More λ-Calculus
Lexical Semantics

LING 571 — Deep Processing Techniques for NLP

October 31st, 2018 🎃

Ryan Georgi

Announcements

- We ***will*** be dropping lowest assignment.
- EC for HW #4 due tonight

Happy Hall🎃ween!

- *Emily's costume today:*

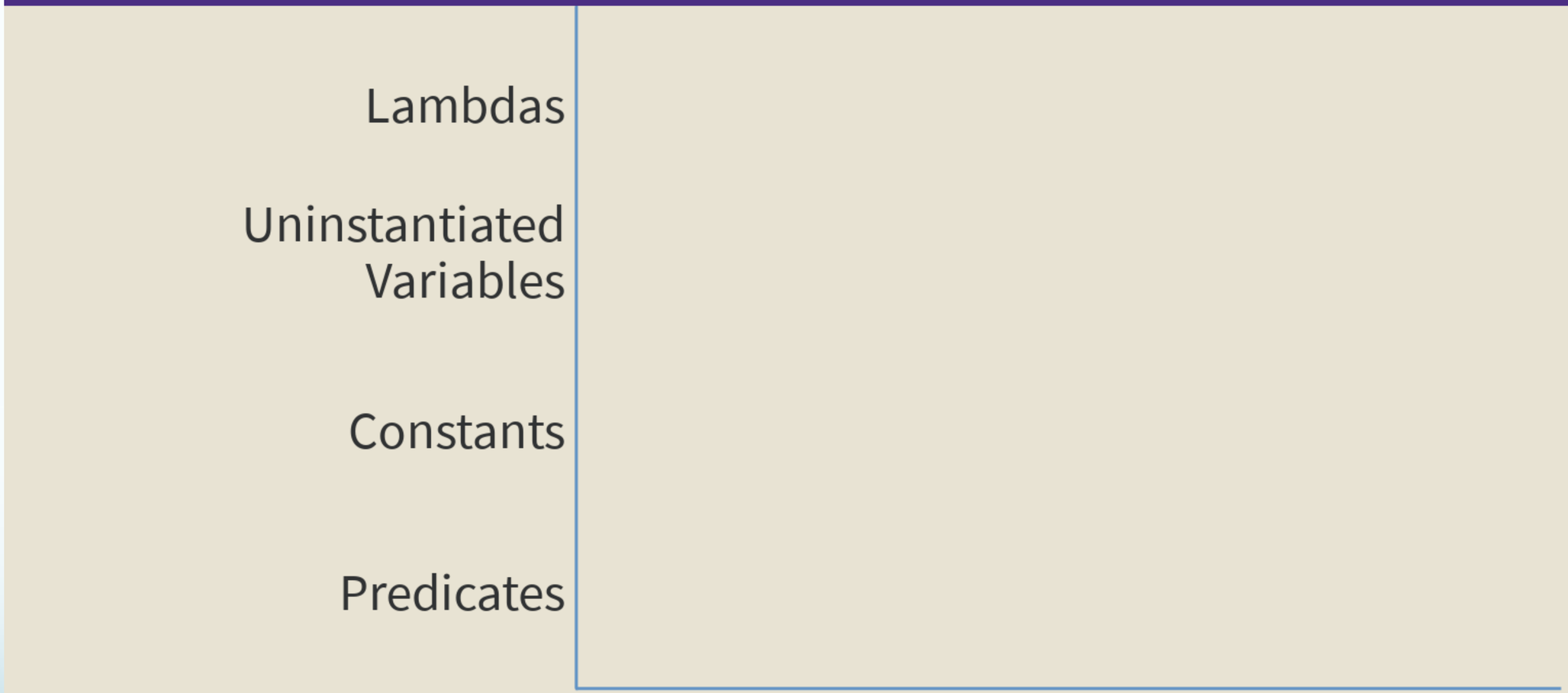
More λ-Calculus

A Few Clarifications

- We will consider "*all*" to be semantically equivalent to "*every*"
- "*nobody*" = "***not** a*"

Target Representations

The target representations should have:



Common Nouns

- $\lambda x.\boldsymbol{Restaurant}(x) \rightarrow$ *'restaurant'*
  - Somewhat similar to the NNP construction
    - λ *var.Predicate(var)*

- But common nouns represent ***relations***, rather than ***constants***
  - Meaning of the noun encoded in the predicate
  - Relate the concept of the noun to a particular instance of variable

Scope Ambiguity, Again

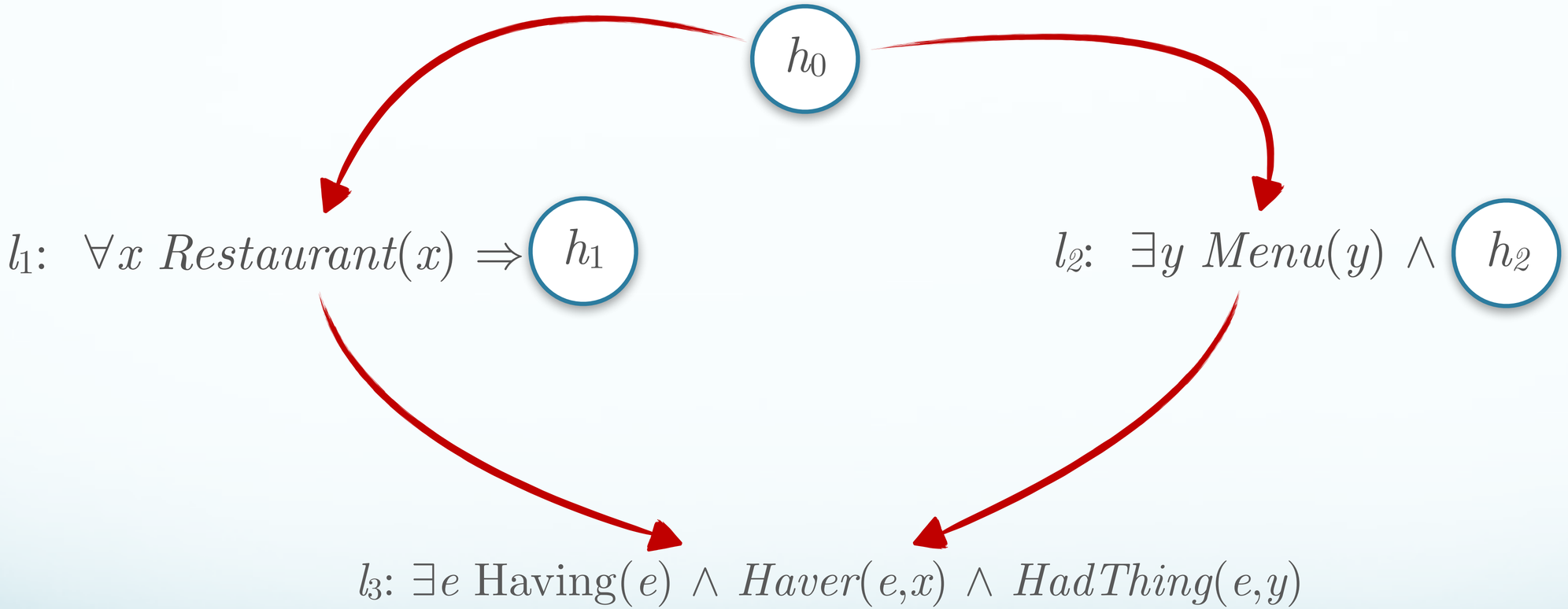
- "*Every restaurant has a menu*"
  - $\forall x\ Restaurant(x) \Rightarrow \exists y\ Menu(y) \wedge \exists e\ \text{Having}(e) \wedge Haver(e,x) \wedge HadThing(e,y)$
  - $\exists y\ Menu(y)\ \forall x\ Restaurant(x) \Rightarrow \exists e\ \text{Having}(e) \wedge Haver(e,x) \wedge HadThing(e,y)$

