NLP

Introduction to NLP

262.

Features and Unification

Need for Feature-based Grammars

- Example (number agreement)
  - *The dogs bites*
- Example (count/mass nouns)
  - *many water*
- Example in French (number and person agreement w/subject)
  - Paul est parti, Michelle est partie, Ils sont partis, Elles sont parties
- Example in French (number and person agreement w/direct object)
  - Je l'ai vu (I saw him), Je l'ai vue (I saw her)
- Idea
  - $S \rightarrow$ NP VP

    (but only if the person of the NP is equal to the person of the VP)

Parameterized Grammars

- Parameterized rules, e.g.,
  - S → NP[person,number,”nominative” VP[person,number]
  - VP[person,number] → V[person,number] NP[person,number,”accusative”
  - NP[“first”,number,”nominative”] → DET[number]N[number]
- Appropriate modifications are needed to the parser

Unification Grammars

- Various unification grammar formalisms
  - LFG, HPSG, FUG
- Handle agreement
  - e.g., number, gender, person
- Unification
  - Two constituents can be combined only if their features can ‘unify’
- Feature structures (FS or FD)
  - Nested structures that represent all features in an attribute-value matrix
  - Values are typed, so GENDER=PLURAL is not allowed
  - FSs can also be represented as graphs (DAG)
  - Feature paths (from root to a node in the graph)

Example in NLTK

```
import nltk;
from __future__ import print_function
from nltk.featstruct import FeatStruct
from nltk.sem.logic import Variable, VariableExpression, Expression

fs1 = FeatStruct(number='singular', person=3)
print (fs1)
[ number = 'singular' ]
[ person = 3          ]

fs2 = FeatStruct(type='NP', agr=fs1)
print (fs2)
[ agr  = [ number = 'singular' ] ]
[        [ person = 3          ] ]
[                                ]
[ type = 'NP'                    ]
```

http://www.nltk.org/howto/featstruct.html

Feature Unification

- Graph-matching
- Recursive definition
  - Two FSs unify if they can be merged into a consistent FS
  - Leaf nodes unify if:
    - They are the same
    - One can “subsume” the other
    - Special case: One or both are blank

Feature Unification

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{PERSON} & 3 \end{bmatrix} \cup \begin{bmatrix} \text{CAT} & \text{NP} \\ \text{NUMBER} & \text{SINGULAR} \end{bmatrix}$$

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{NUMBER} & \text{SINGULAR} \\ \text{PERSON} & 3 \end{bmatrix}$$

Feature Unification

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{PERSON} & 3 \end{bmatrix} \sqcup \begin{bmatrix} \text{CAT} & \text{NP} \\ \text{PERSON} & 1 \end{bmatrix}$$

FAILURE

Example in NLTK

```
fs2 = FeatStruct(type='NP', agr=fs1)
print (fs2)
[ agr  = [ number = 'singular' ] ]
[        [ person = 3          ] ]
[                                ]
[ type = 'NP'                    ]
fs3 = FeatStruct(agr=FeatStruct(number=Variable('?n')),
subj=FeatStruct(number=Variable('?n')))
print(fs3)
[ agr  = [ number = ?n ] ]
[                        ]
[ subj = [ number = ?n ] ]
print(fs2.unify(fs3))
[ agr  = [ number = 'singular' ] ]
[        [ person = 3          ] ]
[                                ]
[ subj = [ number = 'singular' ] ]
[                                ]
[ type = 'NP'                    ]
```

http://www.nltk.org/howto/featstruct.html

Agreement with Features

- S $\rightarrow$ NP VP

  {NP PERSON} = {VP PERSON}

- S $\rightarrow$ Aux NP VP

  {Aux PERSON} = {NP PERSON}

- Verb $\rightarrow$ bites

  {Verb PERSON} = 3

- Verb $\rightarrow$ bite

  {Verb PERSON} = 1

Types in Semantics

- ***e*** – entities, ***t*** – facts
- **<e,t>** : unary predicates – maps entities to facts
- **<e,<e,t>>** : binary predicates
- **<<e,t>,t>** : type-raised entities
- Examples:
  - "Jorge", "he", A123: e
  - "Janice likes cats": t
  - "likes": <e,<e,t>>
  - "likes cats": <e,t>
  - "every person": <<e,t>,t>

Type Coercion

- Programming languages
  - How is it done in your favorite programming language?
- Examples in natural language
  - I had a coffee this morning (-> one cup of coffee)
  - I tried two wines last night (-> two types of wine)
  - I had fish for dinner (-> some fish, not “a fish”)

Subtypes and Selectional Restrictions

- Type hierarchy
  - object > edible object > fruit > banana
  - noun > count noun
  - noun > mass noun
- Selectional restrictions
  - Some verbs can only take arguments of certain types
  - Example: eat + “edible object”, believe + “idea“
- Selectional restrictions and type coercion (metonymy)
  - I have read this title (“title” -> “book”)
  - I like Shakespeare (“Shakespeare” -> “works by Shakespeare”)

Subcategorization with Features

- VP $\rightarrow$ Verb

  {VP SUBCAT} = {Verb SUBCAT}
  {VP SUBCAT} = INTRANS

- VP $\rightarrow$ Verb NP

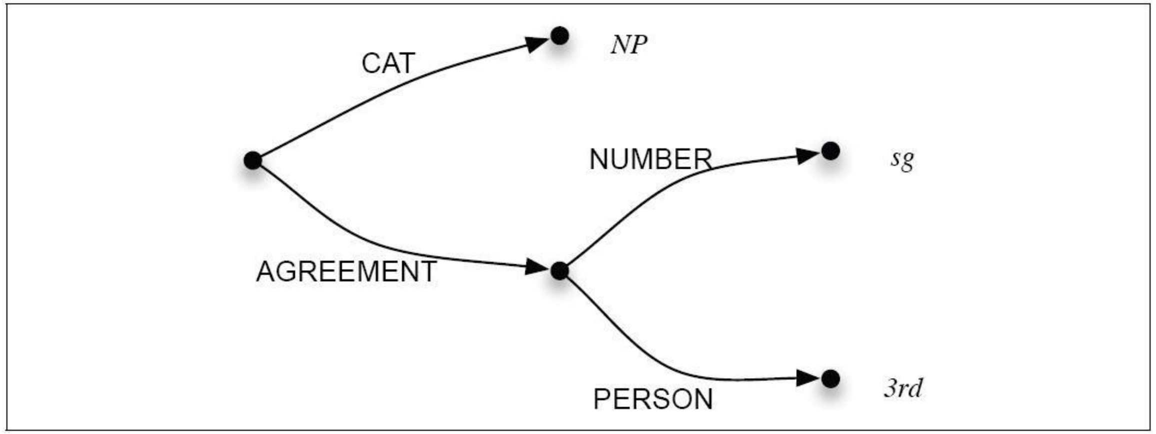
  {VP SUBCAT} = {Verb SUBCAT}
  {VP SUBCAT} = TRANS

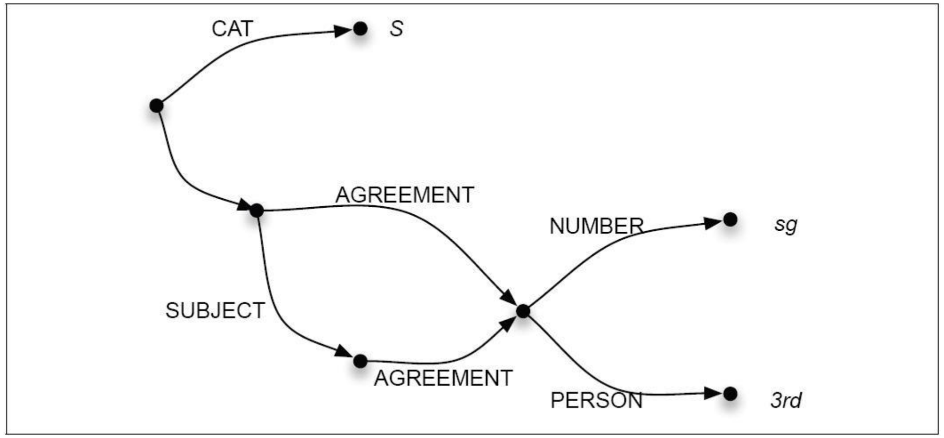
- VP $\rightarrow$ Verb NP NP

  {VP SUBCAT} = {Verb SUBCAT}
  {VP SUBCAT} = DITRANS

Representing FSs as DAGs

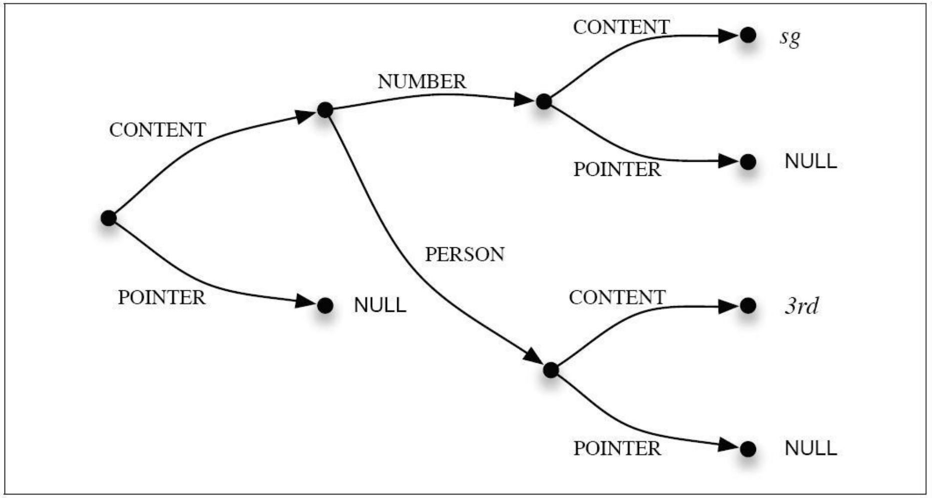
- FS = feature structure
- DAG = directed acyclic graph (not a tree and not an arbitrary graph)

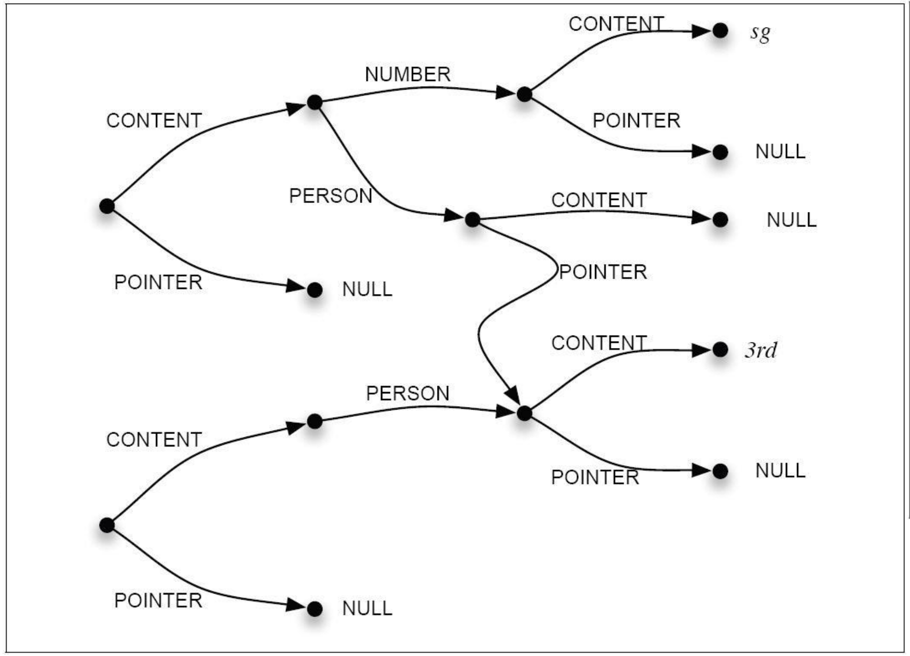


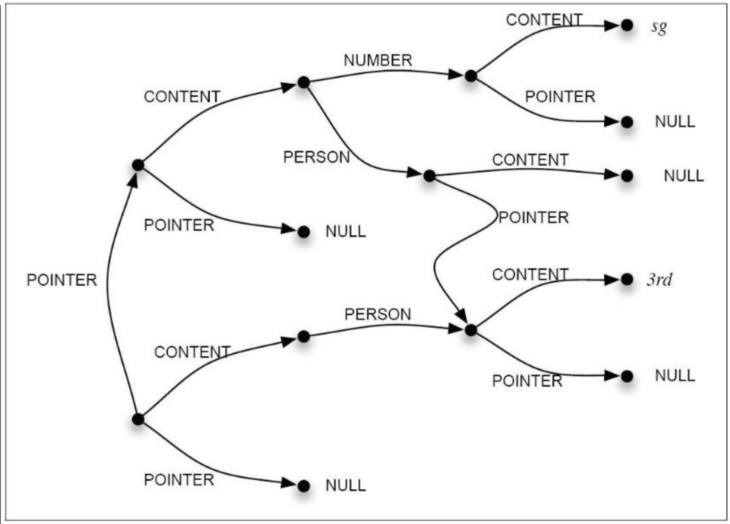


[Example from Jurafsky and Martin]

FS Unification







[Example from Jurafsky and Martin]

