Language Technology
http://cs.lth.se/edan20/
Chapter 14: The Rest

Pierre Nugues

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

October 12, 2023

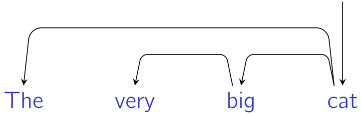
NLP Fields

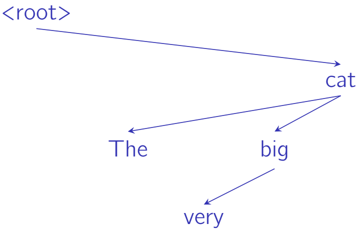
NLP has many other fields. More traditional structures:

- Syntax
- Semantics
- Entities

Dependency Grammars

Dependency grammars (DG) describe the structure in term of links





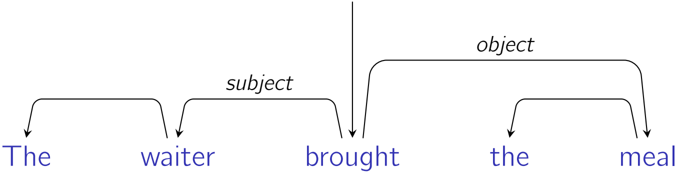
Each word has a head or “régissant” except the root of the sentence.
A head has one or more modifiers or dependents:
Cat is the head of *big* and *the*; *big* is the head of *very*.
DG can be more versatile with a flexible word order language like German, Russian, or Latin.

Dependencies and Grammatical Functions

The dependency structure generally reflects the traditional syntactic representation

The links can be annotated with grammatical function labels.

In a simple sentence, it corresponds to the subject and the object



Probably a more natural description to tie syntax to semantics

Annotation: CoNLL-U (simplified)

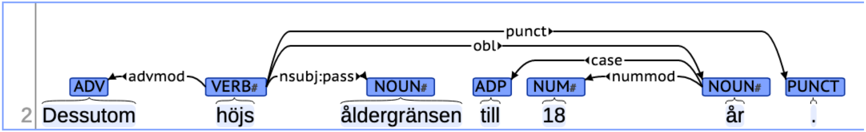
CoNLL-U is an attempt to unify the grammatical annotation across human languages.

ID	FORM	LEMMA	UPOS	HEAD	DEPREL
1	Dessutom	dessutom	ADV	2	advmod
2	höjs	höja	VERB	0	root
3	åldergränsen	åldergräns	NOUN	2	nsubj:pass
4	till	till	ADP	6	case
5	18	18	NUM	6	nummod
6	år	år	NOUN	2	obl
7	.	.	PUNCT	2	punct

Corpora available in many languages:
`https://universaldependencies.org/`

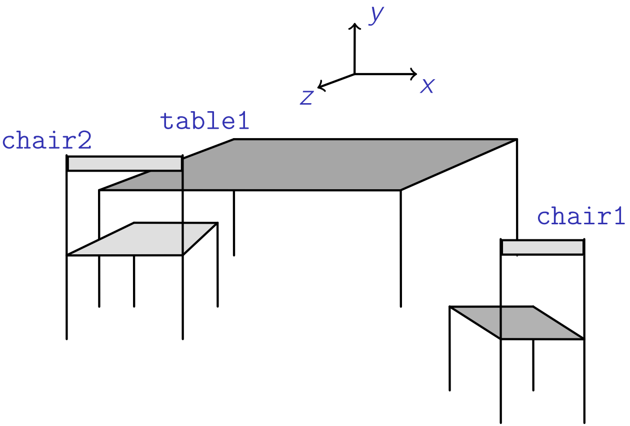
Visualizing Dependencies

Using *conllu.js* (`http://spyysalo.github.io/conllu.js/`):