- For the homework, only need to come up with one.
  - How is this handled in practice?

Scope Ambiguity, Again

- "Hole semantics" via Bos, (1996)
  - h_n ← this is a "hole"
- We will use this to stand in for portions of the quantifier statements
- We will then "plug" the "holes" *…why, semanticists, why?*
  - Following the rules of ***dominance constraints***
  - An expression with hole h must be filled by a subordinate expression.

Scope Ambiguity, Again



see J&M 2nd ed, p. 597

Negation

- "No vegetarian restaurant serves meat."
  - $\neg(\exists x\, VegetarianRestaurant(x) \wedge Serves(x, Meat))$
- "All vegetarian restaurants do not serve meat."
  - $\forall x\ VegetarianRestaurant(x) \Rightarrow \neg Serves(x, Meat)$

- *These are semantically equivalent!*
  - $\neg$[IF P, THEN Q] $\Leftrightarrow$ P AND NOT Q

- For NLTK, use the hyphen/minus character: '-'

‘John booked a flight’

- Target representation:
  - $\exists x\ Flight(x) \wedge (\exists e Booked(\mathrm{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x))$

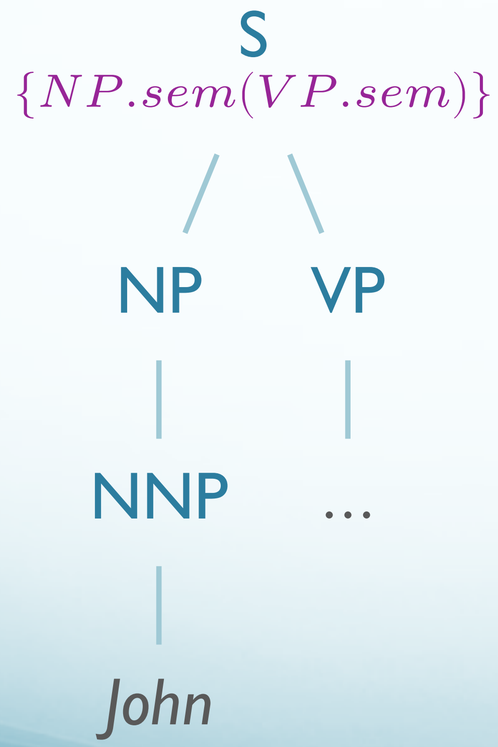
‘John booked a flight’

- $\exists x\ Flight(x) \wedge (\exists eBooked(\mathrm{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x))$

$$S \rightarrow NP\ VP \qquad \{NP.sem(VP.sem)\}$$

‘John booked a flight’

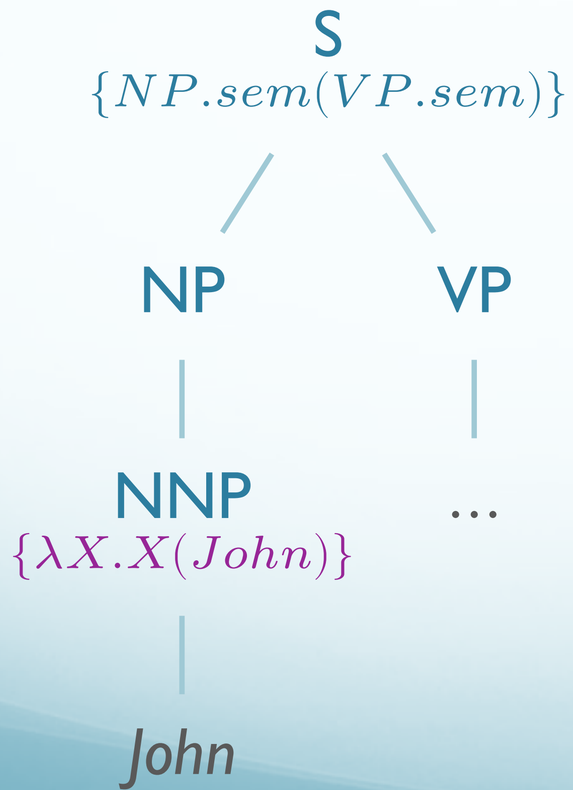
- $\exists x\ Flight(x) \wedge (\exists eBooked(\mathrm{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x))$



$S \rightarrow NP\ VP \qquad \{NP.sem(\ VP.sem)\}$

‘John booked a flight’

