Semantic Role Labeling (SRL)

- Characterize clauses as relations with roles:

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

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

  The letter quotes Black as saying that \text{ Judge white and Navajo ranchers} 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
SRL Example

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.  

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

(wsj_1723)
PropBank Example

aim.01
sense: intend, plan
rules: Arg0: aimer, plamer
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
rules: 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))...
Path Features

- 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>
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>WHADVP</td>
<td>ADVP</td>
<td><em>T</em></td>
<td>8,620</td>
<td>WH trace (e.g., the woman who you saw <em>T</em>)</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
<td><em>U</em></td>
<td>7,478</td>
<td>Empty units (e.g., S 25 <em>U</em> )</td>
</tr>
<tr>
<td>O</td>
<td></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>1,063</td>
<td>Moved clauses (e.g., Sam had to go, Sasha explained <em>T</em>)</td>
</tr>
<tr>
<td>WHNP</td>
<td>O</td>
<td></td>
<td>2,492</td>
<td>WH-trace (e.g., Sam explained how to leave <em>T</em>)</td>
</tr>
<tr>
<td>WHNP</td>
<td>O</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>6393</td>
<td>(S (NP (-NONE- *)) VP)</td>
</tr>
<tr>
<td>5005</td>
<td>7895</td>
<td>(SBAR (-NONE- 0) S)</td>
</tr>
<tr>
<td>5312</td>
<td>5336</td>
<td>(SBAR WHNP 1 (S (NP (NONE &quot;T^1&quot;) VP))</td>
</tr>
<tr>
<td>4474</td>
<td>4717</td>
<td>(NP QP (-NONE- *T^1))</td>
</tr>
<tr>
<td>1082</td>
<td>1082</td>
<td>(NP S \ UU (-NONE- &quot;U&quot;))</td>
</tr>
<tr>
<td>1327</td>
<td>1593</td>
<td>(VP VBNL+ (NP (-NONE- *)) PP)</td>
</tr>
<tr>
<td>700</td>
<td>700</td>
<td>(AD-VP QP (-NONE- &quot;U&quot;))</td>
</tr>
<tr>
<td>662</td>
<td>1219</td>
<td>(SBAR (WHNP-1 (-NONE- U)) (S (NP (-NONE- &quot;T^1&quot;) VP))</td>
</tr>
<tr>
<td>618</td>
<td>635</td>
<td>(S S -1 , NP (VP VBD (SBAR (-NONE- 0) (S (-NONE- &quot;T^1&quot;)))) .)</td>
</tr>
<tr>
<td>499</td>
<td>512</td>
<td>(SVN &quot; S-1 , &quot; (VP VBEZ (S (-NONE- &quot;T^1&quot;)) NP .)</td>
</tr>
<tr>
<td>361</td>
<td>369</td>
<td>(SVN &quot; S-1 , &quot; (VP VBDZ (S (-NONE- &quot;T^-1&quot;)) NP .)</td>
</tr>
<tr>
<td>352</td>
<td>350</td>
<td>(S NP-1 (VP VBEZ (S (NP (-NONE- &quot;T^-1&quot;) VP))</td>
</tr>
<tr>
<td>346</td>
<td>273</td>
<td>(S NP-1 (VP AUX (VP VBNL+ (NP (-NONE- *)) PP)))</td>
</tr>
<tr>
<td>422</td>
<td>46/</td>
<td>(VP VBDL+ (NP (-NONE- *)) PP)</td>
</tr>
<tr>
<td>269</td>
<td>275</td>
<td>(S &quot; S-1 , &quot; NP (VP VBD (S (-NONE- &quot;T^-1&quot;))) .)</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)

## Results

<table>
<thead>
<tr>
<th>Empty node POS</th>
<th>Label</th>
<th>Section 23</th>
<th>Parser output</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Overall)</td>
<td>0.93</td>
<td>0.85 0.88</td>
<td>0.85 0.74 0.79</td>
</tr>
<tr>
<td>NP</td>
<td>*</td>
<td>0.95 0.87 0.91</td>
<td>0.86 0.79 0.82</td>
</tr>
<tr>
<td>NP</td>
<td><em>T</em></td>
<td>0.93 0.88 0.91</td>
<td>0.85 0.77 0.81</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0.94 0.99 0.96</td>
<td>0.86 0.89 0.88</td>
</tr>
<tr>
<td></td>
<td><em>U</em></td>
<td>0.92 0.98 0.95</td>
<td>0.87 0.96 0.92</td>
</tr>
<tr>
<td>S</td>
<td><em>T</em></td>
<td>0.98 0.83 0.90</td>
<td>0.97 0.81 0.88</td>
</tr>
<tr>
<td>ADVN</td>
<td><em>T</em></td>
<td>0.91 0.52 0.66</td>
<td>0.84 0.12 0.56</td>
</tr>
<tr>
<td>SBAR</td>
<td></td>
<td>0.90 0.63 0.74</td>
<td>0.88 0.58 0.70</td>
</tr>
<tr>
<td>WHNP</td>
<td>0</td>
<td>0.75 0.79 0.77</td>
<td>0.48 0.46 0.47</td>
</tr>
</tbody>
</table>

