Carnegie Mellon University
Language
Technologies
Institute

11-411/11-611 Natural Language Processing

Verb and Sentence Semantics

David R. Mortensen and Lori Levin

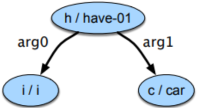
April 4, 2023

Language Technologies Institute

I have a car.

Meaning Representations Link Sentences to States of Affairs

$$\exists e, y\ Having(e) \wedge Haver(e, Speaker) \wedge HadThing(e, y) \wedge Car(y)$$




```
(h / have-01
      arg0: (i / i)
      arg1: (c / car))
```

```
Having:
      Haver:    Speaker
      HadThing: Car
```

Figure 19.1 A list of symbols, two directed graphs, and a record structure: a sampler of meaning representations for *I have a car.*

Two More Meaning Representations

Resource Description Framework Triples

$$\begin{array}{c} \text{Car} \\ \Uparrow \text{POSS-BY} \\ \text{Speaker} \end{array}$$

Predicate-Based Meaning Representation

RESTAURANT(Udipi)
SERVE(Udipi, Indian)
RESTAURANT(MadMex)
SERVE(MadMex, Mexican)

What are Meaning Representations for?

Meaning representations provide an interface between natural language (syntax, morphology, lexicon, etc.,) and extra-linguistic realities.

Learning Objectives

- Understand the criteria for a good MRL (meaning representation language)
- Be able to represent simple English sentences in model-theoretic semantics
- Be able to represent simple English statements in first order logic (FOL)
- Understand the advantages of a Neo-Davidsonian representation of FOL
- Be able to compose the meaning of a sentence from the meanings of the constituents using lambda abstraction and β-reduction.
- Be able to explain the uses and value of semantic role labels

When to use a symbolic representation instead of a neural system

- When you want to control or interpret the representation and the reasoning over the representation, in almost any NLP task.
- Hybrid neuro-symbolic systems are a popular subject of current research.

Desirable Properties (Desiderata) of Meaning Representations

Imagine a Knowledge Base

The KB contains the things we know: **knowledge elements like those from the preceding slide**.

Our knowledge base (KB)

We can query it and use it to make inferences.

TBox and Abox Are the Components of a Traditional KB

TBox (Terminology) contains the knowledge about **categories** or **concepts** in an application domain.

- All cathedrals are churches.
- All churches are buildings.

ABox (Assertions) contains facts about **instances** of categories.

- St. Paul's is a cathedral.
- David likes Haskell.

Desireable Qualities: Verifiability against a Knowledge Base

Udipi serves vegetarian food.

- We need a way to know whether the sentence is true.
- We want to represent the sentence in a meaning representation like *`serve(Udipi, vegetarian food)`* and then
- see whether *`serve(Udipi, vegetarian food)`* matches something in a knowledge base of facts.
- If the meaning representation matches something in a knowledge base, we will say that the sentence *Udipi serves vegetarian food* is true.

Desirable Qualities: Unambiguous Representation

Let’s eat somewhere near campus

eat *at* a place near campus

eat a place near campus

Our MRL should encode each of the bottom possibilities distinctly. It should not be ambiguous in the same way that the English sentence is ambiguous.

Desirable Qualities: Vagueness is allowed

I want to eat Italian food could mean that I want to eat pasta or pizza or prosciutto, etc. It is not necessary to make separate representations of all the things it could mean.

Ambiguity vs Vagueness

Ambiguous: two meaning representations
I want to eat somewhere near campus. (at someplace)
Godzilla wants to eat somewhere near campus. (eat someplace)
I want to eat somewhere near campus and so does Godzilla.

Vague: one meaning representation
I want to eat Italian food. (pasta)
Godzilla wants to eat Italian food (pizza)
I want to eat Italian food and so does Godzilla.

Desirable Qualities: Canonical Form

- “Mad Mex has vegetarian dishes.”
- “They have vegetarian food at Mad Mex.”
- “Vegetarian dishes are served at Mad Mex.”
- “Mad Mex serves vegetarian fare.

Inputs that mean the same thing should have the same meaning representation: SERVE(MadMex, VegetarianFood)

Desirable Qualities: Infererence

“Vegetarians can eat at Mad Mex.”

- We need a way to draw conclusions about things that are not explicitly in the knowledge base using a line of reasoning.
  - If a restaurant serves vegetarian food, vegetarians can eat there. Mad Mex serves vegetarian food. Therefore vegetarians can eat there.