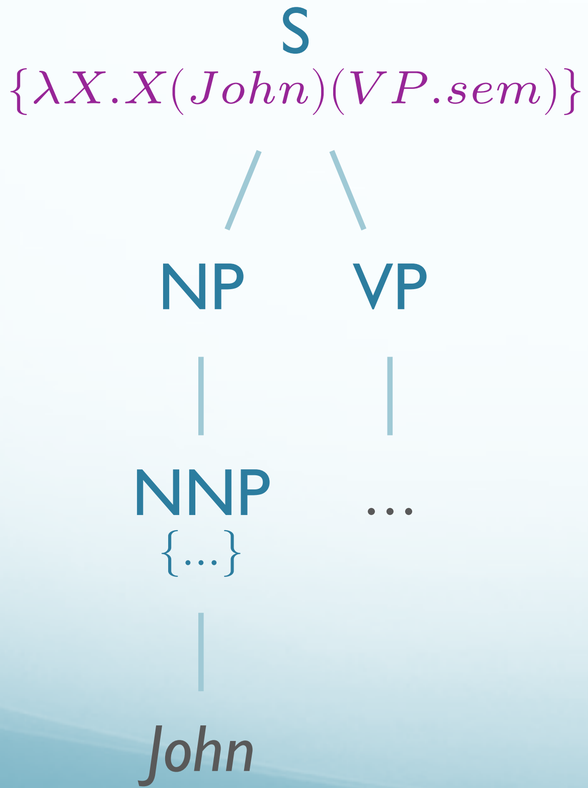
- $\exists x\ Flight(x) \wedge (\exists eBooked(\mathrm{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x))$



$$S \rightarrow NP\ VP \qquad \{NP.sem(VP.sem)\}$$
$$NNP \rightarrow \text{‘}John\text{’} \qquad \{\lambda X.X(John)\}$$
$$NP \rightarrow NNP \qquad \{NNP.sem\}$$
$$VP \rightarrow Verb\ NP \qquad \{Verb.sem(NP.sem)\}$$

‘John booked a flight’

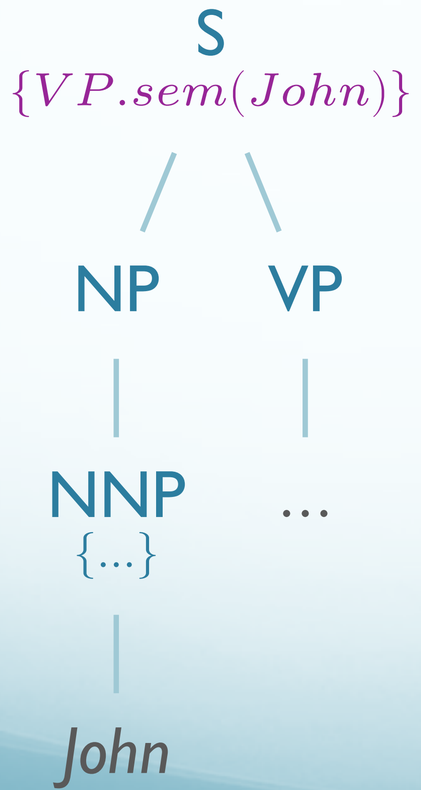
- $\exists x\ Flight(x) \wedge (\exists eBooked(\mathrm{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x))$



$$S \rightarrow NP\ VP \qquad \{NP.sem(VP.sem)\}$$
$$NNP \rightarrow \text{‘}John\text{’} \qquad \{\lambda X.X(John)\}$$
$$NP \rightarrow NNP \qquad \{NNP.sem\}$$
$$VP \rightarrow Verb\ NP \qquad \{Verb.sem(NP.sem)\}$$

‘John booked a flight’

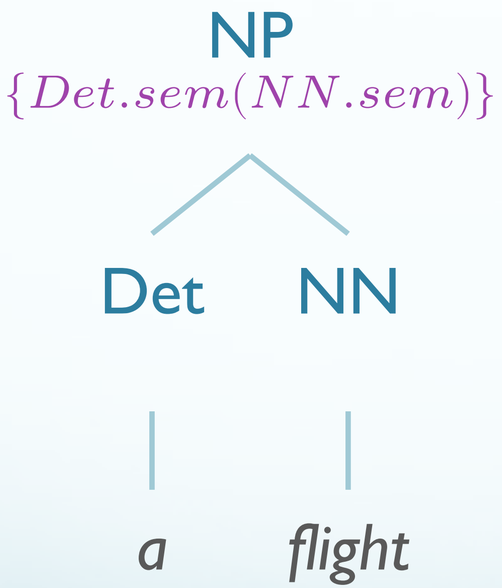
- $\exists x\ Flight(x) \wedge (\exists eBooked(\mathrm{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x))$



$$S \rightarrow NP\ VP \qquad \{NP.sem(VP.sem)\}$$
$$NNP \rightarrow \text{‘}John\text{’} \qquad \{\lambda X.X(John)\}$$
$$NP \rightarrow NNP \qquad \{NNP.sem\}$$
$$VP \rightarrow Verb\ NP \qquad \{Verb.sem(NP.sem)\}$$

‘John booked a flight’

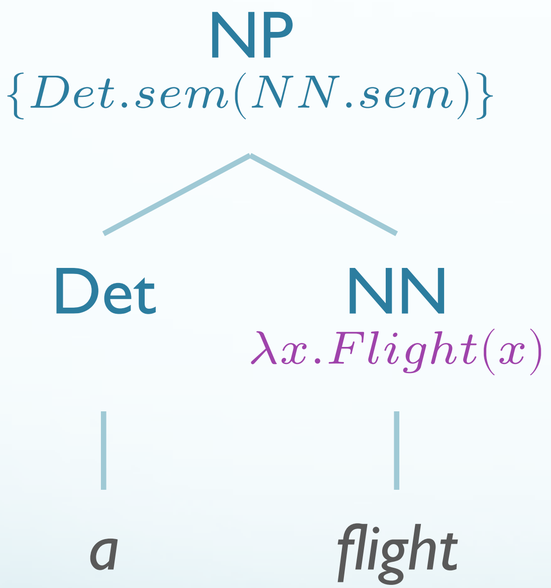
- $\exists x\ Flight(x) \wedge (\exists eBooked(\mathrm{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x))$



$NP \rightarrow Det\ NN$ $\{Det.sem(NN.sem)\}$

‘John booked a flight’

