Semantic Role Labeling (SRL)

- Characterize clauses as relations with roles:
  \[
  \text{[She] \text{blames} [the Government] \text{for failing to do enough to help].}
  \]
  Holman would characterize this as blaming [the poor].
  The letter quotes Black as saying that [the white and Navajo ranchers] misrepresented their livestock losses and blame [everyone] [on the Navajo].
- Want to do 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

SRL Example

- Proposition Bank / 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

- Sales fell to $251.2 million from $278.7 million
  - arg1: Sales
  - rel: fell
  - arg4: to $251.2 million
  - arg5: from $278.7 million

- Many of Wednesday’s winners were losers yesterday as investors quickly took profits and rotated their buying to other issues, traders said.
  - arg1: traders
  - rel: rotated
  - arg2: their buying
  - arg3: to other issues
PropBank Example

aim01  sense: internal plan
roles: Arg0: plan, plan
Arg1: plan, plan
The Central Council of Church Bell Ringers aims "trace" to improve relations with vicars.

aim02  sense: point (weapon) at
roles: Arg0: point
Arg1: aim, etc.
Arg2: target

Bank has been aiming packages at the elderly.

Path Features

Path

<table>
<thead>
<tr>
<th>NP</th>
<th>VB</th>
<th>VP</th>
<th>PP</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRP</td>
<td>VB</td>
<td>VP</td>
<td>PP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Description

- PP: argument/adjunct
- NP: subject
- VB: VP/VP NP: object
- VP: VP/VP NP: subject (embedded VP)
- VP: VP/VP ADVP: adverbial
- NN/NP/VP: PP: prepositional complement of noun

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

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

<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>SB</td>
<td></td>
<td>10,031</td>
<td>Empty nouns (e.g., <em>the man who was</em>)</td>
</tr>
<tr>
<td>NP/PRO</td>
<td>SB</td>
<td></td>
<td>3,032</td>
<td>NP/PRO (e.g., <em>the deep blue sea</em>)</td>
</tr>
<tr>
<td>VP/ADVP</td>
<td>SB</td>
<td></td>
<td>7,120</td>
<td>Empty verbs (e.g., <em>he saw</em>)</td>
</tr>
<tr>
<td>DP</td>
<td>SB</td>
<td></td>
<td>1,531</td>
<td>Empty complements (e.g., <em>the fact that he missed the train</em>)</td>
</tr>
<tr>
<td>DP/ADVP</td>
<td>SB</td>
<td></td>
<td>2,402</td>
<td>Empty distance (e.g., <em>the fact that he missed the train</em>)</td>
</tr>
<tr>
<td>NP</td>
<td>SB</td>
<td></td>
<td>1,021</td>
<td>Empty clauses (e.g., <em>the fact that he missed the train</em>)</td>
</tr>
<tr>
<td>VP</td>
<td>SB</td>
<td></td>
<td>1,799</td>
<td>Empty relative pronouns (e.g., <em>the fact that he missed the train</em>)</td>
</tr>
<tr>
<td>VP/ADVP</td>
<td>SB</td>
<td></td>
<td>772</td>
<td>Empty relative pronouns (e.g., <em>the man who was</em>)</td>
</tr>
</tbody>
</table>

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>Count</th>
<th>Match</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>5106</td>
<td>6/651</td>
<td>GI (SB) -(S-NP) -(SB)</td>
</tr>
<tr>
<td>5908</td>
<td>7/903</td>
<td>GI (SB) -(S-NP) -(SB)</td>
</tr>
<tr>
<td>5122</td>
<td>5/596</td>
<td>GI (SB) -(S-NP) -(SB)</td>
</tr>
<tr>
<td>4977</td>
<td>5/567</td>
<td>GI (SB) -(S-NP) -(SB)</td>
</tr>
<tr>
<td>1127</td>
<td>1/120</td>
<td>GI (SB) -(S-NP) -(SB)</td>
</tr>
<tr>
<td>700</td>
<td>1/210</td>
<td>GI (SB) -(S-NP) -(SB)</td>
</tr>
<tr>
<td>700</td>
<td>1/210</td>
<td>GI (SB) -(S-NP) -(SB)</td>
</tr>
<tr>
<td>639</td>
<td>1/180</td>
<td>GI (SB) -(S-NP) -(SB)</td>
</tr>
<tr>
<td>1127</td>
<td>1/120</td>
<td>GI (SB) -(S-NP) -(SB)</td>
</tr>
<tr>
<td>472</td>
<td>4/528</td>
<td>GI (SB) -(S-NP) -(SB)</td>
</tr>
<tr>
<td>381</td>
<td>3/560</td>
<td>GI (SB) -(S-NP) -(SB)</td>
</tr>
<tr>
<td>352</td>
<td>3/320</td>
<td>GI (SB) -(S-NP) -(SB)</td>
</tr>
<tr>
<td>322</td>
<td>4/407</td>
<td>GI (SB) -(S-NP) -(SB)</td>
</tr>
<tr>
<td>258</td>
<td>2/295</td>
<td>GI (SB) -(S-NP) -(SB)</td>
</tr>
</tbody>
</table>
Results

