Semantic Role Labeling (SRL)

- Characterize clauses as relations with roles:

  \[ \text{Judge}, \text{She} \text{ blam} \text{es } \text{ the Government} \text{ for failing to do enough to help} \].

  Holman would characterize this as blaming \[ \text{ Evaluate the poor} \].

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

- 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
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

fall.01 sense: move downward
roles:
  Arg1: thing falling
  Arg2: extent, distance fallen
  Arg3: start point
  Arg4: end point

Sales fell to $251.2 million from $278.7 million.

arg1: Sales
rel: fell
arg4: to $251.2 million
arg3: from $278.7 million

PropBank Example

rotate.02 sense: shift from one thing to another
roles:
  Arg0: cause 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. (wsj_1723)

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

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

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

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

Results

<table>
<thead>
<tr>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VB</td>
<td>VP</td>
</tr>
<tr>
<td>VB</td>
<td>VP</td>
</tr>
<tr>
<td>VB</td>
<td>VP</td>
</tr>
<tr>
<td>VB</td>
<td>VP</td>
</tr>
<tr>
<td>VB</td>
<td>VP</td>
</tr>
<tr>
<td>NN</td>
<td>NP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Core</th>
<th>Arg/M</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>Acc.</td>
</tr>
<tr>
<td>92.2</td>
<td>80.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Core</th>
<th>Arg/M</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>Acc.</td>
</tr>
<tr>
<td>84.1</td>
<td>66.5</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)
Example: English

Example: German
### Types of Empties

<table>
<thead>
<tr>
<th>Antecedent</th>
<th>POS</th>
<th>Label</th>
<th>Count</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>NP</td>
<td>*</td>
<td>18,334</td>
<td>NP trace (e.g., Sam was seen *)</td>
</tr>
<tr>
<td>WHNP</td>
<td>NP</td>
<td>*</td>
<td>9,812</td>
<td>NP PRO (e.g., * to sleep is nice)</td>
</tr>
<tr>
<td></td>
<td><em>T</em></td>
<td></td>
<td>8,620</td>
<td>WH trace (e.g., the woman who you saw <em>F</em>)</td>
</tr>
<tr>
<td></td>
<td><em>U</em></td>
<td></td>
<td>7,476</td>
<td>Empty units (e.g., S 25 <em>U</em>)</td>
</tr>
<tr>
<td></td>
<td>O</td>
<td></td>
<td>5,635</td>
<td>Empty complementizers (e.g., Sam said 0 Sasha snores)</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
<td><em>T</em></td>
<td>4,004</td>
<td>Moved clauses (e.g., Sam had to go, Sasha explained <em>T</em>)</td>
</tr>
<tr>
<td>WHADVP</td>
<td>ADVP</td>
<td><em>T</em></td>
<td>2,402</td>
<td>WH-trace (e.g., Sam explained how to leave <em>T</em>)</td>
</tr>
<tr>
<td>SBAR</td>
<td></td>
<td></td>
<td>2,033</td>
<td>Empty clauses (e.g., Sam had to go, Sasha explained (SBAR))</td>
</tr>
<tr>
<td>WHNP</td>
<td>O</td>
<td></td>
<td>1,759</td>
<td>Empty relative pronouns (e.g., the woman 0 we saw)</td>
</tr>
<tr>
<td>WHADVP</td>
<td>O</td>
<td></td>
<td>575</td>
<td>Empty relative pronouns (e.g., no reason 0 to leave)</td>
</tr>
</tbody>
</table>

### A Pattern-Matching Approach

- [Johnson 02]
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

Top Patterns Extracted

<table>
<thead>
<tr>
<th>Count</th>
<th>Match</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>5816</td>
<td>6737</td>
<td>(S (NP (¬NONE- *)) VP)</td>
</tr>
<tr>
<td>5605</td>
<td>7895</td>
<td>(S/HAM (¬NONE- 0) S)</td>
</tr>
<tr>
<td>5312</td>
<td>5338</td>
<td>(S/HAM WHNP 1 (S (NP (¬NONE- 'T'-1) VP))</td>
</tr>
<tr>
<td>4441</td>
<td>5917</td>
<td>(NP OP (¬NONE- '<em>T</em>'))</td>
</tr>
<tr>
<td>1082</td>
<td>1082</td>
<td>(NP UU (¬NONE- '<em>U</em>'))</td>
</tr>
<tr>
<td>1227</td>
<td>1593</td>
<td>(VP VBD) (NP (¬NONE- *)) PP)</td>
</tr>
<tr>
<td>700</td>
<td>700</td>
<td>(ADJP OP (¬NONE- '<em>U</em>'))</td>
</tr>
<tr>
<td>662</td>
<td>1219</td>
<td>(S/HAM (WHNP-1 (¬NONE- 0)) (S (NP (¬NONE- 'T'-1) VP))</td>
</tr>
<tr>
<td>618</td>
<td>635</td>
<td>(S S-1) (NP (VP VBD (S/HAM (¬NONE- 0) (S (¬NONE- 'T'-1)))) .)</td>
</tr>
<tr>
<td>499</td>
<td>512</td>
<td>(SINV 'S-1', ' (VP VBZ (S (¬NONE- 'T'-1)))) NP .)</td>
</tr>
<tr>
<td>361</td>
<td>369</td>
<td>(SINV 'S-1', ' (VP VBD (¬NONE- 'T'-1)) NP .)</td>
</tr>
<tr>
<td>352</td>
<td>320</td>
<td>(S NF-1 (VP VBZ (S (¬NONE- 'T'-1)) VP))</td>
</tr>
<tr>
<td>316</td>
<td>273</td>
<td>(S NF-1 (VP AUX (VP VBD) (¬NONE- '*U')) PP))</td>
</tr>
<tr>
<td>322</td>
<td>467</td>
<td>(VP VBD (NP (¬NONE- *)) PP)</td>
</tr>
</tbody>
</table>
| 269   | 275   | (S 'S-1', ' NP (VP VBD (S (¬NONE- 'T'-1)))) .)
A Feature-Based 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)