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).
on('Pierre', table1).
```

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.

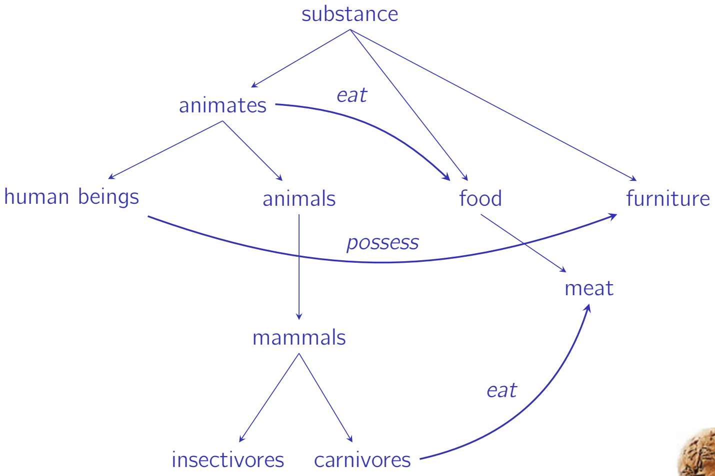
Aristotle, Categories, IV. (trans. E. M. Edghill)

Representation of Categories

expressions

- substance
- quantity
- quality
- relation
- place
- time
- position
- state
- action
- affection

Semantic Networks



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)
```

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 *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.

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].

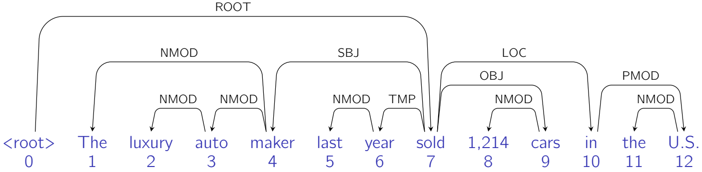
FrameNet uses three annotation levels: Frame elements, Phrase types (categories), and grammatical functions.

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

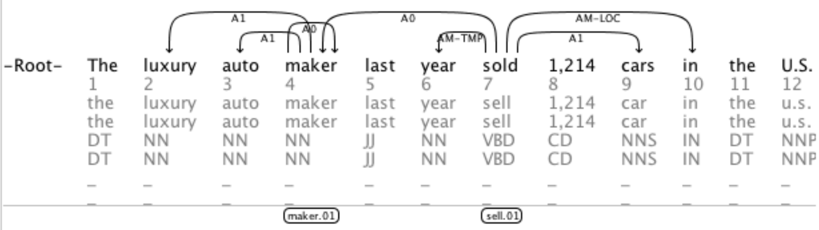
For the verbs, three GFs: Subject (Ext), Object (Obj), Complement (Dep), and Modifier (Mod), i.e. modifying adverbs ended by $-$*ly* or indicating manner

Visualizing Dependencies

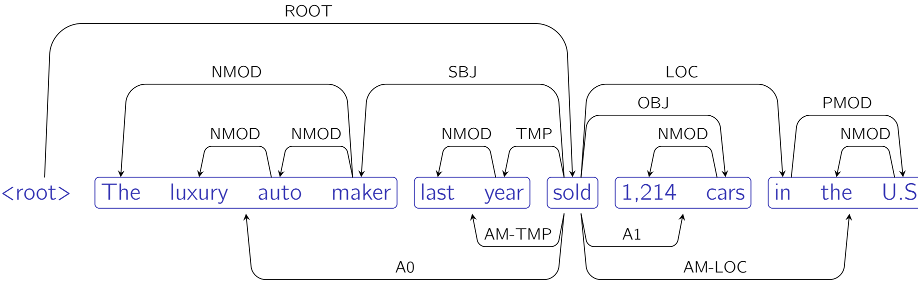
Syntactic dependencies:



Semantic dependencies (predicate–argument structures):



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		

Parsing Pipeline (Old Style)

Input sentence

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

↓ **Predicate identification**

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

maker.?? sell.??

↓ **Predicate sense disambiguation**

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

maker.01 sell.01

↓ **Argument identification**

The luxury auto <u>maker</u> last <u>year</u> **sold** 1,214 <u>cars</u> <u>in</u> the U.S.

sell.01

↓ **Argument labeling**

The luxury auto <u>maker</u> last <u>year</u> **sold** 1,214 <u>cars</u> <u>in</u> the U.S.

