Artificial Intelligence EDAP01

Lecture 10: Semantic Technology

Pierre Nugues

Pierre.Nugues@cs.lth.se
http://cs.lth.se/pierre_nugues/

February 21st, 2025

Semantics

Started with Bréal's work, *Essai de sémantique, science des significations*, 1897, available here:
https://gallica.bnf.fr/ark:/12148/bpt6k50474n.image
Many possible definitions.
In this lecture:

- In natural language processing, a fuzzy concept corresponding roughly to the interface between language and the world;
- The "semantic" web: A graph of objects.

Words and Meaning

Words form an interface to the world and parts of speech are traditionally related to rough categories:

- Common nouns to objects and
- Verbs to actions

Semantics is the study of (nonexhaustive):

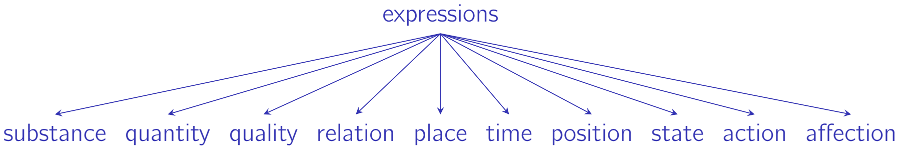
- Classes of words: Groups of words having the same meaning
- Definition: What is a meal? What is table?
- Representation: Relations between the words in a sentence;
- Reasoning: The meal is on the table. Is it cold?

Categories of Words

Expressions, which are in no way composite, signify substance, quantity, quality, relation, place, time, position, state, action, or affection. To sketch my meaning roughly, examples of substance are 'man' or 'the horse', of quantity, such terms as 'two cubits long' or 'three cubits long', of quality, such attributes as 'white', 'grammatical'. 'Double', 'half', 'greater', fall under the category of relation; 'in the market place', 'in the Lyceum', under that of place; 'yesterday', 'last year', under that of time. 'Lying', 'sitting', are terms indicating position, 'shod', 'armed', state; 'to lance', 'to cauterize', action; 'to be lanced', 'to be cauterized', affection.

Aristotle, Categories, IV. (trans. E. M. Edghill)

Representation of Categories



Classes

Organize the words in relation to the world:

- Synonymy/Antonymy
- Polysemy
- Hyponyms/Hypernyms `is_a(tree, plant), life form, entity`
- Meronyms/Holonyms `part_of(leg, table)`
- Grammatical cases: [$_{nominative}$ *I*] *broke* [$_{accusative}$ *the window*] [$_{ablative}$ *with a hammer*]
- Semantic cases: [$_{actor}$ *I*] *broke* [$_{object}$ *the window*] [$_{instrument}$ *with a hammer*]
- Case ambiguity (*The window broke*/ *I broke the window*)

Definitions

Short texts describing a word:

- A **genus** or superclass using a hypernym.
- Specific attributes to differentiate it from other members of the superclass. This part of the definition is called the *differentia specifica*.

bank (1.1): **a land** sloping up along each side of a canal or a river.

hedgehog: **a small animal** with stiff spines covering its back.

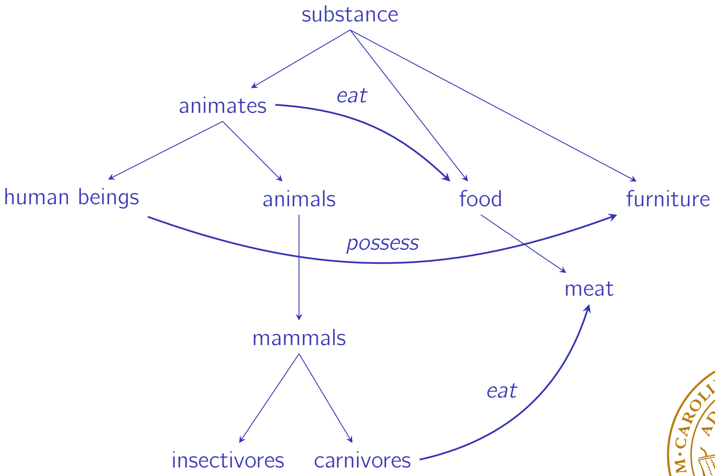
waiter: **a person** employed to serve customers at their table in a restaurant, etc.

