Semantic Role Labeling (SRL)

- Characterize clauses as relations with roles:
  \[ \text{She} \text{blames the Government for failing to do enough to help}. \]
  Holmes would characterize this as blaming [agent, the poor].
  The letter quotes Black as saying that [negate, white and Navajo ranchers]
  misrepresent their livestock losses and blame [increase, everything].
- 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 lot easier than parsing

**SRL Example**

**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**

**PropBank Example**

<table>
<thead>
<tr>
<th>role</th>
<th>sense</th>
<th>arg1</th>
<th>arg2</th>
<th>arg3</th>
<th>arg4</th>
</tr>
</thead>
<tbody>
<tr>
<td>fall.01</td>
<td>sense: move downward</td>
<td>Sales</td>
<td>unit</td>
<td>to $251.2 million</td>
<td>from $270.7 million</td>
</tr>
</tbody>
</table>

Many of Wednesday’s winners were losers yesterday as investors quickly took profits and rotated their buying to other issues, traders said. (negb.1723)
PropBank Example

aim.01  sense: intend, plan
  roles: Arg0: agent, planner
  Arg1: plan, event

The Central Council of Church Bell Ringers aims *trace* to
improve relations with vicars.

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

Humans have been aiming packages at the elderly.

arg0: hands
arg1: packages
arg2: at the elderly

Shared Arguments

(PP-NNB) (NP-internal) (NN-debt)

(VP-VBZ has)

(VP-VBN forced)

(VP-NN)

(VP-VBZ

(ADVP-MNR (BE inactively))

Path Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VB/VB/PP</td>
<td>PP argument/adjunct</td>
</tr>
<tr>
<td>VB/VB/PP/NP</td>
<td>subject</td>
</tr>
<tr>
<td>VB/VB/PP/N</td>
<td>object</td>
</tr>
<tr>
<td>VB/VB/PP/NP/N</td>
<td>subject (embedded VP)</td>
</tr>
<tr>
<td>VB/VB/PP/NP/VP</td>
<td>oblique/adjunct</td>
</tr>
<tr>
<td>NP/NP/NP/PP</td>
<td>prepositional complement of noun</td>
</tr>
</tbody>
</table>

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>Feature</th>
<th>Gold</th>
<th>Parse</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRL</td>
<td>10.5</td>
<td>9.1</td>
</tr>
<tr>
<td>Accuracy</td>
<td>87.4</td>
<td>55.0</td>
</tr>
</tbody>
</table>

Empty Elements

- In the PTB, three kinds of empty elements:
  - Null items (usually complementizers)
  - Dislocation (WH-traces, topicalization, relative
    clause and heavy NP extrapolation)
  - 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
### Results

<table>
<thead>
<tr>
<th>Empty node POS</th>
<th>Section 23</th>
<th>Parser output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FOS</td>
<td>Label</td>
<td>Section 23</td>
</tr>
<tr>
<td>(Overall)</td>
<td>0.93</td>
<td>0.87</td>
</tr>
<tr>
<td>NP</td>
<td>0.05</td>
<td>0.01</td>
</tr>
<tr>
<td>NP</td>
<td>?</td>
<td>0.93</td>
</tr>
<tr>
<td>Person</td>
<td>0.94</td>
<td>0.99</td>
</tr>
<tr>
<td>People can and do spend entire courses on this topic</td>
<td>0.95</td>
<td>0.97</td>
</tr>
</tbody>
</table>

### 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.**

### 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

### 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

### Truth-Conditional Semantics

- **Linguistic expressions:**
  - "Bob sings"
- **Logical translations:**
  - sings(bob)
  - Could be p.1218(e_397)
- **Denotation:**
  - [bob] = some specific person (in some context)
  - [sings(bob)] = ????
- **Types on translations:**
  - bob: e (for entity)
  - sings(bob): 1 (for truth-value)
Truth-Conditional Semantics

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

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

- So what about verbs (and verb phrases)?
  - \( \text{sings} \) must combine with \( \text{bob} \) to produce \( \text{sings(bob)} \)
  - The \( \lambda \)-calculus is a notation for functions whose arguments are not yet filled.
  - \( \text{sings} : \lambda x.\text{sings}(x) \)
  - This is a predicate – a function which takes an entity (type e) and produces a truth value (type t). We can write its type as \( e \rightarrow t \).