Desirable Qualities: Variables

"I'd like to find a restaurant where I can get vegetarian food."

- Should trigger a search in a knowledge base for an x such that SERVE(x, VegetarianFood).

Desirable Qualities: Expressiveness

“Delta flies Boeing 737s from Boston to New York.”

- The meaning representation should be expressive enough to represent everything we want to say.

Self-Imposed Limitation: Literality

Natural language is full of non-literal expressions:

- I heard it from the horse's mouth.
- Meta is hemorrhaging money!
- Dynamically-typed languages are a pain in the ass.
- Why won't you take back what you said?

We are going to represent only the **literal**, **compositional** meanings of these expressions (and leave the metaphorical interpretation up to pragmatics).

Extensional versus Intentional Semantics

Two Approaches to Semantics

- Intentional
  - Views the meanings of words in terms of their (formal) definitions, e.g., *boy* means `[-female, -adult, +human ]`
- Extensional
  - Views meanings of language in terms of claims about the ACTUAL WORLD.
  - For simplicity, we will assume an extensional view of semantics.

(Open Class) Word Meanings are Predicates and Predicates are Sets

The semantic content of *red* (the word) is RED (the predicate).

RED =

The predicate is defined in terms of things in the world, not intrinsic qualities.

Model Theoretic Semantics: based on set membership

RED(x) is the assertion that x $\in$ RED

Properties such as *red* are represented as sets of entities. ***RED(x)*** is true if x $\in$ ***RED***.

Model Theoretic Semantics: based on set membership

Relations such as *LIKE* are sets of $\langle x, y \rangle$ pairs where x likes y

- Asserting that Alan likes David or LIKE(Alan, David) is asserting that $\langle$Alan, David$\rangle$ is an element of the set LIKE
- Such assertions can be either true or false depending on whether $\langle$Alan, David$\rangle$ is in the set LIKE.

- **Domain:** Noah, Karen, Rebecca, Frederick, Green Mango, Casbah, Udipi, Thai, Mediterranean, Indian
  - n, k, r, f, g, c, u, t, m, i
- **Properties:** Green Mango and Udipi are crowded; Casbah is expensive
- **Relations:** Karen likes Green Mango, Frederick likes Casbah, everyone likes Udipi, Green Mango serves Thai, Casbah serves Mediterranean, and Udipi serves Indian
  - Crowded = g, u
  - Expensive = c
  - Likes = (k, g), (f, c), (n, u), (k, u), (r, u), (f, u)
  - Serves = (g, t), (c, m), (u, i)

What Elements Must We Represent?

A meaning representation language should let us represent the following:

objects like restaurants, people, noodles, etc.,

properties like crowded, expensive, cozy, etc.,

relations like SERVES in SERVES(Casbah, Mediterranean) or BROTHER in BROTHER(Jared, David)

Another way of dividing things up is into the following:

entities (Constants or one-place predicates that encode participants in events)

events (May be one-place, two-place,..., or n-place predicates that encode STATES OF AFFAIRS)

relations (two-place, ..., n-place predicates that encode relationships among events, entities, and relations.)

All MRLs will need a way of representing these elements.

First Order Logic

Elements of First-Order Logic

entities Named elements like David or Udipi

functions Take arguments and return an entity

- $\mathrm{Cuisine}(\mathrm{Udipi}) = \mathrm{Indian}$
- $\mathrm{Cuisine}(\mathrm{Casbah}) = \mathrm{Mediterranean}$

predicates Take arguments and evaluate as either True or False

- $\mathrm{Crowded}(\mathrm{Udipi})$ (T)
- $\mathrm{Expensive}(\mathrm{Udipi})$ (F)
- $\mathrm{Like}(\mathrm{Alan}, \mathrm{David})$ (T)

variables Elements that are non-named and which are bound by quantifiers. **An essential ingredient in FOL.**

Formulas in FOL

predicates symbols for predicates, followed by arguments in parentheses: $\mathrm{ADMIRE}(\mathrm{David}, \mathrm{Alan})$.

negation $\neg$ negates any formula.

connectives binary connectives join any two formulas:

- $\implies$ means “implies”
- $\wedge$ means “and”
- $\vee$ means “or”

quantifiers quantify over variables in the scope of a formula. For example, if ϕ is a formula and x is a variable then $\forall x \phi$ (for all x, ϕ holds) and $\exists x \phi$ (there exists x such that ϕ) are valid formulas.