Lexical Database

```
%% is_a(?Word, ?Hypernym)
is_a(hedgehog, insectivore).
is_a(cat, feline).
is_a(feline, carnivore).
is_a(insectivore, mammal).
is_a(carnivore, mammal).
is_a(mammal, animal).
is_a(animal, animate_being).

hypernym(X, Y) :- is_a(X, Y).
hypernym(X, Y) :- is_a(X, Z), hypernym(Z, Y).
```

Semantic Networks



An Example: WordNet

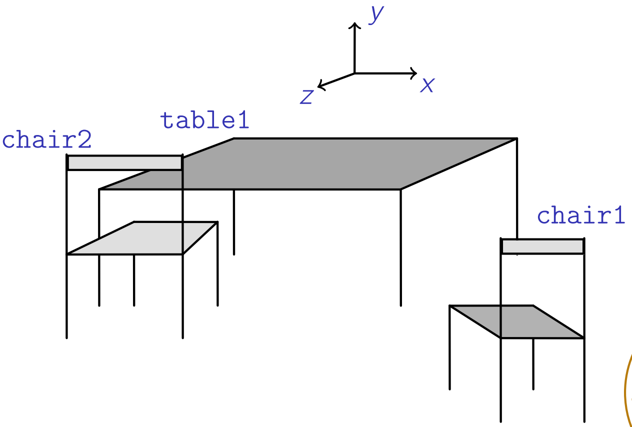
Nouns	hyponyms/hypernyms
	synonyms/antonyms
	meronyms
Adjectives	synonyms/antonyms
	relational fraternal –> brother
Verbs	Semantic domains (body function, change, communication, perception, contact, motion, creation, possession, competition, emotion, cognition, social interaction, weather)
	Synonymy, Antonymy: (rise/fall, ascent/descent, live/die)
	"Entailment": succeed/try, snore/sleep

URL: `https://wordnet.princeton.edu/`

The State of Affairs

Semantics is used to describe the state of affairs

Two people at a table, Pierre and Socrates, and a robot waiter.



Formal Semantics

Its goal is to:

- Represent the state of affairs.
- Translate phrases or sentences such as *The robot brought the meal* or *the meal on the table* into logic formulas
- Solve references: Link words to real entities
- Reason about the world and the sentences.

A way to represent things and relations is to use first-order predicate calculus (FOPC) and predicate–argument structures

Predicates

Constants:

```
% The people:
  'Socrates'.
  'Pierre'.

% The chairs:
  chair1.     % chair #1
  chair2.     % chair #2

% The unique table:
  table1.     % table #1
```

Predicates to encode properties:

```
person('Pierre').
person('Socrates').

object(table1).
object(chair1).
object(chair2).

chair(chair1).
chair(chair2).
table(table1).
```

Predicates to encode relations:

```
in_front_of(chair1, table1).
```

Prolog

Prolog is a natural tool to do first-order predicate calculus

- Things, either real or abstract, are mapped onto constants – or atoms: `'Socrates'`, `'Pierre'`, `chair1`, `chair2`.
- Predicates can encode properties: `person('Pierre')`, `person('Socrates')`, `object(table1)`, `object(chair1)`.
- Predicates can encode relations: `in_front_of(chair1, table1)`, `on('Pierre', table1)`.
- Variables unify with objects

Querying the State of Affairs

```
Constants:
?- table(chair1).
false.
?- chair(chair2).
true.
```

Variables:

```
?- chair(X).
X = chair1;
X = chair2
```

Conjunctions:

```
?- chair(X), table(Y), in_front_of(X, Y).
X = chair1, Y = table1
```

Semantic Parsing and Logical Forms

Semantic parsing links words and sentences to logic.
A semantic parser maps sentences onto predicate-argument structures (or logical forms)

I would like to book a late flight to Boston

```
would(like_to(i,
   book(i,
       np_pp(a(late(flight)),
           X^to(X, boston)))))
```

Semantic Interpretation

Question:

What is the earliest flight from Boston to Atlanta?

Modeling a flight from Boston to Atlanta:

$$\exists x(\mathit{flight}(x) \wedge \mathit{from}(x, \mathit{Boston}) \wedge \mathit{to}(x, \mathit{Atlanta}) \wedge \exists y(\mathit{time}(y) \wedge \mathit{departs}(x, y)))$$

Finding the earliest flight:

$$\arg\min_{y} \exists x(\mathit{flight}(x) \wedge \mathit{from}(x, \mathit{Boston}) \wedge \mathit{to}(x, \mathit{Atlanta}) \wedge \\ \exists y(\mathit{time}(y) \wedge \mathit{departs}(x, y)))$$

The Spoken Language Translator (SLT) is an example of application.
It uses the logical form as a universal representation, independent from the language.
It converts sentences from and to this representation.

