Resembles a preposition but is used in combination with a verb → phrasal verb (*rule out, turn down*)

Closed classes
Determiners and conjunctions

- Determiners
  - Articles: *a, an, the*
  - *This, that*
- Conjunctions
  - Join two phrases, clauses, or sentences
  - Coordinating conjunctions
    - Join two elements of equal status
    - *And, or, but*
  - Subordinating conjunctions
    - Some sort of embedded status
    - *That* in *"I thought that you might like some milk"*

Closed classes
Pronouns, auxiliary verbs

- Pronouns
  - Shorthand for referring to noun phrases, entities or events
  - Personal pronouns: *you, she, me,* etc.
  - Possessive pronouns: *my, your, his, her,* etc.
  - Wh-pronouns: *what, who, whom, whoever*
- Auxiliary verbs
  - Copula (*be*): connects subject with adjective or predicate nominals (*he is a duck*)
  - *Do, have*: for verb conjugations
  - Modals (*can, may, must*): indicator of mood

Closed classes
Others

- Interjections
  - *Oh, ah, hey, alas, uh, um*
- Negatives
  - *No, not*
- Politeness markers
  - *Please, thank you*
- Greetings
  - *Hello, goodbye*
- Existential there

Tagsets for English

- Implementations of the above word classes, to be used by actual POS taggers
- The most common one is the Penn Treebank tagset, containing 45 tags.
- Let's look at it
- Exercise!

Tagset Problems

- Some distinctions are quite hard for both humans and machines
- Between prepositions (IN), particles (RP), and adverbs (RB):
  - Mrs. Shaefer never got **around** to joining.
  - All we gotta do is go **around** the corner.
  - Chateau Petrus costs **around** 250.

Tagset Problems

- Some distinctions are quite hard for both humans and machines
- Between prepositions (IN), particles (RP), and adverbs (RB):
  - Mrs. Shaefer never got **around** to joining. **RP**
  - All we gotta do is go **around** the corner. **IN**
  - Chateau Petrus costs **around** 250. **RB**

Tagset Problems

- Noun modifiers, adjectives (JJ*) or nouns (NN*)?
  - **cotton** sweater
  - **income-tax** return
  - the **Gramm-Rudman** act
  - **Chinese** cooking
  - **Pacific** waters

Tagset Problems

- Noun modifiers, adjectives (JJ*) or nouns (NN*)?
  - **cotton** sweater — **NN**
  - **income-tax** return — **JJ**
  - the **Gramm-Rudman** act — **NNP**
  - **Chinese** cooking — **NN**
  - **Pacific** waters — **NN**

Tagset Problems

- Distinguishing between past participles (VBN) and adjectives (JJ)
  - They were **married** by the Justice of the Peace yesterday.
  - At the time, she was already **married**.

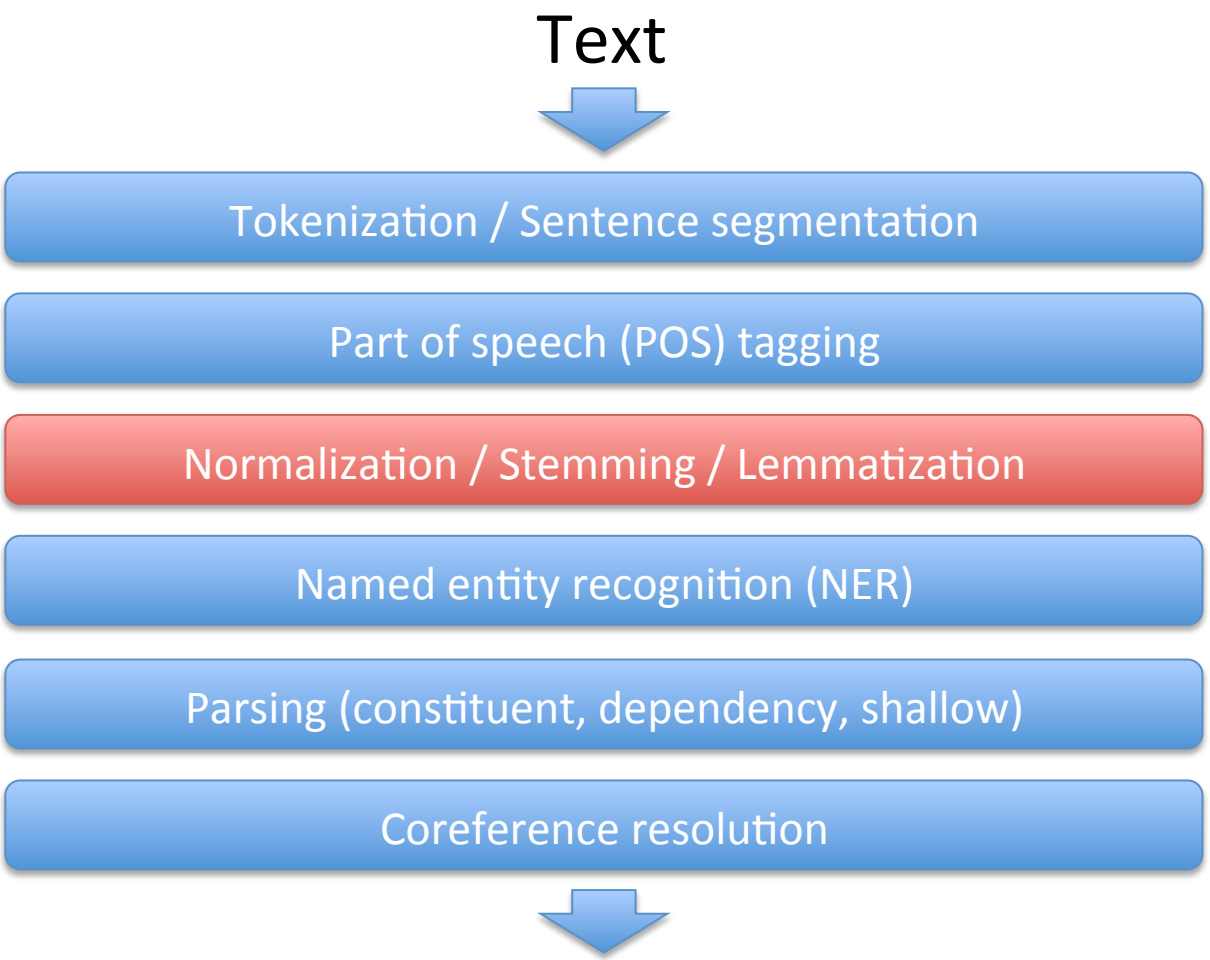
Tagset Problems

- Distinguishing between past participles (VBN) and adjectives (JJ)
  - They were **married** by the Justice of the Peace yesterday. **VBN**
  - At the time, she was already **married**. **JJ**

NLTK example

```
>>> import nltk
>>> text = nltk.word_tokenize("And now for something completely different")
>>> nltk.pos_tag(text)
[('And', 'CC'), ('now', 'RB'), ('for', 'IN'), ('something', 'NN'), ('completely', 'RB'), ('different', 'JJ')]
```

