### Semantic Role Labeling (SRL)

- Characterize clauses as relations with roles:
  - for example, the sentence "The Government blames the poor for failing to do enough to help" would be characterized as blaming the poor.
  - The letter quotes Black as saying that "white and Navajo ranchers interpret their livestock losses and blame everyone."

- Want to more than which NP is the subject (but not much more):
  - Relations like subject are syntactic, relations like agent or message are semantic

- Typical pipeline:
  - Parse, then label roles
  - Almost all errors locked in by parser
  - Really, SRL is quite a bit easier than parsing

---

### PropBank / FrameNet

- FrameNet: roles shared between verbs
- PropBank: each verb has its own roles
- PropBank more used, because it's layered over the treebank (and so has greater coverage, plus parses)
- Note: some linguistic theories postulate even fewer roles than FrameNet (e.g. 5-20 total: agent, patient, instrument, etc.)

---

### PropBank Example

**rotate:02**

- Sense: shift from one thing to another
- Arg1: thing being changed
- Arg2: extent
- 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.

- sent1: winners
- sent2: yesterday
- sent3: investors
- sent4: profits
- sent5: buying
- sent6: other issues

---

### PropBank Example

**fall:01**

- Sense: move downward
- Arg1: start point
- Arg2: end point
- Arg3: thing falling
- Arg4: extent, distance fallen

Sales fell to $251.2 million from $279.7 million.

- sent1: sales
- sent2: fell
- sent3: $251.2 million
- sent4: $279.7 million
PropBank Example

aim.01 sense: introd., plan
roles: Arg0: actor, planner
Arg1: plan, event
The Central Council of Church Bell Ringers aims "trace" to improve relations with vicars.
arg0: The Central Council of Church Bell Ringers
arg1: "trace" to improve relations with vicars

aim.02 sense: point (weapon) at
roles: Arg0: aim, etc.
Arg1: weapon, etc.
Arg2: target
Guns have been aiming packages at the elderly.
arg0: guns
arg1: packages
arg2: the elderly

Path Features

<table>
<thead>
<tr>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VB/VP/PP</td>
<td>PP argument-subject</td>
</tr>
<tr>
<td>VB/VP/PP/NP</td>
<td>subject</td>
</tr>
<tr>
<td>VB/VP/OP</td>
<td>object</td>
</tr>
<tr>
<td>VB/VP/VP/S/NP</td>
<td>subject (embedded VP)</td>
</tr>
<tr>
<td>VB/VP/ADJP/V</td>
<td>adverbial adjunct</td>
</tr>
<tr>
<td>NN/NP/NP/PP</td>
<td>prepositional complement of noun</td>
</tr>
</tbody>
</table>

Shared Arguments

(NP-SBJ (Jmassive) (J internal) (NN SB-obj))
(VP (VBN forced)
	(NP-SBJ: (DT the) (NN SB-obj))
	(VP (VBN forced)
		(NP-SBJ: (DT the) (NN SB-obj))
		(VP (VBN forced)
			)

Results

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

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

<table>
<thead>
<tr>
<th></th>
<th>FORT</th>
<th>AGUC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.5</td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>0.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>FORT</th>
<th>AGUC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.1</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Interaction with Empty Elements

Empty Elements

- In the PTB, three kinds of empty elements:
  - Null items (usually complementizers)
  - Dislocation (WH-traces, topicalization, relative clause and heavy NP extraposition)
  - Control (raising, passives, control, shared argumentation)

- Need to reconstruct these (and resolve any indexation)
Types of Empties

Pattern-Matching Details
- Something like transformation-based learning
- Extract patterns
  - Details: transitive verb marking, auxiliaries
  - Details: legal subtrees
- Rank patterns
  - Pruning ranking: by correct / match rate
  - Application priority: by depth
- Pre-order traversal
- Greedy match

A Pattern-Matching Approach

[Johnson 02]

Top Patterns Extracted

<table>
<thead>
<tr>
<th>Command</th>
<th>Match</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>5170</td>
<td>5208</td>
<td>CO CON ( MEM &quot;-2&quot; ) Vs</td>
</tr>
<tr>
<td>7078</td>
<td>6745</td>
<td>( CON: ( WM ( WM 6 ) )</td>
</tr>
<tr>
<td>3075</td>
<td>6365</td>
<td>( CON: ( WM ( WM 6 ) )</td>
</tr>
<tr>
<td>645</td>
<td>5686</td>
<td>( CON: ( WM ( WM 6 ) )</td>
</tr>
<tr>
<td>669</td>
<td>5687</td>
<td>( CON: ( WM ( WM 6 ) )</td>
</tr>
<tr>
<td>770</td>
<td>6701</td>
<td>( CON: ( WM ( WM 6 ) )</td>
</tr>
<tr>
<td>472</td>
<td>5726</td>
<td>( CON: ( WM ( WM 6 ) )</td>
</tr>
<tr>
<td>294</td>
<td>278</td>
<td>( CON: ( WM ( WM 6 ) )</td>
</tr>
</tbody>
</table>
Results

A Machine-Learning Approach

- [Levy and Manning 04]
- Build two classifiers:
  - First one predicts where empties go
  - Second one predicts if/where they are bound
- Use syntactic features similar to SRL (paths, categories, heads, etc)

Semantic Interpretation

- Back to meaning!
  - A very basic approach to computational semantics
  - Truth-theoretic notion of semantics (Tarskian)
  - Assign a "meaning" to each word
  - Word meanings combine according to the parse structure
  - We'll spend about an hour!
- What's NLP and what isn't?
  - Designing meaning representations?
  - Computing those representations?
  - Reasoning with them?
- Supplemental reading will be on the web page.