Semantics and Reasoning (I)

A semantic representation is necessary to link automated reasoning to language.

Consider the sentence:

The caterpillar ate the hedgehog.

Is it true?

Semantics and Reasoning (II)

The caterpillar ate the hedgehog.

First you need a representation:
In logic:

$$\exists(X, Y), caterpillar(X) \land hedgehog(Y) \land ate(X, Y).$$

With Prolog:

```
?- is_a(X, caterpillar), is_a(Y, hedgehog), ate(X, Y).
```

Semantics and Reasoning (III)

Reasoning (inference):
You would need a rule to describe common sense:

```
ate(X, Y) :- is_a(X, X_c), is_a(Y, Y_c), predator(X_c, Y_c).
```

Your sentence would be untrue because the query:

```
?- predator(X, hedgehog).
X = foxes, eagles, car drivers, ...
```

but no caterpillar.

Resolving References: exists

We can apply generic conversion to quantifiers, for instance with:

A hedgehog has a nest

```
a(X, hedgehog(X), a(Y, nest(Y), have(X, Y)).

?- hedgehog(X), a(Y, nest(Y), have(X, Y)).

exists(X, Property1, Property2) :-
  Property1,
  Property2,
  !.
```

Resolving References: all

All hedgehogs have a nest

```
all(X, hedgehog(X), a(Y, nest(Y), have(X, Y)).
```

There is no hedgehog, which has no nest

```
all(X, Property1, Property2) :-
  \+
  (Property1,
  \+ Property2),
  Property1,
  !.
```

Modeling Information with a Graph Language

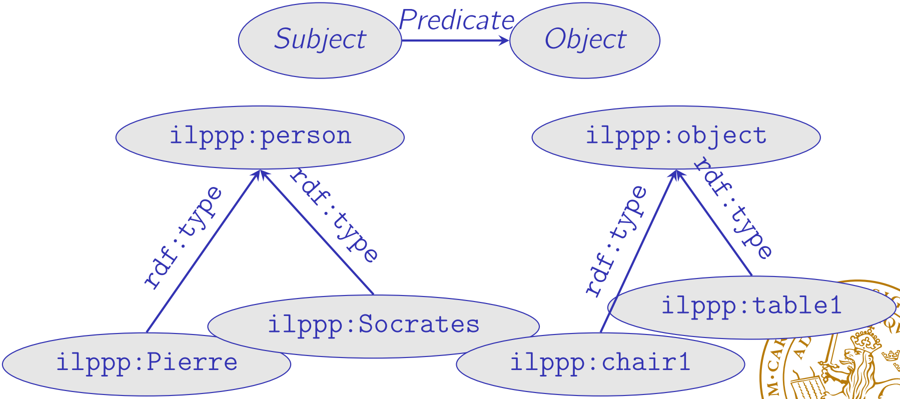
To represent the state of affairs, a machine needs a computer language.
Here we will use a graph to model information, for instance such as:

- Things, either real or abstract: `'Socrates'`, `'Pierre'`, `chair1`, `chair2`.
- Properties: `person('Pierre')`, `person('Socrates')`, `object(table1)`, `object(chair1)`.
- Relations: `in_front_of(chair1, table1)`, `on('Pierre', table1)`.

The Resource Description Framework (RDF) from the semantic web is a data model created for this.
Equivalent to Prolog, but way more verbose.

RDF Triples

RDF is a popular graph format to encode knowledge.
It consists of triples:



RDF

RDF has many syntactic variants: XML, N3, Turtle

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix ilppp: <http://cs.lth.se/nlp/02/book#>.

ilppp:Pierre rdf:type ilppp:person.
ilppp:Socrates rdf:type ilppp:person.

ilppp:table1 rdf:type ilppp:object.
ilppp:chair1 rdf:type ilppp:object.
ilppp:chair2 rdf:type ilppp:object.