Typical NLP pipeline

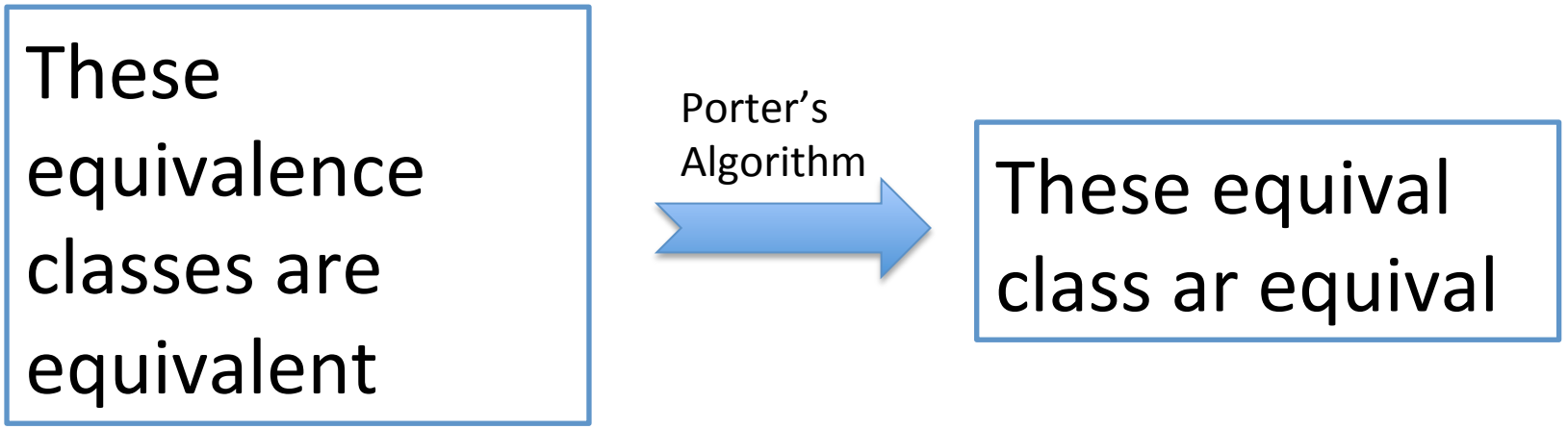


Normalization

- When searching a document, a query containing “USA” should match documents containing “U.S.A.”
- Implicit equivalence classes
  - Removing dots: “U.S.A.” ➔ ”USA”
  - Remove hyphens: “anti-discriminatory” ➔ “antidiscriminatory”
- Explicit equivalence classes
  - Different spellings: “color” and “colour”
  - Synonyms: “car” and “automobile”
- Case folding
  - USA ➔ usa

Stemming

- Reduce words to a common base form



Porter's Stemming Algorithm

- 5 phases of word reductions applied sequentially
  - sses → ss caresses → caress
  - ies → i ponies → poni
  - s → cats → cat
- Rules sensitive to the measure of a word
  - (m > 1) ement → replacement → replac
  - Does not change cement!

Porter's Stemming Algorithm

- Pros
  - Very fast
  - Does not require POS information
- Cons

Errors of commission		Errors of omission	
organization	organ	European	Europe
doing	doe	analysis	analyzes
numerical	numerous	noise	noisy
policy	police	sparse	sparsity

Lemmatization

- Grouping together the different inflected forms of a word so they can be analyzed as a single item.
- “walk”, “walked”, “walks”, “walking” ➔ lemma:”walk”

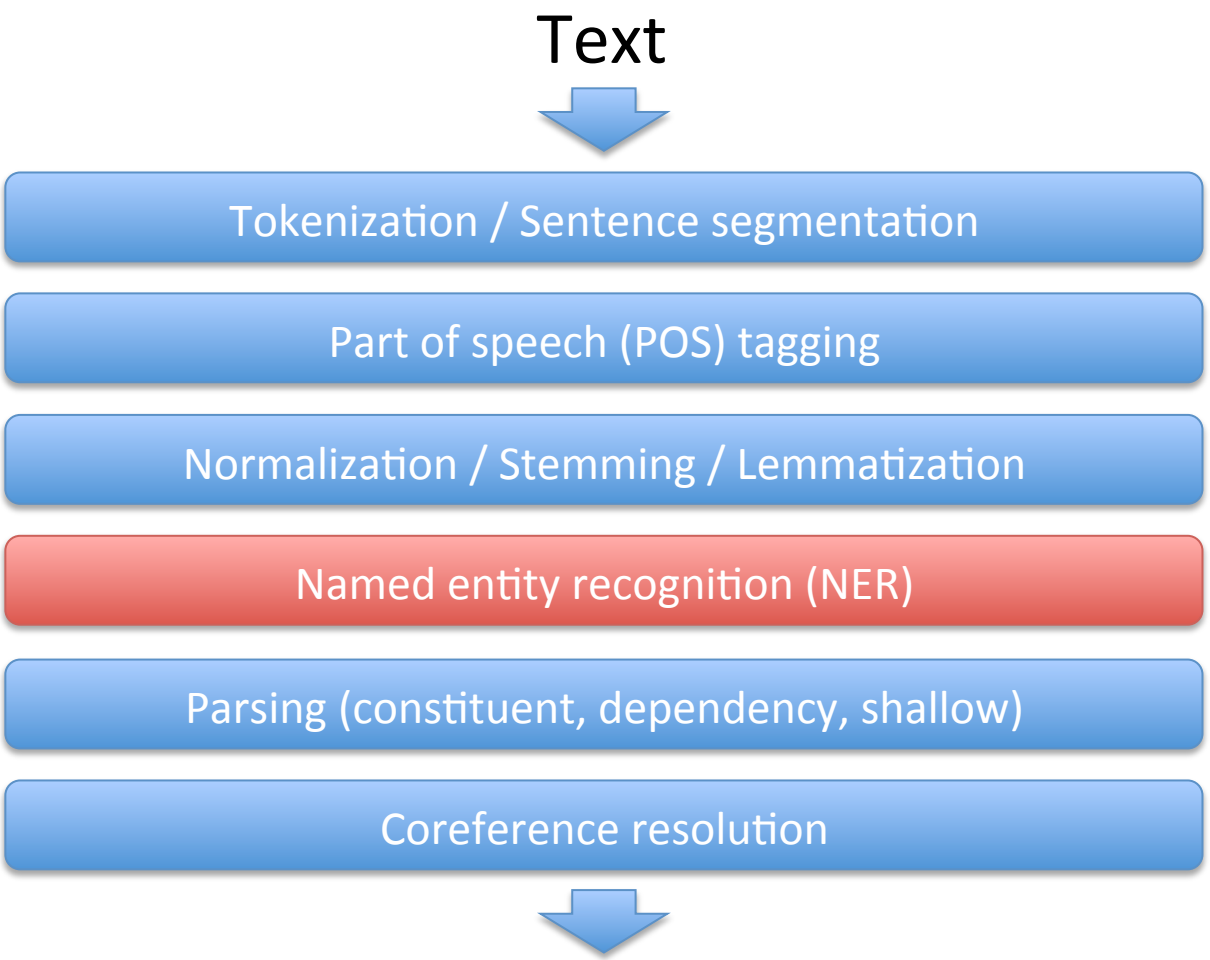
Lemmatization

- Implemented using lexical dictionaries such as WordNet
- Pros
  - More accurate than stemming
- Cons
  - Slower than stemming
  - May require POS tags

NLTK example

```
>>> import nltk
>>> text = nltk.word_tokenize("And now for something completely different")
>>> porter = nltk.PorterStemmer()
>>> [porter.stem(t) for t in text]
['And', 'now', 'for', 'someth', 'complet', 'differ']
>>> wnl = nltk.WordNetLemmatizer()
>>> [wnl.lemmatize(t) for t in text]
['And', 'now', 'for', 'something', 'completely', 'different']
```

Typical NLP pipeline



Named entity recognition

- Detection and classification of named entities in text
- In reality, a good NER will identify:
  - Named entities
  - Numeric entities
  - Temporal expressions
- Can be viewed as the first end-user NLP application.
- But because many NERs are available out of the box, in this class we will consider NER as a tool to be used to construct higher-level applications.

