Semantic Role Labeling (SRL)

- Characterize clauses as relations with roles:
  \[ \text{She blames the Government for failing to do enough to help.} \]
  Holmae would characterize this as blaming [\text{the poor}].
  The letter quotes Black as saying that [\text{white and Nacao ranchers}]
  interpreted their livestock losses and blamed [\text{everyone}]
  \text{[e.g.,} \text{on coyotes].}
- Want to go beyond 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

<table>
<thead>
<tr>
<th>role</th>
<th>sense: shift from one thing to another</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arg1</td>
<td>thing being changed</td>
</tr>
<tr>
<td>Arg2</td>
<td>start point</td>
</tr>
<tr>
<td>Arg3</td>
<td>new thing</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.

Sales fell to $251.2 million from $278.7 million
arg1: Sales
arg2: fell
arg3: to $251.2 million
arg4: from $278.7 million

SRL Example

<table>
<thead>
<tr>
<th>role</th>
<th>sense: move downward</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arg1</td>
<td>thing falling</td>
</tr>
<tr>
<td>Arg2</td>
<td>extent, distance fallen</td>
</tr>
<tr>
<td>Arg4</td>
<td>end point</td>
</tr>
</tbody>
</table>

Sales fell to $251.2 million from $278.7 million
arg1: Sales
arg2: fell
arg3: to $251.2 million
arg4: from $278.7 million

PropBank Example

<table>
<thead>
<tr>
<th>role</th>
<th>sense: move downward</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arg1</td>
<td>thing falling</td>
</tr>
<tr>
<td>Arg2</td>
<td>extent, distance fallen</td>
</tr>
<tr>
<td>Arg4</td>
<td>end point</td>
</tr>
</tbody>
</table>

Sales fell to $251.2 million from $278.7 million
arg1: Sales
arg2: fell
arg3: to $251.2 million
arg4: from $278.7 million
PropBank Example

aim.01
  sense: intrepid, plan
  roles:
  Arg0: aim, planner
  Arg1: plan, attempt
  The Central Council of Church Bell Ringers aims "trace" to
  improve relations with vicars.
  arg1: The Central Council of Church Bell Ringers
  arg2: "trace" to improve relations with vicars