A0 AM-TMP sell.01 A1 AM-LOC

Semantic Parsing As a Tagging Operation

We can also apply a technique similar to that in chunking (Zhou and Xu, 2015):
Starting from the segments:

	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		

We annotate the arguments with the IOB2 tagset (Begin, Inside, Outside):

	The	luxury	auto	maker	last	year	sold	1,214	cars	in	the	U.S.
maker.01	O	B-ARG1	I-ARG1	B-ARG0	O	O	O	O	O	O	O	O
sell.01	B-ARG0	I-ARG0	I-ARG0	I-ARG0	B-TMP	I-TMP	B-V	B-ARG1	I-ARG1	B-LOC	I-LOC	I-LOC

Semantic Parsing as a Tagging Operation (II)

The annotated corpus:

	The	luxury	auto	maker	last	year	sold	1,214	cars	in	the	U.S.
maker.01	O	B-ARG1	I-ARG1	B-ARG0	O	O	O	O	O	O	O	O
sell.01	B-ARG0	I-ARG0	I-ARG0	I-ARG0	B-TMP	I-TMP	B-V	B-ARG1	I-ARG1	B-LOC	I-LOC	I-LOC

Collecting the features from Zhou and Xu (2015):

1. The input is the word sequence and the output is the tag sequence: sequence-to-sequence learning;
2. The features are similar to those used for chunking:
   - The current word;
   - The predicate (from a previous detection);
   - The predicate context (three words centered on the predicate);
   - if the current word is in the predicate context;
3. The process is repeated as many times as there are predicates in the sentence.

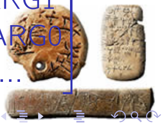
LSTMs and transformers yield even better results

Semantic Parsing as a Tagging Operation (III)

The annotated corpus:

	The	luxury	auto	maker	last	year	sold	1,214	cars	in	the	U.S.
maker.01	O	B-ARG1	I-ARG1	B-ARG0	O	O	O	O	O	O	O	O
sell.01	B-ARG0	I-ARG0	I-ARG0	I-ARG0	B-TMP	I-TMP	B-V	B-ARG1	I-ARG1	B-LOC	I-LOC	I-LOC

$$\mathbf{X} = \begin{bmatrix} \text{The} & \text{sell.01} & \text{year sold 1,214} & 0 \\ \text{luxury} & \text{sell.01} & \text{year sold 1,214} & 0 \\ \text{auto} & \text{sell.01} & \text{year sold 1,214} & 0 \\ \text{maker} & \text{sell.01} & \text{year sold 1,214} & 0 \\ \ldots & \ldots & \ldots & \ldots \\ \text{The} & \text{maker.01} & \text{auto maker last} & 0 \\ \text{luxury} & \text{maker.01} & \text{auto maker last} & 0 \\ \text{luxury} & \text{maker.01} & \text{auto maker last} & 0 \\ \text{auto} & \text{maker.01} & \text{auto maker last} & 1 \\ \ldots & \ldots & \ldots & \ldots \end{bmatrix}; \mathbf{y} = \begin{bmatrix} \text{B-ARG0} \\ \text{I-ARG0} \\ \text{I-ARG0} \\ \text{I-ARG0} \\ \ldots \\ \text{O} \\ \text{B-ARG1} \\ \text{I-ARG1} \\ \text{B-ARG0} \\ \ldots \end{bmatrix}$$



Reference and Named Entities

Named entities are entities uniquely identifiable by their name.

Some definitions/ clarifications:

- Named entity recognition (NER): a partial parsing task, see Chap. 10;
- 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`

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 a mention, the ambiguity of a mention: $P(E|M)$

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)

Disambiguation of Named Entities

Given:

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

Disambiguation is a classification problem dealing with mention-entity pairs:

Mention	Entity	Q number	T/F
Saussure	Ferdinand de Saussure	Q13230	1
Saussure	Henri de Saussure	Q123776	0
Saussure	René de Saussure	Q13237	0
...			