### Performance on gold trees

<table>
<thead>
<tr>
<th>ID</th>
<th>Rel</th>
<th>Combo</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>R</td>
<td>F1</td>
</tr>
<tr>
<td>WSJ(full)</td>
<td>92.0</td>
<td>82.9 87.2</td>
</tr>
<tr>
<td>WSJ(sm)</td>
<td>92.3</td>
<td>79.5 85.5</td>
</tr>
<tr>
<td>NEGRA</td>
<td>73.9</td>
<td>64.6 69.0</td>
</tr>
</tbody>
</table>

### Performance on parsed trees

<table>
<thead>
<tr>
<th>ID</th>
<th>Combo</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>WSJ(full)</td>
<td>24.3</td>
</tr>
<tr>
<td>WSJ(sm)</td>
<td>24.3</td>
</tr>
<tr>
<td>NEGRA</td>
<td>20.9</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 like?"
    - "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:**
  - \( \text{sings}(\text{bob}) \)
  - Could be \( p_{1218}(e_{397}) \)

- **Denotation:**
  - \( [\text{bob}] = \text{some specific person (in some context)} \)
  - \( [\text{sings}(\text{bob})] = ??? \)

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

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

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

- **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 \(e \rightarrow t\).
  - 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)
  - \(\text{VP} : \lambda x.\alpha(x) \land \beta(x) \rightarrow \text{VP} : \alpha \land : \emptyset \quad \text{VP} : \beta\) (intersection)

- **Example:**
  \[\text{S} \quad \text{NP} \quad \text{VP} \quad \lambda x.\text{sings(x)} \land \text{dances(x)}(\text{bob})\]
  \[\text{NP} \quad \text{Bob} \quad \text{bob}\]
  \[\text{VP} \quad \text{sings} \quad \lambda y.\text{sings(y)}\]
  \[\text{and} \quad \text{VP} \quad \text{dances} \quad \lambda z.\text{dances(z)}\]
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:**
  - likes : \( \lambda x. \lambda y. \text{likes}(y,x) \)
  - Two-place predicates of type \( e \rightarrow (e \rightarrow t) \).
  - likes Amy : \( \lambda y. \text{likes}(y,\text{Amy}) \) is just like a one-place predicate.

- **Quantifiers:**
  - What does "Everyone" mean here?
  - Everyone : \( \lambda f. \forall x.f(x) \)
  - 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": ate(bob,waffle)
  - "Amy ate a waffle": ate(amy,waffle)

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

Grounding

- 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 bought to the appropriate entry in a database

- Meaning postulates
  - Insist, e.g. \( \forall x,y.\text{likes}(y,x) \rightarrow \text{knows}(y,x) \)
  - 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)
  - \( \exists e : \text{dance}(e) \land \text{agent}(e, \text{alice}) \land (\text{time}(e) < \text{now}) \)
- Event variables let you talk about non-trivial tense / aspect structures
  - "Alice had been dancing when Bob sneezed"
  - \( \exists e, e' : \text{dance}(e) \land \text{agent}(e, \text{alice}) \land \text{sneeze}(e') \land \text{agent}(e', \text{bob}) \land (\text{start}(e) < \text{start}(e') \land \text{end}(e) = \text{end}(e')) \land (\text{time}(e') < \text{now}) \)

Adverbs

- What about adverbs?
  - "Bob sings terribly"
  - terribly(sings(bob))?
  - (terribly(sings))(bob)?
  - \( \exists e \text{ present}(e) \land \text{type}(e, \text{singing}) \land \text{agent}(e, \text{bob}) \land \text{manner}(e, \text{terrible}) \)?
  - It’s really not this simple..
Propositional Attitudes

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

- 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

Trickier Stuff

- Non-Intersective Adjectives
  - green ball : \( \lambda x. [\text{green}(x) \land \text{ball}(x)] \)
  - fake diamond : \( \lambda x. [\text{fake}(x) \land \text{diamond}(x)] \) ? \( \lambda x. [\text{fake}(\text{diamond}(x))] \)

- Generalized Quantifiers
  - the : \( \lambda f. [\text{unique-member}(f)] \)
  - all : \( \lambda f. \lambda g \left[ \forall x. f(x) \rightarrow g(x) \right] \)
  - most?

- 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!
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**
  - $S \rightarrow NP \text{ loves } NP$
  - $S[\text{sem=}\text{loves}(x,y)] \rightarrow NP[\text{sem=}x] \text{ loves } NP[\text{sem=}y]$
  - Meaning of $S$ depends on meaning of NPs

- **TAG version**:

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

- Gaping hole warning!
- Big difference between statistical disambiguation and statistical reasoning.

\[ \text{The scout saw the enemy soldiers with night goggles.} \]

- With probabilistic parsers, can say things like “72% belief that the PP attaches to the NP.”
- That means that \textit{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?)

\[ \begin{align*}
  John \vdash \text{NP} : & \text{john} \\
  shares \vdash \text{NP} : & \text{shares} \\
  buys \vdash (S\backslash NP)/NP : & \lambda x.\lambda y.\text{buys}\,xy \\
  sleep \vdash S\backslash NP : & \lambda x.\text{sleep}\,x \\
  \text{well} \vdash (S\backslash NP)\backslash(S\backslash NP) : & \lambda f.\lambda x.\text{well}\,(f\,x) \\
\end{align*} \]