ilppp:chair1 ilppp:in_front_of ilppp:table1.
ilppp:Pierre ilppp:on ilppp:table1.
```

RDF Properties

Property name	domain	range
rdfs:isDefinedBy	rdfs:Resource	rdfs:Resource
rdf:subject	rdf:Statement	rdfs:Resource
rdf:predicate	rdf:Statement	rdf:Property
rdf:object	rdf:Statement	not specified
rdf:type	rdfs:Resource	rdfs:Class
rdfs:member	rdfs:Container	not specified
rdfs:subClassOf	rdfs:Class	rdfs:Class
rdf:value	rdfs:Resource	not specified
rdfs:subPropertyOf	rdf:Property	rdf:Property
rdfs:comment	rdfs:Resource	rdfs:Literal
rdfs:label	rdfs:Resource	rdfs:Literal
rdfs:domain	rdf:Property	rdfs:Class
rdfs:range	rdf:Property	rdfs:Class
rdfs:seeAlso	rdfs:Resource	rdfs:Resource

DBpedia, Yago, Wikidata, and Freebase

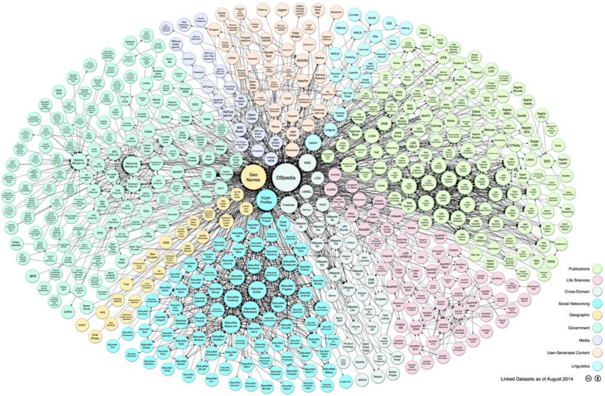
Graph databases consisting of billions of RDF triples.
Coming from a variety of sources such as Wikipedia infoboxes:

```
{{Infobox settlement
| official_name          = Busan Metropolitan City
...
| area_total_km2         = 767.35
...
| population_total       = 3,525,913
...
}}
```

DBpedia: The result of a systematic triple extraction from infoboxes

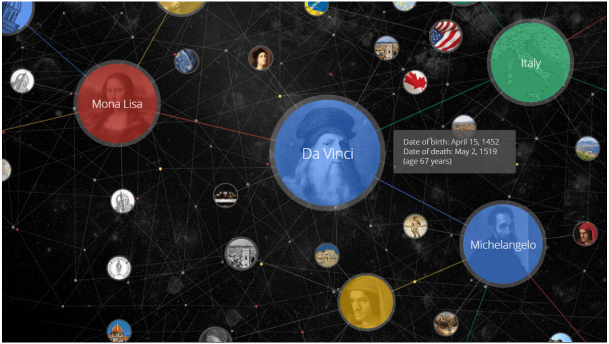
```
dbpedia:Busan foaf:name "Busan Metropolitan City"@en .
dbpedia:Busan dbo:populationTotal "3525913".
dbpedia:Busan dbo:areaTotal "7.6735E8" .
```

Graph Visualization



URL: `https://lod-cloud.net/`

Google Knowledge Graph



URL: https://www.youtube.com/watch?v=TJfrNo3Z-DU
URL: https://www.youtube.com/watch?v=mmQl6VGvX-c
URL: https://developers.google.com/knowledge-graph/

RDF and SPARQL

SPARQL: A query language for RDF

In many ways, very similar to Prolog.

Prolog:

```
?- object(X), object(Y), in_front_of(X, Y).
X = chair1,
Y = table1.
```

SPARQL:

```
SELECT ?x ?y
WHERE
{
  ?x rdf:type ilppp:object.
  ?y rdf:type ilppp:object.
  ?x ilppp:in_front_of ?y
}
```

SPARQL Endpoint for DBpedia

A network service accepting SPARQL queries such as:

```
prefix dbo: <http://dbpedia.org/ontology/>
prefix foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?entity ?population
WHERE
{
  ?entity foaf:name "Busan Metropolitan City"@en.
  ?entity dbo:populationTotal ?population.
}
```

Address of the DBpedia endpoint: `http://dbpedia.org/sparql`

DBpedia

The DBpedia query returns:

Variables	entity	population
Values	`http://dbpedia.org/resource/Busan`	3525913

where `http://dbpedia.org/resource/Busan` or `dbpedia:Busan` is a unique entity name based on the Wikipedia web addresses (URI nomenclature).

Wikidata

Wikidata provides another endpoint based on Wikipedia data:
`https://query.wikidata.org`

```
SELECT ?entity ?population
WHERE
{
  ?entity rdfs:label "Busan"@en .
  ?entity wdt:P1082 ?population.