Examples of FOL

1. $\exists x.\text{RESTAURANT}(x) \land \text{SERVE}(x, \text{VegetarianFood})$
2. $\forall x.\text{MEAT}(x) \implies \neg\text{EAT}(\text{Lori}, x)$
3. $\exists x, y.\text{PERSON}(x) \land \text{RESTAURANT}(y) \land \neg\text{HASVISITED}(\text{x}, \text{y})$
4. $\exists x.\text{PERSON}(x) \land \forall y.(\text{RESTAURANT}(y) \implies \neg\text{HASVISITED}(\text{x}, \text{y}))$
5. $\exists x.\text{RESTAURANT}(\text{x}) \land \forall \text{y}.(\text{PERSON}(\text{y}) \land \text{HASVISITED}(\text{x}, \text{y}))$

- Nouns correspond to one-place predicates:
  RESTAURANT(x) is true if x is a member of the set of restaurants
- Adjectives correspond to one-place predicates:
  VEGETARIAN(x) is true if x is a member of the set of things that are vegetarian
- Verbs correspond to one-place, two-place, or three-place predicates
  DINE(x) as in Noah dined.
  EAT(x, y) as in Noah ate American food.
  GIVE(x, y, z) as in The bad sushi gave Noah a stomach ache.

Modus Ponens

As individual facts are added to a knowledge base, modus ponens can be used to fire applicable implication rules

An example of Modus Ponens

$$\frac{\begin{array}{l}\mathrm{CATHOLIC}(\mathrm{StPauls}) \\ \forall x.\mathrm{CATHOLIC}(x) \implies \mathrm{SERVES}(x, \mathrm{Communion})\end{array}}{\mathrm{SERVES}(\mathrm{StPauls}, \mathrm{Communion})}$$

COMMUNION is the ritual meal (the body and blood of Christ) celebrated by some Christians.

Modus Ponens: Friend or Foe?

- Advantages of Modus Ponens
  - Fire rules to prove implications of propositions
  - Simple, elegant, and deterministic
- Disadvantages of Modus Ponens
  - Proves many irrelevant propositions
  - Knowledge base grows exponentially

Advantages of FOL

- Flexible and expressive
- Well-understood
- Widely used

Neo-Davidsonian Representations address a shortcoming of FOL

https://en.wikipedia.org/wiki/Donald_Davidson_(philosopher)

In Neo-Davidsonian Representations, Events are Treated as Entities

Shortcoming of FOL: predicates don't have a fixed number of arguments.

- It was thrown.
- He threw it.
- He threw it from the bridge.
- He threw it from the bridge with his teeth.
- He threw it from the bridge to the volleyball court.

Neo-Davidsonian representations combine FOL with Semantic Roles and thus avoid this problem:

- Each event is treated as a variable (e.g., e)
- Semantic roles are treated as binary relations between events and entities (e.g., AGENT(e , David))

Neo-Davidsonian Representations: Example

- David threw the midterms from Pausch Bridge to the hillside below
  - THROW(David, midterms, PauschBridge, hillside)
  - $\exists$e.THROW(e) $\wedge$ AGENT(e, David) $\wedge$ THEME(e, midterms) $\wedge$ SOURCE(e, PauschBridge) $\wedge$ GOAL(e, hillside)
- The midterms were thrown from Pausch Bridge
  - THROW(midterms, PauschBridge)
  - $\exists$e.THROW(e) $\wedge$ THEME(e, midterms) $\wedge$ SOURCE(e, PauschBridge)

The Lambda Calculus

Lambdas are a Simple Notation for Functions and Function Application

- Lambda calculus is important to some types of semantic parsing (especially those using FOL)
- Lambda expressions form a simple notation for functions and application

$\lambda n.n+1$	a function from n to $n+1$
$(\lambda n.n+1)3$	apply that function to 3 (yielding 4)

- Lambdas are anonymous functions

Turn a statement into a function:

- **Statement:** loves(david, linguistics)
- **Functions:** (formed by lambda abstraction)
    - $\lambda x.loves(x, linguistics)$
    - $\lambda x.loves(david, x)$
    - $\lambda x.\lambda y.loves(x, y)$

Function Application

- $\lambda x.loves(x, linguistics)$ (Jelinek)
  loves(Jelinek, linguistics) = false
- $\lambda x.loves(david, x)$ (linguistics)
  loves(david, linguistics) = true
- $\lambda x.\lambda y.loves(x, y)$ (david) (linguistics)
  loves(david, linguistics) = true