Named entity types

Type	Tag	Sample categories
People	PER	Individual, fictional characters, small groups
Organization	ORG	Companies, agencies, political parties, religious groups, sports teams
Location	LOC	Physical extents, mountains, lakes, seas
Geo-political entity	GPE	Countries, states, provinces, counties
Facility	FAC	Bridges, buildings, airports
Vehicle	VEH	Planes, trains, automobiles

Named entity examples

Type	Example
People	**Turing** is often considered the father of computer science.
Organization	The **IPCC** said it is likely that future cyclones will be more intense.
Location	The **Mt. Sanitas** loop hike begins at the base of **Sunshine Canyon**.
Geo-political entity	**Palo Alto** will raise parking fees.
Facility	Drivers were advised to consider the **Lincoln Tunnel**.
Vehicle	The updated **Mini Cooper** retains its charm and agility.

Numeric entities

Type	Tag	Example
Money	MONEY	This laptop costs **$450**.
Number	NUMBER	He was the **42nd** president.

Temporal expressions

Type	Tag	Example
Time	TIME	This class starts at **8:30am**.
Date	DATE	The first lecture is on **August 26, 2013**.

Differences between NER systems

- Stanford's CoreNLP recognizes:
  - PERSON, LOCATION (collapsing locations and GPEs), ORGANIZATION, and MISC (other miscellaneous names, including vehicles and facilities)
  - MONEY, NUMBER
  - DATE, TIME
- NLTK recognizes:
  - PERSON, ORGANIZATION, GPE

Ambiguities in NER

Name	Possible categories
Washington	Person, Location, Political Entity, Organization, Facility
Downing St.	Location, Organization
Louis Vuitton	Person, Organization, Product

Washington was born into slavery on the farm of Burroughs.
Washington went up 2 games to 1 in the four-games series.
Blair arrived in **Washington** for what may be his last state visit.
In June, **Washington** passed a primary seatbelt law.
The **Washington** had proved to be a leaky ship.

Ambiguities in NER

Name	Possible categories
Washington	Person, Location, Political Entity, Organization, Facility
Downing St.	Location, Organization
Louis Vuitton	Person, Organization, Product

Washington was born into slavery on the farm of Burroughs.	**PER**
Washington went up 2 games to 1 in the four-games series.	**ORG**
Blair arrived in **Washington** for what may be his last state visit.	**LOC**
In June, **Washington** passed a primary seatbelt law.	**GPE**
The **Washington** had proved to be a leaky ship.	**FAC**

NER as sequence labeling

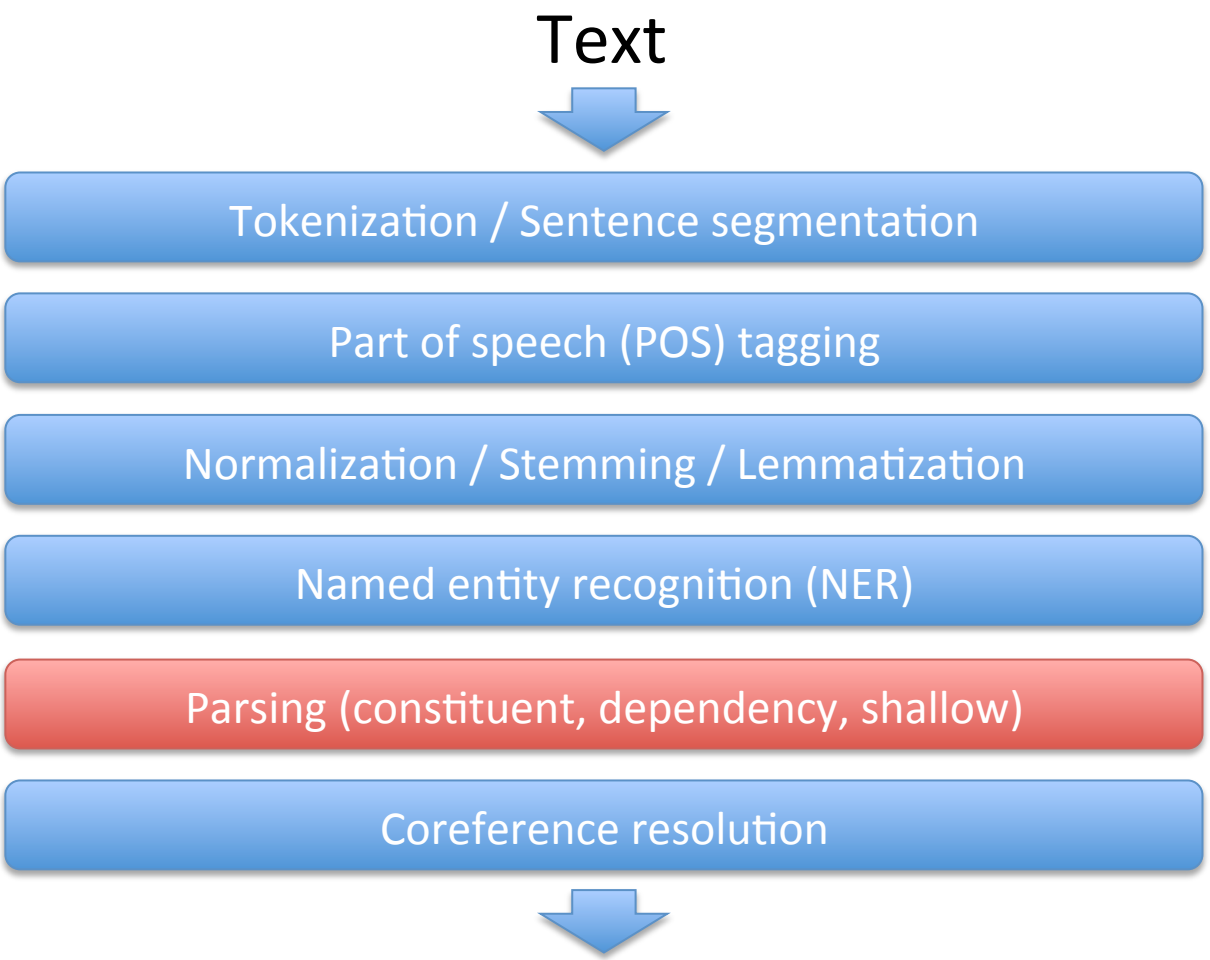
- Most entity mentions span multiple sentences
- Dedicated representations
  - Most common: IOB
  - Stanford: IO
  - UIUC: BILOU

Words	Label
American	B-ORG
Airlines	I-ORG
,	O
a	O
unit	O
of	O
AMR	B-ORG
Corp.	I-ORG
,	O
immediately	O
matched	O
the	O
move	O
,	O
Tim	B-PER
Wagner	I-PER
said	O
.	O