Unification Procedure

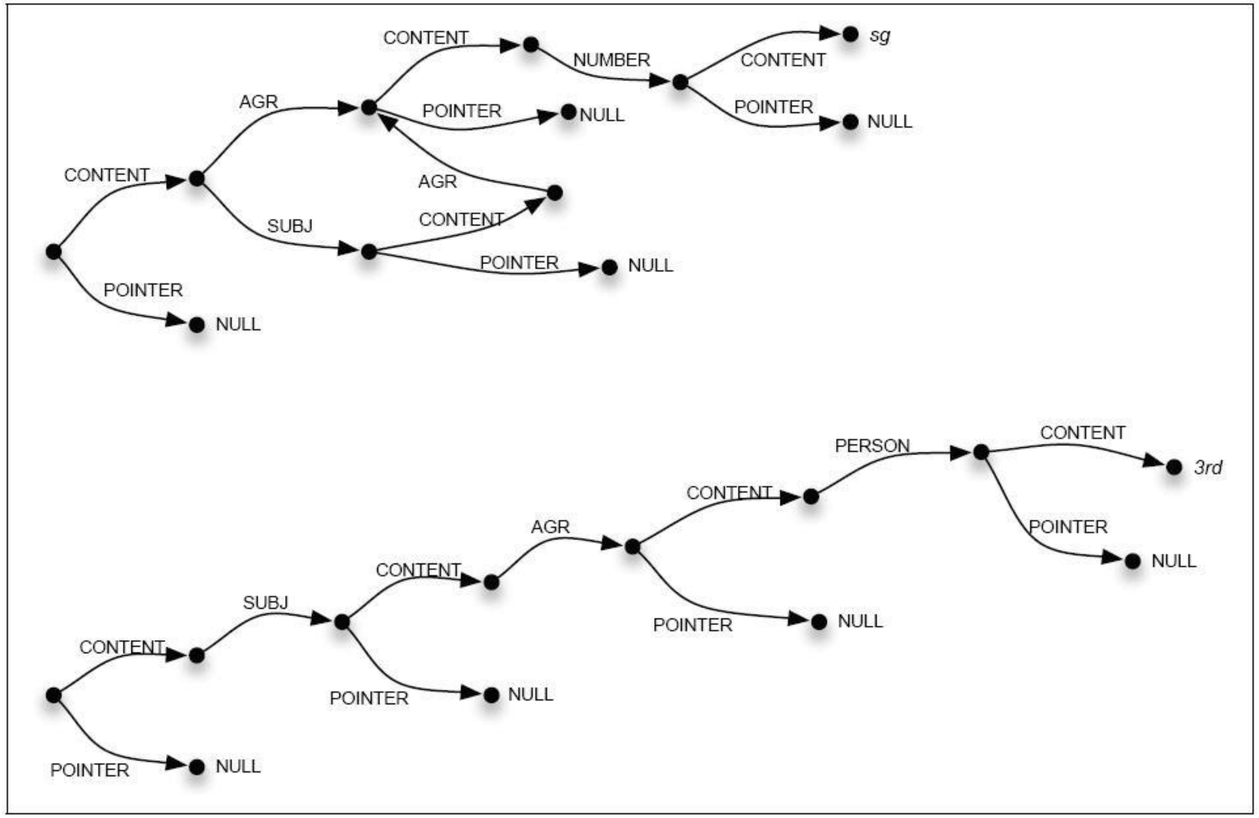
```
function UNIFY(f1-orig, f2-orig) returns f-structure or failure

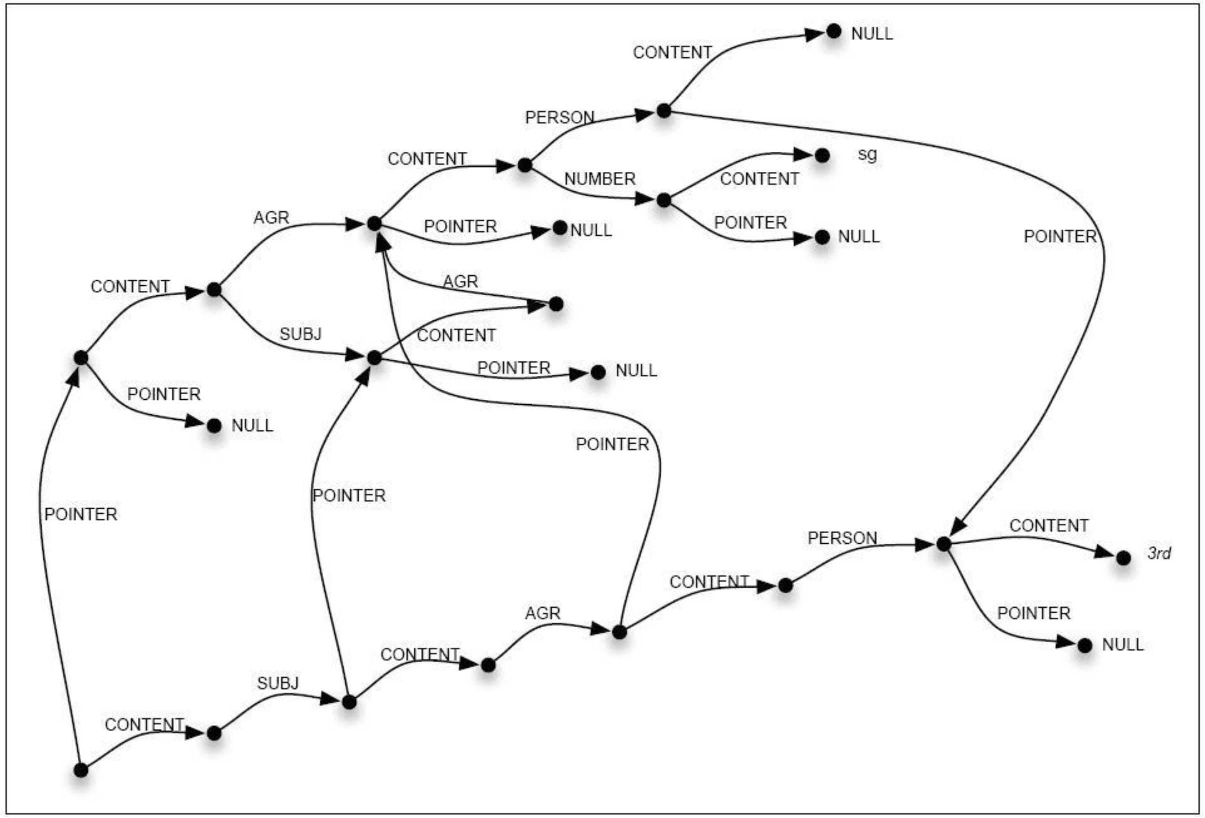
 f1 ← Dereferenced contents of f1-orig
 f2 ← Dereferenced contents of f2-orig

 if f1 and f2 are identical then
    f1.pointer ← f2
    return f2
 else if f1 is null then
   f1.pointer ← f2
   return f2
 else if f2 is null then
   f2.pointer ← f1
   return f1
 else if both f1 and f2 are complex feature structures then
    f2.pointer ← f1
    for each f2-feature in f2 do
      f1-feature ← Find or create a corresponding feature in f1
      if UNIFY(f1-feature.value, f2-feature.value) returns failure then
         return failure
    return f1
 else return failure
```

[Example from Jurafsky and Martin]

FS Unification





[Example from Jurafsky and Martin]

Subsumption

- Unification of a more general concept with a more specific concept
- “undefined” is the most general concept
- “fail” is the least general concept

Subcategorization

Noun Phrase Types		
There	nonreferential there	**There** *is still much to learn*
It	nonreferential it	**It** *was evident that my ideas*
NP	noun phrase	*As he was relating* **his story**
Preposition Phrase Types		
PP	preposition phrase	*couch their message* **in terms**
PPing	gerundive PP	*censured him* **for not having intervened**
PPpart	particle	*turn it* **off**
Verb Phrase Types		
VPbrst	bare stem VP	*she could* **discuss it**
VPto	to-marked infin. VP	*Why do you want* **to know***?*
VPwh	wh-VP	*it is worth considering* **how to write**
VPing	gerundive VP	*I would consider* **using it**
Complement Clause types		
Sfin	finite clause	*maintain* **that the situation was unsatisfactory**
Swh	wh-clause	*it tells us* **where we are**
Sif	whether/if clause	*ask* **whether Aristophanes is depicting a**
Sing	gerundive clause	*see* **some attention being given**
Sto	to-marked clause	*know* **themselves to be relatively unhealthy**
Sforto	for-to clause	*She was waiting* **for him to make some reply**
Sbrst	bare stem clause	*commanded* **that his sermons be published**
Other Types		
AjP	adjective phrase	*thought it* **possible**
Quo	quotes	*asked* **"What was it like?"**

[Example from Jurafsky and Martin]

Subcategorization

Subcat	Example
Quo	asked [$_{Quo}$ "What was it like?"]
NP	asking [$_{NP}$ a question]
Swh	asked [$_{Swh}$ what trades you're interested in]
Sto	ask [$_{Sto}$ him to tell you]
PP	that means asking [$_{PP}$ at home]
Vto	asked [$_{Vto}$ to see a girl called Evelyn]
NP Sif	asked [$_{NP}$ him] [$_{Sif}$ whether he could make]
NP NP	asked [$_{NP}$ myself] [$_{NP}$ a question]
NP Swh	asked [$_{NP}$ him] [$_{Swh}$ why he took time off]

[Example from Jurafsky and Martin]

NLP

Introduction to NLP

264.

Combinatory Categorial Grammar (CCG)

Combinatory Categorial Grammar (CCG)

- Complex types
  - E.g., *X/Y* and *X\Y*
  - These take an argument of type *Y* and return an object of type *X*.
  - X/Y – means that Y should appear on the right
  - X\Y – means that Y should appear on the left

Structure of CCG

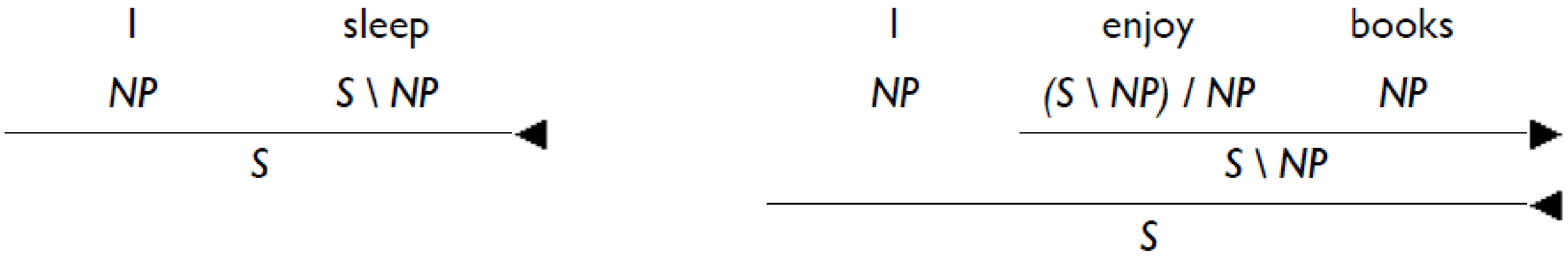
- Categories
- Combinatory rules
- Lexicon

CCG Rules

- Function composition
  - X/Y   Y/Z   ->   X/Z
  - X\Y   Z\X   ->   Z\Y
  - X/Y   Y\Z   ->   X\Z
  - X/Y   Z\X   ->   Z/Y
- Type raising
  - X -> Y/(Y\X)
  - X -> Y\(Y/X)
- Coordination

Example

I	NP
books	NP
sleep	S \ NP
enjoy	(S \ NP) / NP



Example from Jonathan Kummerfeld, Aleka Blackwell, and Patrick Littell

Expressive power

- CCGs can generate the language $a^n b^n c^n d^n$, $n>0$
- Interesting examples:
  - I like New York
  - I like and hate New York
  - I like and would rather be in New York
  - I gave a book to Chen and a laptop to Jorge
  - I want Chen to stay and Jorge to leave
  - I like and Chen hates, New York
    - Where are the verb phrases?

Examples from Steedman 1996

(6) *Forward Application:* ($>$)

$$X/Y : f \quad Y : a \quad \Rightarrow \quad X : fa$$

(7) *Backward Application:* ($<$)

$$Y : a \quad X\backslash Y : f \quad \Rightarrow \quad X : fa$$

They yield derivations like the following:

(8)

$$\begin{array}{ccc}
\text{Mary} & \text{likes} & \text{musicals} \\
\hline
NP_{3sm} : mary' & (S\backslash NP_{3s})/NP : like' & NP : musicals' \\
\cline{2-3}
 & \multicolumn{2}{c}{S\backslash NP_{3s} : like'\,musicals' \quad >} \\
\hline
\multicolumn{3}{c}{S : like'\,musicals'\,mary' \quad <}
\end{array}$$

Examples from Steedman 1996

(9) *Coordination:* $(<\&>)$

$$X \quad conj \quad X \quad \Rightarrow \quad X$$

(10)

$$\begin{array}{ccccc}
\text{I} & \text{loathe} & \text{and} & \text{detest} & \text{opera} \\
\hline
NP & (S\backslash NP)/NP & CONJ & (S\backslash NP)/NP & NP \\
 & \multicolumn{3}{c}{\overline{\quad(S\backslash NP)/NP\quad}^{<\&>}} & \\
 & \multicolumn{4}{c}{\overline{\quad S\backslash NP\quad}^{>}} \\
\multicolumn{5}{c}{\overline{\quad S\quad}^{<}}
\end{array}$$

(13) *Forward Composition:* $(>B)$

$$X/Y : f \quad Y/Z : g \quad \Rightarrow \quad X/Z : \lambda x.f(gx)$$

(14)

$$\begin{array}{cccccc}
\text{I} & \text{requested} & \text{and} & \text{would} & \text{prefer} & \text{musicals} \\
\hline
NP & (S\backslash NP)/NP & CONJ & (S\backslash NP)/VP : will' & VP/NP : prefer' & NP \\
 & & & \multicolumn{2}{c}{\overline{\quad(S\backslash NP)/NP : \lambda x.\lambda y.will'(prefer'x)y\quad}^{>B}} & \\
 & \multicolumn{4}{c}{\overline{\quad(S\backslash NP)/NP\quad}^{<\&>}} & \\
 & \multicolumn{5}{c}{\overline{\quad S\backslash NP\quad}^{>}} \\
\multicolumn{6}{c}{\overline{\quad S\quad}^{<}}
\end{array}$$

Examples from Steedman 1996

(18)

introduce	Bill	to Sue	and	Harry to George
$(VP/PP)/NP$	$(VP/PP)\backslash((VP/PP)/NP)$ $<T$	$VP\backslash(VP/PP)$ $<T$	$CONJ$	$VP\backslash((VP/PP)/NP)$ $<B$

$$\dfrac{(VP/PP)\backslash((VP/PP)/NP) \quad VP\backslash(VP/PP)}{VP\backslash((VP/PP)/NP)}{<}B$$

$$\dfrac{VP\backslash((VP/PP)/NP) \quad CONJ \quad VP\backslash((VP/PP)/NP)}{VP\backslash((VP/PP)/NP)}{<}\&{>}$$

$$\dfrac{(VP/PP)/NP \quad VP\backslash((VP/PP)/NP)}{VP}{<}$$

CCG in NLTK

```
from nltk.ccg import chart, lexicon
lex = lexicon.parseLexicon('''
:- S, NP, N, VP
Det :: NP/N
Pro :: NP
Modal :: S\\NP/VP
TV :: VP/NP
DTV :: TV/NP
the => Det
that => Det
that => NP
I => Pro
you => Pro
we => Pro
chef => N
cake => N
children => N
dough => N
will => Modal
should => Modal
might => Modal
must => Modal
and => var\\.,var/.,var
to => VP[to]/VP
without => (VP\\VP)/VP[ing]
be => TV
cook => TV
eat => TV
cooking => VP[ing]/NP
give => DTV
is => (S\\NP)/NP
prefer => (S\\NP)/NP
which => (N\\N)/(S/NP)
persuade => (VP/VP[to])/NP
''')
```

http://www.nltk.org/howto/ccg.html

CCG in NLTK

```
parser = chart.CCGChartParser(lex, chart.DefaultRuleSet)
for parse in parser.parse("you prefer that cake".split()):
    chart.printCCGDerivation(parse)
    break
```

```
 you    prefer      that   cake
 NP   ((S\NP)/NP)  (NP/N)   N
-----
 NP
     -------------
      ((S\NP)/NP)
                  --------
                   (NP/N)
                          ------
                            N
                  --------------->
                        NP
     ---------------------------->
              (S\NP)
--------------------------------<
              S
```

http://www.nltk.org/howto/ccg.html

```
for parse in parser.parse("that is the cake which you prefer".split()):
    chart.printCCGDerivation(parse)
    break
```

```
 that      is        the    cake      which       you    prefer
  NP  ((S\NP)/NP)  (NP/N)   N   ((N\N)/(S/NP))  NP  ((S\NP)/NP)
------
  NP
      -------------
       ((S\NP)/NP)
                   --------
                    (NP/N)
                           ------
                             N
                                 ----------------
                                  ((N\N)/(S/NP))
                                                 -----
                                                  NP
                                                 ----->T
                                              (S/(S\NP))
                                                      -------------
                                                       ((S\NP)/NP)
                                                 ------------------>B
                                                       (S/NP)
                                 ---------------------------------->
                                               (N\N)
                           ----------------------------------------<
                                              N
                   ------------------------------------------------>
                                          NP
      ------------------------------------------------------------->
                                 (S\NP)
-------------------------------------------------------------------<
                                 S
```

http://www.nltk.org/howto/ccg.html

CCG

- NACLO problem from 2014 (in two parts)
- Authors: Jonathan Kummerfeld, Aleka Blackwell, and Patrick Littell
- http://www.nacloweb.org/resources/problems/2014/N2014-O.pdf
- http://www.nacloweb.org/resources/problems/2014/N2014-OS.pdf
- http://www.nacloweb.org/resources/problems/2014/N2014-P.pdf
- http://www.nacloweb.org/resources/problems/2014/N2014-PS.pdf

CCG Parsing

- CKY works fine
- http://openccg.sourceforge.net/

Parse 1:

Fruit	flies	like	a	banana
N	S[dcl]\NP	((S\NP)\(S\NP))/NP	NP[nb]/N	N
NP			NP[nb]	
		(S\NP)\(S\NP)		
	S[dcl]\NP			
S[dcl]				

Parse 2:

Fruit	flies	like	a	banana
N	(S[dcl]\NP)/PP	PP/NP	NP[nb]/N	N
NP			NP[nb]	
		PP		
	S[dcl]\NP			
S[dcl]				

Parse 3:

Fruit	flies	like	a	banana
N	S[dcl]\NP	(S\S)/NP	NP[nb]/N	N
NP			NP[nb]	
S[dcl]		S\S		
S[dcl]				

Parse 4:

Fruit	flies	like	a	banana
N	N\N	(S[dcl]\NP)/NP	NP[nb]/N	N
N			NP[nb]	
NP		S[dcl]\NP		
S[dcl]				

Parse 5:

Fruit	flies	like	a	banana
N/N	N	(S[dcl]\NP)/NP	NP[nb]/N	N
N			NP[nb]	
NP		S[dcl]\NP		
S[dcl]				

http://4.easy-ccg.appspot.com/do_parse?sentence=Fruit+flies+like+a+banana&nbest=5

Exercise

- How do you represent the following categories in CCG
  - Nouns
  - Adjectives
  - Articles
  - Prepositions
  - Transitive verbs
  - Intransitive verbs

Exercise

- How do you represent the following categories in CCG
  - Nouns N
  - Adjectives N/N
  - Articles NP/N
  - Prepositions (NP\NP)/NP
  - Transitive verbs (S\NP)/NP
  - Intransitive verbs S\NP

CCG for Dutch Cross-Serial Dependencies

… because I$_1$ Cecilia$_2$ Henk$_3$ the hippopotamuses$_4$ saw$_1$ help$_2$ feed$_{3,4}$.

dat	*ik*	*Cecilia*	*Henk*	*de nijlpaarden*	*zag*	*helpen*	*voeren*
	NP_1	NP_2	NP_3	NP_4	$((S\backslash NP_1)\backslash NP_2)/VP$	$(VP\backslash NP_3)/VP$	$VP\backslash NP_4$

$$\frac{(VP\backslash NP_3)/VP \qquad VP\backslash NP_4}{(VP\backslash NP_3)\backslash NP_4} >\mathbf{B}_\times$$

$$\frac{((S\backslash NP_1)\backslash NP_2)/VP \qquad (VP\backslash NP_3)\backslash NP_4}{(((S\backslash NP_1)\backslash NP_2)\backslash NP_3)\backslash NP_4} >\mathbf{B}^2_\times$$

$$\frac{NP_4 \qquad (((S\backslash NP_1)\backslash NP_2)\backslash NP_3)\backslash NP_4}{((S\backslash NP_1)\backslash NP_2)\backslash NP_3} <$$

$$\frac{NP_3 \qquad ((S\backslash NP_1)\backslash NP_2)\backslash NP_3}{(S\backslash NP_1)\backslash NP_2} <$$

$$\frac{NP_2 \qquad (S\backslash NP_1)\backslash NP_2}{S\backslash NP_1} <$$

$$\frac{NP_1 \qquad S\backslash NP_1}{S} <$$

[Example from Stephen Clark]

CCGBank

- Hockenmaier and Steedman (2005)

Sentence 1

```
{S[dcl] {S[dcl] {NP {NP {NP {NP {N {N/N  Pierre}
                                   {N  Vinken}}}
                            {,  ,}}
                       {NP\NP {S[adj]\NP {NP {N {N/N  61}
                                                {N  years}}}
                                         {(S[adj]\NP)\NP  old}}}}
                   {,  ,}}
              {S[dcl]\NP {(S[dcl]\NP)/(S[b]\NP)  will}
                         {S[b]\NP {S[b]\NP {(S[b]\NP)/PP {((S[b]\NP)/PP)/NP  join}
                                                         {NP {NP[nb]/N  the}
                                                             {N  board}}}
                                           {PP {PP/NP  as}
                                               {NP {NP[nb]/N  a}
                                                   {N {N/N  nonexecutive}
                                                      {N  director}}}}}
                                  {(S\NP)\(S\NP) {((S\NP)\(S\NP))/N  Nov.}
                                                 {N  29}}}}}
   {.  .}}
```

Pierre	(N/N)	Vinken
61	(N/N)	years
old	((S[adj]\NP)\NP)	*Vinken* years
will	((S[dcl]\NP)/(S[b]\NP))	Vinken join
join	(((S[b]\NP)/PP)/NP)	*Vinken* as board
the	(NP[nb]/N)	board
as	(PP/NP)	director
a	(NP[nb]/N)	director
nonexecutive	(N/N)	director
Nov.	(((S\NP)\(S\NP))/N)	join 29

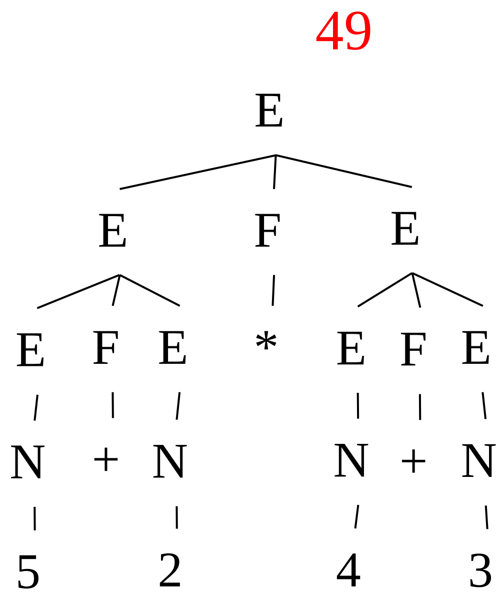
NLP

Introduction to NLP

362.
First Order Logic

Semantics

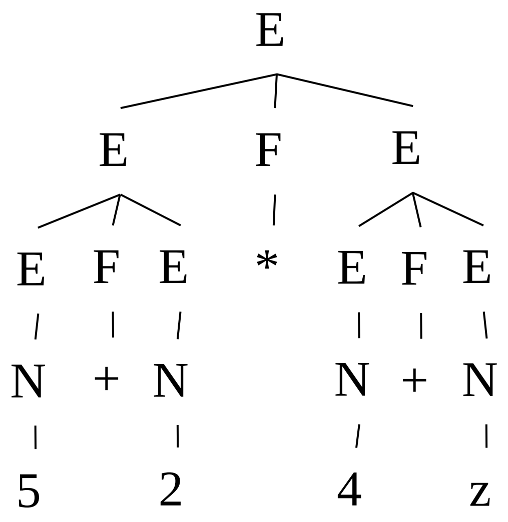
- What is the meaning of: (5+2)*(4+3)?
- Parse tree



Semantics

- What if we had (5+2)*(4+z)?

mult(add(5,2),add(4,z))



What about (English) sentences?

- Socrates is a human.
- Every human is mortal.

Representing Meaning

- Goal
  - Capturing the meaning of linguistic utterances using formal notation
- Linguistic meaning
  - "It is 8 pm"
- Pragmatic meaning
  - "It is time to leave"
- Semantic analysis:
  - Assign each word a meaning
  - Combine the meanings of words into sentences
- *I bought a book*:

  ∃ *x,y*: Buying(*x*) ^ Buyer(*speaker,x*) ^ BoughtItem(*y,x*) ^ Book(*y*)

  Buying (Buyer=speaker, BoughtItem=book)


LOOK SIMBA, THIS IS LINGUISTICS. YOU MAY STUDY ANY PART OF THE LAND.
WHAT ABOUT THAT DARK SHADOWY PLACE?
THAT IS SEMANTICS. YOU MUST NEVER GO THERE.
zipmeme

First Order Logic

- Used to represent
  - Objects – Martin the cat
  - Relations – Martin and Moses are brothers
  - Functions – Martin's age

First Order Logic

- *Formula* $\rightarrow$ AtomicFormula | Formula Connective Formula
  | Quantifier Variable Formula | ¬ Formula | (Formula)
- AtomicFormula $\rightarrow$ Predicate (Term…)
- Term $\rightarrow$ Function (Term…) | Constant | Variable
- Connective $\rightarrow$ ∧ | ∨ | ⇒
- Quantifier $\rightarrow$ ∀ | ∃
- Constant $\rightarrow$ M | Martin
- Variable $\rightarrow$ *x* | *y* | …
- Predicate $\rightarrow$ Likes | Eats | …
- Function $\rightarrow$ AgeOf | ColorOf | …

Types

- Base types
  - e (entity) – objects
  - t (truth values)
- Complex types
  - If a is a type and b is a type, then a→b is a type.
  - (a→b)(a)=b
- Example
  - Type of *Mary* = e
  - Type of *sleeps* = e→t
  - Type of *sleeps(Mary)* = t
  - Type of *^ = t→t*
  - Type of *^(sleeps(Mary))* = t
  - * *^(sleeps)* - not well typed

Lambda Expressions

- Example
  - inc(x) = λx x+1
  - then inc(4) = (λx x+1)(4) = 5
- Example
  - add(x,y) = λx,λy(x+y)
  - then add(3,4) = (λx,λy(x+y))(3)(4)= (λy 3+y)(4) = 3+4 = 7
- Useful for semantic parsing (see later)

Lambda Expressions

- $\lambda x.\mathit{like}(x,\mathrm{Mary})$
- $\lambda x.\mathit{like}(\mathrm{Mary},x)$
- $\lambda x.(\lambda y.\mathit{like}(x,y))$
- $\lambda P.P(\mathrm{Mary})$
  - property is true of Mary

Lambda Expressions

- [λx.sleeps(x)](Mary)=sleeps(Mary)
- [λx.likes(Mary,x)](John)=likes(Mary,John)
- [λx.likes(x,y)](Mary)=likes(Mary,Mary)

Example

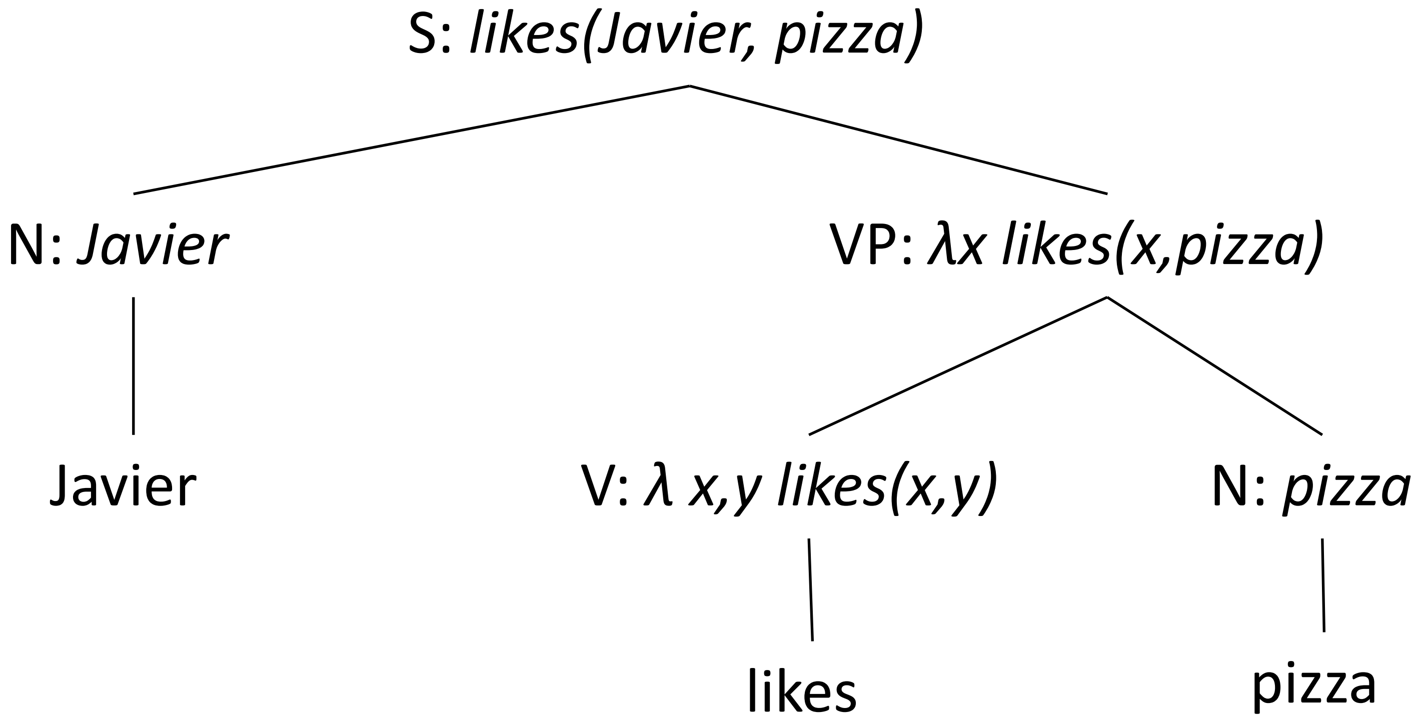
- Input
  - Javier likes pizza
- Output
  - *like(Javier, pizza)*

Example

```
S  -> NP VP   {VP.Sem(NP.Sem)}     t
VP -> V NP    {V.Sem(NP.Sem)}      <e,t>
NP -> N       {N.Sem}              e
V  -> likes   {λ x,y likes(x,y)    <e,<e,t>>
N  -> Javier  {Javier}             e
N  -> pizza   {pizza}              e
```

Semantic Parsing (preview)

- Associate a semantic expression with each node



NLP

Introduction to NLP

363.