Lambda Expressions in Different Programming Languages

Python:	`lambda n: n + 1`
Javascript:	`n => n + 1`
Java:	`(n) -> return n + 1`
C#:	`n => n + 1`
Haskell:	`\n -> n + 1`

The Basic Components of Lambda Calculus

Basic components of lambda calculus

Variables	x
Abstraction	$(\lambda x.M)$, where M is a lambda term
Application	(MN), where M and N are lambda terms

Importantly, **variables can be functions** (lambda expressions can be applied to other lambda expressions). Functions that take functions as arguments are called HIGHER-ORDER FUNCTIONS.

You Can Combine Lambdas with FOL

- The semantics of syntactic expressions can be represented in lambda calculus
- Simple example—words:

knotty	$\lambda x.Knotty(x)$
problems	$\lambda x.Problem(x)$

- We convert them to functions because $Knotty(x)$ (predicate with argument(s)) is not a lambda term

β-Reduction is Function Application

Given an expression (λx.M)N, yield a new expression with every occurance of x in M replaced by N.

- $(\lambda x.x2 + 2 \cdot x + 2)4 \rhd 42 + 2 \cdot 4 + 2$
- $(\lambda x.\text{Study}(x, \text{LambdaCalculus}) \Rightarrow \text{Pass}(x, NLP))\text{David} \rhd$ $\text{Study}(\text{David}, \text{LambdaCalculus}) \Rightarrow \text{Pass}(\text{David}, \text{NLP})$

Compositional Semantics

Given lambda calculus (with beta-reduction), compositionality, and context free grammars that pair semantic formulae with syntactic ones, it is possible to compute semantic representations in FOL from syntactic parses

Compositionality is the notion that the meaning of a whole is a simple function of the meanings of its parts. Language is often, though not always, compositional. **Certain frameworks allow computers to build up semantic representations from syntactic representations by assuming compositionality**.

The Relationship between Syntax and Semantics Can Be Very General

Phrase	Semantics
eat some rice	$\lambda y.\exists x.Rice(x) \wedge Eat(y,x)$
drink some water	$\lambda y.\exists x.Water(x) \wedge Drink(y,x)$
praise some friends	$\lambda y.\exists x.Friend(x) \wedge Praise(y,x)$
ridicule some lawyers	$\lambda y.\exists x.Lawyer(x) \wedge Ridicule(y,x)$

This is a consequence of compositionality.

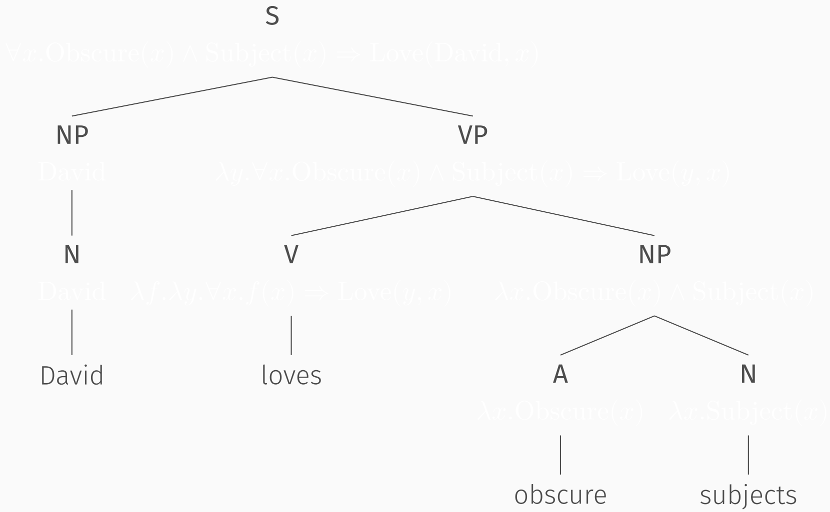
Syntactic Rules Can Be Paired with Smeantic Rules