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

Path Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VB/VF/V</td>
<td>PP argument/object/adjunct</td>
</tr>
<tr>
<td>VB/VF/V</td>
<td>subject</td>
</tr>
<tr>
<td>VB/VF/V</td>
<td>subject (embedded VP)</td>
</tr>
<tr>
<td>VB/VF/A</td>
<td>adverbial adjunct</td>
</tr>
<tr>
<td>NN/IN/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></th>
<th>gold</th>
<th>hi gs</th>
<th>hi acs</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-meas</td>
<td>92.4</td>
<td>79.9</td>
<td>82.3</td>
</tr>
<tr>
<td>Acc</td>
<td>94.1</td>
<td>80.0</td>
<td>84.4</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 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>Const</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>PP</td>
<td>**</td>
<td>2.000</td>
<td>Finite verb string (e.g., **am **am nicht <strong>am)</strong></td>
</tr>
<tr>
<td>N</td>
<td>SB</td>
<td>**</td>
<td>1.476</td>
<td>Empty, syntactic string (e.g., <strong>am or does not exist)</strong></td>
</tr>
<tr>
<td>N</td>
<td>SB</td>
<td>0</td>
<td>1.476</td>
<td>Empty, syntactic string (e.g., <strong>am or does not exist)</strong></td>
</tr>
<tr>
<td>N</td>
<td>SB</td>
<td>**</td>
<td>0</td>
<td>Empty, syntactic string (e.g., <strong>am or does not exist)</strong></td>
</tr>
<tr>
<td>N</td>
<td>SB</td>
<td>0</td>
<td>0</td>
<td>Empty, syntactic string (e.g., <strong>am or does not exist)</strong></td>
</tr>
<tr>
<td>N</td>
<td>SB</td>
<td>**</td>
<td>0</td>
<td>Empty, syntactic string (e.g., <strong>am or does not exist)</strong></td>
</tr>
<tr>
<td>N</td>
<td>SB</td>
<td>0</td>
<td>0</td>
<td>Empty, syntactic string (e.g., <strong>am or does not exist)</strong></td>
</tr>
<tr>
<td>N</td>
<td>SB</td>
<td>**</td>
<td>0</td>
<td>Empty, syntactic string (e.g., <strong>am or does not exist)</strong></td>
</tr>
<tr>
<td>N</td>
<td>SB</td>
<td>0</td>
<td>0</td>
<td>Empty, syntactic string (e.g., <strong>am or does not exist)</strong></td>
</tr>
<tr>
<td>N</td>
<td>SB</td>
<td>**</td>
<td>0</td>
<td>Empty, syntactic string (e.g., <strong>am or does not exist)</strong></td>
</tr>
<tr>
<td>N</td>
<td>SB</td>
<td>0</td>
<td>0</td>
<td>Empty, syntactic string (e.g., <strong>am or does not exist)</strong></td>
</tr>
<tr>
<td>N</td>
<td>SB</td>
<td>**</td>
<td>0</td>
<td>Empty, syntactic string (e.g., <strong>am or does not exist)</strong></td>
</tr>
<tr>
<td>N</td>
<td>SB</td>
<td>0</td>
<td>0</td>
<td>Empty, syntactic string (e.g., <strong>am or does not exist)</strong></td>
</tr>
<tr>
<td>N</td>
<td>SB</td>
<td>**</td>
<td>0</td>
<td>Empty, syntactic string (e.g., <strong>am or does not exist)</strong></td>
</tr>
<tr>
<td>N</td>
<td>SB</td>
<td>0</td>
<td>0</td>
<td>Empty, syntactic string (e.g., <strong>am or does not exist)</strong></td>
</tr>
<tr>
<td>N</td>
<td>SB</td>
<td>**</td>
<td>0</td>
<td>Empty, syntactic string (e.g., <strong>am or does not exist)</strong></td>
</tr>
<tr>
<td>N</td>
<td>SB</td>
<td>0</td>
<td>0</td>
<td>Empty, syntactic string (e.g., <strong>am or does not exist)</strong></td>
</tr>
<tr>
<td>N</td>
<td>SB</td>
<td>**</td>
<td>0</td>
<td>Empty, syntactic string (e.g., <strong>am or does not exist)</strong></td>
</tr>
<tr>
<td>N</td>
<td>SB</td>
<td>0</td>
<td>0</td>
<td>Empty, syntactic string (e.g., <strong>am or does not exist)</strong></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>Comm</th>
<th>Match</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>348</td>
<td>1378</td>
<td>C C 3  ( SOME &quot;x&quot;)  X</td>
</tr>
<tr>
<td>554</td>
<td>924</td>
<td>C C 3  ( SOME &quot;x&quot;)  X</td>
</tr>
<tr>
<td>554</td>
<td>924</td>
<td>C C 3  ( SOME &quot;x&quot;)  X</td>
</tr>
<tr>
<td>554</td>
<td>924</td>
<td>C C 3  ( SOME &quot;x&quot;)  X</td>
</tr>
<tr>
<td>554</td>
<td>924</td>
<td>C C 3  ( SOME &quot;x&quot;)  X</td>
</tr>
<tr>
<td>554</td>
<td>924</td>
<td>C C 3  ( SOME &quot;x&quot;)  X</td>
</tr>
<tr>
<td>554</td>
<td>924</td>
<td>C C 3  ( SOME &quot;x&quot;)  X</td>
</tr>
<tr>
<td>554</td>
<td>924</td>
<td>C C 3  ( SOME &quot;x&quot;)  X</td>
</tr>
<tr>
<td>554</td>
<td>924</td>
<td>C C 3  ( SOME &quot;x&quot;)  X</td>
</tr>
<tr>
<td>554</td>
<td>924</td>
<td>C C 3  ( SOME &quot;x&quot;)  X</td>
</tr>
<tr>
<td>554</td>
<td>924</td>
<td>C C 3  ( SOME &quot;x&quot;)  X</td>
</tr>
<tr>
<td>554</td>
<td>924</td>
<td>C C 3  ( SOME &quot;x&quot;)  X</td>
</tr>
</tbody>
</table>

Example: English

Example: German
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:**
  - $sings(bob)$
  - $\lambda y. y.sings(y)$

- **Denotation:**
  - $\lambda$ is 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 \(\equiv \text{bob} \rightarrow ??\)

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

- **So what about verbs (and verb phrases)?**
  - \(\lambda\text{-calculus}\) 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 \(e \rightarrow t\).