Knowledge Representation

Knowledge Representation

- Ontologies
- Categories and objects
- Events
- Times
- Beliefs

Knowledge Representation

- Object
  - Martin the cat
- Categories
  - Cat
- Ontology
  - Mammal includes Cat, Dog, Whale
  - Cat includes PersianCat, ManxCat
- ISA relation
  - ISA (Martin,Cat)
- AKO relation
  - AKO (PersianCat,Cat)
- HASA relation
  - HASA (Cat, Tail)

Semantics of FOL

- FOL sentences can be assigned a value of *true* or *false.*
  - *ISA(Milo,Cat) = true*
- *Milo is younger than Martin*
  - *<(AgeOf(Milo),AgeOf(Martin)) = true*
  - *=(AgeOf(Milo),AgeOf(Martin)) = false*

Examples with Quantifiers

- All cats eat fish

  $\forall x{:}ISA(x,Cat) \Rightarrow EatFish(x)$

Representing Events

- Martin ate
- Martin ate in the morning
- Martin ate fish
- Martin ate fish in the morning

One Possible Representation

- FOL representations
  - Eating1(Martin)
  - Eating2(Martin,Morning)
  - Eating3(Martin,Fish)
  - Eating4(Martin,Fish,Morning)
- Meaning postulates
  - Eating4(x,y,z) -> Eating3(x,y)
  - Eating4(x,y,z) -> Eating2(x,z)
  - Eating4(x,y,z) -> Eating1(x)

Example from Jurafsky and Martin

Second Possible Representation

- Eating4(x,y,z)
  - With some arguments unspecified
- Problems
  - Too many commitments
  - Hard to combine Eating4(Martin,Fish,z) with Eating4(Martin,y,Morning)

Example from Jurafsky and Martin

Third Possible Representation

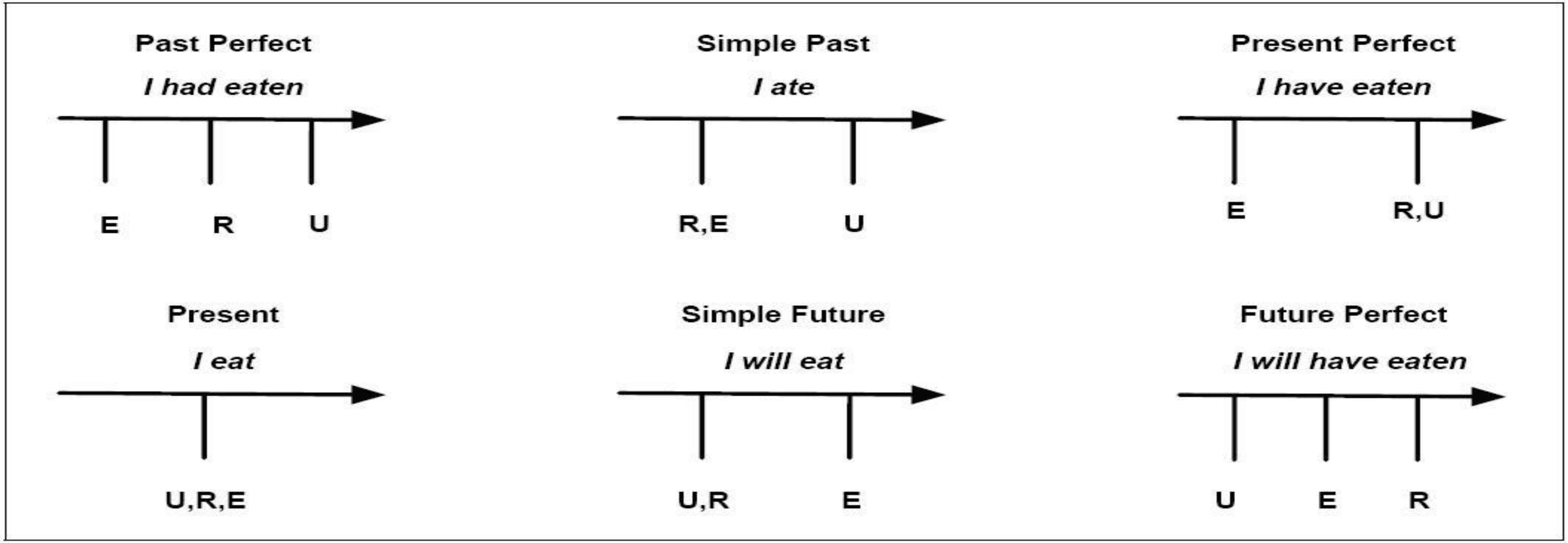
- Reification
  - $\exists$ e: ISA(e,Eating) $\wedge$ Eater(e,Martin) $\wedge$ Eaten(e,Fish)

Example from Jurafsky and Martin

Representing Time

- Example
  - Martin went from the kitchen to the yard
  - ISA(e,Going) ^ Goer(e,Martin) ^ Origin (e,kitchen) ^ Target (e,yard)
- Issue
  - no tense information: past? present? future?
- Fluents
  - A predicate that is true at a given time: T(f,t)

Representing Time



Example from Jurafsky and Martin

Representing time

- $\exists$ *i,e,w,t: Isa(w,Arriving)* $\wedge$ *Arriver(w,Speaker)* $\wedge$ *Destination(w,NewYork)* $\wedge$ *IntervalOf(w,i)* $\wedge$ *EndPoint(i,e)* $\wedge$ *Precedes (e,Now)*
- $\exists$ *i,e,w,t: Isa(w,Arriving)* $\wedge$ *Arriver(w,Speaker)* $\wedge$ *Destination(w,NewYork)* $\wedge$ *IntervalOf(w,i)* $\wedge$ *MemberOf(i,Now)*
- $\exists$ *i,e,w,t: Isa(w,Arriving)* $\wedge$ *Arriver(w,Speaker)* $\wedge$ *Destination(w,NewYork)* $\wedge$ *IntervalOf(w,i)* $\wedge$ *StartPoint(i,s)* $\wedge$ *Precedes (Now,s)*

Example from Jurafsky and Martin

Aspect

- Stative
  - I know my departure gate
- Activity
  - John is flying
    (no particular end point)
- Accomplishment
  - Sally booked her flight
    (natural end point and result in a particular state)
- Achievement
  - She found her gate
- Figuring out statives:
  - I am needing the cheapest fare.
  - I am wanting to go today.
  - Need the cheapest fare!

Example from Jurafsky and Martin

Representing Beliefs

- Example
  - Milo believes that Martin ate fish
- One possible representation
  - $\exists$ e,b: ISA(e,Eating) $\wedge$ Eater(e,Martin) $\wedge$ Eaten(e,Fish) $\wedge$ ISA(b,Believing) $\wedge$ Believer(b,Milo) $\wedge$ Believed(b,e)
- However this implies (by dropping some of the terms) that "Martin ate fish" (without the Belief event)
- Modal logic
  - Possibility, Temporal Logic, Belief Logic

Representing Beliefs

- Want, believe, imagine, know: all introduce hypothetical worlds
- I believe that Mary ate British food.
- Reified example:
  - *∃ u,v: Isa(u,Believing) ∧ Isa(v,Eating) ∧ Believer (u,Speaker) ∧ BelievedProp(u,v) ∧ Eater(v,Mary) ∧ Eaten(v,BritishFood)*

  However this implies also*:*
  - *∃ u,v: Isa(v,Eating) ∧ Eater(v,Mary) ∧ Eaten(v,BritishFood)*
- Modal operators:
  - *Believing(Speaker,Eating(Mary,BritishFood))* - not FOPC! – predicates in FOPC hold between objects, not between relations.
  - *Believes(Speaker, ∃ v: ISA(v,Eating) ∧ Eater(v,Mary) ∧ Eaten(v,BritishFood))*

NLP

Introduction to NLP

364.

Inference

Modus Ponens

- Modus ponens:

$$\frac{\alpha \qquad \alpha \Rightarrow \beta}{\beta}$$

- Example:

$$\frac{Cat(Martin) \qquad \forall x: Cat(x) \Rightarrow EatsFish(x)}{EatsFish(Martin)}$$

Inference

- Forward chaining
  - as individual facts are added to the database, all derived inferences are generated
- Backward chaining
  - starts from queries
  - Example: the Prolog programming language
- Prolog example
  - ```
    father(X, Y) :- parent(X, Y), male(X).
    parent(john, bill).
    parent(jane, bill).
    female(jane).
    male (john).
    ?- father(M, bill).
    ```
    ```

The Kinship Domain

- Brothers are siblings

  $\forall x,y \; Brother(x,y) \Rightarrow Sibling(x,y)$

- One's mother is one's female parent

  $\forall m,c \; Mother(c) = m \Leftrightarrow (Female(m) \wedge Parent(m,c))$

- "Sibling" is symmetric

  $\forall x,y \; Sibling(x,y) \Leftrightarrow Sibling(y,x)$

Universal Instantiation

- Every instantiation of a universally quantified sentence is entailed by it:

$$\frac{\forall v\ \alpha}{\text{Subst}(\{v/g\}, \alpha)}$$

  for any variable *v* and ground term *g*

- E.g., $\forall x\ Cat(x) \land Fish(y) \Rightarrow Eats(x,y)$ yields:

  $Cat(Martin) \land \text{Fish(Blub)} \Rightarrow Eats(Martin,Blub)$

Existential Instantiation

- For any sentence α, variable *v*, and constant symbol *k* that does not appear elsewhere in the knowledge base:

$$\frac{\exists v\ \alpha}{\text{Subst}(\{v/k\}, \alpha)}$$

- E.g., $\exists x\ Cat(x) \wedge EatsFish(x)$ yields:

$$Cat(C_1) \wedge EatsFish(C_1)$$

provided C_1 is a new constant symbol, called a Skolem constant

Unification

- If a substitution θ is available, unification is possible
- Examples:
  - p=Eats(x,y), q=Eats(x,Blub), possible if θ = {y/Blub}
  - p=Eats(Martin,y), q=Eats(x,Blub), possible if θ = {x/Martin,y/Blub}
  - p=Eats(Martin,y), q=Eats(y,Blub), fails because Martin$\neq$Blub
- Subsumption
  - Unification works not only when two things are the same but also when one of them subsumes the other one
  - Example: All cats eat fish, Martin is a cat, Blub is a fish

NLP

Introduction to NLP

365.

Semantic Parsing

Semantic Parsing

- Converting natural language to a logical form
  - e.g., executable code for a specific application
- Example:
  - Airline reservations
  - Geographical query systems

Stages of Semantic Parsing

- Input
  - Sentence
- Syntactic Analysis
  - Syntactic structure
- Semantic Analysis
  - Semantic representation

Compositional Semantics

- Add semantic attachments to CFG rules
- Compositional semantics
  - Parse the sentence syntactically
  - Associate some semantics to each word
  - Combine the semantics of words and non-terminals recursively
  - Until the root of the sentence

Example

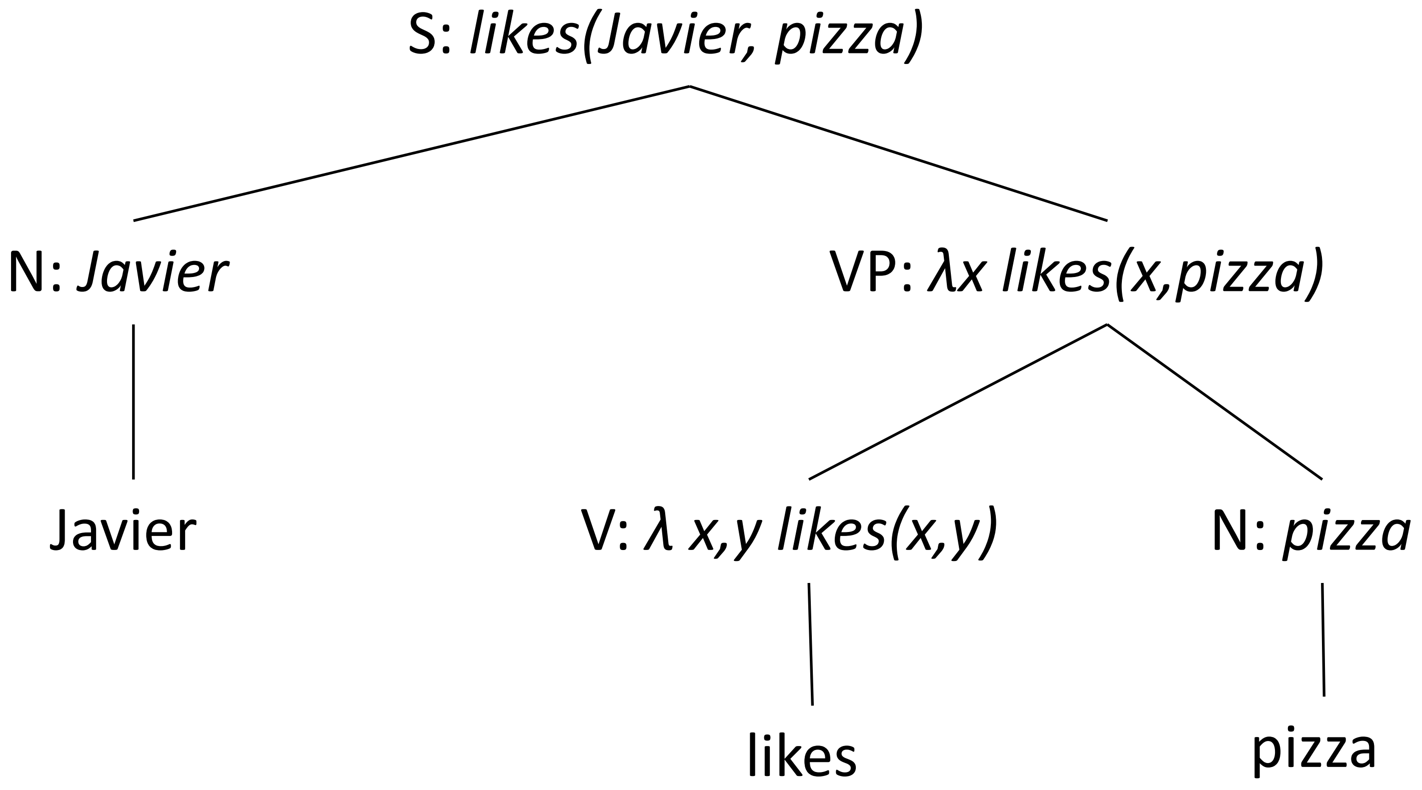
- Input
  - Javier likes pizza
- Output
  - *like(Javier, pizza)*

Example