SYNTAX

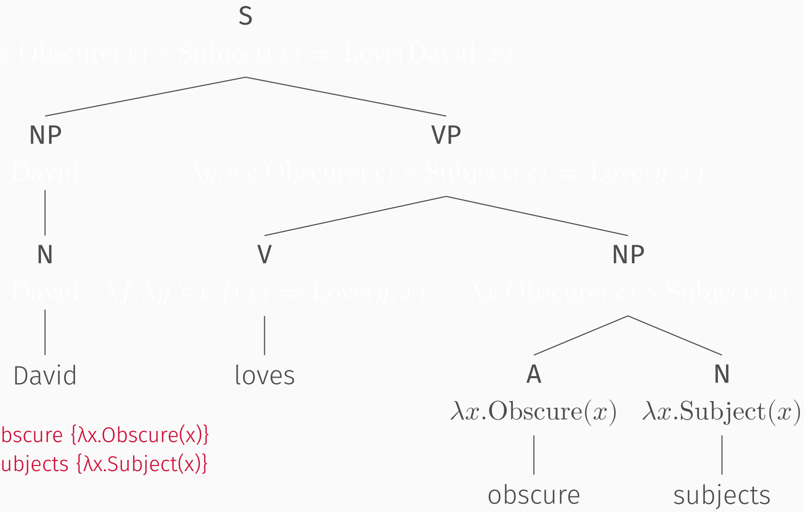
SEMANTICS

$NP \rightarrow A\ N$	$\{\lambda x.A.\text{sem}(x)$	$\cap\ N.\text{sem}(x)\}$
	A.sem refers to the lambda term associated with the syntactic node A	**N.sem(x)** refers to β-reduction of the lambda term N.sem by applying it to x

N → David {David}
V → loves {$\lambda f.\lambda y.\forall x.f(x) \Rightarrow Love(y, x)$}
A → obscure {$\lambda x.Obscure(x)$}
N → subjects {$\lambda x.Subject(x)$}
NP → N { N.sem }
NP → A N {$\lambda x.$ A.sem(x) $\wedge$ N.sem(x)}
VP → V NP { V.sem(NP.sem) }
S → NP VP { VP.sem(NP.sem) }

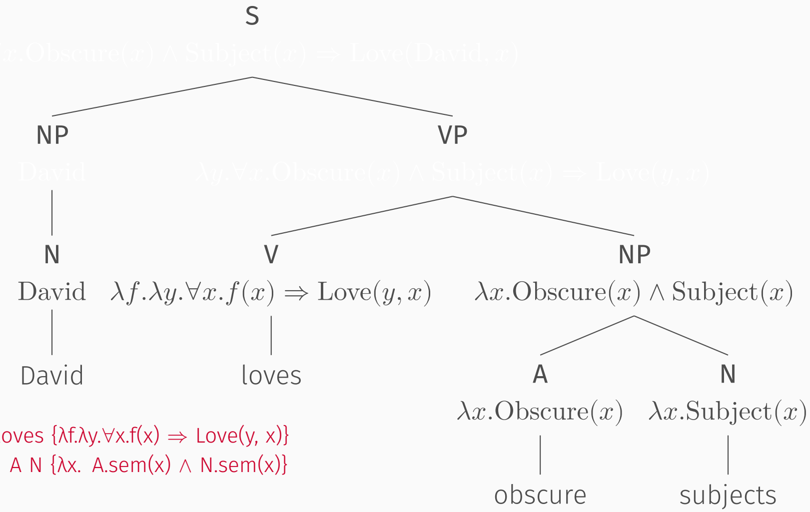

S
NP
VP
N
V
NP
David
loves
A
N
obscure
subjects

Building Semantic Representations Compositionally



A → obscure {λx.Obscure(x)}
N → subjects {λx.Subject(x)}

Building Semantic Representations Compositionally



V → loves {λf.λy.∀x.f(x) ⇒ Love(y, x)}
NP → A N {λx. A.sem(x) ∧ N.sem(x)}

S

$\forall x.\text{Obscure}(x) \land \text{Subject}(x) \Rightarrow \text{Love}(\text{David}, x)$

NP

David

N

David

David

VP

$\lambda y.\forall x.\text{Obscure}(x) \land \text{Subject}(x) \Rightarrow \text{Love}(y, x)$