NLTK example

```
>>> import nltk
>>> text = nltk.word_tokenize(“Mr. John Smith from IBM visited China on 2010-01-31.")
>>> tagged = nltk.pos_tag(text)
>>> nltk.ne_chunk(tagged)
Tree('S', [Tree('PERSON', [('Mr.', 'NNP')]), Tree('PERSON', [('John', 'NNP'), ('Smith', 'NNP')]), ('from', 'IN'), Tree('ORGANIZATION', [('IBM', 'NNP')]), ('visited', 'VBD'), Tree('GPE', [('China', 'NNP')]), ('on', 'IN'), ('2010-01-31', 'CD'), ('.', '.')])
```

Typical NLP pipeline



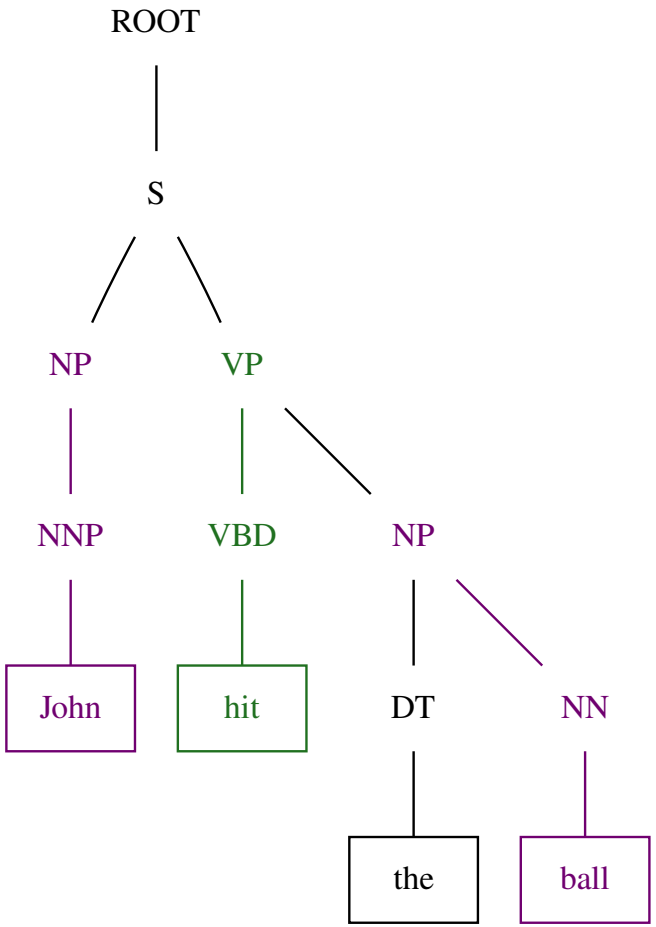
Syntax representations

- Syntactic parsing
  - The task of recognizing a sentence and assigning a syntactic structure to it.
- Multiple structures possible
  - Constituency-based
  - Dependency-based
  - Shallow

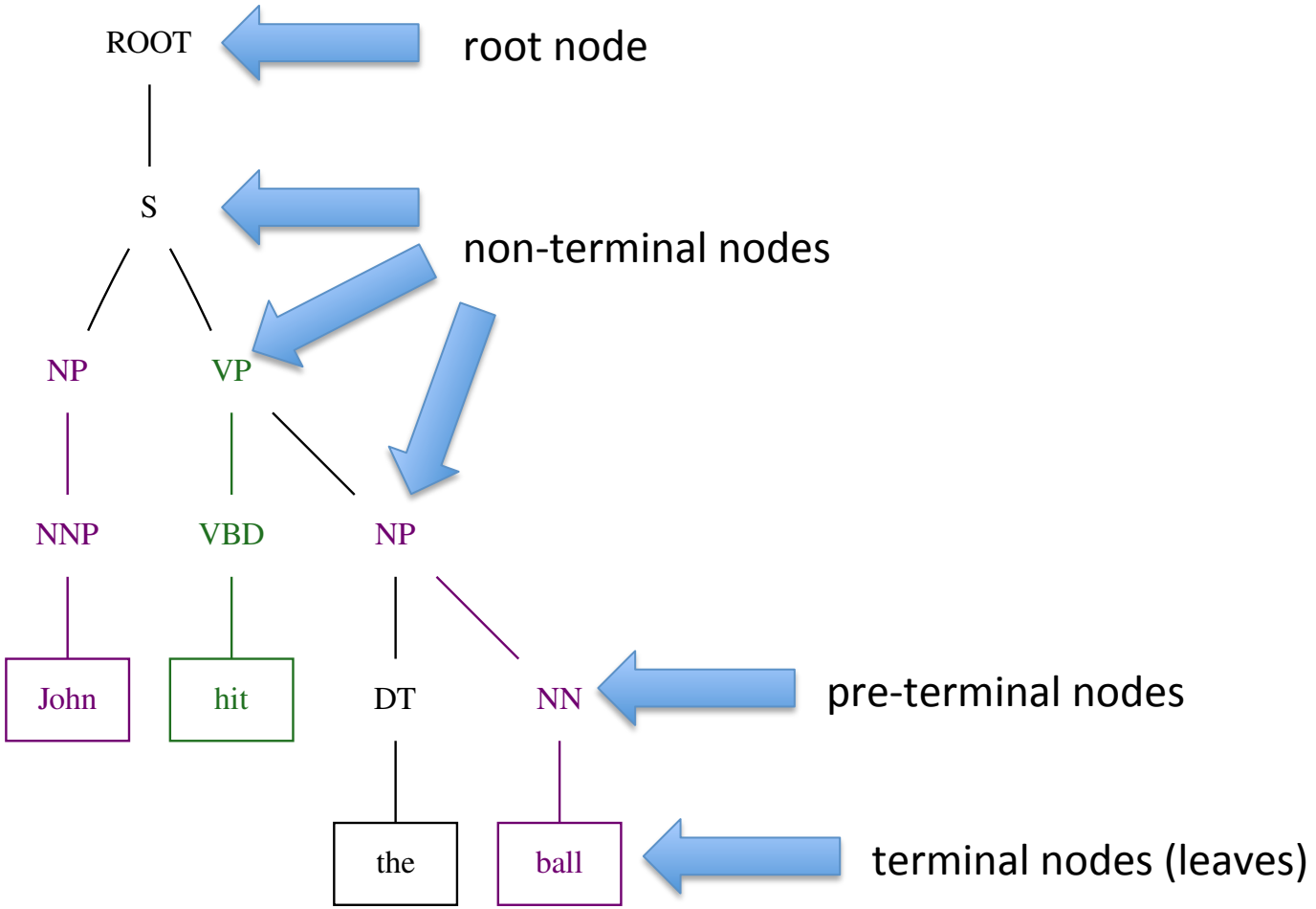
Constituency-based syntax

- Constituent parse tree
  - Ordered, rooted tree that stores the output of some grammar applied to the input sentence
  - Most grammars these days are Probabilistic Context Free Grammars (PCFG)

