COMP 786 (Fall 2020)
Natural Language Processing

Week 6: Semantic Parsing (Semantic Role Labeling, Lambda-Calculus, CCG, DCS, etc.)

Mohit Bansal

(various slides adapted/borrowed from courses by Dan Klein, Artzi-FitzGerald-Zettlemoyer CCG tutorial, JurafskyMartin-SLP3, others)
Announcements

- Coding-HW1 (on word vector training+evaluation + visualization) has been released (and details emailed) last week – due Sep23 midnight! Stay after class today to get more info/demo.

- Midterm project presentation coming up in 2 weeks (look out for details in email).
Semantic Role Labeling
Semantic Role Labeling (SRL)

Who did what to whom at where?

The police officer detained the suspect at the scene of the crime

Agent Predicate Theme Location

[SLP3 book]
Role-based relations for the different clauses in the sentence:

\[ \text{Judge } \text{She } \text{blames } \text{Evaluate } \text{the Government } \text{Reason } \text{for failing to do enough to help} \].

Holman would characterise this as \text{blaming } \text{Evaluate } \text{the poor}.

The letter quotes Black as saying that \[ \text{Judge } \text{white and Navajo ranchers } \text{misrepresent their livestock losses and blame } \text{Reason } \text{everything } \text{Evaluate } \text{on coyotes} \].

More semantic relations (e.g., agent, reason, message) than just subject/object style syntactic roles

Typical traditional pipelines involves POS-tagging and parsing, and then features extracted on those (plus NER, etc.)...but then several errors caused by wrong parse!
Semantic Role Labeling (SRL)

PropBank / FrameNet

FrameNet: roles shared between verbs
PropBank: each verb has its own roles
PropBank more used, because it’s layered over the t

Note: some linguistic theories postulate even fewer

roles than FrameNet (e.g. 5-20 total: agent, patient, instrume
PropBank vs. FrameNet

- PropBank has each verb get its own roles, whereas FrameNet shares roles between verbs (e.g., argue and banter in figure below)

- PropBank more convenient w.r.t. being layered over Treebank parses (and hence more coverage)
PropBank Roles

- Based on Dowty, 1991: roles are verb-sense specific in PropBank (role definitions depend on specific verb and relation to other roles)

- Each verb sense has numbered arguments e.g., ARG-0, ARG-1, etc.
  - ARG-0 is usually PROTO-AGENT
  - ARG-1 is usually PROTO-PATIENT
  - ARG-2 is usually benefactive, instrument, attribute
  - ARG-3 is usually start point, benefactive, instrument, attribute
  - ARG-4 is usually end point (e.g., for move or push style verbs)

(ARG-2,3,4 onwards not very consistent and highly depend on specific verb and its sense in the sentence, hence labeling of PropBank is tricky)
Sales fell to $251.2 million from $278.7 million.

arg1: Sales
rel: fell
arg4: to $251.2 million
arg3: from $278.7 million
rotate.02 sense: shift from one thing to another
roles: Arg0: causer of shift
Arg1: thing being changed
Arg2: old thing
Arg3: new thing

Many of Wednesday’s winners were losers yesterday as investors quickly took profits and rotated their buying to other issues, traders said.

arg0: investors
rel: rotated
arg1: their buying
arg3: to other issues
PropBank Example 3

aim.01  
sense: intend, plan
roles:  Arg0: aimer, planner
       Arg1: plan, intent

The Central Council of Church Bell Ringers aims *trace* to improve relations with vicars.  
(wsj_0089)
arg0:  The Central Council of Church Bell Ringers
rel:  aims
arg1:  *trace* to improve relations with vicars

aim.02  
sense: point (weapon) at
roles:  Arg0: aimer
       Arg1: weapon, etc.
       Arg2: target

Banks have been aiming packages at the elderly.
arg0:  Banks
rel:  aiming
arg1:  packages
arg2:  at the elderly
Shared Arguments

(NP-SBJ (JJ massive) (JJ internal) (NN debt) )
  (VP (VBZ has)
    (VP (VBN forced)
      (S
        (NP-SBJ-1 (DT the) (NN government) )
        (VP
          (VP (TO to)
            (VP (VB borrow)
              (ADVP-MNR (RB massively) )))...
PropBank Parsing

PropBanking a Sentence

A sample parse tree

Analysts have been expecting a GM-Jaguar pact that would give the U.S. car maker an eventual 30% stake in the British company.