V

$\lambda f.\lambda y.\forall x.f(x) \Rightarrow \text{Love}(y, x)$

loves

NP

$\lambda x.\text{Obscure}(x) \land \text{Subject}(x)$

A

$\lambda x.\text{Obscure}(x)$

obscure

N

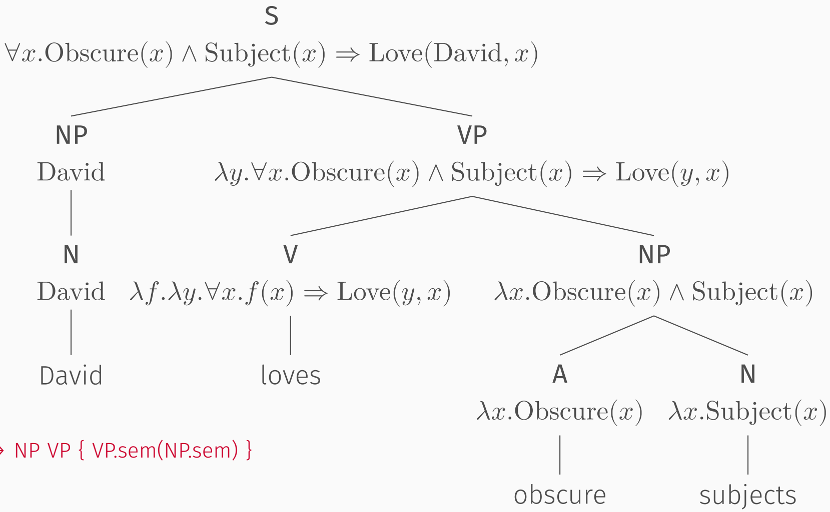
$\lambda x.\text{Subject}(x)$

subjects

V → loves {λf.λy.∀x.f(x) ⇒ Love(y, x)}

NP → N { N.sem }

Building Semantic Representations Compositionally



S → NP VP { VP.sem(NP.sem) }

Semantic Roles

Five Sentences with Semantic Roles Held Constant

- David tossed **the exams** from *Pausch Bridge* to THE HILLSIDE BELOW.
- **The exams** were tossed by David from *Pausch Bridge* to THE HILLSIDE BELOW.
- It was David who tossed **the exams** from *Pausch Bridge* to THE HILLSIDE BELOW.
- **The exams** were thrown from *Pausch Bridge*.
- **The exams** were thrown to THE HILLSIDE BELOW.

Traditional Semantic Roles

- In the linguistics literature, one sees a number of common terms for semantic roles
    - Agent
    - Patient
    - Theme
    - Force
    - Experiencer
    - Stimulus
    - Recipient
    - Source
    - Goal
    - etc.

  These have their place, and are useful to know if you want to understand what a semantic role is, but are not widely used in NLP In NLP, we tend to use finer-grained (and sometimes cryptically named) semantic role labels

Traditional Semantic Roles: Examples

David tossed **the exams** from *Pausch Bridge* to THE HILLSIDE BELOW.

David	AGENT
the exams	PATIENT
Pausch Bridge	SOURCE
THE HILLSIDE BELOW	GOAL

Semantic Role Labeling

Semantic Role Labeling: The Task

Input: a sentence, paragraph, or document.

Output: for each predicate*, labeled spans identifying each of its arguments.

*Predicates are sometimes identified in the input, sometimes not.

Propbank: an SRL Resource

- Corpus (PTB) with propositions annotated
    - Predicates (verbs)
    - Arguments (semantic roles)
- Semantic roles are Arg0, Arg1, etc., each with a description
    - Arg0 is typically the most agent-like argument
    - Labels for other arguments are somewhat arbitrary

“Agree” in PropBank

- arg0: agreer
- arg1: proposition
- arg2: other entity agreeing

- The group agreed it wouldn’t make an offer.
- Usually John agrees with Mary on everything.

“Fall (move downward)” in PropBank

- arg1: logical subject, patient, thing falling
- arg2: extent, amount fallen
- arg3: starting point
- arg4: ending point
- argM-loc: medium

- Sales fell to $251.2 million from $278.8 million.
- The average junk bond fell by 4.2%.
- The meteor fell through the atmosphere, crashing into Cambridge.

FrameNet: another SRL Resource

- A **semantic frame** is a schematic representation of a situation involving various participants, and other conceptual roles
- In **FrameNet**, frames—not verbs—are first-class citizens
  - To a first approximation, verbs that relate to the same situation belong to the same frame
  - Roles are given fine-grained labels that are specific to the frame, but not the verb
  - Frames can center around words other than verbs