Constituency-based parse tree



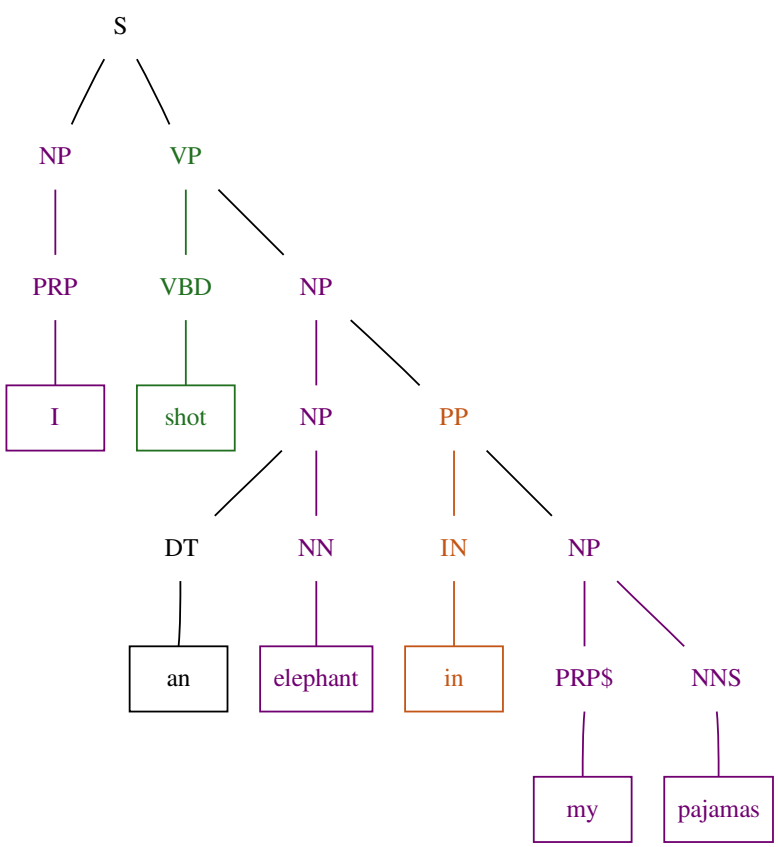
Constituency-based parse tree

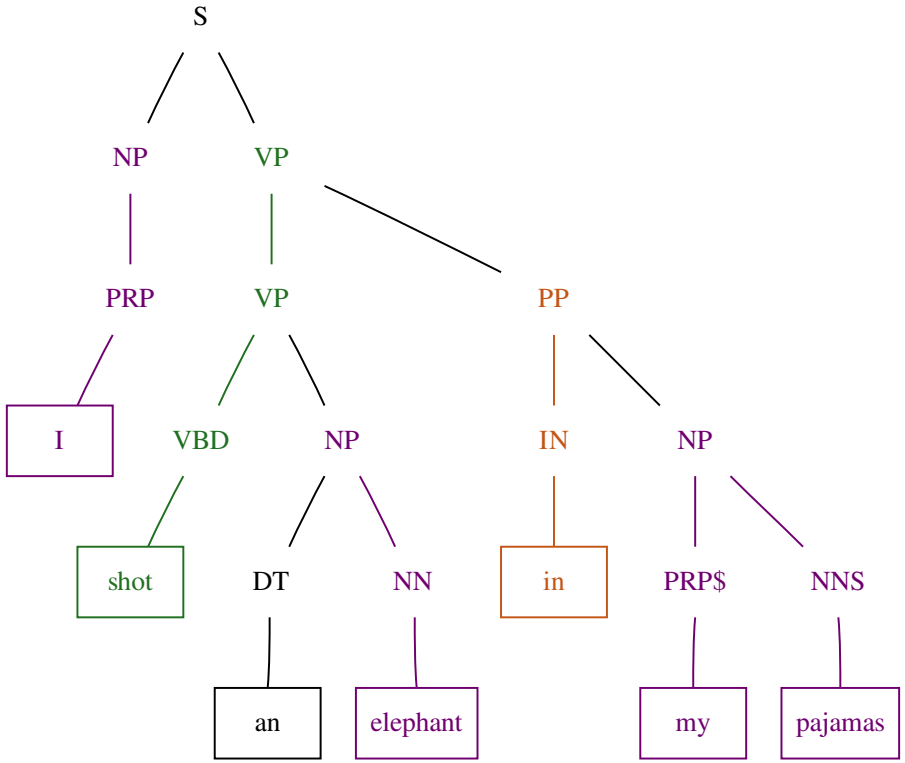


Structural ambiguity

Attachment ambiguity

"One morning I shot an elephant in my pajamas. How he got into my pajamas I don't know."





Structural ambiguity
Coordination ambiguity

- "old men and women"
  - [old [men and women]]
  - [[old men] and [women]]
- President Kennedy today pushed aside other White House business to devote all his time to working on the Berlin crisis address he will deliver tomorrow night to the American people over nationwide television and radio.
  - [nationwide [television and radio]] or [[nationwide television] and [radio]]?
  - Other?

Treebanks

- Treebank = collection of parse trees examples to be given to a parser for training
- The most famous one is the Penn Treebank
  - Contains over 45,000 annotated sentences
  - A *de facto* standard for most work in parsing

Most common Penn Treebank tags

Tag	Definition
S	Simple clause (sentence)
SBAR	Relative clause and subordinate clause
SBARQ	*Wh*-questions
SQ	Yes/no questions, clause inside SBARQ,
SINV	Sentences with subject-auxiliary inversion
ADJP	Adjective phrase
ADVP	Adverbial phrase
NP	Noun phrase
PP	Prepositional phrase
QP	Quantifier phrase (inside NP)
VP	Verb phrase
WHNP	*Wh*-noun phrase
WHPP	*Wh*-prepositional phrase

Most common Penn Treebank tags

Tag	Example
S	Casey threw the ball.
SBAR	The person **who threw the ball**.
SBARQ	Who threw the ball?
SQ	Who **threw the ball**? **Did Casey throw the ball?**
SINV	Never had I seen such a place.
ADJP	the **best and brightest** students
ADVP	He ran **quickly**.
NP	the higher prices
PP	Casey threw the ball **to Jane**.
QP	**from 10 to 15** monkeys
VP	Casey **threw the ball**.
WHNP	The person **who** threw the ball.
WHPP	of which, by whose authority

All Penn Treebank tags

- 26 constituent tags
- View them all on the course web page.

- Example!

Matching patterns over constituent trees

- Several NLP applications can be reduced to matching certain syntactic patterns over constituent trees
- Tregex and Tsurgeon
  - Tregex: matches patterns over trees
  - Tsurgeon: tree manipulation

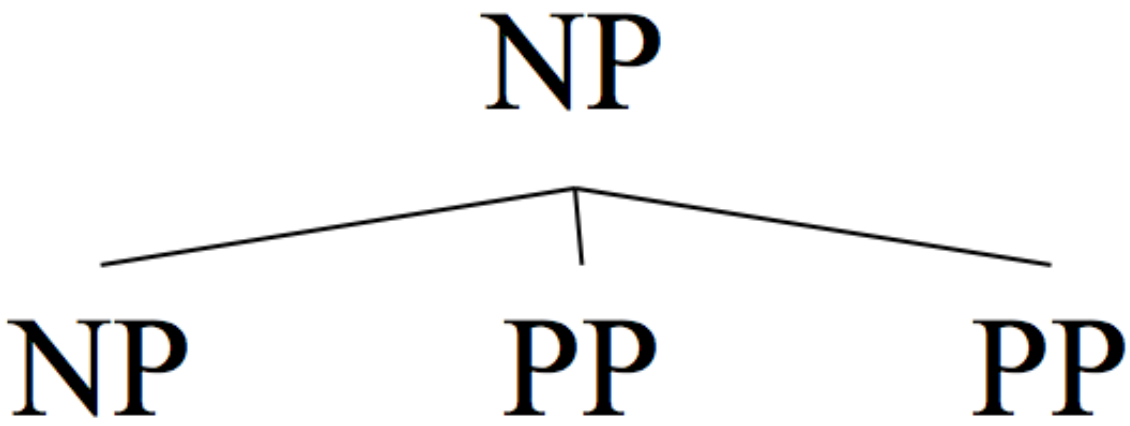
Tregex node-node relations

`A << B`	A dominates B
`A < B`	A immediately dominates B
`A <<: B`	B is a unary descendent of A
`A $++ B`	A is a left sister of B
`A $+ B`	A is the immediate left sister of B
`A <+(C) B`	A dominates B through a chain of nodes each of which matches description C
`A . B`	A immediately precedes B
`A .+(C) B`	A precedes B through a chain of nodes each of which matches description C
`A <# B`	B is the head daughter of A
`A <<# B`	B heads A (through transitive closure of <#)