- **Adjectives?**
  - \(\text{NP} \rightarrow \text{VP} \rightarrow \text{S}\)
  - \(\lambda y.\text{sings}(y)\)
  - \(\lambda z.\text{dances}(z)\)

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” \(\rightarrow\) “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”: \(\text{ate(amy,waffle)}\)

  \(\exists x : \text{waffle}(x) \land \text{ate(bob,x)}\)

  - What about “every”?

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

  \(\text{Example:}\)
  - \(\text{S NP VP}\)
  - \(\text{Everyone VBP NP}\)
  - \(\text{Amy likes }\lambda x.\lambda y.\text{likes}(y,x)\)
  - \(\lambda x.\lambda y.\text{likes}(y,amy)\)
  - \(\lambda f.\forall x.f(x)\)

Other Cases

- **Transitive verbs:**
  - \(\lambda x.\lambda y.\text{likes}(y,x)\)
  - 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.

  \(\text{Quantifiers:}\)
  - What does “Everyone” mean here?
    - \(\forall x,y.\text{likes}(y,x)\)
  - 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”: \(\text{ate(bob,waffle)}\)
  - “Amy ate a waffle”: \(\text{ate(amy,waffle)}\)

  \(\exists x : \text{waffle}(x) \land \text{ate(bob,x)}\)

  - 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

  \(\text{Meaning postulates}\)
  - 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”: \( \text{danced(alice)} \)  
  
  \( \exists e \colon \text{dance(e)} \wedge \text{agent(e, alice)} \wedge (\text{time(e)} < \text{now}) \)

- Event variables let you talk about non-trivial tense / aspect structures
  - “Alice had been dancing when Bob sneezed”
  
  \( \exists e, e' \colon \text{dance(e)} \wedge \text{agent(e, alice)} \wedge \text{sneeze(e')}) \wedge \text{agent(e', bob)} \wedge (\text{start(e)} < \text{start(e')}) \wedge (\text{end(e')} < \text{now}) \)

Adverbs

- What about adverbs?
  
  - “Bob sings terribly”
    
    \( \text{terribly(sings(bob))} \)  
    
    \( \text{(terribly(sings))}(bob) \)  
    
    \( \exists e \colon \text{present(e)} \wedge \text{type(e, singing)} \wedge \text{agent(e, bob)} \wedge \text{manner(e, terrible)} \)  
    
    - 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

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)

Adverbs

- What about adverbs?
  
  - “Bob sings terribly”
  
  - terribly(sings(bob))?
  
  - \( \exists e \colon \text{present(e)} \wedge \text{type(e, singing)} \wedge \text{agent(e, bob)} \wedge \text{manner(e, terrible)} \)  

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

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)

Adverbs

- What about adverbs?
  
  - “Bob sings terribly”
  
  - terribly(sings(bob))?
  
  - \( \exists e \colon \text{present(e)} \wedge \text{type(e, singing)} \wedge \text{agent(e, bob)} \wedge \text{manner(e, terrible)} \)  

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

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)
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?)

- **Examples**
  
  \[ \text{Ahn} \rightarrow \text{NP} : \text{johl}^! \]
  \[ \text{shares} \rightarrow \text{NP} : \text{shares}^! \]
  \[ \text{bnya} \rightarrow (S[NP])\text{/NP} : \text{x} \times \text{y} \times \text{bnya}^! \text{x} \]
  \[ \text{sleeps} \rightarrow S[NP] : \text{x} \times \text{x} \times \text{sleeps}^! \text{x} \]
  \[ \text{well} \rightarrow (S[NP])\text{/NP} : \text{x} \times \text{x} \times \text{well}^! \text{x} \]

- **Diagram**
  
  ![Diagram](attachment:image.png)