(S (NP-SBJ Analysts) (VP have (VP been (VP expecting (NP (NP a GM-Jaguar pact) (S (NP-SBJ *T*-1) (VP would (VP give (NP the U.S. car maker) (NP (NP an eventual (ADJP 30 %) stake) (PP-LOC in (NP the British company)))))))))))))

Martha Palmer 2013
The same parse tree PropBanked

have been expecting

S
Arg0 (NP-SBJ Analysts)
(VP have
(VP been
(VP expecting
Arg1 (NP (NP a GM-Jaguar pact)
(SBAR (WHNP-1 that)
(S Arg0 (NP-SBJ *T*-1)
(VP would
(VP give
Arg2 (NP the U.S. car maker)
Arg1 (NP (NP an eventual (ADJP 30 %) stake)
(PP-LOC in (NP the British
company)

that would give

Arg0

*T*-1

Arg2

an eventual 30% stake in the
British company

expect(Analysts, GM-J pact)
give(GM-J pact, US car maker, 30% stake)
Data Size

Annotated PropBank Data

- Penn English TreeBank, OntoNotes 5.0.
  - Total ~2 million words
- Penn Chinese TreeBank
- Hindi/Urdu PropBank
- Arabic PropBank

2013 Verb Frames Coverage
Count of word sense (lexical units)

<table>
<thead>
<tr>
<th>Language</th>
<th>Final Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>10,615*</td>
</tr>
<tr>
<td>Chinese</td>
<td>24,642</td>
</tr>
<tr>
<td>Arabic</td>
<td>7,015</td>
</tr>
</tbody>
</table>

From Martha Palmer 2013 Tutorial
Recall that the difference between these two models of semantic roles is that FrameNet (22.27) employs many frame-specific frame elements as roles, while PropBank (22.28) uses a smaller number of numbered argument labels that can be interpreted as verb-specific labels, along with the more general ARGM labels. Some examples:

(22.27) [You] can’t [blame] [the program] [for being unable to identify it]

(22.28) [The San Francisco Examiner] issued [a special edition] [yesterday]

A simplified semantic role labeling algorithm is sketched in Fig. 22.4. While there are a large number of algorithms, many of them use some version of the steps in this algorithm. Most algorithms, beginning with the very earliest semantic role analyzers (Simmons, 1973), begin by parsing, using broad-coverage parsers to assign a parse to the input string. Figure 22.5 shows a parse of (22.28) above. The parse is then traversed to find all words that are predicates. For each of these predicates, the algorithm examines each node in the parse tree and decides the semantic role (if any) it plays for this predicate. This is generally done by supervised classification. Given a labeled training set such as PropBank or FrameNet, a feature vector is extracted for each node, using feature templates described in the next subsection. A 1-of-N classifier is then trained to predict a semantic role for each constituent given these features, where N is the number of potential semantic roles plus an extra NONE role for non-role constituents. Most standard classification algorithms have been used (logistic regression, SVM, etc). Finally, for each test sentence to be labeled, the classifier is run on each relevant constituent. We give more details of the algorithm after we discuss features.

```plaintext
function SEMANTICROLELABEL(words) returns labeled tree

parse ← PARSE(words)
for each predicate in parse do
    for each node in parse do
        featurevector ← EXTRACTFEATURES(node, predicate, parse)
        CLASSIFYNODE(node, featurevector, parse)
```

Features for Semantic Role Labeling

A wide variety of features can be used for semantic role labeling. Most systems use some generalization of the core set of features introduced by Gildea and Jurafsky (2000). A typical set of basic features are based on the following feature templates (demonstrated on the NP-SBJ constituent The San Francisco Examiner in Fig. 22.5):

- The governing predicate, in this case the verb issued. The predicate is a critical feature since labels are defined only with respect to a particular predicate.