Tregex example

Tregex Pattern:

```
NP=np < (NP $+ (PP $+ PP=pp2))
```



Dependency syntax

- All nodes are terminal
- Does not acknowledge the distinction between terminal and non-terminal nodes.
- Simpler, fewer nodes (no non-terminals)
- Faster to generate. Good linear-time algorithms exist.
  - See shift-reduce parsers such as maltparser.org
- No functionality lost for many applications.

Generating dependencies

- Most dependency representations are created from constituency-based trees by creating binary relations between constituent head words.
- Head word
  - Most important word in a constituent
  - Identified using linguistic heuristics

Head word identification rules

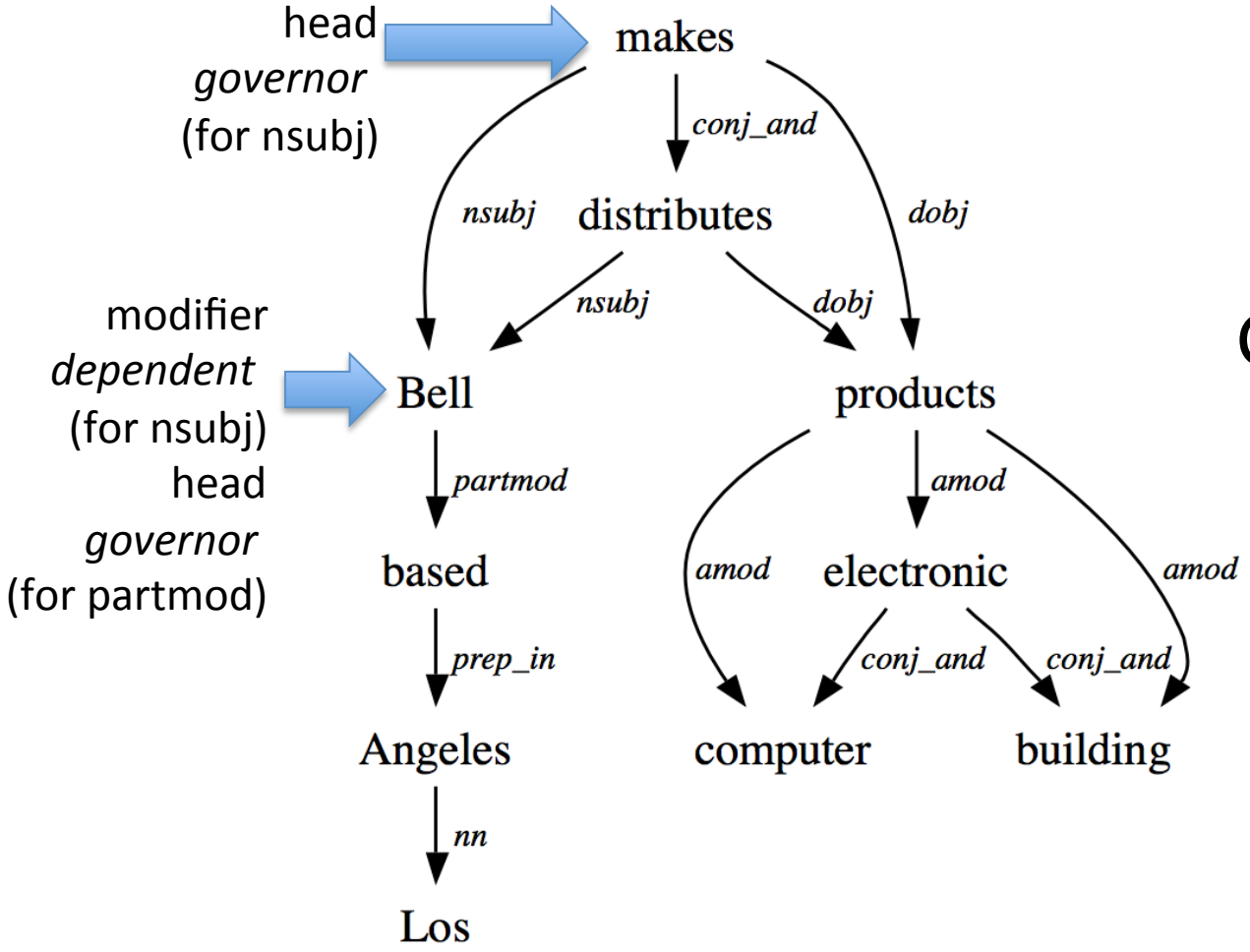
ADJP	←	NNS QP NN $ ADVP JJ VBN VBG ADJP JJR NP JJS DT FW RBR RBS SBAR RB
ADVP	→	RB RBR RBS FW ADVP TO CD JJR JJ IN NP JJS NN
CONJP	→	CC RB IN
FRAG	→	(NN* | NP) W* SBAR (PP | IN) (ADJP | JJ) ADVP RB
INTJ	←	*
LST	→	LS :
NAC, NP, NX, WHNP	←	(NN* | NX) NP-ε JJR CD JJ JJS RB QP NP
PP, WHPP	→	IN TO VBG VBN RP FW
PRN	→	S* N* W* PP|IN ADJP|JJ* ADVP|RB*
PRT	→	RP
QP	←	$ IN NNS NN JJ RB DT CD NCD QP JJR JJS
RRC	→	VP NP ADVP ADJP PP
S	←	VP *-PRD S SBAR ADJP UCP NP
SBAR	←	S SQ SINV SBAR FRAG IN DT
SBARQ	←	SQ S SINV SBARQ FRAG
SINV	←	VBZ VBD VBP VB MD VP *-PRD S SINV ADJP NP
SQ	←	VBZ VBD VBP VB MD *-PRD VP SQ
UCP	→	*
VP	→	VBD VBN MD VBZ VB VBG VBP VP *-PRD ADJP NN NNS NP
WHADJP	←	CC WRB JJ ADJP
WHADVP	→	CC WRB
X	→	*

Converting from a constituency-based representation to dependencies

- Live example!

Stanford dependencies

Bell, based in Los Angeles, makes and distributes electronic, computer and building products.



Graph not tree!

Important Stanford dependencies: nsubj and nsubjpass

***nsubj*: nominal subject**

A nominal subject is a noun phrase which is the syntactic subject of a clause. The governor of this relation might not always be a verb: when the verb is a copular verb, the root of the clause is the complement of the copular verb, which can be an adjective or noun.

"Clinton defeated Dole"	*nsubj*(defeated, Clinton)
"The baby is cute"	*nsubj*(cute, baby)

***nsubjpass*: passive nominal subject**

A passive nominal subject is a noun phrase which is the syntactic subject of a passive clause.