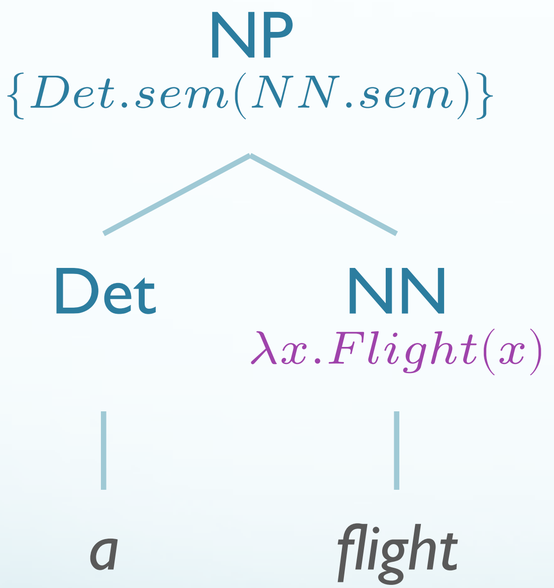
- $\exists x\ Flight(x) \wedge (\exists eBooked(\mathrm{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x))$



$$NP \rightarrow Det\ NN \qquad \{Det.sem(NN.sem)\}$$
$$NN \rightarrow \text{‘flight’} \qquad \{\lambda x.Flight(x)\}$$

‘John booked a flight’

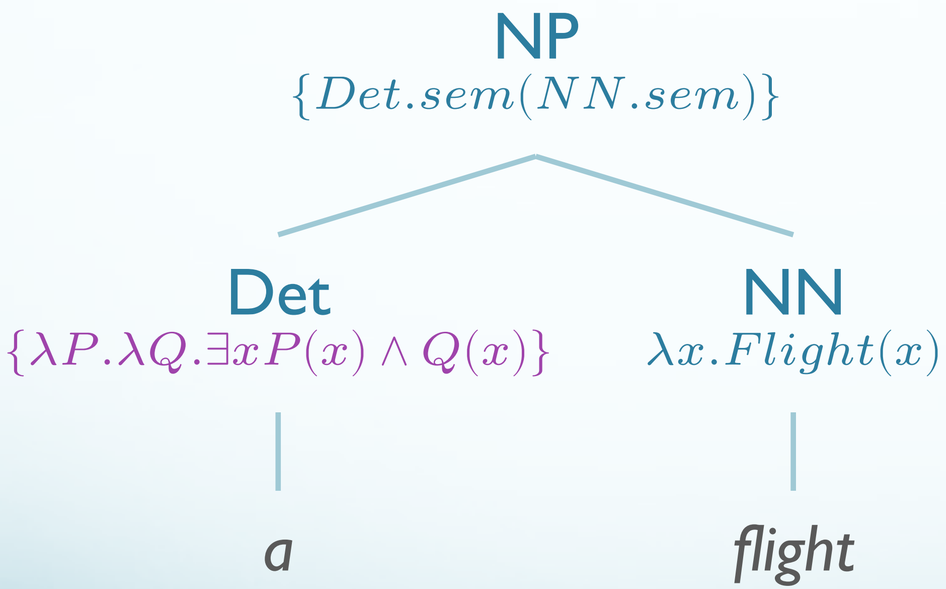
- $\exists x\ Flight(x) \wedge (\exists eBooked(\text{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x))$



$$NP \rightarrow Det\ NN \qquad \{Det.sem(NN.sem)\}$$
$$NN \rightarrow \text{‘}flight\text{’} \qquad \{\lambda x.Flight(x)\}$$
$$Det \rightarrow \text{‘}a\text{’} \qquad \{\ \lambda P.\lambda Q.\exists x\ P(x) \wedge Q(x)\ \}$$

‘John booked a flight’

- $\exists x\ Flight(x) \wedge (\exists eBooked(\text{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x))$



$$NP \rightarrow Det\ NN \qquad \{Det.sem(NN.sem)\}$$
$$NN \rightarrow \text{‘}flight\text{’} \qquad \{\lambda x.Flight(x)\}$$
$$Det \rightarrow \text{‘}a\text{’} \qquad \{\ \lambda P.\lambda Q.\exists x\ P(x) \wedge Q(x)\ \}$$

‘John booked a flight’

- $\exists x\ Flight(x) \wedge (\exists eBooked(\text{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x))$

NP
$\{Det.sem(NN.sem)\}$

$$NP \rightarrow Det\ NN \qquad \{Det.sem(NN.sem)\}$$
$$NN \rightarrow \text{‘}flight\text{’} \qquad \{\lambda x.Flight(x)\}$$
$$Det \rightarrow \text{‘}a\text{’} \qquad \{\ \lambda P.\lambda Q.\exists x\ P(x) \wedge Q(x)\ \}$$

‘John booked a flight’

- $\exists x\ Flight(x) \wedge (\exists eBooked(\text{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x))$

NP

$\{Det.sem(NN.sem)\}$

$\{\lambda P.\lambda Q.\exists x P(x) \wedge Q(x)(\lambda x.Flight(x))\}$

$NP \rightarrow\ Det\ NN$	$\{Det.sem(NN.sem)\}$
$NN \rightarrow\ \text{‘}flight\text{’}$	$\{\lambda x.Flight(x)\}$
$Det \rightarrow\ \text{‘}a\text{’}$	$\{\ \lambda P.\lambda Q.\exists x\ P(x) \wedge Q(x)\ \}$

‘John booked a flight’

- $\exists x\ Flight(x) \wedge (\exists e Booked(\text{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x))$

NP

$\{Det.sem(NN.sem)\}$

$\{\lambda P.\lambda Q.\exists x P(x) \wedge Q(x)(\lambda x.Flight(x))\}$

$\{\lambda Q.\exists x(\lambda x.Flight(x))(x) \wedge Q(x)\}$

$NP \rightarrow Det\ NN$	$\{Det.sem(NN.sem)\}$
$NN \rightarrow$ ‘flight’	$\{\lambda x.Flight(x)\}$
$Det \rightarrow$ ‘a’	$\{\ \lambda P.\lambda Q.\exists x\ P(x) \wedge Q(x)\ \}$

‘John booked a flight’

- $\exists x\ Flight(x) \wedge (\exists e Booked(\mathrm{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x))$

NP

$\{Det.sem(NN.sem)\}$

$\{\lambda P.\lambda Q.\exists x P(x) \wedge Q(x)(\lambda x.Flight(x))\}$

$\{\lambda Q.\exists x(\lambda x.Flight(x))(x) \wedge Q(x)\}$

$\{\lambda Q.\exists x Flight(x) \wedge Q(x)\}$

$NP \rightarrow Det\ NN$	$\{Det.sem(NN.sem)\}$
$NN \rightarrow$ ‘flight’	$\{\lambda x.Flight(x)\}$
$Det \rightarrow$ ‘a’	$\{\ \lambda P.\lambda Q.\exists x\ P(x) \wedge Q(x)\ \}$