```
S  -> NP VP   {VP.Sem(NP.Sem)}      t
VP -> V NP    {V.Sem(NP.Sem)}       <e,t>
NP -> N       {N.Sem}               e
V  -> likes   {λ x,y likes(x,y)     <e,<e,t>>
N  -> Javier  {Javier}              e
N  -> pizza   {pizza}               e
```

Semantic Parsing

- Associate a semantic expression with each node



Grammar with Semantic Attachments

Grammar Rule	**Semantic Attachment**
$S \rightarrow NP\ VP$	$\{NP.sem(VP.sem)\}$
$NP \rightarrow Det\ Nominal$	$\{Det.sem(Nominal.sem)\}$
$NP \rightarrow ProperNoun$	$\{ProperNoun.sem\}$
$Nominal \rightarrow Noun$	$\{Noun.sem\}$
$VP \rightarrow Verb$	$\{Verb.sem\}$
$VP \rightarrow Verb\ NP$	$\{Verb.sem(NP.sem)\}$
$Det \rightarrow every$	$\{\lambda P.\lambda Q.\forall x P(x) \Rightarrow Q(x)\}$
$Det \rightarrow a$	$\{\lambda P.\lambda Q.\exists x P(x) \wedge Q(x)\}$
$Noun \rightarrow restaurant$	$\{\lambda r.Restaurant(r)\}$
$ProperNoun \rightarrow Matthew$	$\{\lambda m.m(Matthew)\}$
$ProperNoun \rightarrow Franco$	$\{\lambda f.f(Franco)\}$
$ProperNoun \rightarrow Frasca$	$\{\lambda f.f(Frasca)\}$
$Verb \rightarrow closed$	$\{\lambda x.\exists e Closed(e) \wedge ClosedThing(e,x)\}$
$Verb \rightarrow opened$	$\{\lambda w.\lambda z.w(\lambda x.\exists e Opened(e) \wedge Opener(e,z)$ $\wedge Opened(e,x))\}$

Example from Jurafsky and Martin

Using CCG (Steedman 1996)

- CCG representations for semantics
  - *ADJ:* $\lambda x.tall(x)$
  - (S\NP)/ADJ : $\lambda f.\lambda x.f(x)$
  - *NP: YaoMing*

YaoMing   is   tall

$$\cfrac{\cfrac{}{\begin{array}{c}NP\\ YaoMing\end{array}} \quad \cfrac{\cfrac{}{\begin{array}{c}(S\backslash NP)/ADJ\\ \lambda f.\lambda x.f(x)\end{array}} \quad \cfrac{}{\begin{array}{c}ADJ\\ \lambda x.tall(x)\end{array}}}{\begin{array}{c}S\backslash NP\\ \lambda x.tall(x)\end{array}}>}{\begin{array}{c}S\\ Tall\ (YaoMing)\end{array}}<$$

CCG Parsing

- Example:
  - https://bitbucket.org/yoavartzi/spf
- Tutorial by Artzi, FitzGerald, Zettlemoyer
  - http://yoavartzi.com/pub/afz-tutorial.acl.2013.pdf

GeoQuery (Zelle and Mooney 1996)

What is the capital of the state with the largest population?
`answer(C, (capital(S,C), largest(P, (state(S), population(S,P))))).`

What are the major cities in Kansas?
`answer(C, (major(C), city(C), loc(C,S), equal(S,stateid(kansas)))).`

Type	Form	Example
country	countryid(Name)	countryid(usa)
city	cityid(Name, State)	cityid(austin,tx)
state	stateid(Name)	stateid(texas)
river	riverid(Name)	riverid(colorado)
place	placeid(Name)	placeid(pacific)

Form	Predicate
capital(C)	C is a capital (city).
city(C)	C is a city.
major(X)	X is major.
place(P)	P is a place.
river(R)	R is a river.
state(S)	S is a state.
capital(C)	C is a capital (city).
area(S,A)	The area of S is A.
capital(S,C)	The capital of S is C.
equal(V,C)	variable V is ground term C.
density(S,D)	The (population) density of S is P
elevation(P,E)	The elevation of P is E.
high_point(S,P)	The highest point of S is P.
higher(P1,P2)	P1's elevation is greater than P2's.
loc(X,Y)	X is located in Y.
low_point(S,P)	The lowest point of S is P.
len(R,L)	The length of R is L.
next_to(S1,S2)	S1 is next to S2.
size(X,Y)	The size of X is Y.
traverse(R,S)	R traverses S.

Zettlemoyer and Collins (2005)

a) What states border Texas

$$\lambda x.state(x) \wedge borders(x, texas)$$

b) What is the largest state

$$\arg\max(\lambda x.state(x), \lambda x.size(x))$$

c) What states border the state that borders the most states

$$\lambda x.state(x) \wedge borders(x, \arg\max(\lambda y.state(y),$$
$$\lambda y.count(\lambda z.state(z) \wedge borders(y, z))))$$

Utah	:=	NP
Idaho	:=	NP
borders	:=	$(S\backslash NP)/NP$

Utah	:=	$NP : utah$
Idaho	:=	$NP : idaho$
borders	:=	$(S\backslash NP)/NP : \lambda x.\lambda y.borders(y, x)$

a)

$$\begin{array}{ccc}
\text{Utah} & \text{borders} & \text{Idaho} \\
\hline
NP & (S\backslash NP)/NP & NP \\
utah & \lambda x.\lambda y.borders(y,x) & idaho \\
 & \multicolumn{2}{c}{\overline{\qquad\qquad\qquad\qquad}^{>}} \\
 & \multicolumn{2}{c}{(S\backslash NP)} \\
 & \multicolumn{2}{c}{\lambda y.borders(y, idaho)} \\
\multicolumn{3}{c}{\overline{\qquad\qquad\qquad\qquad\qquad\qquad}^{<}} \\
\multicolumn{3}{c}{S} \\
\multicolumn{3}{c}{borders(utah, idaho)}
\end{array}$$

b)

$$\begin{array}{cccc}
\text{What} & \text{states} & \text{border} & \text{Texas} \\
\hline
(S/(S\backslash NP))/N & N & (S\backslash NP)/NP & NP \\
\lambda f.\lambda g.\lambda x.f(x) \wedge g(x) & \lambda x.state(x) & \lambda x.\lambda y.borders(y,x) & texas \\
\multicolumn{2}{c}{\overline{\qquad\qquad\qquad\qquad}^{>}} & \multicolumn{2}{c}{\overline{\qquad\qquad\qquad\qquad}^{>}} \\
\multicolumn{2}{c}{S/(S\backslash NP)} & \multicolumn{2}{c}{(S\backslash NP)} \\
\multicolumn{2}{c}{\lambda g.\lambda x.state(x) \wedge g(x)} & \multicolumn{2}{c}{\lambda y.borders(y, texas)} \\
\multicolumn{4}{c}{\overline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}^{>}} \\
\multicolumn{4}{c}{S} \\
\multicolumn{4}{c}{\lambda x.state(x) \wedge borders(x, texas)}
\end{array}$$

Zettlemoyer and Collins (2005)

states	:=	$N : \lambda x.state(x)$
major	:=	$N/N : \lambda f.\lambda x.major(x) \wedge f(x)$
population	:=	$N : \lambda x.population(x)$
cities	:=	$N : \lambda x.city(x)$
rivers	:=	$N : \lambda x.river(x)$
run through	:=	$(S\backslash NP)/NP : \lambda x.\lambda y.traverse(y, x)$
the largest	:=	$NP/N : \lambda f.\arg\max(f, \lambda x.size(x))$
river	:=	$N : \lambda x.river(x)$
the highest	:=	$NP/N : \lambda f.\arg\max(f, \lambda x.elev(x))$
the longest	:=	$NP/N : \lambda f.\arg\max(f, \lambda x.len(x))$

Figure 6: Ten learned lexical items that had highest associated parameter values from a randomly chosen development run in the Geo880 domain.

Zettlemoyer and Collins (2005)

- PCCG learning
- Lexicon Λ, parameter vector θ
- GENLEX

Rules		Categories produced from logical form
Input Trigger	Output Category	$\arg\max(\lambda x.state(x) \wedge borders(x, texas), \lambda x.size(x))$
constant c	$NP : c$	$NP : texas$
arity one predicate p_1	$N : \lambda x.p_1(x)$	$N : \lambda x.state(x)$
arity one predicate p_1	$S\backslash NP : \lambda x.p_1(x)$	$S\backslash NP : \lambda x.state(x)$
arity two predicate p_2	$(S\backslash NP)/NP : \lambda x.\lambda y.p_2(y, x)$	$(S\backslash NP)/NP : \lambda x.\lambda y.borders(y, x)$
arity two predicate p_2	$(S\backslash NP)/NP : \lambda x.\lambda y.p_2(x, y)$	$(S\backslash NP)/NP : \lambda x.\lambda y.borders(x, y)$
arity one predicate p_1	$N/N : \lambda g.\lambda x.p_1(x) \wedge g(x)$	$N/N : \lambda g.\lambda x.state(x) \wedge g(x)$
literal with arity two predicate p_2 and constant second argument c	$N/N : \lambda g.\lambda x.p_2(x, c) \wedge g(x)$	$N/N : \lambda g.\lambda x.borders(x, texas) \wedge g(x)$
arity two predicate p_2	$(N\backslash N)/NP : \lambda x.\lambda g.\lambda y.p_2(x, y) \wedge g(x)$	$(N\backslash N)/NP : \lambda g.\lambda x.\lambda y.borders(x, y) \wedge g(x)$
an arg max / min with second argument arity one function f	$NP/N : \lambda g.\arg\max / \min(g, \lambda x.f(x))$	$NP/N : \lambda g.\arg\max(g, \lambda x.size(x))$
an arity one numeric-ranged function f	$S/NP : \lambda x.f(x)$	$S/NP : \lambda x.size(x)$

Figure 3: The rules that define GENLEX. We use the term *predicate* to refer to a function that returns a truth value; *function* to refer to all other functions; and *constant* to refer to constants of type e. Each row represents a rule. The first column lists the triggers that identify some sub-structure within a logical form L, and then generate a category. The second column lists the category that is created. The third column lists example categories that are created when the rule is applied to the logical form at the top of this column.

Dong and Lapata (2016)

Jobs This benchmark dataset contains 640 queries to a database of job listings. Specifically, questions are paired with Prolog-style queries. We used the same training-test split as Zettlemoyer and Collins (2005) which contains 500 training and 140 test instances. Values for the variables company, degree, language, platform, location, job area, and number are identified.

Geo This is a standard semantic parsing benchmark which contains 880 queries to a database of U.S. geography. Geo has 880 instances split into a training set of 680 training examples and 200 test examples (Zettlemoyer and Collins, 2005). We used the same meaning representation based on lambda-calculus as Kwiatkowski et al. (2011). Values for the variables city, state, country, river, and number are identified.

Atis This dataset has 5,410 queries to a flight booking system. The standard split has 4,480 training instances, 480 development instances, and 450 test instances. Sentences are paired with lambda-calculus expressions. Values for the variables date, time, city, aircraft code, airport, airline, and number are identified.

Dataset	Length	Example
Jobs	9.80	*what microsoft jobs do not require a bscs?*
	22.90	answer(company(J,'microsoft'),job(J),not((req_deg(J,'bscs'))))
Geo	7.60	*what is the population of the state with the largest area?*
	19.10	(population:i (argmax $0 (state:t $0) (area:i $0)))
Atis	11.10	*dallas to san francisco leaving after 4 in the afternoon please*
	28.10	(lambda $0 e (and (>(departure_time $0) 1600:ti) (from $0 dallas:ci) (to $0 san_francisco:ci)))
Ifttt	6.95	*Turn on heater when temperature drops below 58 degree*
	21.80	TRIGGER: Weather - Current_temperature_drops_below - ((Temperature (58)) (Degrees_in (f))) ACTION: WeMo_Insight_Switch - Turn_on - ((Which_switch? ("")))

Table 1: Examples of natural language descriptions and their meaning representations from four datasets. The average length of input and output sequences is shown in the second column.

Method	Accuracy
COCKTAIL (Tang and Mooney, 2001)	79.4
PRECISE (Popescu et al., 2003)	88.0
ZC05 (Zettlemoyer and Collins, 2005)	79.3
DCS+L (Liang et al., 2013)	90.7
TISP (Zhao and Huang, 2015)	85.0
Seq2Seq	87.1
− attention	77.9
− argument	70.7
Seq2Tree	90.0
− attention	83.6

Table 2: Evaluation results on Jobs.

Dong and Lapata (2016)

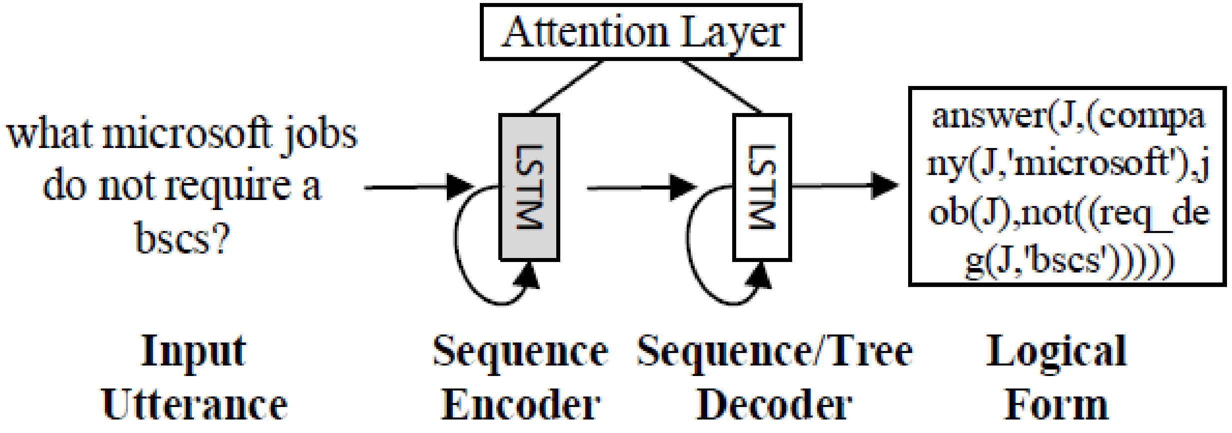


Figure 1: Input utterances and their logical forms are encoded and decoded with neural networks. An attention layer is used to learn soft alignments.

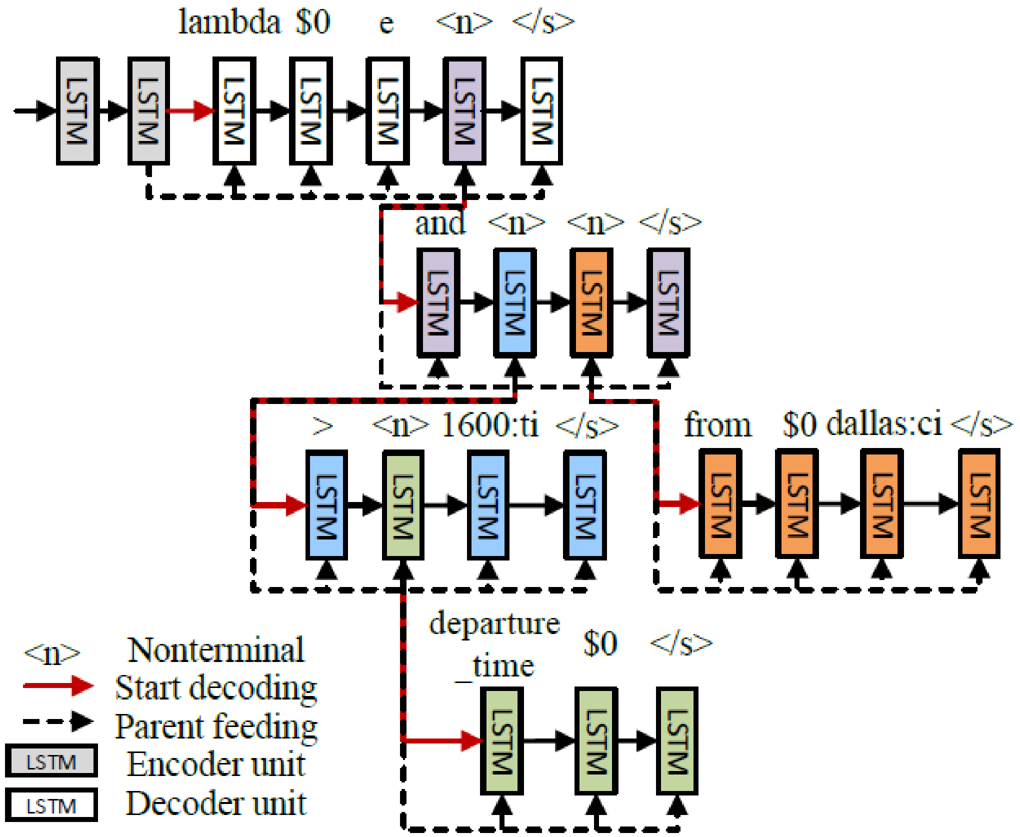
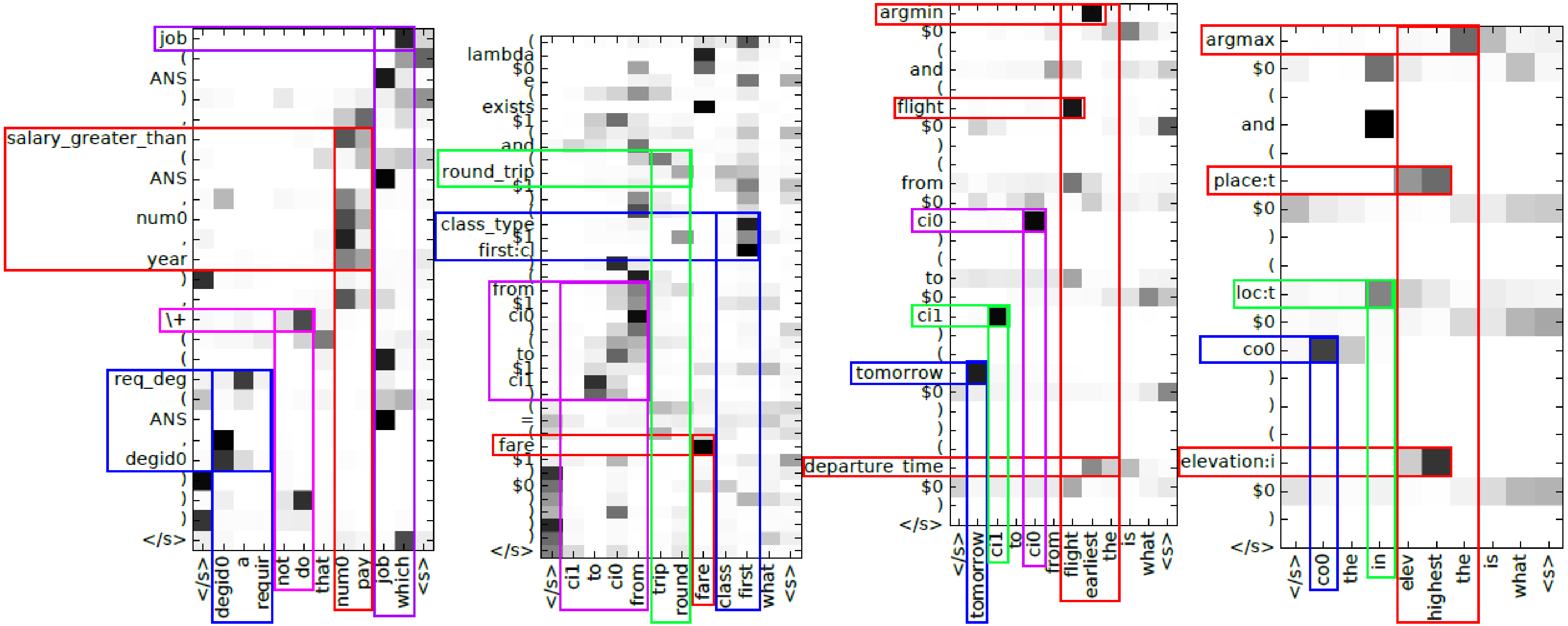


Figure 3: Sequence-to-tree (SEQ2TREE) model with a hierarchical tree decoder.

Dong and Lapata (2016)



(a) which jobs pay num0 that do not require a degid0

(b) what's first class fare round trip from ci0 to ci1

(c) what is the earliest flight from ci0 to ci1 tomorrow

(d) what is the highest elevation in the co0

Figure 6: Alignments (same color rectangles) produced by the attention mechanism (darker color represents higher attention score). Input sentences are reversed and stemmed. Model output is shown for SEQ2SEQ (a, b) and SEQ2TREE (c, d).

Dong and Lapata (2018)

Dataset	Length	Example
GEO	7.6	x : *which state has the most rivers running through it?*
	13.7	y : (argmax $0 (state:t $0) (count $1 (and (river:t $1) (loc:t $1 $0))))
	6.9	a : (argmax#1 state:t@1 (count#1 (and river:t@1 loc:t@2)))
ATIS	11.1	x : *all flights from dallas before 10am*
	21.1	y : (lambda $0 e (and (flight $0) (from $0 dallas:ci) (< (departure_time $0) 1000:ti)))
	9.2	a : (lambda#2 (and flight@1 from@2 (< departure_time@1 ?)))
DJANGO	14.4	x : *if length of bits is lesser than integer 3 or second element of bits is not equal to string 'as' ,*
	8.7	y : if len(bits) < 3 or bits[1] != 'as':
	8.0	a : if len (NAME) < NUMBER or NAME [NUMBER] != STRING :
WIKISQL	17.9	Table schema: ‖*Pianist*‖*Conductor*‖*Record Company*‖*Year of Recording*‖*Format*‖
	13.3	x : *What record company did conductor Mikhail Snitko record for after 1996?*
	13.0	y : `SELECT` *Record Company* `WHERE` (*Year of Recording* > *1996*) `AND` (*Conductor* = *Mikhail Snitko*)
	2.7	a : `WHERE` > `AND` =

Table 1: Examples of natural language expressions x, their meaning representations y, and meaning sketches a. The average number of tokens is shown in the second column.

Dong and Lapata 2018

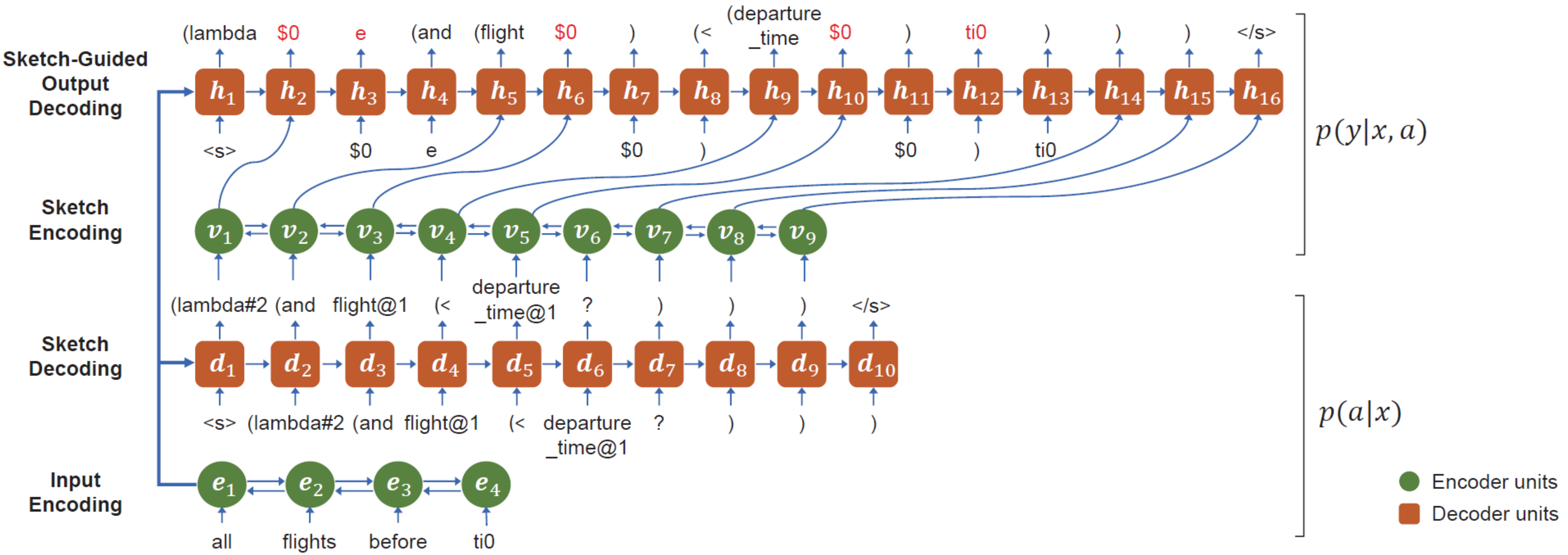


Figure 1: We first generate the meaning sketch a for natural language input x. Then, a fine meaning decoder fills in the missing details (shown in red) of meaning representation y. The coarse structure a is used to guide and constrain the output decoding.

Dong and Lapata 2018

Method	GEO	ATIS
ZC07 (Zettlemoyer and Collins, 2007)	86.1	84.6
UBL (Kwiatkowksi et al., 2010)	87.9	71.4
FUBL (Kwiatkowski et al., 2011)	88.6	82.8
GUSP++ (Poon, 2013)	—	83.5
KCAZ13 (Kwiatkowski et al., 2013)	89.0	—
DCS+L (Liang et al., 2013)	87.9	—
TISP (Zhao and Huang, 2015)	88.9	84.2
SEQ2SEQ (Dong and Lapata, 2016)	84.6	84.2
SEQ2TREE (Dong and Lapata, 2016)	87.1	84.6
ASN (Rabinovich et al., 2017)	85.7	85.3
ASN+SUPATT (Rabinovich et al., 2017)	87.1	85.9
ONESTAGE	85.0	85.3
COARSE2FINE	88.2	87.7
− sketch encoder	87.1	86.9
+ oracle sketch	93.9	95.1

Table 2: Accuracies on GEO and ATIS.

Introduction to NLP

Abstract Meaning Representation

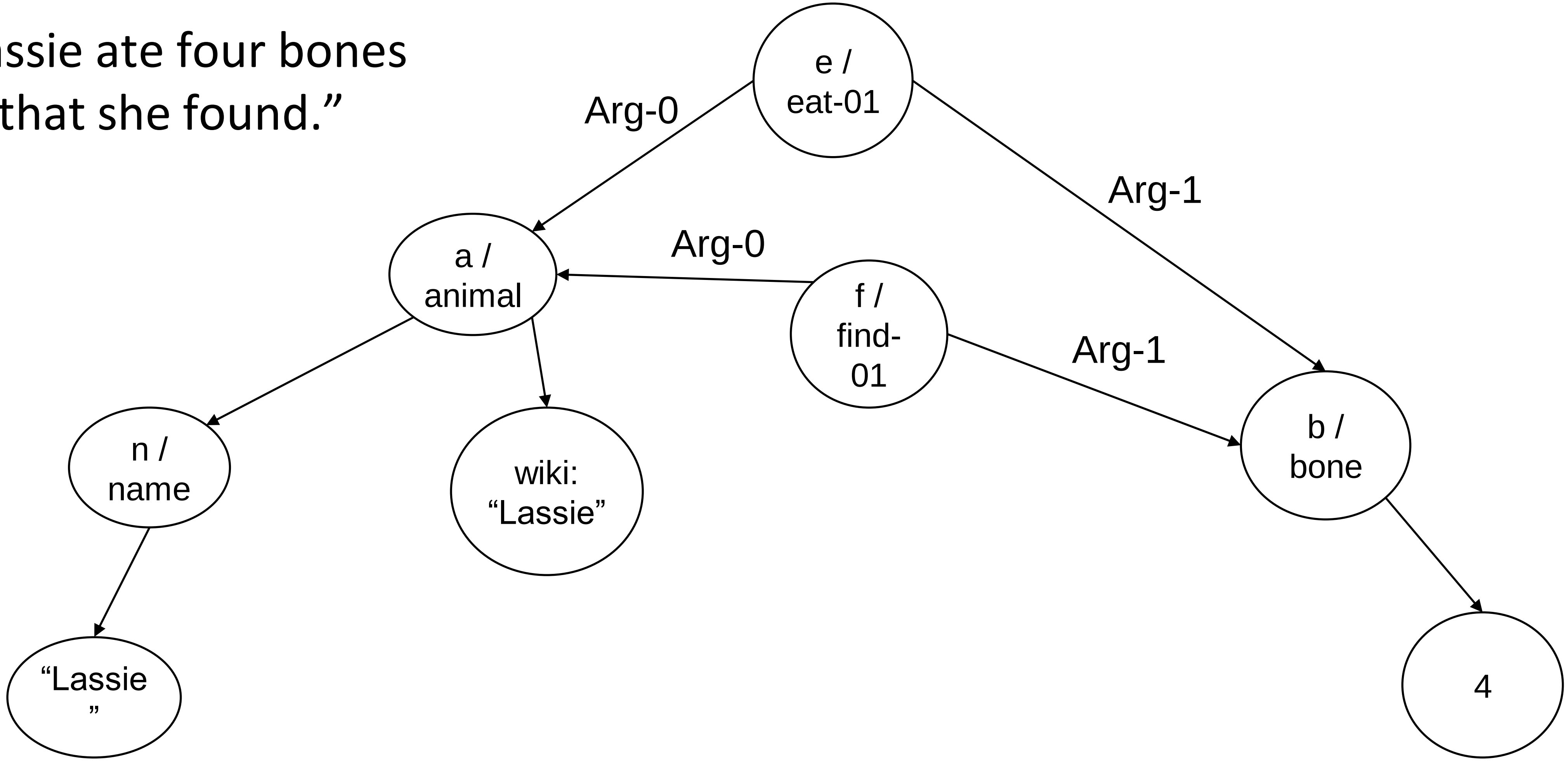
Abstract Meaning Representation (AMR)

- http://amr.isi.edu/
- Single structure that includes:
  - Predicate-Argument Structure
  - Named Entity Recognition
  - Coreference Resolution
  - Wikification

[slide from Jonathan Kummerfeld]

Example

“Lassie ate four bones that she found.”



[slide from Jonathan Kummerfeld]

Example

About 14,000 people fled their homes at the weekend after a local tsunami warning was issued, the UN said on its Web site