- The phrase type of the constituent, in this case, NP (or NP-SBJ). Some semantic roles tend to appear as NPs, others as Ss or PPs, and so on.
SRL Features

Features

Headword of constituent
Examiner

Headword POS
NNP

Voice of the clause
Active

Subcategorization of pred
VP -> VBD NP PP

Named Entity type of constit
ORGANIZATION

First and last words of constit
The, Examiner

Linear position, clause re: predicate
before
Path-based Features for SRL

Path Features
- Path from target to filler
- Filler's syntactic type, headword, case
- Target's identity
- Sentence voice, etc.
- Lots of other second-order features

Results
- Gold vs parsed source trees
- SRL is fairly easy on gold trees
- Harder on automatic parses

<table>
<thead>
<tr>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VB↑VP↓PP</td>
<td>PP argument/adjunct</td>
</tr>
<tr>
<td>VB↑VP↑S↓NP</td>
<td>subject</td>
</tr>
<tr>
<td>VB↑VP↓NP</td>
<td>object</td>
</tr>
<tr>
<td>VB↑VP↑VP↑S↓NP</td>
<td>subject (embedded VP)</td>
</tr>
<tr>
<td>VB↑VP↓ADVP</td>
<td>adverbial adjunct</td>
</tr>
<tr>
<td>NN↑NP↑NP↓PP</td>
<td>prepositional complement of noun</td>
</tr>
</tbody>
</table>
Some SRL Results

- So major feature categories in traditional feature-based SRL models were:
  - Headword, syntactic type, case, etc. of candidate node/constituent
  - Linear and tree path from predicate target to node
  - Active vs. passive voice
  - Second order and higher order features

- Accuracy for such feature-based SRL models then highly depends on accuracy of underlying parse tree!
  - So quite high SRL results when using ground-truth parses
  - Much lower results with automatically-predicted parses!
A bi-LSTM approach to semantic role labeling. Most actual networks are much deeper than shown in this figure; 3 to 4 bi-LSTM layers (6 to 8 total LSTMs) are common. The input is a concatenation of an embedding for the input word and an embedding of a binary variable which is 1 for the predicate to 0 for all other words. After He et al. (2017).
“The police are thoroughly investigating the cause of the accident.”
Semantic Roles Beyond Verbs

Not just verbs: NomBank

Meyers et al. 2004

Figure from Jiang and Ng 2006
Semantic Roles Beyond Verbs

Additional Issues for nouns

- **Features:**
  - Nominalization lexicon (employment → employ)
  - Morphological stem
    - Healthcare, Medicate → care

- **Different positions**
  - Most arguments of nominal predicates occur inside the NP
  - Others are introduced by support verbs
  - Especially light verbs “X made an argument”, “Y took a nap”
Austria, once expected to waltz smoothly into the European Union, is elbowing its partners treading on toes and pogo-dancing in a most un-Viennese manner.
PropBank vs. FrameNet Representations

(a) A phrase-structure tree taken from the Penn Treebank and annotated with PropBank predicate-argument structures. The verbs *created* and *pushed* serve as predicates in this sentence. Dotted arrows connect each predicate to its semantic arguments (bracketed phrases).

(b) A partial depiction of frame-semantic structures for the same sentence. The words in bold are targets, which instantiate a (lemmatized and part-of-speech–tagged) lexical unit and evoke as an argument. Every frame annotation is enclosed in a distinct shape, and its argument labels are shown together on the same vertical tier below the sentence. See text for explanation of abbreviations.

Phrase-structure syntax trees from the Wall Street Journal section of the Penn Treebank (Marcus, Marcinkiewicz, and Santorini 1993) annotated with predicate-argument structures for verbs. In Figure 2(a), the syntax tree for the sentence is marked with various semantic roles. The two main verbs in the sentence, *created* and *pushed*, are the predicates. For the former, the constituent *more than 1.2 million jobs* serves as the semantic role ARG1 and the constituent *In that time* serves as the role ARGM-TMP. Similarly for the latter verb, roles ARG1, ARG2, ARGM-DIR, and ARGM-TMP are shown in the figure.

PropBank defines core roles ARG0 through ARG5, which received different interpretations for different predicates. Additional modifier roles ARGM-* include ARGM-TMP (temporal) and ARGM-DIR (directional), as shown in Figure 2(a). The PropBank representation therefore has a small number of roles, and the training data set comprises some 40,000 sentences, thus making the semantic role labeling task an attractive one from the perspective of machine learning.

There are many instances of influential work on semantic role labeling using PropBank conventions. Pradhan et al. (2004) present a system that uses support vector machines (SVMs) to identify the arguments in a syntax tree that can serve as semantic roles, followed by classification of the identified arguments to role names via a collection of binary SVMs. Punyakanok et al. (2004) describe a semantic role labeler that uses integer linear programming for inference and uses several global constraints to find the best.