"Dole was defeated by Clinton"	*nsubjpass*(defeated, Dole)

Important Stanford dependencies: dobj, iobj, pobj

***dobj*: direct object**

The direct object of a VP is the noun phrase which is the (accusative) object of the verb.

"She gave me a raise"	*dobj*(gave, raise)
"They win the lottery"	*dobj*(win, lottery)

***iobj*: indirect object**

The indirect object of a VP is the noun phrase which is the (dative) object of the verb.

"She gave me a raise"	*iobj*(gave, me)

***pobj*: object of a preposition**

The object of a preposition is the head of a noun phrase following the preposition, or the adverbs "here" and "there". (The preposition in turn may be modifying a noun, verb, etc.) Unlike the Penn Treebank, we here define cases of VBG quasi-prepositions like "including", "concerning", etc. as instances of *pobj*. (The preposition can be called a FW for "pace", "versus", etc. It can also be called a CC – but we don't currently handle that and would need to distinguish from conjoined prepositions.) In the case of preposition stranding, the object can precede the preposition (e.g., "What does CPR stand for?").

"I sat on the chair"	*pobj*(on, chair)

Stanford dependencies

- Approximately 53 overall
- See the *Stanford typed dependencies manual* for details (on the course website)

Matching patterns over dependency graphs

- Often, it is more straightforward to write your own code over this simple dependency representation
- For complex patterns there is Semgrex
  - A language similar to Tregex, but running over dependency graphs rather than constituent trees

Semgrex syntax: Nodes

- Nodes are represented as {attr1:value1;attr2:value2;…}
  - Attributes are regular strings; values can be strings or regular expressions marked by "/"s
    - {lemma:run;pos:/VB.*/} => any verb form of the word "run"
  - {} is any node in the graph
  - {$} is any root in the graph
- Descriptions can be negated with !
  - !{lemma:boy} => any word that isn't "boy"

Slide by Chloe Kiddon

Semgrex syntax: Relations

- Relationships between nodes can be specified
- Relations in Semgrex have two parts: the relation symbol and the relation type: i.e. <nsubj
  - **A <reln B** : A is the dependent of a *reln* relation with B
  - **A >reln B** : A is the governor of a *reln* relation with B
  - **A <<reln B** : There is some node in a dep->gov chain from A that is the dependent of a *reln* relation with B
  - **A >>reln B** : There is some node in a gov>dep chain from A that is the governor of a *reln* relation with B
- Relation types can be regular strings or regular expressions encased by "/"

Slide by Chloe Kiddon

Semgrex syntax: Building complex expressions

- Relations can be strung together for “and”
  - All relations are relative to first node in string
  - {} >nsubj {} >dobj {}
    - “A node that is the governor of both an nsubj relation and a dobj relation”
  - & symbol is optional: {} >nsubj {} & >dobj {}
- Nodes can be grouped w/ parentheses
  - {pos:NN} @ ({} <nsubj {})
    - “A noun that is aligned to a node that is the dependent of an nsubj relation ”
  - Not the same as {pos:NN} @ {} <nsubj {}

Semgrex syntax

- See the entire tutorial on the course website

Other dependency representations CoNLL-2008 and -2009

- 37 syntactic dependencies
- Always a tree for English!
- Some parsers use these instead of the Stanford dependencies

- Approximately 38 semantic dependencies (derived from semantic roles)
  - We probably won't cover these in this class
- See the *The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies* paper for details

CoNLL-2008 syntactic dependencies

The key dependencies for most applications

Label	Freq.	Description
NMOD	324834	Modifier of nominal
P	135260	Punctuation
PMOD	115988	Modifier of preposition
SBJ	89371	Subject
OBJ	66677	Object
ROOT	49178	Root
ADV	47379	General adverbial
NAME	41138	Name-internal link
VC	35250	Verb chain
COORD	31140	Coordination
DEP	29456	Unclassified
TMP	26305	Temporal adverbial or nominal modifier
CONJ	24522	Second conjunct (dependent on conjunction)
LOC	18500	Locative adverbial or nominal modifier
AMOD	17868	Modifier of adjective or adverbial
PRD	16265	Predicative complement
APPO	16163	Apposition

Parsers with CoNLL dependencies

- Malt
  - http://www.maltparser.org/
  - Very fast (O(n)), performance a couple F1 points below the state of the art
  - Shift-reduce algorithm
  - There are NLTK wrappers for it
- MSTParser
  - http://sourceforge.net/projects/mstparser/
  - Not so fast (at least O(n2)) but has state of the art performance
  - Maximum spanning tree (MST) algorithm
- Ensemble of Malt parsers
  - http://www.surdeanu.info/mihai/ensemble/
  - Very fast, with state of the art performance
  - Supports both CoNLL and (basic, tree-based) Stanford dependencies

Shallow syntax, or chunking

- Text chunking: dividing a text in syntactically correlated sequences of words.
- Chunking << dependency or constituent syntax
  - But it is sufficient for some applications
  - Can be implemented with really fast algorithms

Shallow syntax example

He reckons the current account deficit will narrow to only #1.8 billion in September.

[NP He] [VP reckons] [NP the current account deficit]
[VP will narrow] [PP to] [NP only # 1.8 billion] [PP in]
[NP September] .

- This problem reduces to sequence tagging, similarly to NER!
- But the IOB (or more complex) representation is preferred here. Why?