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:
  - \( S : (\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \beta \) (function application)
  - \( \alpha : \beta \rightarrow \gamma \rightarrow \delta \rightarrow \gamma \) (intersection)

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

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" \( \rightarrow \) "Who sings?" + "Bob"
    - Chain together facts and use them for comprehension

Other Cases

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

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

Indefinites

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

  - Can’t be right!
    - \( \lambda x.\text{waffle}(x) \land \text{ate}(\text{bob},x) \)
    - What does the translation of "a" have to be?
    - What about "the"?
    - What about "every"?

Grounding

- **Grounding**
  - So why does the translation \( \text{likes} : \lambda x.\lambda y.\text{likes}(y,x) \) 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. \( \forall x.\forall y.\text{likes}(y,x) \rightarrow \text{knows}(y,x) \)
    - 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" : $\text{danced}(\text{alice})$.
  - $\exists e: \text{dance}(e) \land \text{agent}(e, \text{alice}) \land (\text{time}(e) < \text{now})$.
- Event variables let you talk about non-trivial tense / aspect structures.
  - "Alice had been dancing when Bob sneezed".
    - $\exists e, e': \text{dance}(e) \land \text{agent}(e, \text{alice}) \land \text{sneeze}(e') \land \text{agent}(e', \text{bob}) \land (\text{start}(e) < \text{start}(e')) \land (\text{end}(e) = \text{end}(e'))$.

**Adverbs**

- What about adverbs?
  - "Bob sings terribly".
  - $\text{terribly(sings)}(\text{bob})$?
  - $\exists e: \text{present}(e) \land \text{type}(e, \text{singing}) \land \text{agent}(e, \text{bob}) \land \text{manner}(e, \text{terrible})$?

**Propositional Attitudes**

- "Bob thinks that I am a gummi bear".
  - $\text{thinks}($bob, $\text{gummi}(\text{me}))$?
  - $\text{thinks}($bob, "I am a gummi bear")?
  - $\text{thinks}($bob, $\ ^\text{gummi}(\text{me}))$?
- 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.

**Trickier Stuff**

- **Non-Intersective Adjectives**
  - green ball : $\lambda x. (\text{green}(x) \land \text{ball}(x))$.
  - fake diamond : $\lambda x. (\text{fake}(x) \land \text{diamond}(x))$.
- **Generalized Quantifiers**
  - the: $\lambda f. \text{unique-member}(f)$.
  - all: $\lambda f. \lambda g: \forall x.f(x) \rightarrow g(x)$.
- **Generics**
  - "Cats like naps".
  - "The players scored a goal".
- **Pronouns (and bound anaphora)**
  - "If you have a dime, put it in the meter."

**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).

**Implementation, TAG, Idioms**

- Add a "sem" feature to each context-free rule:
  - $S \rightarrow \text{NP} \text{loves} \text{NP}$.
  - $S[\text{sem} = \text{loves}(x,y)] \rightarrow \text{NP}[\text{sem} = x] \text{loves} \text{NP}[\text{sem} = y]$.
  - Meaning of $S$ depends on meaning of NPs.

- **TAG version**:
  - $S \rightarrow X \text{loves}(y,Z) \rightarrow X \text{died}(x)$.

- **Template filling**: $S[\text{sem} = \text{showflights}(x,y)] \rightarrow \text{I want a flight from NP[sem=x] to NP[sem=y]}$. 
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?)

\[
\text{John} \rightarrow \text{NP} : \text{John}^{	ext{NP}} \rightarrow \text{NP} : \text{John}^{	ext{NP}}
\]

\[
\text{shares} \rightarrow \text{NP} : \text{shares}^{	ext{NP}} \rightarrow \text{NP} : \text{shares}^{	ext{NP}}
\]

\[
\text{hugs} \rightarrow (\text{S} \\text{(NP)}) / (\text{NP}) : \lambda x. \text{hugs}(xy)
\]

\[
\text{sleeps} \rightarrow (\text{S} \\text{(NP)}) / (\text{NP}) : \lambda x. \text{sleeps}(x)
\]

\[
\text{welf} \rightarrow (\text{S} \\text{(NP)}) / (\text{S} \\text{(NP)}) : \lambda f, x. \text{welf}(fx)
\]