In that time more than 1.2 million jobs have been *created* and the official jobless rate has been *pushed* below 17% from 21%.
Compositional Semantics
Compositional Semantics I: Logic form

- Logic-form based (lambda calculus), Semantic Parsing

- Useful for Q&A, IE, grounding, comprehension tasks (summarization, reading tasks)

- A lot of focus has been on KB-based Question Answering in this direction (where input-output training data is question-answer pairs, and latent intermediate representation is the question’s semantic parse, which is ‘executed’ against the KB to get the answer)
Question Answering

- Initial approaches to Q&A: pattern matching, pattern learning, query rewriting, information extraction
- Next came a large-scale, open-domain IE system like IBM Watson
Deep Q&A: Semantic Parsing

Complex, free-form, multi-clause questions

The Black Sea is an inland sea located between far-southeastern Europe and the far-western edges of the continent of Asia and the country of Turkey. It’s bordered by Turkey, and by the countries of Bulgaria, Romania, Ukraine, Russia and Georgia.

Black Sea - World Atlas
www.worldatlas.com/aatlas/infopage/blacksea.htm
Deep Q&A: Semantic Parsing

- Complex, free-form, multi-clause questions

Google search results for: what are the capitals of the countries that border the baltic sea

**Important cities along the Baltic include:**

- The Russian cities of St. Petersburg and Kaliningrad.
- Stockholm, capital of Sweden.
- Copenhagen, capital of Denmark.
- Oslo, capital of Norway.
- Helsinki, capital of Finland.
- Tallinn, capital of Estonia.
- Riga, capital of Latvia.

**Baltic Sea - Encyclopedia of Earth**

[www.eoearth.org/article/Baltic_Sea](http://www.eoearth.org/article/Baltic_Sea)
Semantic Parsing: Logic forms

Parsing with logic (booleans, individuals, functions) and lambda forms

Sentence
\( \text{loves}(\text{john}, \text{mary}) \)

Noun Phrase
\( \text{john} \)

Verb Phrase
\( \lambda x. \text{loves}(x, \text{mary}) \)

Name
\( \text{john} \)

Verb
\( \lambda y. \lambda x. \text{loves}(x, y) \)

Noun Phrase
\( \text{mary} \)

“John”
\( \text{john} \)

“loves”
\( \lambda y. \lambda x. \text{loves}(x, y) \)

“Mary”
\( \text{mary} \)

Truth-Conditional Semantics

- Examples like “Bob sings”
- Logical translation of this will be something like: \( \text{sings(bob)} \)
- Types on these translations are entities (e) and truth-values (t), e.g.:
  
  \[
  \text{bob: e} \\
  \text{sings(bob): t}
  \]
Truth-Conditional Semantics

- For verbs (and verb phrases), \texttt{sings} combines with \texttt{bob} to produce \texttt{sings(bob)}

- In general, we use lambda-calculus or \(\lambda\)-calculus, i.e., a notation for functions whose arguments have not yet been filled/resolved/satisfied

- \(\lambda x.\text{sings}(x)\)

- This is a ‘predicate’, i.e., a function which take an entity (type \(e\)) and produces a truth value (type \(t\)), denoted as \(e \to t\)
Compositional Semantics

Now after we have these meanings for words, we want to combine them into meaning for phrases and sentences.

For this, we associate a combination rule with each grammar rule of the parse tree, e.g.:

- **S**: \( \beta(\alpha) \rightarrow NP: \alpha \)  \( VP: \beta \) *(function application)*
- **VP**: \( \lambda x . \alpha(x) \land \beta(x) \rightarrow VP: \alpha \)  and: \( \emptyset \)  \( VP: \beta \) *(intersection)*

Example:

- \( S \rightarrow NP \rightarrow VP \rightarrow VP \):
- \( S \rightarrow [\lambda x.\text{sings}(x) \land \text{dances}(x)](bob) \)
- \( NP \rightarrow VP \rightarrow VP \):
- \( \text{Bob} \rightarrow \text{bob} \rightarrow \text{sings} \rightarrow \lambda y.\text{sings}(y) \)
- \( \text{VP} \rightarrow \text{dances} \rightarrow \lambda z.\text{dances}(z) \)
- \( \text{VP} \rightarrow \text{and} \rightarrow \lor x.\text{sings}(x) \land \text{dances}(x) \)
Transitive Verbs & Quantifiers