```
(s / say-01
  :ARG0 (g / organization
          :name (n / name
                  :op1 "UN"))
  :ARG1 (f / flee-01
          :ARG0 (p / person
                  :quant (a / about
                           :op1 14000))
          :ARG1 (h / home :poss p)
          :time (w / weekend)
          :time (a2 / after
                  :op1 (w2 / warn-01
                         :ARG1 (t / tsunami)
                         :location (l / local))))
  :medium (s2 / site
            :poss g
            :mod (w3 / web)))
```

Status of AMR

- AMR currently lacks
  - Multilingual consideration
  - Quantifier scope
  - Co-references across sentences
  - Grammatical number, tense, aspect, quotation marks
  - Many noun-noun or noun-adjective relations
  - Many detailed frames, e.g. Earthquake (with roles for magnitude, epicenter, casualties, etc)

[slide from Jonathan Kummerfeld]

AMR Parsing (Wang et al. 2015,16)

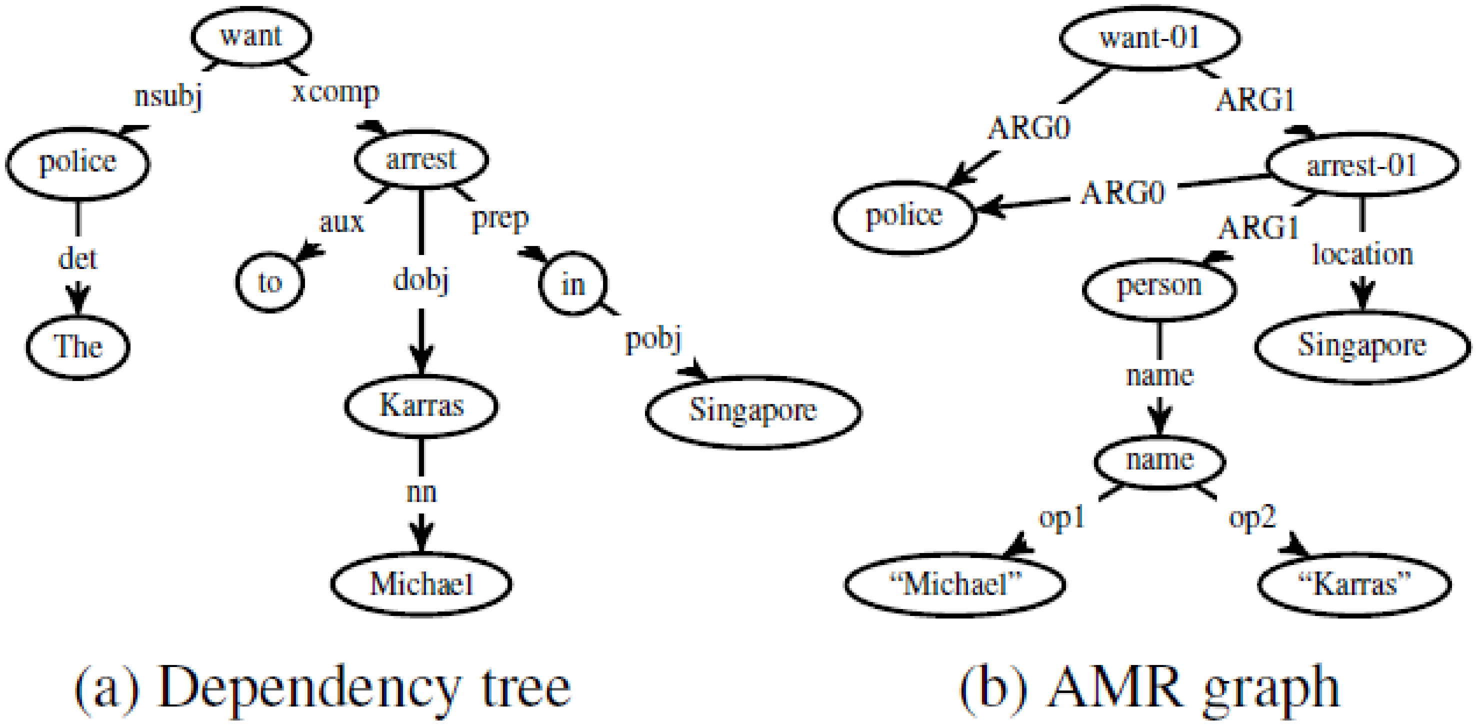


Figure 1: Dependency tree and AMR graph for the sentence, "The police want to arrest Micheal Karras in Singapore."

AMR Parsing (Wang et al. 2015,16)

Action	Current state $\Rightarrow$ Result state	Assign labels	Precondition
NEXT EDGE-l_r	$(\sigma_0\|\sigma', \beta_0\|\beta', G) \Rightarrow (\sigma_0\|\sigma', \beta', G')$	$\delta[(\sigma_0, \beta_0) \to l_r]$	β is not empty
SWAP-l_r	$(\sigma_0\|\sigma', \beta_0\|\beta', G) \Rightarrow (\sigma_0\|\beta_0\|\sigma', \beta', G')$	$\delta[(\beta_0, \sigma_0) \to l_r]$	
REATTACH$_k$-l_r	$(\sigma_0\|\sigma', \beta_0\|\beta', G) \Rightarrow (\sigma_0\|\sigma', \beta', G')$	$\delta[(k, \beta_0) \to l_r]$	
REPLACE HEAD	$(\sigma_0\|\sigma', \beta_0\|\beta', G) \Rightarrow (\beta_0\|\sigma', \beta = CH(\beta_0, G'), G')$	NONE	
REENTRANCE$_k$-l_r	$(\sigma_0\|\sigma', \beta_0\|\beta', G) \Rightarrow (\sigma_0\|\sigma', \beta_0\|\beta', G')$	$\delta[(k, \beta_0) \to l_r]$	
MERGE	$(\sigma_0\|\sigma', \beta_0\|\beta', G) \Rightarrow (\tilde{\sigma}\|\sigma', \beta', G')$	NONE	
NEXT NODE-l_c	$(\sigma_0\|\sigma_1\|\sigma', [], G) \Rightarrow (\sigma_1\|\sigma', \beta = CH(\sigma_1, G'), G')$	$\gamma[\sigma_0 \to l_c]$	β is empty
DELETE NODE	$(\sigma_0\|\sigma_1\|\sigma', [], G) \Rightarrow (\sigma_1\|\sigma', \beta = CH(\sigma_1, G'), G')$	NONE	

Table 1: Transitions designed in our parser. $CH(x, y)$ means getting all node x's children in graph y.

AMR Parsing (Wang et al. 2015,16)

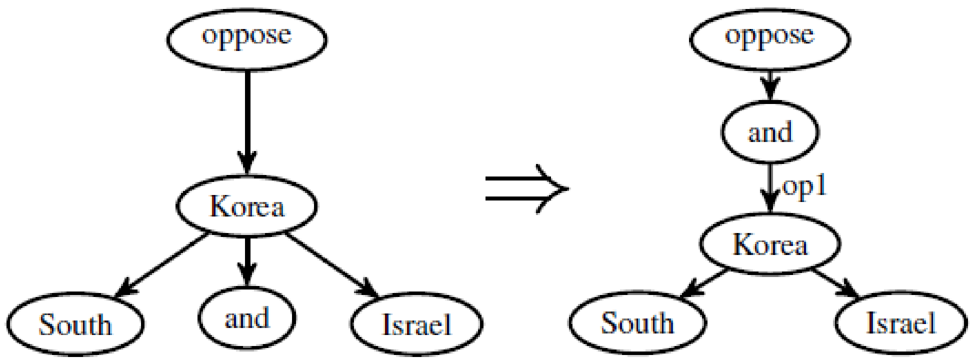


Figure 4: SWAP action

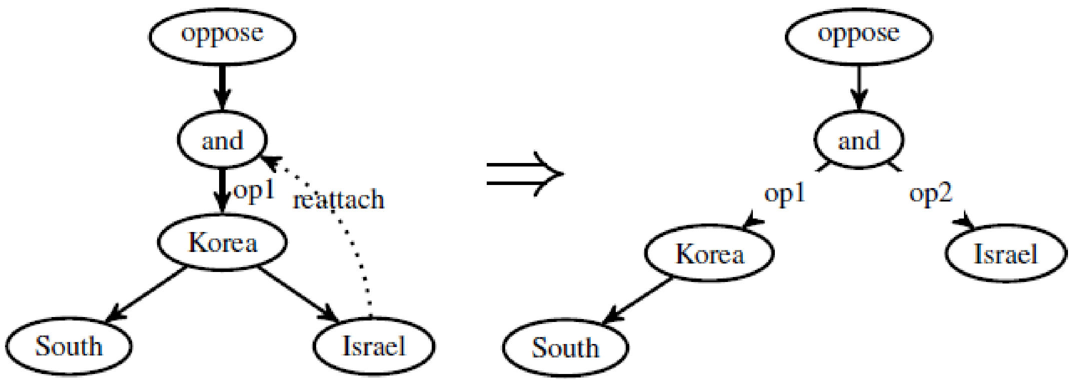


Figure 5: REATTACH action

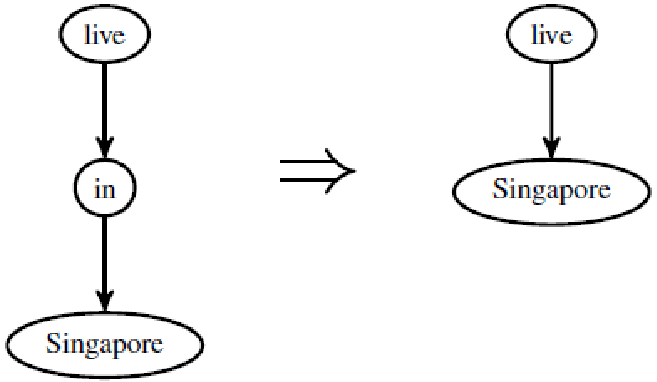


Figure 6: REPLACE-HEAD action

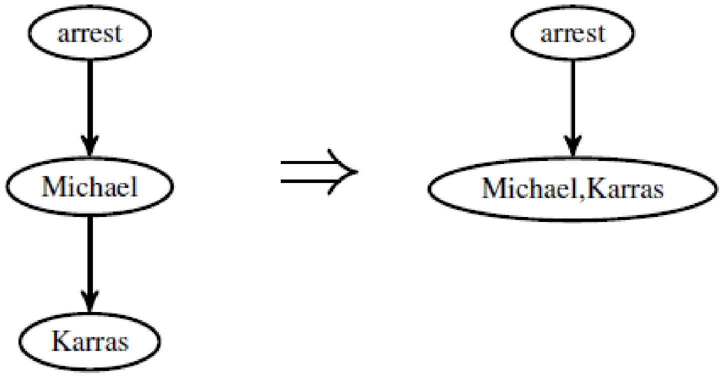
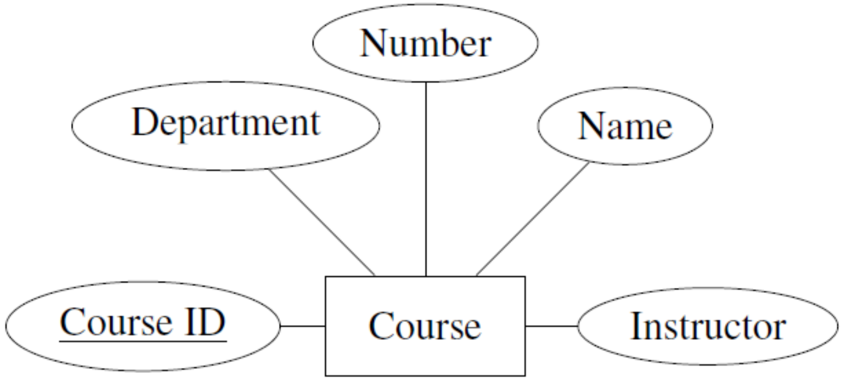


Figure 8: MERGE action

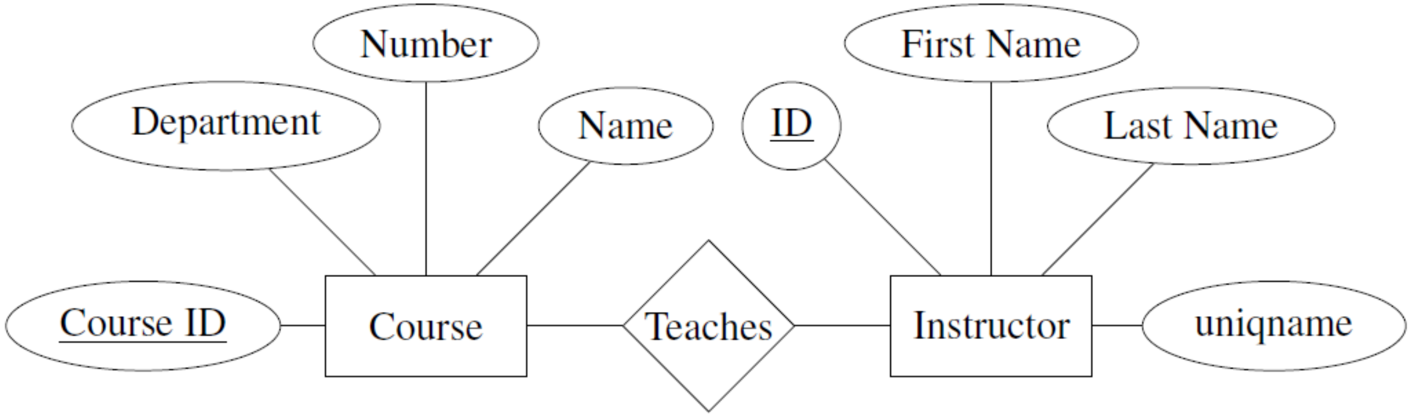
Introduction to NLP

Natural Language to SQL

NL to SQL



(a) Single Table Schema

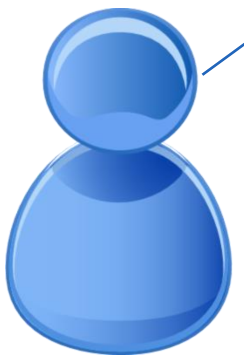


(b) Multi-Table Schema

Figure 1: Two possible schemas for a database that could answer, "Who teaches Discrete Mathematics?"

[Finegan-Dollak et al]

Example: Text-to-SQL



“Who teaches NLP?”