Entailment and Presupposition

- Some notions worth knowing:
  - Entailment:
    - A entails B if A being true necessarily implies B is true
    - "Twitchy is a big mouse" → "Twitchy is a mouse"
    - "Twitchy is a big mouse" → "Twitchy is big"
    - "Twitchy is a big mouse" → "Twitchy is furry"
  - Presupposition:
    - A presupposes B if A is only well-defined if B is true
    - "The computer in the corner is broken" presupposes that there is a (salient) computer in the corner

Meaning

- "Meaning"
  - What is meaning?
    - "The computer in the corner."
    - "Bob likes Alice."
    - "I think I am a gummi bear."
  - Knowing whether a statement is true?
    - Knowing the conditions under which it's true?
    - Being able to react appropriately to it?
      - "Who does Bob like?"
      - "Close the door."
  - A distinction:
    - Linguistic (semantic) meaning
    - Speaker (pragmatic) meaning
  - Today: assembling the semantic meaning of sentence from its parts

Truth-Conditional Semantics

- Linguistic expressions:
  - "Bob sings"
  - Logical translations:
    - sings(bob)
    - Could be $p_{1218}(e_{397})$
  - Denotation:
    - $[\text{bob}] = \text{some specific person (in some context)}$
    - $[\text{sings(bob)}] = ? ? ?$
- Types on translations:
  - bob : e
  - sings(bob) : t
Truth-Conditional Semantics

- Proper names:
  - Refer directly to some entity in the world
  - Bob: \( \text{bob} \) (Proper name)

- Sentences:
  - Are either true or false (given how the world actually is)
  - Bob sings: \( \text{sings(bob)} \)

- So what about verbs (and verb phrases)?
  - \( \lambda \)-calculus is a notation for functions whose arguments are not yet filled.
  - \( \lambda x. \text{sings}(x) \)

- Adjectives:
  - \( \text{NP} \) Bob \( \text{VP} \) \( \lambda y. \text{sings}(y) \) \( \text{sings(bob)} \)

Compositional Semantics

- So now we have meanings for the words
- How do we know how to combine words?
- Associate a combination rule with each grammar rule:

  - \( S : (\alpha \times \beta) \rightarrow \alpha \times \beta \) (Function application)
  - \( \text{VP} : \lambda x. (x) \times \beta \times \alpha \) and \( \text{NP} : \alpha \) (intersection)

  - Example:
    - \( S \) \( \text{NP} \) \( \text{VP} \)
    - \( \text{Bob} \) \( \text{VP} \) \( \text{and} \) \( \text{VP} \)
    - \( \text{sings} \) \( \lambda x. \text{sings}(x) \)
    - \( \text{dances} \) \( \lambda x. \text{dances}(x) \)
    - \( \text{sings(bob)} \land \text{dances(bob)} \)

Denotation

- What do we do with logical translations?
  - Translation language (logical form) has fewer ambiguities
  - Can check truth value against a database
    - Denotation ("evaluation") calculated using the database
  - More usefully: assert truth and modify a database
  - Questions: check whether a statement in a corpus entails the (question, answer) pair:
    - "Bob sings and dances" → "Who sings?" + "Bob"
  - Chain together facts and use them for comprehension

Other Cases

- Transitive verbs:
  - \( \lambda x. \lambda y. \text{likes}(y,x) \)
  - Two-place predicates of type \( e \rightarrow (e \rightarrow t) \)
  - \( \lambda y. \text{likes}(y,\text{Amy}) \)
    - \( \text{is just like a one-place predicate.} \)

- Quantifiers:
  - What does "Everyone" mean here?
    - \( \lambda x. \forall x. \text{likes}(y,x) \)
  - Mostly works, but some problems
    - \( \text{Have to change our NP/VP rule.} \)
    - \( \text{Won't work for "Amy likes everyone."} \)
    - \( \text{"Everyone likes someone."} \)
    - \( \text{This gets tricky quickly!} \)

Indefinites

- First try
  - "Bob ate a waffle": \( \text{ate(bob, waffle)} \)
  - "Amy ate a waffle": \( \text{ate(amy, waffle)} \)

- Can’t be right!
  - \( \exists x : \text{waffle}(x) \land \text{ate(bob,x)} \)
  - \( \text{What about \"the\"?} \)
  - \( \text{What about \"every\"?} \)