- Transitive verbs example is ‘like’ predicate:
  - $\lambda x. \lambda y. \text{likes}(y, x)$

- These are two-place predicates hence $e \rightarrow (e \rightarrow t)$

- Whereas ‘likes Amy’ = $\lambda y. \text{likes}(y, \text{amy})$ is just a one-place predicate because $x$ has been satisfied/resolved
Transitive Verbs & Quantifiers

- What about the ‘everyone’ quantifier, e.g., “Everyone likes Amy”?
  
  Everyone = $\lambda f. \forall x. f(x)$

- See example figure on how this works →

- Gets tricky for examples like: “Amy likes everyone” and “Everyone likes someone”
Indefinites

- If we say “Bob ate a waffle” and “Amy ate a waffle”, then using:
  \[ \text{ate}(bob, \text{waffle}) \]
  \[ \text{ate}(amy, \text{waffle}) \]
- Doesn’t seem correct for ‘a waffle’
- More correct seems to use ‘there exists’ operator:
  \[ \exists x: \text{waffle}(x) \land \text{ate}(bob, x) \]
- And what about ‘the’ and ‘every’?
Tense and Events

- We need **event** variables because just verbs don’t get us far!

Example: “Bob sang”

- sang(bob)?

- \( \exists e: \text{singing}(e) \land \text{agent}(e, \text{bob}) \land (\text{time}(e) < \text{now}) \)

Hence, these event variable \( e \) help us represent complex tense and aspect structures:

Example: “Bob had been singing when Mary coughed”

- \( \exists e, e': \text{singing}(e) \land \text{agent}(e, \text{bob}) \land \text{coughing}(e') \land \text{agent}(e', \text{mary}) \land \\
  (\text{start}(e) < \text{start}(e') \land \text{end}(e) = \text{end}(e')) \land \\
  (\text{time}(e') < \text{now}) \)
Adverbs

- Example: “Bob sings terribly”
- terribly(sings(bob))?  
- (terribly(sings))(bob)\
- \(\exists e: \text{present}(e) \land \text{type}(e, \text{singing}) \land \text{agent}(e, \text{bob}) \land \text{manner}(e, \text{terrible})\)\
- Gets tricky pretty quickly…
CCG Parsing

- Combinatory Categorial Grammars:
  - Each category encodes an argument sequence (fwd/bwd slashes specify argument order/direction)
  - Closely related to lambda calculus
  - Captures both syntactic and semantic info
  - Naturally allows meaning representation and semantic parsing

John ⊲ NP
shares ⊲ NP
buys ⊲ (S\NP)/NP
sleeps ⊲ S\NP
well ⊲ (S\NP)\( (S\NP) \\

```
John (S\NP)/NP \ NP
buys shares
```
CCG Parsing

Given training examples with paired sentences/questions and their logical-form lambda calculus,

Input: List one way flights to Prague.

Output: $\lambda x.\text{flight}(x) \land \text{one\_way}(x) \land \text{to}(x, \text{PRG})$

This is a tricky learning problem because the derivations are not annotated, so we learn lexicon and parameters for a weighted CCG (e.g., based on [Zettlemoyer and Collins, 2005])
## CCG Lexicon

<table>
<thead>
<tr>
<th>Words</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>flights</td>
<td>( N : \lambda x. \text{flight}(x) )</td>
</tr>
<tr>
<td>to</td>
<td>( (\text{N\N})/\text{NP} : \lambda x. \lambda f. \lambda y. f(x) \land to(y,x) )</td>
</tr>
<tr>
<td>Prague</td>
<td>( \text{NP} : \text{PRG} )</td>
</tr>
<tr>
<td>New York city</td>
<td>( \text{NP} : \text{NYC} )</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Combinator Rules

Application Unary Rules:

- \( X/Y : f \quad Y : a \quad \Rightarrow \quad X : f(a) \)
- \( Y : a \quad X\Y : f \quad \Rightarrow \quad X : f(a) \)

Composition Rules:

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

Type Raising

Crossed Composition
## CCG Parsing Example

<table>
<thead>
<tr>
<th>Show me</th>
<th>flights</th>
<th>to</th>
<th>Prague</th>
</tr>
</thead>
<tbody>
<tr>
<td>S/N</td>
<td>λf.f</td>
<td>N</td>
<td>(N\N)/NP</td>
</tr>
<tr>
<td></td>
<td>λx.flight(x)</td>
<td>λy.λf.λx.f(y)∧to(x,y)</td>
<td>NP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PRG</td>
</tr>
<tr>
<td>N\N</td>
<td></td>
<td>N</td>
<td>λf.λx.f(x)∧to(x,PRG)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>λx.flight(x)∧to(x,PRG)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>S</td>
<td>S</td>
<td>λx.flight(x)∧to(x,PRG)</td>
</tr>
</tbody>
</table>
Weighted CCG

Given a log-linear model with a CCG lexicon \( L \), a feature vector \( f \), and weights \( w \), the best parse is

\[
y^* = \arg\max_y w \cdot f(x, y)
\]

Where \( y \) is the set of all parses for sentence \( x \) based on lexicon \( L \).
Lexicon Problem and Factored Lexicons

- Lexicon is key component of CCG
- But same word often paired with many different categories
- Difficult to learn with limited sentence-logicform data
- Factored Lexicons is one solution: lexical entries share info; decomposition leads to more compact lexicons

\[
\text{the house dog} \quad \text{house} \vdash ADJ : \lambda x. of(x, vy.\text{house}(y))
\]

\[
\text{the dog of the house} \quad \text{house} \vdash N : \lambda x.\text{house}(x)
\]

\[
\forall x.\text{dog}(x) \land of(x, vy.\text{house}(y))
\]

\[
\text{the garden dog} \quad \text{garden} \vdash ADJ : \lambda x. of(x, vy.\text{garden}(y))
\]

\[
\forall x.\text{dog}(x) \land of(x, vy.\text{garden}(y))
\]
Weak Supervision

- Instead of relying on sentence-logicform pairs as training data, we can learn from query-answer pairs.

- Logical forms are latent, and we can check which one gets the correct answer on being executed against a knowledge base (KB).

What is the largest state that borders Texas?

\[
\text{New Mexico}
\]

\[
\text{argmax}(\lambda x. \text{state}(x) \\
\quad \land \text{border}(x, TX), \lambda y. \text{size}(y))
\]

\[
\text{argmax}(\lambda x. \text{river}(x) \\
\quad \land \text{in}(x, TX), \lambda y. \text{size}(y))
\]

New Mexico [Correct]
Rio Grande [Incorrect]

[Clarke et al. 2010; Liang et al. 2011]
Weak Supervision

- Learning from Instruction-Demonstration Pairs

at the chair, move forward three steps past the sofa

Some examples from other domains:

- Sentences and labeled game states [Goldwasser and Roth 2011]
- Sentences and sets of physical objects [Matuszek et al. 2012]
Weak Supervision

Learning from Conversation Logs

**SYSTEM** how can I help you? *(OPEN_TASK)*

**USER** i’d like to fly to new york

**SYSTEM** flying to new york. *(CONFIRM: from(fl, ATL)) leaving what city? *(ASK: \(\lambda x.\text{from}(fl, x)\))

**USER** from boston on june seven with american airlines

**SYSTEM** flying to new york. *(CONFIRM: to(fl, NYC)) what date would you like to depart boston? *(ASK: \(\lambda x.\text{date}(fl, x) \wedge \text{to}(fl, BOS)\))

**USER** june seventh

[CONVERSATION CONTINUES]
Other Semantic Parsing Ideas

- Various recent ideas/extensions:
  - Dependency-based compositional semantics (DCS)
  - Bootstrapping w/ conversations
  - On-the-fly ontology matching
  - Question answering on Freebase
  - Paraphrasing
  - Unsupervised SP (clustering lambda forms)
  - Grounded USP (via databases)

Dependency-based Compositional Semantics (DCS)

Figure 1: Our probabilistic model: a question $\mathbf{x}$ is mapped to a latent logical form $z$, which is then evaluated with respect to a world $w$ (database of facts), producing an answer $y$. We represent logical forms $z$ as labeled trees, induced automatically from $(\mathbf{x}, y)$ pairs.