```
SELECT I.NAME
FROM INSTRUCTOR AS I,
OFFERING_INSTRUCTOR AS OI,
COURSE_OFFERING AS O, SEMESTER AS S,
COURSE AS C
WHERE OI.INSTRUCTOR_ID=I.INSTRUCTOR_ID
AND O.OFFERING_ID=OI.OFFERING_ID
AND O.SEMESTER=S.SEMESTER_ID
AND O.COURSE_ID=C.COURSE_ID
AND C.NAME=”NLP"
AND S.YEAR=2016 AND S.SEMESTER="FA"
```

All About SQL

Course

Course ID	Dept.	Number	Name	Credits
1	EECS	203	Discrete Math	4
2	LING	137	Epic Grammar Fails	3
3	EECS	595	NLP	4

Instructor

Instructor ID	First Name	Last Name
1	Dragomir	Radev
2	Walter	Lasecki
3	Ezra	Keshet
4	Rada	Mihalcea

Course Offering

Course ID	Instructor ID	Year	Semester
3	1	2016	Fall
3	4	2017	Fall
2	3	2018	Winter

```
SELECT C.CREDITS
FROM COURSE AS C
WHERE C.NAME = “NLP”;
```

How many credits is NLP?

All About SQL

Course

Course ID	Dept.	Number	Name	Credits
1	EECS	203	Discrete Math	4
2	LING	137	Epic Grammar Fails	3
3	EECS	595	NLP	4

Instructor

Instructor ID	First Name	Last Name
1	Dragomir	Radev
2	Walter	Lasecki
3	Ezra	Keshet
4	Rada	Mihalcea

Course Offering

Course ID	Instructor ID	Year	Semester
3	1	2016	Fall
3	4	2017	Fall
2	3	2018	Winter

Who teaches NLP?

```
SELECT I.FIRST_NAME, I.LAST_NAME
FROM INSTRUCTOR AS I,
   COURSE AS C,
   COURSE_OFFERING AS CO
WHERE C.NAME = "NLP"
AND C.COURSE_ID = CO.COURSE_ID
AND CO.INSTRUCTOR_ID =
   I.INSTRUCTOR_ID;
```

All About SQL

Course

Course ID	Dept.	Number	Name	Credits
1	EECS	203	Discrete Math	4
2	LING	137	Epic Grammar Fails	3
3	EECS	595	NLP	4

Instructor

Instructor ID	First Name	Last Name
1	Dragomir	Radev
2	Walter	Lasecki
3	Ezra	Keshet
4	Rada	Mihalcea

Course Offering

Course ID	Instructor ID	Year	Semester
3	1	2016	Fall
3	4	2017	Fall
2	3	2018	Winter

What course is worth the most credits?

```
SELECT C1.NAME
FROM COURSE AS C1
WHERE C1.CREDITS =
    (SELECT MAX C2.CREDITS
    FROM COURSE AS C2);
```

More Complicated SQL

Which countries in Europe have at least 3 car manufacturers?

```
SELECT T1.country_name
FROM countries AS T1 JOIN continents
AS T2 ON T1.continent = T2.cont_id
JOIN car_makers AS T3 ON
T1.country_id = T3.country
WHERE T2.continent = 'Europe'
GROUP BY T1.country_name
HAVING COUNT(*) >= 3
```

What is the average life expectancy in the countries where English is not the official language?

```
SELECT AVG(life_expectancy)
FROM country
WHERE name NOT IN
    (SELECT T1.name
     FROM country AS T1 JOIN
     country_language AS T2
     ON T1.code = T2.country_code
     WHERE T2.language = "English"
       AND T2.is_official = "T")
```

Seq2SQL datasets are scarce

- Compared to other large datasets such as ImageNet for object recognition, building a decent seq2SQL dataset is even more time-consuming
- Hard to find many databases with multiple tables online
- Annotation requires very specific knowledge in databases

Traditional Seq2SQL datasets

Traditional 9 seq2SQL datasets: ATIS, Geo, Scholar, etc. + Advising

- Pros
  - SQL queries cover complex SQL structures and components
- Cons
  - The number of labeled queries is small (< 500)
    - Paraphrase about 4-10 natural language questions for each SQL query.
    - The total # of question-SQL pairs: ~500 -> ~5,000
  - Each of datasets contains SQL queries only to a single database

GEO Query

which state has the most rivers running through it?

```
argmax $0
 (state:t $0)
 (count $1 (and
            (river:t $1)
            (loc:t $1 $0)))
```

Lambda Calculus Logical Form

ATIS

Show me flights from Pittsburgh to Seattle

```
lambda $0 e
  (and (flight $0)
  (from $0 pittsburgh:ci)
  (to $0 seattle:ci))
```

Lambda Calculus Logical Form

JOBS

what microsoft jobs do not require a bscs?