‘John booked a flight’

- $\exists x\ Flight(x) \wedge (\exists e Booked(\mathrm{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x))$

NP

$\{Det.sem(NN.sem)\}$

$\{\lambda P.\lambda Q.\exists x P(x) \wedge Q(x)(\lambda x.Flight(x))\}$

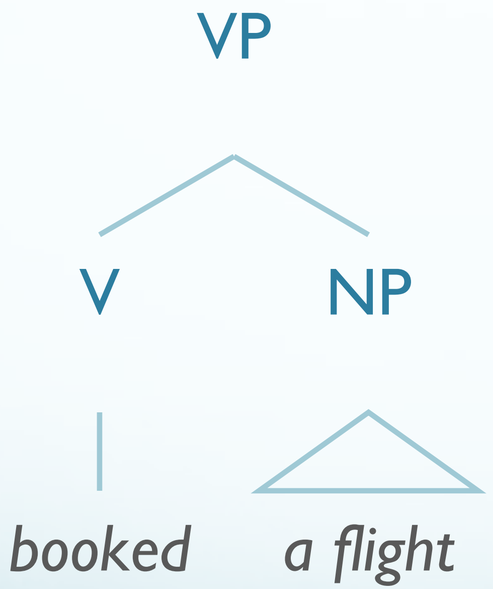
$\{\lambda Q.\exists x(\lambda x.Flight(x))(x) \wedge Q(x)\}$

$\{\lambda Q.\exists x Flight(x) \wedge Q(x)\}$

$NP \rightarrow Det\ NN$	$\{Det.sem(NN.sem)\}$
$NN \rightarrow$ ‘*flight*’	$\{\lambda x.Flight(x)\}$
$Det \rightarrow$ ‘*a*’	$\{\ \lambda P.\lambda Q.\exists x\ P(x) \wedge Q(x)\ \}$
‘*a flight*’	$\{\ \lambda Q.\exists x\ Flight(x) \wedge Q(x)\ \}$

‘John booked a flight’

- $\exists x\ Flight(x) \wedge (\exists eBooked(\mathrm{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x))$

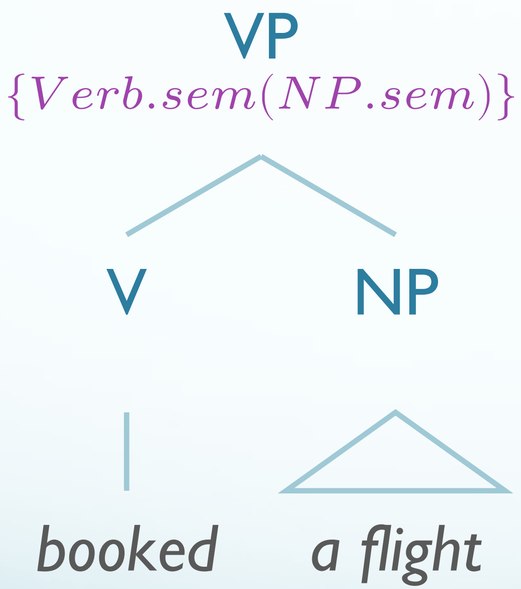


$VP \rightarrow\ Verb\ NP$ $\quad\quad$ $\{Verb.sem(NP.sem)\}$

‘$a\ flight$’ $\quad\quad$ $\{\ \lambda Q.\exists x\ Flight(x) \wedge Q(x)\ \}$

‘John booked a flight’

- $\exists x\ Flight(x) \wedge (\exists eBooked(\mathrm{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x))$

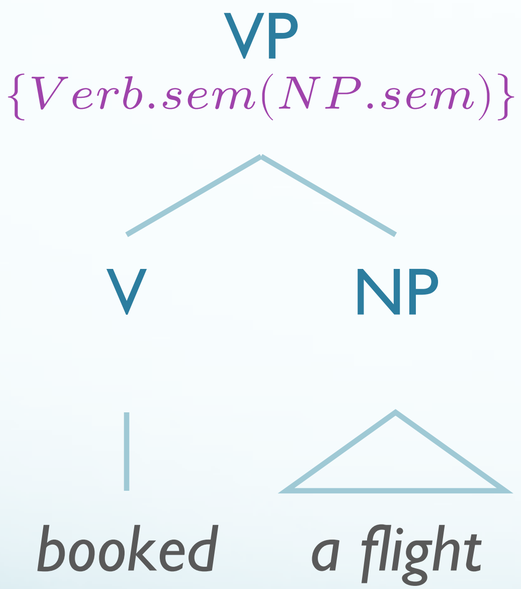


$VP \rightarrow\ Verb\ NP$ $\quad\quad$ $\{Verb.sem(NP.sem)\}$

‘a flight’ $\quad\quad$ $\{\ \lambda Q.\exists x\ Flight(x) \wedge Q(x)\ \}$

‘John booked a flight’

- $\exists x\ Flight(x) \wedge (\exists e Booked(\text{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x))$



$Verb \rightarrow$ ‘booked’

$\{\lambda W.\lambda z.\, W(\lambda y.\exists e Booked(\text{e}) \wedge Booker(e,z) \wedge BookedThing(e,y))\}$

$VP \rightarrow\ Verb\ NP$ $\quad\quad$ $\{Verb.sem(NP.sem)\}$

‘a flight’ $\quad\quad$ $\{\ \lambda Q.\exists x\ Flight(x) \wedge Q(x)\ \}$

‘John booked a flight’

- $\exists x\ Flight(x) \wedge (\exists e Booked(\mathrm{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x))$

$Verb.sem(NP.sem)$

$\lambda W.\lambda z.\ W(\lambda y.\exists e Booked(\mathrm{e}) \wedge Booker(e,z) \wedge BookedThing(e,y))(\lambda Q.\exists x\ Flight(x) \wedge Q(x))$