The Frame change_position_on_a_scale

Core roles	
ATTRIBUTE	scalar property that the ITEM possesses
DIFFERENCE	distance by which an ITEM changes its position
FINAL_STATE	ITEM's state after the change
FINAL_VALUE	position on the scale where ITEM ends up
INITIAL_STATE	ITEM's state before the change
INITIAL_VALUE	position on the scale from which the ITEM moves
ITEM	entity that has a position on the scale
VALUE_RANGE	portion of the scale along which values of ATTRIBUTE fluctuate
Some non-core roles ...	
DURATION	length of time over which the change occurs
SPEED	rate of change of the value
GROUP	the group in which an ITEM changes the value of an ATTRIBUTE

“Triggers” for the change_position Frame

- **Verbs**: advance, climb, decline, decrease, diminish, dip, double, drop, dwindle, edge, explode, fall, fluctuate, gain, grow, increase, jump, move, mushroom, plummet, reach, rise, rocket, shift, skyrocket, slide, soar, swell, swing, triple, tumble
- **Nouns**: decline, decrease, escalation, explosion, fall, fluctuation, gain, growth, hike, increase, rise, shift, tumble
- **Adverb**: increasingly

Examples from FrameNet: change position on a scale

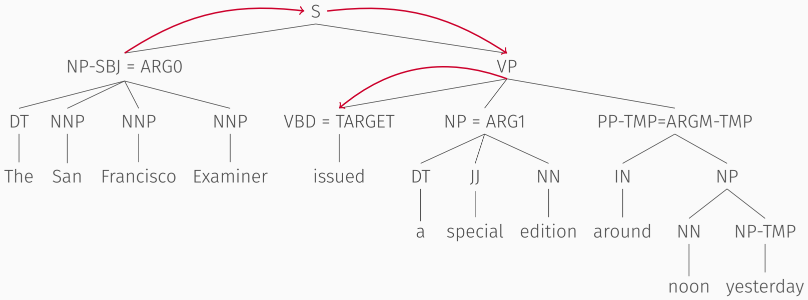
- Item: I fear **this service** will DIMINISH in quality.
- Initial and final values: Microsoft shares FELL **from 12 3/8 to 7 5/8**.
- Initial state: Diesels have INCREASED **from having a 20% market share in 1995 to just over 30%** in 2004.
- Attribute: Oil ROSE **in price** by 2%.
- Difference: Oil ROSE in price **by 2%**.
- Correlated variable: The amount of power INCREASES **with the frequency of the laser**.

An Old-Timey Approach to SRL

In some sense, SRL is a classification task.

- **Classes**: roles defined by the event type
- **Things to be classified**: the entities (encoded as noun phrases) participating in the event
- **Features**: linguistic properties of the noun phrases, path features based on the relative position of the predicate/verb and the NP, etc.

Path Features Represent the Path from Predicate to Target



Other Features used for semantic role labeling

For a phrase that you want to label, e.g., *The San Francisco Examiner*

- Phrase type: *The San Francisco Examiner* is a noun phrase
- The predicate: *issued*
- The head of the phrase: *Examiner*
- Part of speech of the head: NNP
- The voice: *issued* is in the active voice
- The position: *The San Francisco Examiner* is before *issued*
- How many arguments does the predicate need? *Issued* needs a subject and an object
- Named Entity Type: *The San Francisco Examiner* is an ORG (organization)
- The first and last words: *The* and *Examiner*

SRL and the Modern Day

In the last five years, it has been shown that traditional, linguistically informed, SRL models are not competitive with modern, end-to-end neural models (He, et al. 2017). Modern approaches treat SRL as more of a sequence labeling problem and can leverage the architectures that have been developed for such tasks.

However, other research (as by CMU's Emma Strubell) have shown that newer syntactically-aware SRL systems may be more efficient and may generalize better than the end-to-end models that are considered SOTA.

We will talk more about Semantic Role Labeling in a Subsequent Lecture

Slot-Filler Representations

When FOL is too Powerful, Try Slot-Filler Representations

- What if you want semantics to be as simple as filling out a form?

  Having

  Haver: **David**
  HadThing: **Car**

- This representations are computationally more tractable than FOL and can be represented with general-purpose data structures (e.g., hash maps/associative arrays/dictionaries)
- If you think this representation looks kind of like PropBank, you're not wrong!

PropBank and FrameNet are Slot-Filler Compatible

- Both Propbank and FrameNet define frames
    - Each frame corresponds to a verb (PropBank) or a “situation” (FrameNet)
    - The slots (fields) correspond to semantic roles within the corresponding type of event
    - The fillers are the entities that participate in the event by performing those roles
    - In some slot-filler representations, fillers can also be events or relations

Questions?