```
answer(
  company(J,'microsoft'),
  job(J),
  not((req deg(J,'bscs'))))
```

Prolog-style Program

Credit to Pengcheng Yin

ATIS (Price, 1990; Dahl et al., 1994) User questions for a flight-booking task, manually annotated. We use the modified SQL from Iyer et al. (2017), which follows the data split from the logical form version (Zettlemoyer and Collins, 2007).

GeoQuery (Zelle and Mooney, 1996) User questions about US geography, manually annotated with Prolog. We use the SQL version (Popescu et al., 2003; Giordani and Moschitti, 2012; Iyer et al., 2017), which follows the logical form data split (Zettlemoyer and Collins, 2005).

Restaurants (Tang and Mooney, 2000; Popescu et al., 2003) User questions about restaurants, their food types, and locations.

Scholar (Iyer et al., 2017) User questions about academic publications, with automatically generated SQL that was checked by asking the user if the output was correct.

Academic (Li and Jagadish, 2014) Questions about the Microsoft Academic Search (MAS) database, derived by enumerating every logical query that could be expressed using the search page of the MAS website and writing sentences to match them. The domain is similar to that of Scholar, but their schemas differ.

Yelp and IMDB (Yaghmazadeh et al., 2017) Questions about the Yelp website and the Internet Movie Database, collected from colleagues of the authors who knew the type of information in each database, but not their schemas.

WikiSQL (Zhong et al., 2017) A large collection of automatically generated questions about individual tables from Wikipedia, paraphrased by crowd workers to be fluent English.

Advising (This Work) Our dataset of questions over a database of course information at the University of Michigan, but with fictional student records. Some questions were collected from the EECS department Facebook page and others were written by CS students with knowledge of the database who were instructed to write questions they might ask in an academic advising appointment.

Dialog2SQL Data Creation

Our Complex and Cross-Domain Text-to-SQL Dataset: Spider

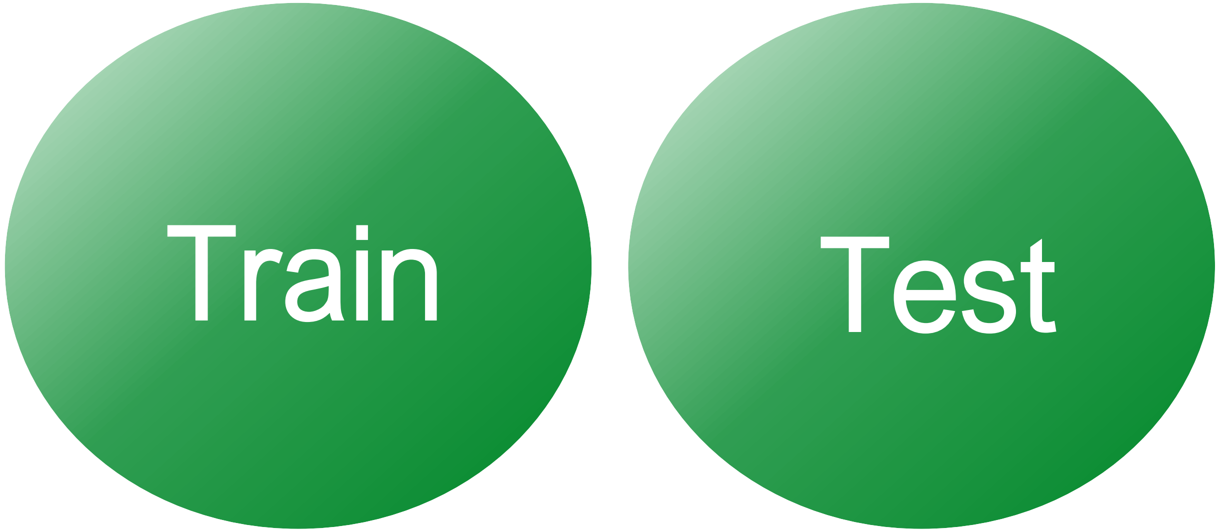
Dataset	# Q	# SQL	# DB	# Table /DB	ORDER BY	GROUP BY	NESTED	HAVING	LIMIT
ATIS	5,280	947	1	32	0	5	315	0	0
GeoQuery	877	247	1	6	20	46	167	9	20
Scholar	817	193	1	7	75	100	7	20	1
Academic	196	185	1	15	23	40	7	18	23
IMDB	131	89	1	16	10	6	1	0	10
Yelp	128	110	1	7	18	21	0	4	18
Advising	3,898	208	1	10	15	9	22	0	11
Restaurants	378	378	1	3	0	0	4	0	0
WikiSQL	80,654	77,840	26,521	1	0	0	0	0	0
Spider (original)	10,181	5,693	200	5.1	1335	1491	844	388	903

Figure: Comparisons of text-to-SQL datasets

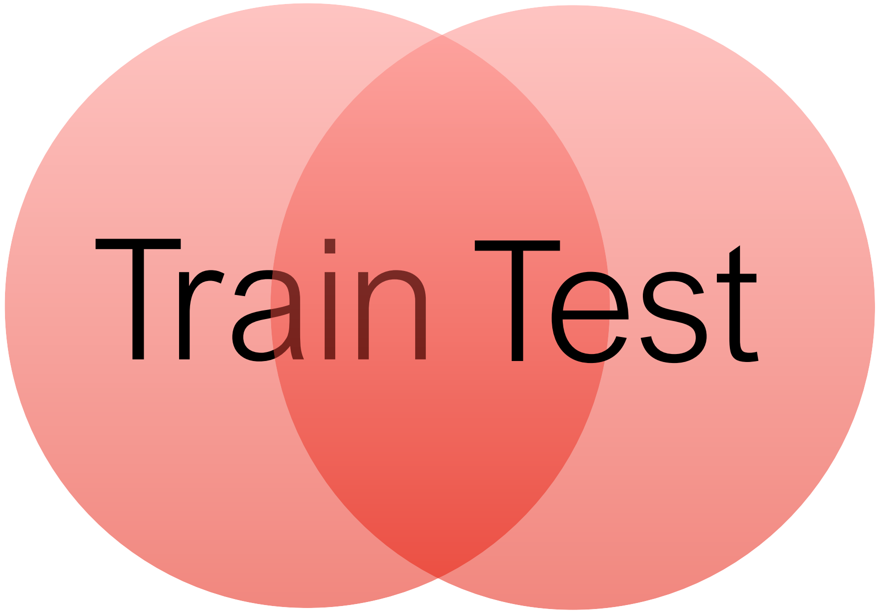
Standard Practice in ML

- Split dataset into train set, test set, optional development (dev) set.
- No training example can also appear in test set.

Good Split



Bad Split



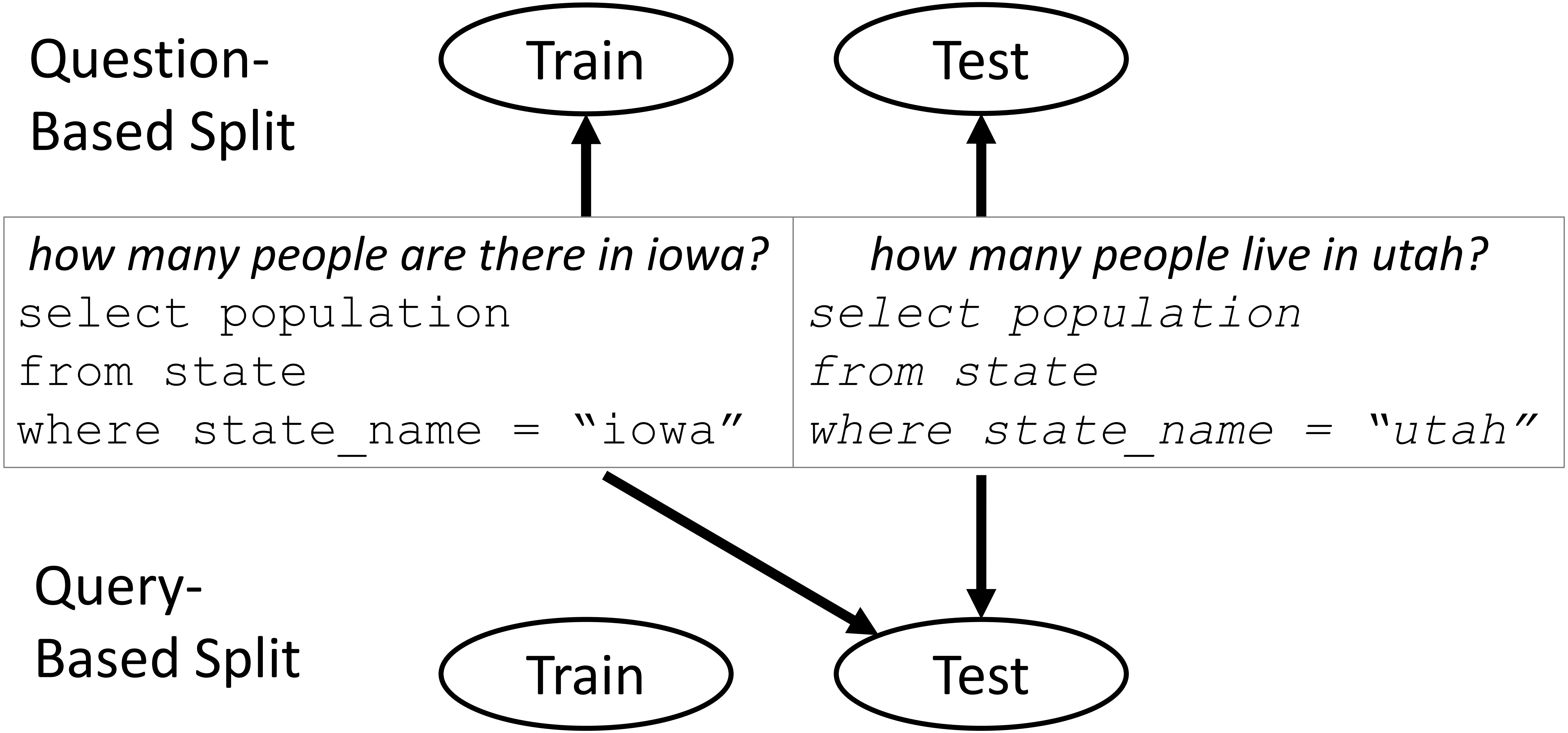
How Do We Define an Example?

how many people are there in iowa?

```
select population
from state
where state_name = “iowa”
```

how many people live in utah?

```
select population
from state
where state_name = “utah”
```



Question-
Based Split
Train
Test
how many people are there in iowa?
select population
from state
where state_name = "iowa"
how many people live in utah?
select population
from state
where state_name = "utah"
Query-
Based Split
Train
Test

Seq2SQL data – WikiSQL (Salesforce)

- The first realistic seq2SQL task definition on the top of WikiSQL makes it the most popular seq2SQL dataset
- Databases in the test set do not appear in the train/dev set, which requires model to generalize to new databases
- https://github.com/salesforce/WikiSQL

WikiSQL Animated GIF

https://beta.techcrunch.com/wp-content/uploads/2017/08/unnamed2.gif

WikiSQL Example

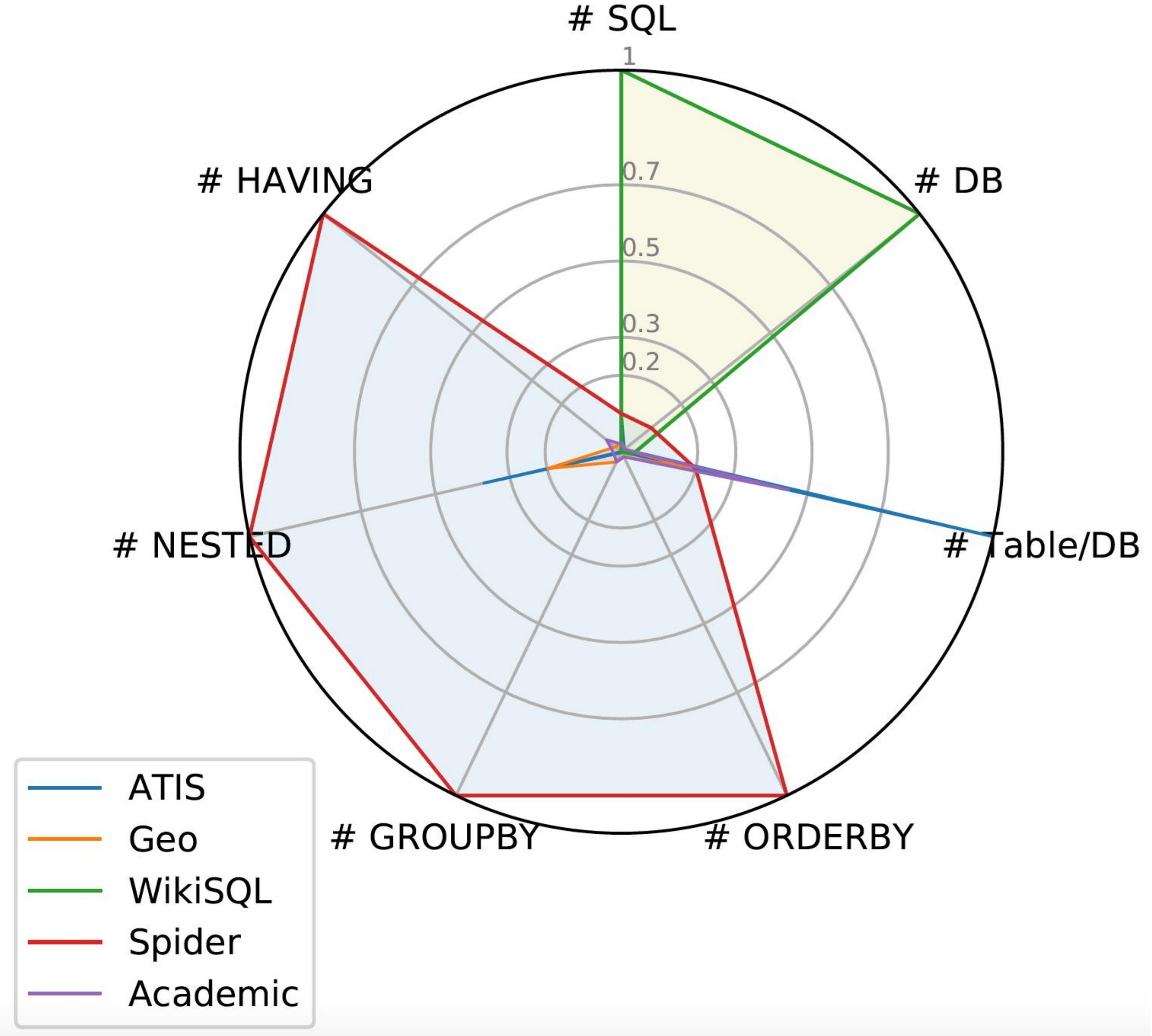
- https://github.com/salesforce/WikiSQL

Seq2SQL data – WikiSQL (Salesforce paper)

- WikiSQL - Pros
  - The number of SQL queries and databases is huge (>20,000)
  - Databases in the test set do not appear in the train/dev set, which requires model to generalize to new databases
- WikiSQL - Cons
  - SQL queries are generated by templates and paraphrased by Turkers
  - All databases have only one table - not a full relational database
  - SQL only contains SELECT and WHERE. No GROUP BY/Nested queries etc.

Seq2SQL data - Spider

- WikiSQL is great. But it has limited SQL coverage and a very simple schema, which makes the task simple and less interesting



[Yu et al. 2018]

Seq2SQL data - Yale Spider

- SQL labels cover almost all important SQL components
- Each database has multiple tables and several foreign keys
- It is currently the only large-scale complex and cross-domain semantic parsing and text-to-SQL dataset!
- Check it out!!!
- Our Blog
  - Project Page: https://yale-lily.github.io/spider
  - Github Page: https://github.com/taoyds/spider

Query Difficulty

Easy

What is the number of cars with more than 4 cylinders'

```
SELECT COUNT(*)
FROM cars_data
WHERE cylinders > 4
```

Meidum

For each stadium, how many concerts are there?

```
SELECT T2.name, COUNT(*)
FROM concert AS T1 JOIN stadium AS T2
ON T1.stadium_id = T2.stadium_id
GROUP BY T1.stadium_id
```

Hard

Which countries in Europe have at least 3 car manufacturers?

```
SELECT T1.country_name
FROM countries AS T1 JOIN continents
AS T2 ON T1.continent = T2.cont_id
JOIN car_makers AS T3 ON
T1.country_id = T3.country
WHERE T2.continent = 'Europe'
GROUP BY T1.country_name
HAVING COUNT(*) >= 3
```

Extra Hard

What is the average life expectancy in the countries where English is not the official language?

```
SELECT AVG(life_expectancy)
FROM country
WHERE name NOT IN
   (SELECT T1.name
    FROM country AS T1 JOIN
    country_language AS T2
    ON T1.code = T2.country_code
    WHERE T2.language = "English"
      AND T2.is_official = "T")
```

NLP