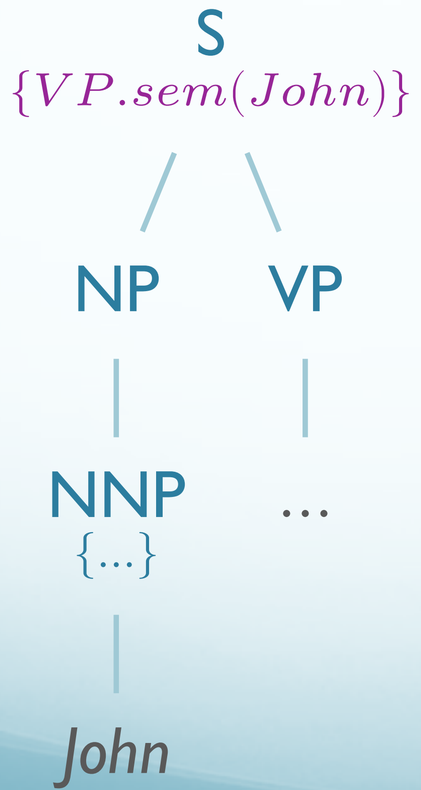
$\lambda z.(\lambda Q.\exists x\ Flight(x) \wedge Q(x))(\lambda y.\exists e Booked(\mathrm{e}) \wedge Booker(e,z) \wedge BookedThing(e,y))$

$\lambda z.\exists x\ Flight(x) \wedge (\lambda y.\exists e Booked(\mathrm{e}) \wedge Booker(e,z) \wedge BookedThing(e,y))(x)$

$\lambda z.\exists x\ Flight(x) \wedge (\exists e Booked(\mathrm{e}) \wedge Booker(e,z) \wedge BookedThing(e,\boldsymbol{x})$

‘John booked a flight’

- $\exists x\ Flight(x) \wedge (\exists eBooked(\mathrm{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x))$



S

‘booked a flight'

$VP.sem(John)$

$\lambda z.\exists x\ Flight(x) \wedge (\exists eBooked(\mathrm{e}) \wedge Booker(e,z) \wedge BookedThing(e,\boldsymbol{x})$

‘John booked a flight’

- $\exists x\ Flight(x) \wedge (\exists e Booked(\mathrm{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x))$

S	$VP.sem(John)$
‘booked a flight'	$\lambda z.\exists x\ Flight(x) \wedge (\exists e Booked(\mathrm{e}) \wedge Booker(e,z) \wedge BookedThing(e,\boldsymbol{x})$

$\boldsymbol{\lambda z}.\exists x\ Flight(x) \wedge (\exists e Booked(\mathrm{e}) \wedge Booker(e,z) \wedge BookedThing(e,x)(\boldsymbol{John})$

$\exists x\ Flight(x) \wedge (\exists e Booked(\mathrm{e}) \wedge Booker(e,\boldsymbol{John}) \wedge BookedThing(e,x)$

'John booked a flight'

$Det \rightarrow$ *'a'*	$\{ \lambda P.\lambda Q.\exists x\ P(x) \wedge Q(x) \}$
$Det \rightarrow$ *'every'*	$\{ \lambda P.\lambda Q.\forall x\ P(x) \Rightarrow Q(x) \}$
$NN \rightarrow$ *'flight'*	$\{\lambda x.Flight(x)\}$
$Verb \rightarrow$ *'booked'*	$\{\lambda W.\lambda z.\, W(\lambda y.\exists e Booked(\mathrm{e}) \wedge Booker(e,z) \wedge BookedThing(e,y))\}$
$NNP \rightarrow$ *'John'*	$\{\lambda X.X(John)\}$
$NP \rightarrow NNP$	$\{NNP.sem\}$
$NP \rightarrow Det\ NN$	$\{Det.sem(NN.sem)\}$
$S \rightarrow NP\ VP$	$\{NP.sem(VP.sem)\}$
$VP \rightarrow Verb\ NP$	$\{Verb.sem(NP.sem)\}$

‘John booked no flight’

- $\neg(\exists x\ Flight(x) \wedge (\exists e Booked(\mathrm{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x))$
- $\forall x Flight(x) \Rightarrow \neg(\exists e Booked(\mathrm{e}) \wedge Booker(e,\ John) \wedge BookedThing(e,\ x)$

‘John booked a flight’

$Det \rightarrow$ ‘no’	$\{ \lambda P.\lambda Q.\neg\exists x\ P(x) \wedge Q(x) \mid \lambda P.\lambda Q.\forall x\ P(x) \Rightarrow \neg Q(x)\}$
$Det \rightarrow$ ‘a’	$\{ \lambda P.\lambda Q.\exists x\ P(x) \wedge Q(x)\ \}$
$Det \rightarrow$ ‘$every$’	$\{ \lambda P.\lambda Q.\forall x\ P(x) \Rightarrow Q(x)\ \}$
$NN \rightarrow$ ‘$flight$’	$\{\lambda x.Flight(x)\}$
$Verb \rightarrow$ ‘$booked$’	$\{\lambda W.\lambda z.\, W(\lambda y.\exists e Booked(\mathrm{e}) \wedge Booker(e,z) \wedge BookedThing(e,y))\}$
$NNP \rightarrow$ ‘$John$’	$\{\lambda X.X(John)\}$
$NP \rightarrow NNP$	$\{NNP.sem\}$
$NP \rightarrow Det\ NN$	$\{Det.sem(NN.sem)\}$
$S \rightarrow NP\ VP$	$\{NP.sem(VP.sem)\}$
$VP \rightarrow Verb\ NP$	$\{Verb.sem(NP.sem)\}$

What is the purpose of creating a dummy predicate for a lambda expression?

To help compose it with things to the left.	1
To help compose it with things to the right.	2
1 or 2, depending on the position in the tree	3
1 or 2, depending on the needs of the lambda function	4

Start the presentation to see live content. Still no live content? Install the app or get help at **PollEv.com/app**

Total Results

Other Lambda Calculus

Adjectives

- Similar to nouns, but with an extra conjunction and dummy predicate:
  - "red" $= \lambda P\, \lambda x(red(x) \wedge P(x))$

Definite Article

- $a = \lambda P.\lambda Q.\exists x(P(x) \wedge Q(x))$
- $the = \lambda P.\lambda Q.\exists x(\forall y(\mathrm{P}(y) \Leftrightarrow x{=}y \wedge \mathrm{Q}(y)))$
- Roughly:
  - *"For everything in the set of things* $P(y)$ (y is a thing in the set denoted by P)...
  - *...we are discussing one thing that could be in that set* (y)
  - *...and happens to be the thing we are discussing* (x)

Definite Article

- $the = \lambda P.\lambda Q.\exists x(\forall y(\mathrm{P}(y) \Leftrightarrow x = y \wedge \mathrm{Q}(y)))$
  - Bertrand Russel, "*On Denoting*" (1905).
  - The definite article isn't exactly the same as a constant (like "*John*")
  - Rather, it picks out a set of items from a set (the generic NN), and makes a strong assertion:

    A) The book arrived.