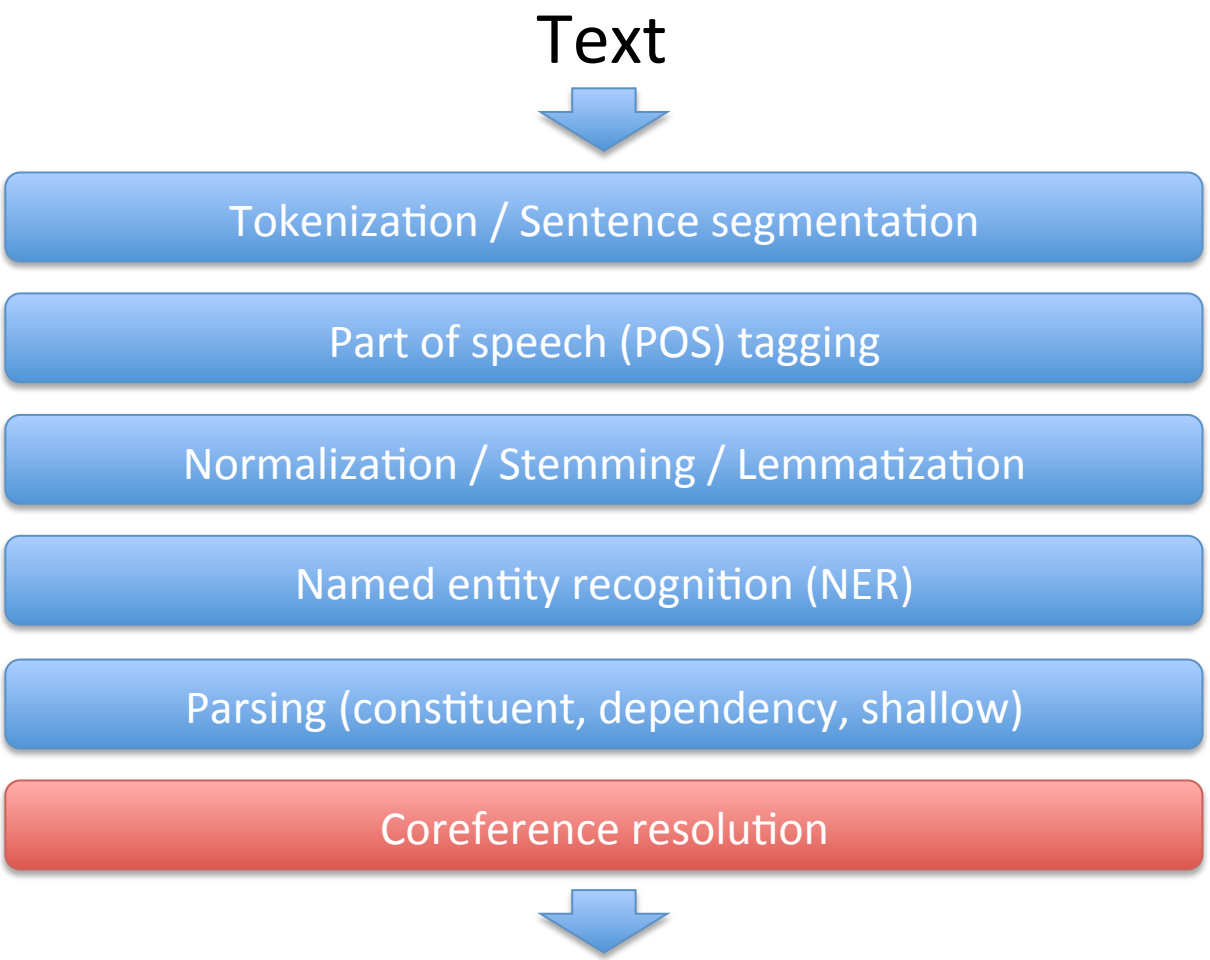
Shallow syntax example

```
He          PRP   B-NP
reckons     VBZ   B-VP
the         DT    B-NP
current     JJ    I-NP
account     NN    I-NP
deficit     NN    I-NP
will        MD    B-VP
narrow      VB    I-VP
to          TO    B-PP
only        RB    B-NP
#           #     I-NP
1.8         CD    I-NP
billion     CD    I-NP
in          IN    B-PP
September   NNP   B-NP
.           .     O
```

Shallow syntax tools

- Included in OpenNLP
- Not included in StanfordNLP or NLTK
- But can be easily added to both. NLTK even provides nice documentation for this. See chapter 7.2 in the NLTK book.

Typical NLP pipeline



Coreference resolution

Victoria Chen, Chief Financial Officer of MegaBucks Banking Corp. since 2004, saw her pay jump 20%, to $1.3 million, as the 37-year-old became the Denver-based financial services company's president. It has been years since she came to Megabucks from rival LotsaBucks.

Coreference resolution

Victoria Chen, Chief Financial Officer of MegaBucks Banking Corp. since 2004, saw her pay jump 20%, to $1.3 million, as the 37-year-old became the Denver-based financial services company's president. It has been years since she came to Megabucks from rival LotsaBucks.

Coreferring expressions: *Victoria Chen*, *Chief Financial Officer of MegaBucks Banking Corp. since 2004*, *her*, *the 37-year-old*, *the Denver-based financial services company's president*
Referent: **Victoria Chen**

Coreference resolution

Victoria Chen, Chief Financial Officer of MegaBucks Banking Corp. since 2004, saw her pay jump 20%, to $1.3 million, as the 37-year-old became the Denver-based financial services company's president. It has been years since she came to Megabucks from rival LotsaBucks.

Coreferring expressions: *MegaBucks Banking Corp.*, *the Denver-based financial services company, MegaBucks*
Referent: **MegaBucks Banking Corp.**

Definition

- Coreference resolution
  - Finding expressions that corefer, i.e., referring expressions that refer to the same entity
- Subtasks
  - Pronominal anaphora resolution: finding the antecedent for a single pronoun
  - Event resolution: coreference resolution when the referring expressions point to events rather than entities

Types of referring expressions

- Indefinite noun phrases
  - Mrs. Martin was so very kind as to send Mrs. Goddard *a beautiful goose*.
  - He had gone round one day to bring her *some walnuts*.
  - I saw *this beautiful Ford Falcon* today.

Typically evoke a representation for a new entity in the discourse

Types of referring expressions

- Definite noun phrases
  - It concerns a white stallion which I have sold to an officer. But the pedigree of *the white stallion* was not fully established.

Refer to entities that are identifiable to the hearer. That is, they were previously mentioned in text.

Types of referring expressions

- Pronouns
  - Emma smiled and chatted as cheerfully as *she* could.

Pronoun referents have a high degree of activation or **salience** in the discourse.

Pronouns usually refer to entities introduced no further than a few sentences back.

Pronouns are usually **anaphoric** (they appear after the referent was introduced) but can be **cataphoric** (the opposite).

Types of referring expressions

- Demonstrative pronouns
  - I just bought a copy of Thoreau's *Walden*. I had bought one five years ago. *That one* had been very tattered; *this one* was in much better condition.

This = proximal demonstrative
That = distal demonstrative (further away)

Types of referring expressions

- Names
  - *Miss Woodhouse* certainly had not done him justice.
  - *International Business Machines* sought patent compensation from Amazon; *IBM* had previously sued other companies.

Names can refer to both new and old entities in the discourse.

Features for coreference resolution

- String match
- Agreement
  - Person, number, gender, animacy, NE class
- Syntactic constructs
  - Appositives, predicate nominatives, etc.
- Grammatical role
  - Subject more salient than object
- Repeated mention
  - Billy Bones had been thinking about a glass of rum ever since the pirate ship dicked. *He* hobbled over to the Old Parrot bar. Jim went with him. *He* called for a glass of rum.
- Parallelism of syntax
- Selectional restrictions
  - John parked his car in the garage after driving *it* for hours.

Coreference resolution tools

- Included in CoreNLP and OpenNLP
- Not included in NLTK

Why is coreference resolution important?

Ethel Kennedy says that when the family gathered for Thanksgiving she wanted the children to know what a real turkey looked like. So she sent ***her son****, Robert F. Kennedy Jr., to a farm to buy two birds.*

From *Coreference for Learning to Extract Relations: Yes, Virginia, Coreference Matters*

Why is coreference resolution important?

	P+	P-	R+	R-	R*	F+	F-
attendSchool (1)	83	**97**	**49**	16	27	**62**	27
GPEEmploy(2)	91	**96**	**29**	3	3	**44**	5
GPELeader (3)	87	**99**	**48**	28	30	**62**	43
hasBirthPlace (4)	87	**97**	**57**	37	53	**69**	53
hasChild (5)	**70**	60	**37**	17	11	**48**	27
hasSibling (6)	**73**	69	**67**	17	17	**70**	28
hasSpouse (7)	61	**96**	**72**	22	31	**68**	36
ORGEmploys(8)	**92**	82	**22**	4	7	**35**	7
ORGLeader (9)	88	**97**	**73**	32	42	**80**	48
hasBirthDate (10)	**90**	85	**45**	13	32	**60**	23

Table 1: Precision, Recall, and F scores

From *Coreference for Learning to Extract Relations: Yes, Virginia, Coreference Matters*

Readings

- Please see the recommended readings for Lecture 2 on the course web site: http://www.surdeanu.info/mihai/teaching/ista555-spring15/readings.php