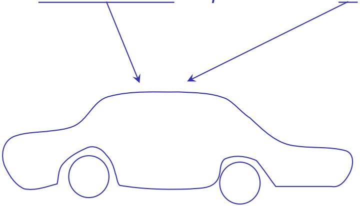
Feature vectors represent pair of mentions and entities:

- Cosine similarity between the mention context and the named entity page in Wikipedia and bag-of-word vectors of the mention context
- Training set built from Wikipedia markup:
  `[[Ferdinand_de_Saussure|Saussure]]`

Coreference

[$_{\text{entity1}}$ Garcia Alvarado], 56, was killed when [$_{\text{entity2}}$ a bomb] placed by [$_{\text{entity3}}$ urban guerrillas] on [$_{\text{entity4}}$ his vehicle] exploded as [$_{\text{entity5}}$ it] came to [$_{\text{entity6}}$ a halt] at [$_{\text{entity7}}$ an intersection] in [$_{\text{entity8}}$ downtown] [$_{\text{entity9}}$ San Salvador].

on **his vehicle** *exploded as* **it** *came to a halt*



Coreference Annotation: CoNLL 2011 simplified

0	“	“	...	-
1	Vandenberg	NNP		(8 |(0)
2	and	CC		-
3	Rayburn	NNP		(23) |8)
4	are	VBP		-
5	heroes	NNS		-
6	of	IN		-
7	mine	NN		(15)
8	,	,		-
9	”	”		-
10	Mr.	NNP		(15
11	Boren	NNP		15)
12	says	VBZ		-
13	,	,		-
14	referring	VBG		-
15	as	RB		-
16	well	RB		-
17	to	IN		-
18	Sam	NNP		(23
19	Rayburn	NNP		-
20	,	,		-
21	the	DT		-
22	Democratic	JJ		-
23	House	NNP		-
24	speaker	NN		-
25	who	WP		-
26	cooperated	VBD		-
27	with	IN		-
28	President	NNP		-
29	Eisenhower	NNP		23)
30	.	.		-

Entities and mentions:

$e_0 = \{Vandenberg\}$

$e_8 = \{Vandenberg\ and\ Rayburn\}$

$e_{15} = \{mine, Mr.\ Boren\}$

$e_{23} =$
$\{Rayburn, Sam\ Rayburn\ `,'\ the\ Democratic\ House\ speaker\ who\ cooperated\ with\ President\ Eisenhower\}$

Coreference Chains

In the MUC competitions, coreference is defined as symmetric and transitive:

- If A is coreferential with B, the reverse is also true.
- If A is coreferential with B, and B is coreferential with C, then A is coreferential with C.

It forms an equivalence class called a **coreference chain**.

The `TYPE` attribute specifies the link between the anaphor and its antecedent.

`IDENT` is the only possible value of the attribute

Other types are possible such as part, subset, etc.

Solving Coreferences: A Simplistic Method

Coreferences define a class of equivalent references

Backward search with a compatible gender and number

∼90% of the antecedents are in the current or previous sentence

Garcia Alvarado, 56, was killed when ***a bomb*** *placed by urban guerrillas*

$\boxed{2}$

on ***his vehicle*** *exploded as* ***it*** *came to a halt at an intersection in*

$\boxed{1}$

downtown San Salvador

Machine Learning to Solve Coreferences

Instead of manually engineered rules, machine learning uses an annotated corpus and trains the rules automatically.

The coreference solver (classifier)

- Considers pairs of noun phrases (NP_i, NP_j)
- Represents each pair by a feature vector.
- Decides for each pair whether it corefers or not.
- Using the transitivity property, identifies all the coreference chains in the text.

Assignments

You had these assignments:

0. Python and NumPy
1. Word indexing with regular expressions; document comparison with vector similarity;
2. Bayesian language models and prediction (autocomplete);
3. Subword tokenization, statistics, Viterbi, dynamic programming;
4. Classification with feature extraction, logistic regression and feed-forward networks, gradient descent;
5. Sequence-to-sequence classification with recurrent networks (RNN and LSTM);
6. Machine translation with transformers.

Programming interactivity and experimentation through notebooks
Familiarization with the world of research with scientific papers

Röd tråd

A few observations from Tradition:

- I hear and I forget
- I see and I remember
- I do and I understand

Disputed origin: Old Chinese proverb, attributed to Confucius, possibly from Maria Montessori

https://drandrewhuang.wordpress.com/2021/05/24/
tracing-the-origins-of-i-hear-and-i-forget-i-see-and-i-remem