}
```

We can program this extraction
Programs: `https://`
`//github.com/pnugues/ilppp/tree/master/programs/ch14/python`

Querying Wikidata

Extracting cat names

```
query = '''PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?item ?itemLabel WHERE {
    ?item wdt:P31 wd:Q146 .

    OPTIONAL {
        ?item rdfs:label ?itemLabel
        filter (lang(?itemLabel) = "en") .
    }
}'''
```

Querying Wikidata (II)

Sending the query:

```
url = 'https://query.wikidata.org/bigdata/namespace/wdq/sparql
data = requests.get(url,
          params={'query': query, 'format': 'json'}).json()

cats = []
for item in data['results']['bindings']:
    cats.append({
        'id': item['item']['value'],
        'name': item.get('itemLabel', {}).get('value')})

df = pd.DataFrame(cats)
print(len(df))
print(df)
```

Discourse Entities

Mentions (or referring expressions)	**Discourse entities (or referents)**	**Logic properties**
Susan, she, her	`'Susan'`	`'Susan'`
Lyn, she	`'Lyn'`	`'Lyn'`
A Ferrari	`X`	`ferrari(X)`
A lot of trophies	`E`	$E \subset \{X \mid trophy(X)\}$

In RDF, entities can be resources and common nouns can be classes.

Reference and Named Entities

Named entities are entities uniquely identifiable by their name.

Some definitions/ clarifications:

- Named entity recognition (NER): a partial parsing task;
- Reference resolution for named entities: find the entity behind a mention, here a name.

Words	POS	Groups	Named entities
U.N.	NNP	I-NP	I-ORG
official	NN	I-NP	O
Ekeus	NNP	I-NP	I-PER
heads	VBZ	I-VP	O
for	IN	I-PP	O
Baghdad	NNP	I-NP	I-LOC
.	.	O	O

As it is impossible to set a physical link between a real-life object and its mention, we use unique identifiers or tags in the form of URIs instead (from Wikidata,DBpedia, Yago).

Mentions of Named Entities are Ambiguous

Cambridge: England, Massachusetts, or Ontario?

Given the text (from Wikipedia):

> *One of his translators, Roy Harris, summarized* **Saussure***'s contribution to linguistics and the study of language in the following way...*

Which Saussure? *Saussure* has 11 entries in Wikipedia:

- *Ferdinand de Saussure*:
  - Wikidata: `http://www.wikidata.org/wiki/Q13230`
  - DBpedia: `http://dbpedia.org/resource/Ferdinand_de_Saussure`
- *Henri de Saussure*: `http://www.wikidata.org/wiki/Q123776`
- *René de Saussure*: `http://www.wikidata.org/wiki/Q13237`

Göran Persson in Swedish

In Wikipedia, at least four entities can be linked to the name *Göran Persson*:

1. **Göran Persson** (född 1949), socialdemokratisk partiledare och svensk statsminister 1996–2006 (Q53747)
2. **Göran Persson** (född 1960), socialdemokratisk politiker från Skåne (Q5626648)
3. Göran Persson (militär), svensk överste av 1:a graden
4. **Göran Persson** (musiker), svensk proggmusiker (Q6042900)
5. Göran Persson (litterär figur), överkonstapel i 1930-talets Lysekil
6. Göran Persson (skulptör) (född 1956), konstnär representerad i bl.a. Karlskoga
7. **Jöran Persson**, svensk ämbetsman på 1500-talet (Q2625684)

Named Entities and Linked Data

Graph databases are popular devices used to represent named entities, especially the resource description framework (RDF).
Entities are assigned unique resource identifiers (URIs) similar to URLs (as in HTTP addresses) and can be linked to other data sources (Linked data)

Collecting Entity-Mention Pairs from Wikipedia

Wikipedia has a mark up that enables an editor to link a word or phrase to a page:

- `[[Ferdinand_de_Saussure|Saussure]]` or
- `[[target or link|text or label or anchor]]`

In our case, it is an association between a mention and an entity:
`[[Entity|Mention]]`
All the links can be extracted from a wikipedia dump to derive two probabilities:

- The probability of a mention given an entity, how we name things: $P(M|E)$
- The probability of a entity given an mention, the ambiguity of a mention: $P(E|M)$

Lexicons

Words are ambiguous: A same form may have more than one entry and sense.

The *Oxford Advanced Learner's Dictionary* (OLAD) lists five entries for *bank*:

1. *noun*, raised ground
2. *verb*, turn
3. *noun*, organization
4. *verb*, place money
5. *noun*, row or series

and five senses for the first entry.

Cultural Relativity of the Sense (Hjelmslev)

<table>
<tr><th>French</th><th>German</th><th>Danish</th></tr>
<tr><td>arbre</td><td>Baum</td><td rowspan="2">Træ</td></tr>
<tr><td rowspan="2">bois</td><td>Holz</td></tr>
<tr><td rowspan="2">Wald</td><td rowspan="2">Skov</td></tr>
<tr><td>forêt</td></tr>
</table>

<table>
<tr><th>French</th><th>Welsh</th></tr>
<tr><td rowspan="2">vert</td><td>gwyrdd</td></tr>
<tr><td rowspan="3">glas</td></tr>
<tr><td>bleu</td></tr>
<tr><td rowspan="2">gris</td></tr>
<tr><td rowspan="2">llwyd</td></tr>
<tr><td>brun</td></tr>
</table>

Sense Tagging Using the Oxford Advanced Learner's Dictionary (OALD)

Sentence: *The patron ordered a meal*

Words	Definitions	Sense
The patron	**Correct sense:** A customer of a shop, restaurant, theater	**1.2**
	Alternate sense: A person who gives money or support to a person, an organization, a cause or an activity	**1.1**
ordered	**Correct sense:** To request somebody to bring food, drink, etc in a hotel, restaurant etc.	**2.3**
	Alternate senses: To give an order to somebody	**2.1**
	To request somebody to supply or make goods, etc.	**2.2**
	To put something in order	**2.4**
a meal	**Correct sense:** The food eaten on such occasion	**1.2**

Beyond Words: Predicates and Arguments

Dictionaries store information about how words combine with other words to form larger structures.

This information is called valence (cf. valence in chemistry)

In the *Oxford Advanced Learner's Dictionary*, **tell**, sense 1, has the valence patterns:

tell something (to somebody) / tell somebody (something)

as in:

- *I told a lie to him*
- *I told him a lie*

Both have the same predicate–argument representation:

```
tell.01(Speaker: I, Utterance: a lie, Hearer: him)
```

Case Grammar

Verbs have semantic cases (or semantic roles):

- An Agent – Instigator of the action (typically animate)
- An Instrument – Cause of the event or object in causing the event (typically animate)
- A Dative – Entity affected by the action (typically animate)
- A Factitive – Object or being resulting from the event
- A Locative – Place of the event
- A Source – Place from which something moves,
- A Goal – Place to which something moves,
- A Beneficiary – Being on whose behalf the event occurred (typically animate)
- A Time – Time at which the event occurred
- An Object – Entity that is acted upon or that changes, the most general case.

Case Grammar: An Example

```
open(Object, {Agent}, {Instrument})
```

The door opened	Object = *door*
John opened the door	Object = *door* and Agent = *John*
The wind opened the door	Object = *door* and Agent = *wind*
John opened the door with a chisel	Object = *door*, Agent = *John*, and Instrument = *chisel*

Parsing with Cases

The waiter brought the meal to the patron

Identify the verb **bring** and apply constraints:

Case	**Type**		**Value**
Agentive	Animate	(Obligatory)	*The waiter*
Objective (or theme)		(Obligatory)	*the meal*
Dative	Animate	(Optional)	*the patron*
Time		(Obligatory)	past

FrameNet

In 1968, Fillmore wrote an oft cited paper on case grammars.

Later, he started the FrameNet project:

`http://framenet.icsi.berkeley.edu/`

Framenet is an extensive lexical database itemizing the case (or frame) properties of English verbs.

In FrameNet, Fillmore no longer uses universal cases but a set of frames – predicate argument structures – where each frame is specific to a class of words.

The *Impact* Frame

Impact:

bang.v, bump.v, clang.v, clunk.v, collide.v, collision.n, crash.v, crash.n, crunch.v, glancing.a, graze.v, hit.v, hit.n, impact.v, impact.n, plop.v, plough.v, plunk.v, run.v, slam.v, slap.v, smack.v, smash.v, strike.v, thud.v, thump.v

Frame elements:

cause, force, impactee, impactor, impactors, manner, place, result, speed, sub_location, time.

The *Revenge* Frame

15 lexical units (verb, nouns, adjectives):

avenge.v, avenger.n, get back (at).v, get_even.v, retaliate.v, retaliation.n, retribution.n, retributive.a, retributory.a, revenge.n, revenge.v, revengeful.a, revenger.n, vengeance.n, vengeful.a, and vindictive.a.

Five frame elements (FE):

Avenger, Punishment, Offender, Injury, and Injured_party.

The lexical unit in a sentence is called the target.

See also the Universal Verb Index:
`https://verbs.colorado.edu/verb-index/`

Annotation

1. [*<Avenger>* His brothers] **avenged** [*<Injured_party>* him].
2. With this, [*<Avenger>* El Cid] at once **avenged** [*<Injury>* the death of his son].
3. [*<Avenger>* Hook] tries to **avenge** [*<Injured_party>* himself] [*<Offender>* on Peter Pan] [*<Punishment>* by becoming a second and better father].

FrameNet uses three annotation levels: Frame elements, Phrase types (categories), and grammatical functions.

GFs are specific to the target's part-of-speech (i.e. verbs, adjectives, prepositions, and nouns).

For the verbs, three GFs: Subject (Ext), Object (Obj), Complement (Dep), and Modifier (Mod), i.e. modifying adverbs ended by −ly or indicating manner

Propbank

Semantic analysis often uses Propbank instead of Framenet because of Propbank's larger annotated corpus

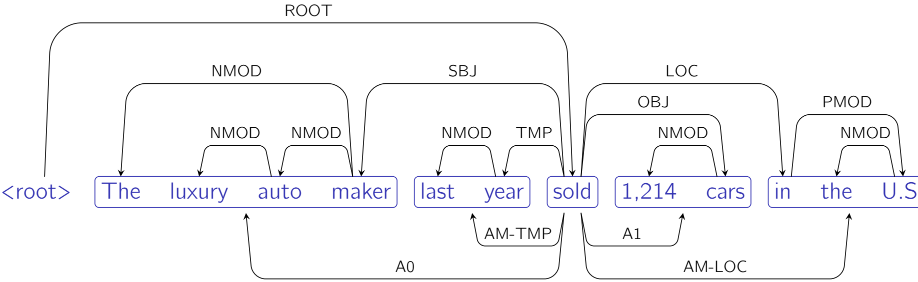
CoNLL 2008 and 2009 used Propbank for their evaluation of semantic parsers.

CoNLL annotation format of the sentence:

The luxury auto maker last year sold 1,214 cars in the U.S.

ID	Form	Lemma	PLemma	POS	PPOS	Feats	PFeats	Head	PHead	Deprel	PDeprel	FillPred	Sense	APred1	APred2
1	The	the	the	DT	DT	_	_	4	4	NMOD	NMOD	_	_	_	_
2	luxury	luxury	luxury	NN	NN	_	_	3	3	NMOD	NMOD	_	_	A1	_
3	auto	auto	auto	NN	NN	_	_	4	4	NMOD	NMOD	_	_	A1	_
4	maker	maker	maker	NN	NN	_	_	7	7	SBJ	SBJ	Y	maker.01	A0	A0
5	last	last	last	JJ	JJ	_	_	6	6	NMOD	NMOD	_	_	_	_
6	year	year	year	NN	NN	_	_	7	7	TMP	TMP	_	_	_	AM-TMP
7	sold	sell	sell	VBD	VBD	_	_	0	0	ROOT	ROOT	Y	sell.01	_	_
8	1,214	1,214	1,214	CD	CD	_	_	9	9	NMOD	NMOD	_	_	_	_
9	cars	car	car	NNS	NNS	_	_	7	7	OBJ	OBJ	_	_	_	A1
10	in	in	in	IN	IN	_	_	7	7	LOC	LOC	_	_	_	AM-LOC
11	the	the	the	DT	DT	_	_	12	12	NMOD	NMOD	_	_	_	_
12	U.S.	u.s.	u.s.	NNP	NNP	_	_	10	10	PMOD	PMOD	_	_	_	_

Alternative Visualization



	The	luxury	auto	maker	last	year	sold	1,214	cars	in	the	U.S.
maker.01		A1		A0								
sell.01	A0				AM-TMP			A1		AM-LOC		

Events

Research on the representation of time, events, and temporal relations dates back the beginning of logic.
It resulted in an impressive number of formulations and models.
A possible approach is to **reify** events: turn them into objects, quantify them existentially, and connect them using predicates

John saw Mary in London on Tuesday

$$\exists \varepsilon [saw(\varepsilon, John, Mary) \wedge place(\varepsilon, London) \wedge time(\varepsilon, Tuesday)],$$

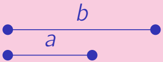
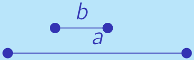
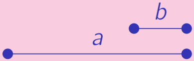
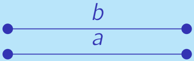
where ε represents the event.

Event Types

Events are closely related to sentence's main verbs Different classifications have been proposed to associate a verb with a type of event, Vendler (1967):

- A state – a permanent property or a usual situation (e.g. *be*, *have*, *know*, *think*);
- An achievement – a state change, a transition, occurring at single moment (e.g. *find*, *realize*, *learn*);
- An activity – a continuous process taking place over a period of time (e.g. *work*, *read*, *sleep*). In English, activities often use the present perfect *-ing*;
- An accomplishment – an activity with a definite endpoint completed by a result (e.g. *write a book, eat an apple*).

Temporal Representation of Events (Allen 1983)

#	Relations	#	Inverse relations	Graphical representations
1.	`before(a, b)`	2.	`after(b, a)`	
3.	`meets(a, b)`	4.	`met_by(b, a)`	
5.	`overlaps(a, b)`	6.	`overlapped_by(b, a)`	
7.	`starts(a, b)`	8.	`started_by(b, a)`	
9.	`during(b, a)`	10.	`contains(a, b)`	
11.	`finishes(b, a)`	12.	`finished_by(a, b)`	
13.	`equals(a, b)`			

TimeML, an Annotation Scheme for Time and Events

TimeML is an effort to unify temporal annotation, based on Allen's (1984) relations and inspired by Vendler's (1967) classification.

TimeML defines the XML elements:

- `TIMEX3` to annotate time expressions (at four o'clock),
- `EVENT`, to annotate the events (he slept),
- "signals".

The `SIGNAL` tag marks words or phrases indicating a temporal relation.

TimeML, an Annotation Scheme for Time and Events (II)

TimeML connects entities using different types of links

Temporal links, `TLINKs`, describe the temporal relation holding between events or between an event and a time.

TimeML elements have attributes. For instance, events have a tense, an aspect, and a class.

The 7 possible classes denote the type of event, whether it is a `STATE`, an instantaneous event (`OCCURRENCE`), etc.

TimeML Example

> *All 75 people on board the Aeroflot Airbus died when it ploughed into a Siberian mountain in March 1994*

(Ingria and Pustejovsky 2004):