    B) A book arrived.
    - A ⊨ B, but B ⊭ A
  - Discussion [link]

Lexical Semantics

Lexical Semantics

- Thus far: $\text{POS} \rightarrow \textit{Word}\ \{\text{sem}\}$
  - Can compose larger semantic formulae bottom-up this way
  - …but we haven't really discussed what a "word" is, semantically.
- Lexical semantics:
  - How do we formally discuss what a "word" is?
  - How do we relate words to one another?
  - How do we differentiate/relate linked senses?

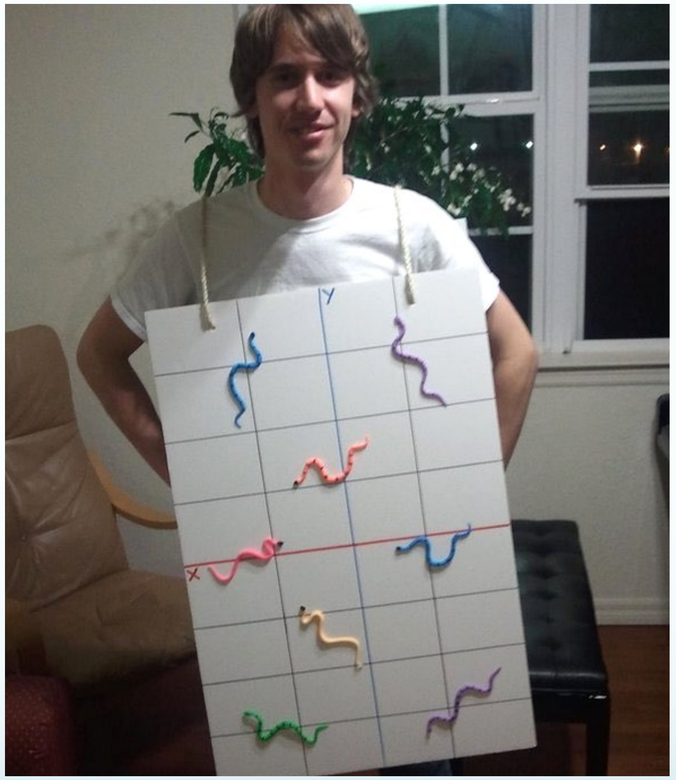
What is a Plant?

- There are more kinds of **plants** and animals in the rainforests than anywhere else on Earth. Over half of the millions of known species of **plants** and animals live in the rainforest. Many are found nowhere else. There are even **plants** and animals in the rainforest that we have not yet discovered.
- The Paulus company was founded in 1938. Since those days the product range has been the subject of constant expansions and is brought up continuously to correspond with the state of the art. We're engineering, manufacturing, and commissioning world-wide ready-to-run **plants** packed with our comprehensive know-how.

Lexical Semantics

…by way of dad-joke Halloween costumes. 🎃

A Ceiling Fan



Snakes on a Plane

(Painful) Examples of ***Homonymy***

Sources of Confusion

Homonymy

Polysemy

Synonymy

Antonymy

[Hypo/Hyper]-nymy

Sources of Confusion: Homonymy

- Words have same form but different meanings
  - Generally same POS, but unrelated meaning
    - $bank_1$ (side of river)
    - $bank_2$ (financial institution)

Sources of Confusion: Homonymy

- Different types of Homonymy:
  - ***Homophones***: same phonology, different orthographic form
    - two
    - to
    - too
  - ***Homographs***: Same orthography, different phonology:
    - "*lead*" (metal)
    - "*lead*" (take somewhere)
- Why do we care?
  - Problem for applications: TTS, ASR transcription, IR

Sources of Confusion: Polysemy

- Multiple RELATED senses
  - e.g. bank: money, organ, blood
- Big issue in lexicography
  - Number of senses
  - Relations between senses
  - Differentiation

Sources of Confusion: Polysemy

- Example: [[*serve*]]
  - *serve breakfast*
  - *serve Philadelphia*
  - *serve time*

Sources of Confusion: Synonymy

- (near) identical meaning
- Substitutability
  - Maintains propositional meaning

Sources of Confusion: Synonymy

- Issues:
  - Also has polysemy!
    - Shades of meaning - other associations
      - *price* vs. *fare*
      - *big* vs. *large*
      - *water* vs. H_20
- Collocational constraints
  - e.g. *babbling brook* vs. **babbling river*
- Register:
  - social factors: e.g. politeness, formality

Sources of Confusion: Antonymy

- Opposition
- Typically ends of a scale
  - *fast* vs. *slow*
  - *big* vs. *little*
- Can be hard to distinguish automatically from synonyms

Sources of Confusion: Hyponomy

- `instanceOf(x, y)` relations:
- More General (***hypernym***) vs. more specific (***hyponym***)
  - *dog* vs. *golden retriever*
  - *fruit* vs. *mango*
- Organize as ontology/taxonomy

Word Sense Disambiguation

- Application of lexical semantics
- Goal: given a word ***in context***, identify the appropriate sense
  - e.g. plants and animals in the rainforest
- Crucial for real syntactic & semantic analysis
  - Correct sense can determine
    - Available syntactic structure
    - Available thematic roles, correct meaning…

Robust Disambiguation

- More to semantics than predicate-argument structure
  - Select sense where predicates underconstrain
- Learning approaches
  - Supervised, bootstrapped, unsupervised
- Knowledge-based approaches
  - Dictionaries, taxonomies
- Contexts for sense selection

There are more kinds of **plants** and animals in the rainforests than anywhere else on Earth. Over half of the millions of known species of **plants** and animals live in the rainforest. Many are found nowhere else. There are even **plants** and animals in the rainforest that we have not yet discovered.