<table>
<thead>
<tr>
<th>POS</th>
<th>Label</th>
<th>Empty node POS Label</th>
<th>Section 23 P</th>
<th>R</th>
<th>f</th>
<th>Parser output P</th>
<th>R</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Overall)</td>
<td></td>
<td></td>
<td>0.93</td>
<td>0.83</td>
<td>0.88</td>
<td>0.85</td>
<td>0.74</td>
<td>0.79</td>
</tr>
<tr>
<td>NP</td>
<td>*</td>
<td>0.95</td>
<td>0.87</td>
<td>0.91</td>
<td>0.86</td>
<td>0.79</td>
<td>0.82</td>
<td></td>
</tr>
<tr>
<td>NP</td>
<td><em>T</em></td>
<td>0.93</td>
<td>0.88</td>
<td>0.91</td>
<td>0.85</td>
<td>0.77</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>NP</td>
<td>T</td>
<td>0.94</td>
<td>0.99</td>
<td>0.96</td>
<td>0.86</td>
<td>0.89</td>
<td>0.88</td>
<td></td>
</tr>
<tr>
<td>NP</td>
<td>T</td>
<td>0.92</td>
<td>0.98</td>
<td>0.95</td>
<td>0.87</td>
<td>0.96</td>
<td>0.92</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>^T*</td>
<td>0.98</td>
<td>0.83</td>
<td>0.90</td>
<td>0.97</td>
<td>0.81</td>
<td>0.88</td>
<td></td>
</tr>
<tr>
<td>ADVP</td>
<td><em>T</em></td>
<td>0.91</td>
<td>0.52</td>
<td>0.66</td>
<td>0.84</td>
<td>0.42</td>
<td>0.56</td>
<td></td>
</tr>
<tr>
<td>SBAR</td>
<td>0.90</td>
<td>0.63</td>
<td>0.74</td>
<td>0.88</td>
<td>0.58</td>
<td>0.70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WHNP</td>
<td>0.75</td>
<td>0.79</td>
<td>0.77</td>
<td>0.48</td>
<td>0.46</td>
<td>0.47</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
  - People can and do spend entire courses on this topic
  - 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 sit next to?”
    - “Close the door.”

- A distinction:
  - Linguistic (semantic) meaning
    - “The door is open.”
  - 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:**
  - \[[\text{bob}]\] = some specific person (in some context)
  - \[[\text{sings(bob)}]\] = ???

- **Types on translations:**
  - \text{bob} : e (for entity)
  - \text{sings(bob)} : t (for truth-value)
Truth-Conditional Semantics

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

- **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 *predicate* – a function which takes an entity (type e) and produces a truth value (type t). We can write its type as \(\text{e} \rightarrow \text{t}\).
  - Adjectives?

---

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:
  - \(\text{S} : \beta(\alpha) \rightarrow \text{NP} : \alpha \quad \text{VP} : \beta\) (function application)
  - \(\text{VP} : \lambda x. \alpha(x) \land \beta(x) \rightarrow \text{VP} : \alpha \quad \text{and} : \emptyset \quad \text{VP} : \beta\) (intersection)
- Example:

  \[
  \text{S} \quad \text{NP} \quad \text{VP} \\
  \quad \text{Bob} \quad \text{sings} \quad \lambda y. \text{sings}(y) \\
  \quad \text{and} \quad \text{dances} \quad \lambda z. \text{dances}(z)
  \]

  \[
  \text{\text{sings(bob)} \land \text{dances(bob)}}
  \]

  \[
  \text{\text{S}} \quad \text{[\lambda x. \text{sings}(x) \land \text{dances}(x)](bob)}
  \]

  \[
  \text{\text{\lambda x. \text{sings}(x) \land \text{dances}(x)}}
  \]