```
All 75 people
<EVENT eid="e7" class="STATE">on board</EVENT>
<MAKEINSTANCE eiid="ei7" eventID="e7" tense="NONE" aspect="NONE"/>
<TLINK eventInstanceID="ei7" relatedToEvent="ei5"
relType="INCLUDES"/>
the Aeroflot Airbus
<EVENT eid="e5" class="OCCURRENCE" >died</EVENT>
<MAKEINSTANCE eiid="ei5" eventID="e5" tense="PAST" aspect="NONE"/>
<TLINK eventInstanceID="ei5" signalID="s2" relatedToEvent="ei6"
relType="IAFTER"/>
```

TimeML Example

> *All 75 people on board the Aeroflot Airbus died when it ploughed into a Siberian mountain in March 1994*

(Ingria and Pustejovsky 2004):

```
<SIGNAL sid="s2">when</SIGNAL>
it
<EVENT eid="e6" class="OCCURRENCE">ploughed</EVENT>
<MAKEINSTANCE eiid="ei6" eventID="e6" tense="PAST" aspect="NONE"/>
<TLINK eventInstanceID="ei6" signalID="s3" relatedToTime="t2"
relType="IS_INCLUDED"/>
<TLINK eventInstanceID="ei6" relatedToEvent="ei4"
relType="IDENTITY"/>
into a Siberian mountain
<SIGNAL sid="s3">in</SIGNAL>
<TIMEX3 tid="t2" type="DATE" value="1994-04">March 1994</TIMEX3>.
```