[Lieng et al., 2013]
Definitions 1 (DCS trees) are labeled with relations. Formally:

\[ \text{where nodes are labeled with predicates, and edges} \]

As another example, output pairs, e.g., \( w \in \text{CA} \), \( x, y \in \text{world} \). We represent functions by a set of tuples; for example, \( \{ P(x, y), \emptyset \} \). We denote \( \{ \} \) just symbols.

The logical forms in DCS are called DCS trees, which is desirable for two reasons: (i) they parallel syntactic dependency trees, which captures the core idea of using trees to represent formal semantics. We then introduce the full framework are trees, which is desirable for two

We first present a basic version (Section 2.1) of a new semantic representation, (DCS), which is both sim-ple as well as sets and tuples formed from other values as an example domain. Let

\[ \text{of phenomena such as quantification, where syntactic and} \]

reasoning on trees (Dechter, 2003). The recurrence \( J \) evaluated on \( v \), \( i, t \) for a node \( z \) with only join relations. Such a simple and streamlined compositional semantics like a syntactic dependency tree with predicates in

\[ \text{The basic version of DCS restricts} \]

\[ \text{suitable for program induction.} \]

\[ \text{Clarke et al. (2010), are simpler but lack the full ex-}

\[ \text{pressive power of lambda calculus.} \]

\[ \text{Table 1: Possible relations appearing on the edges of a} \]

\[ \text{DCS tree. Here, } j, j' \in \{1, 2, \ldots \} \text{ and } i \in \{1, 2, \ldots \}*. \]

\[ \text{Relations } \mathcal{R} \]

\[ \frac{j}{j'} \text{ (join)} \quad E \text{ (extract)} \]

\[ \Sigma \text{ (aggregate)} \quad Q \text{ (quantify)} \]

\[ X_i \text{ (execute)} \quad C \text{ (compare)} \]
Example: *major city in California*

\[ z = \langle \text{city}; \frac{1}{1}: \langle \text{major} \rangle; \frac{1}{1}: \langle \text{loc}; \frac{2}{1}: \langle \text{CA} \rangle \rangle \]

(a) DCS tree

\[ \lambda c \exists m \exists l \exists s. \]

\[ \text{city}(c) \land \text{major}(m) \land \]

\[ \text{loc}(l) \land \text{CA}(s) \land \]

\[ c_1 = m_1 \land c_1 = l_1 \land l_2 = s_1 \]

(b) Lambda calculus formula

(c) Denotation: \( [z]_w = \{ \text{SF, LA}, \ldots \} \)

Figure 2: (a) An example of a DCS tree (written in both the mathematical and graphical notation). Each node is labeled with a predicate, and each edge is labeled with a relation. (b) A DCS tree \( z \) with only join relations encodes a constraint satisfaction problem. (c) The denotation of \( z \) is the set of consistent values for the root node.
California borders which states?  Alaska borders no states.  Some river traverses every city.  city traversed by no rivers

(a) Extraction (E)  (b) Quantification (Q)  (c) Quantifier ambiguity (Q, Q)  (d) Quantification (Q, E)

state bordering the most states  state bordering more states than Texas  state bordering the largest state  Every state’s largest city is major.

(e) Superlative (c)  (f) Comparative (c)  (g) Superlative ambiguity (c)  (h) Quantification+Superlative (Q, c)

[Liang et al., 2013]
Semantic Parsing on Freebase

Mapping questions to answers via latent logical forms. To narrow down the logical predicate space, they use a (i) coarse alignment based on Freebase and a text corpus and (ii) a bridging operation that generates predicates compatible with neighboring predicates.
Semantic Parsing via Paraphrasing

For each candidate logical form (red), they generate canonical utterances (purple). The model is trained to paraphrase the input utterance (green) into the canonical utterances associated with the correct denotation (blue).

[Berant and Liang, 2014]
The main challenge in semantic parsing is the mismatch between language and the knowledge base. (a) Traditional: map utterances directly to logical forms, (b) Kwiatkowski et al. (2013): map utterance to intermediate, underspecified logical form, then perform ontology matching to handle the mismatch, (c) Berant and Liang (2014): generate intermediate, canonical text utterances for logical forms, then use paraphrase models.

[Kwiatkowski et al., 2013; Berant and Liang, 2014]