Empty node | Section 23 | Parser output
--- | --- | ---
POS | Label | P | R | f | P | R | f
--- | --- | --- | --- | --- | --- | --- | ---
(Overall) | | 0.93 | 0.83 | 0.88 | 0.85 | 0.74 | 0.79
NP | * | 0.95 | 0.87 | 0.91 | 0.86 | 0.79 | 0.82
NP | *T* | 0.93 | 0.88 | 0.91 | 0.85 | 0.77 | 0.81
0 | 0.94 | 0.99 | 0.98 | 0.86 | 0.89 | 0.88
S | *T* | 0.92 | 0.96 | 0.95 | 0.87 | 0.96 | 0.92
ADVP | *T* | 0.98 | 0.83 | 0.90 | 0.97 | 0.81 | 0.88
SBAR | 0.91 | 0.52 | 0.60 | 0.84 | 0.42 | 0.56
WNP | 0.75 | 0.79 | 0.77 | 0.48 | 0.46 | 0.47

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
  - People can and do spend entire courses on this topic
  - We’ll spend 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”
  - 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:
  - `[bob]` = some specific person (in some context)
  - `[sings(bob)]` = ???

- Types on translations:
  - `bob` : `e` (for entity)
  - `sings(bob)` : `t` (for truth-value)
Truth-Conditional Semantics

- **Proper names:**
  - Refer directly to some entity in the world
  - Bob : bob \[\text{[bob]}\] → ???

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

- **So what about verbs (and verb phrases)?**
  - Must combine with bob to produce sings(bob)

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

- 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 \(→\) 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 : \(\beta(\alpha) \rightarrow \alpha \rightarrow \beta\) (function application)
  - VP : \(\lambda x.\alpha(x) \rightarrow \beta(x) \rightarrow \alpha \rightarrow \beta\) (intersection)

- **Example:**

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

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**

Indefinites

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

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

Other Cases

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

- **Quantifiers:**
  - What does “Everyone” mean here?
  - \(\exists x, y.\text{likes}(y,x)\)
  - 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!

\[\forall x.\lambda y.\text{likes}(y,x)\]

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

Adverbs

- What about adverbs?
  - “Bob sings terribly”
    - terribly(sings)(bob)
    - $\exists e, e' : present(e) \land type(e, singing) \land agent(e, bob) \land manner(e, terrible) \land (time(e') < now)$

Propositional Attitudes

- “Bob thinks that I am a gummi bear”
  - thinks(bob, gummi(me))
  - Thinks(bob, “I am a gummi bear”)
  - thinks(bob, $^\text{gummi}(me)$)
- 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 : $\lambda x. [\text{fake}(x) \land \text{diamond}(x)]$
- Generalized Quantifiers
  - the : $\lambda f. [\text{unique-member}(f)]$
  - all : $\lambda f. [\forall x. f(x) \rightarrow g(x)]$
  - could do with more general second order predicates, too (why worse?)
    - th(\text{cat}, \text{meows}), all(\text{cat}, \text{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!

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 pumpkin 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
  - $S \rightarrow NP$ loves NP
  - S[sem=loves(x,y)] \rightarrow NP[sem=x] loves NP[sem=y]
  - Meaning of S depends on meaning of NPs
- TAG version:
  - Template filling: S[sem=showflights(x,y)] \rightarrow I want a flight from NP[sem=x] to NP[sem=y]
Modeling Uncertainty

- Gaping hole warning!
- Big difference between the syntax and semantics models presented here.
  
  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 symbol 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?)

\[ \text{John} \vdash \text{NP} : \text{John}! \]
\[ \text{shares} \vdash \text{NP} : \text{shares}! \]
\[ \text{bys} \vdash (\text{S}(\text{NP}))/\text{NP} : \text{lambda lambda}_y \text{y}_y \text{y}_y \text{y}_x \]
\[ \text{sleeps} \vdash \text{S}(\text{NP}) : \text{lambda sleep}_x \text{x} \]
\[ \text{well} \vdash (\text{S}(\text{NP}))(\text{S}(\text{NP})) : \text{lambda lambda}_x \text{well}(f) \]

\[ \begin{array}{c}
\text{NP} \\
\text{S} \\
\text{John} (\text{S}(\text{NP}))/\text{NP} \\
\text{Bys} \\
\text{Shares}
\end{array} \]