Grounding

- So why does the translation \( \lambda x. \lambda y. \text{likes}(y,x) \) have anything to do with actual liking?
  - It doesn’t (unless the denotation model says so)
  - Sometimes that’s enough: wire up \( \text{bought} \) to the appropriate entry in a database

- Meaning postulates
  - Insist, e.g. \( \forall x. \lambda y. \text{likes}(y,x) \rightarrow \text{knows}(y,x) \)
  - This gets into lexical semantics issues

- Statistical version?
Tense and Events

- In general, you don’t get far with verbs as predicates
- Better to have event variables e
  - "Alice danced" : danced(alice) ∧ (time(e) < now)
- Event variables let you talk about non-trivial tense / aspect structures
  - "Alice had been dancing when Bob sneezed" :
    ∃ e, e' : dance(e) ∧ agent(e, alice) ∧ sneeze(e') ∧ agent(e', bob) ∧ (start(e) < start(e')) ∧ (end(e') < now)

Adverbs

- What about adverbs?
  - “Bob sings terribly”
    terribly(sings(bob))?
    (terribly(sings))(bob)?
  - ∃ e : present(e) ∧ type(e, singing) ∧ manner(e, terrible)?

Propositional Attitudes

- “Bob thinks that I am a gummi bear”
  - thinks(bob, gummi(me))?
  - thinks(bob, "I am a gummi bear")?
  - thinks(bob, ^gummi(me))?
- Usual solution involves intensions (^X) which are, roughly, the set of possible worlds (or conditions) in which X is true
- Hard to deal with computationally
  - Modeling other agents models, etc
  - Can come up in simple dialog scenarios, e.g., if you want to talk about what your bill claims you bought vs. what you actually bought

Multiple Quantifiers

- Quantifier scope
  - Groucho Marx celebrates quantifier order ambiguity:
    "In this country a woman gives birth every 15 min. Our job is to find that woman and stop her."
- Deciding between readings
  - "Bob bought a pumpkin every Halloween"
  - "Bob put a warning in every window"
- Multiple ways to work this out
  - Make it syntactic (movement)
  - Make it lexical (type-shifting)

Trickier Stuff

- Non-Intersective Adjectives
  - green ball : λ x. [green(x) ∧ ball(x)]
  - fake diamond : λ x. [fake(x) ∧ diamond(x)]
- Generalized Quantifiers
  - the : λ f. [unique-member(f)]
  - ∃ : λ a : λ b : [λ x. f(x)] → a = b
  - all : λ a : λ b : [λ x. f(x)] → a = b
  - Could do with more general second order predicates, too (why worse?)
    - the(cats, meows), all(cats, meows)
- Generics
  - "Cats like naps"
  - "The players scored a goal"
- Pronouns (and bound anaphora)
  - "If you have a dime, put it in the meter."
- ... the list goes on and on!

Implementation, TAG, Idioms

- Add a "sem" feature to each context-free rule
  - S → NP loves NP
  - S[sem=loves(x,y)] → NP[sem=x] loves NP[sem=y]
  - Meaning of S depends on meaning of NPs
- TAG version:
  - I want a flight from NP[sem=x] to NP[sem=y]

Template filling:
S[sem=showflights(x,y)] → I want a flight from NP[sem=x] to NP[sem=y]
Modeling Uncertainty

- Gaping hole warning!
- Big difference between statistical disambiguation and statistical reasoning.
  
  The scout saw the enemy soldiers with night goggles.
  
  - With probabilistic parsers, can say things like “72% belief that the PP attaches to the NP.”
  - That means that probably the enemy has night vision goggles.
  - However, you can’t throw a logical assertion into a theorem prover with 72% confidence.
  - Not clear humans really extract and process logical statements symbolically anyway.
  - Use this to decide the expected utility of calling reinforcements?
  
  - In short, we need probabilistic reasoning, not just probabilistic disambiguation followed by symbolic reasoning!

CCG Parsing

- Combinatory Categorial Grammar
  
  Fully (mono-) lexicalized grammar
  
  Categories encode argument sequences
  
  Very closely related to the lambda calculus
  
  Can have spurious ambiguities (why?)

\[
\begin{array}{c}
\text{John}\vdash\text{NP} : \text{john} \\
\text{shares}\vdash\text{NP} : \text{shares} \\
\text{hugs}\vdash (S)(NP)/NP : \lambda x.\lambda y.\text{hugs}x \\
\text{sleeps}\vdash S(NP) : \lambda x.\text{sleeps}x \\
\text{welf}\vdash (S)(NP)/(S)(NP) : \lambda f.\lambda x.\text{welf}(f)x \\
\end{array}
\]

\[
\begin{array}{c}
\text{NP} \\
S \\
NP \\
NP \\
\end{array}
\]

\[
\begin{array}{c}
\text{hugs} \\
\text{shares} \\
\end{array}
\]