Biological Example

The Paulus company was founded in 1938. Since those days the product range has been the subject of constant expansions and is brought up continuously to correspond with the state of the art. We're engineering, manufacturing and commissioning world-wide ready-to-run **plants** packed with our comprehensive know-how. Our Product Range includes pneumatic conveying systems for carbon, carbide, sand, lime and many others. We use reagent injection in molten metal for the…

Industrial Example

Label the First Use of "Plant"

Roadmap

- Lexical Semantics
  - Motivation & Definitions
  - Word Senses
  - Tasks:
    - **Word sense disambiguation**
    - Word sense similarity
  - Distributional Similarity

Disambiguation: Features

- Part of Speech
  - Of word and neighbors
- Morphologically simplified form
- Words in neighborhood
  - How big is "neighborhood?"
  - Is there a single optimal size? Why?

Disambiguation: Features

- (Possibly shallow) Syntactic analysis
  - predicate-argument relations
  - modification (complements)
  - phrases
- Collocation
  - words in specific relation
    - Predicate-Argument, or $(+/-)1$ word index
- Co-occurrence
  - bag of words

Disambiguation: Evaluation

- Ideally, end-to-end evaluation with WSD component
  - Demonstrate real impact of technique in system
  - Difficult, expensive, still application specific
- Typically intrinsic, sense-based
  - Accuracy, precision, recall
  - SENSEVAL/SEMEVAL: all words, lexical sample

WSD Evaluation

- Baseline:
  - Most frequent sense
- Ceiling:
  - Human inter-rater agreement
    - 75-80% fine
    - 90% coarse

Roadmap

- Lexical Semantics
  - Motivation & Definitions
  - Word Senses
  - Tasks:
    - Word sense disambiguation
    - **Word sense similarity**
  - Distributional Similarity

Word Sense Similarity

- Synonymy:
  - True propositional substitutability is rare, slippery
- Word similarity (semantic distance)
  - Looser notion, more flexible

Word Sense Similarity

- Appropriate to applications:
  - IR, summarization, MT, essay scoring
    - Don't need binary +/– synonym decision
    - Want terms/documents that have high ***similarity***
- Approaches:
  - Distributional
  - Thesaurus-based

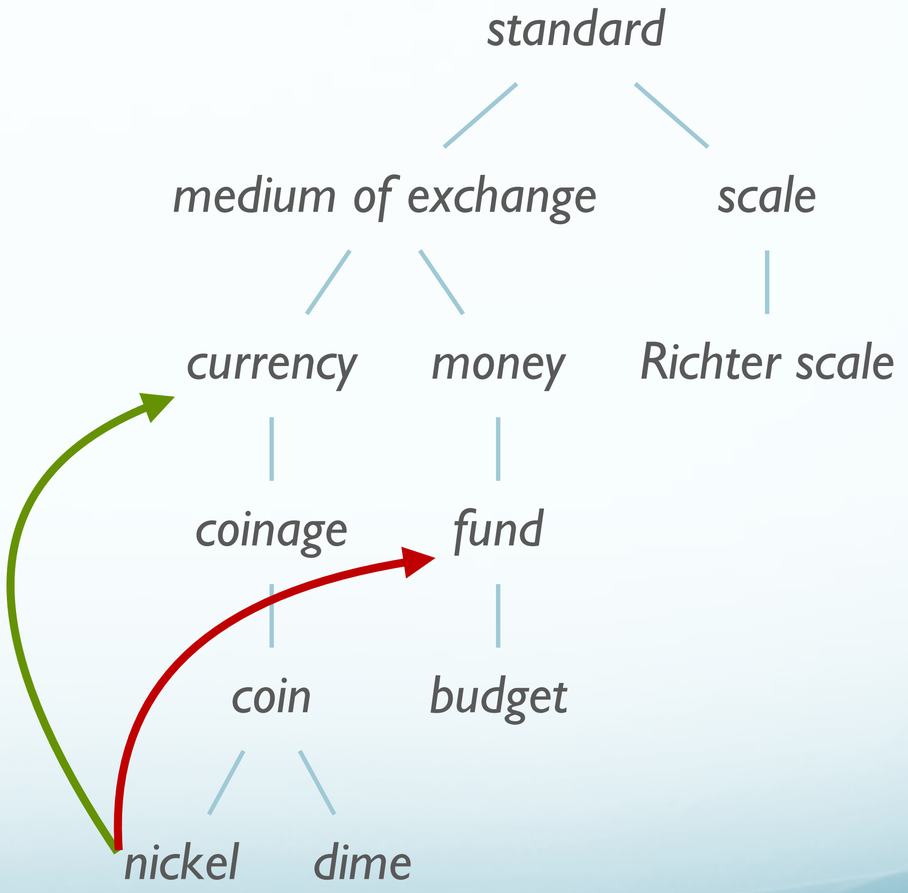
Similarity vs. Relatedness

- Similarity:
  - *car, bicycle*
  - *nickel < coin < currency*

- Related:
  - *car, gasoline*
  - *coin, budget*

Thesaurus-Based:

- Build ontology of senses
  - e.g. WordNet
  - Use distance to infer similarity/relatedness:



Roadmap

- Lexical Semantics
  - Motivation & Definitions
  - Word Senses
  - Tasks:
    - Word sense disambiguation
    - Word sense similarity
  - **Distributional Similarity**

Distributional Similarity

- "You shall know a word by the company it keeps!" (*Firth, 1957*)
  - A bottle of *tezgüino* is on the table.
  - Everybody likes *tezgüino*.
  - *Tezgüino* makes you drunk.
  - We make *tezgüino* from corn.
- Tezguino; corn-based alcoholic beverage. (From *Lin, 1998a*)

Distributional Similarity

- Represent 'company' of word such that similar words will have similar representations
  - 'Company' = context
- Word represented by context feature vector
  - Many alternatives for vector
- Initial representation:
  - 'Bag of words' binary feature vector
  - Feature vector length N, where N is size of vocabulary
    - $f_i{=}1$ if $word_i$ within window size w of $word_0$

Context Feature Vector

	arts	boil	data	function	large	sugar	summarized	water
Apricot	0	1	0	0	1	1	0	1
Pineapple	0	1	0	0	1	1	0	1
Digital	0	0	1	1	1	0	1	0
Information	0	0	1	1	1	0	1	0

Distributional Similarity Questions

- What is the right neighborhood?
  - What is the context?
- How should we weight the features?
- How can we compute similarity between vectors?