Natural Language Processing

Lecture 2
Linguistics Structure and the Traditional Pipeline

2021

Linguistic structure
Representation levels
Grammars
Parsing and generation
Parsing and pipeline
Pipeline tasks

Linguistic structure

Representation levels

Linguistic structure
Representation levels
Grammars
Parsing and generation
Parsing and pipeline
Pipeline tasks

Natural languages are very complex sign systems, and their signs (words, phrases, sentences etc.) have dramatically more internal structure than ordinary symbols. Linguists typically distinguish at least the following four levels of linguistic representation in a linguistic sign:[1]

- phonological structure: the level of individual sounds, or, in written language, written symbols, letters;
- morphological structure: the level of *morphemes*, i.e., minimal meaningful linguistic units, and their organization into *words*;

[1] The discussion of representation levels and grammars in this section closely follows Marcus Kracht's Intoduction to Linguistics, which is highly recommended.

Representation levels cont.

Linguistic structure
Representation levels
Grammars
Parsing and generation
Parsing and pipeline
Pipeline tasks

- syntactic structure: the level at which words are organized into well formed sentences;
- semantic structure: the level of *meaning*, i.e., what the linguistic sign *signifies*.

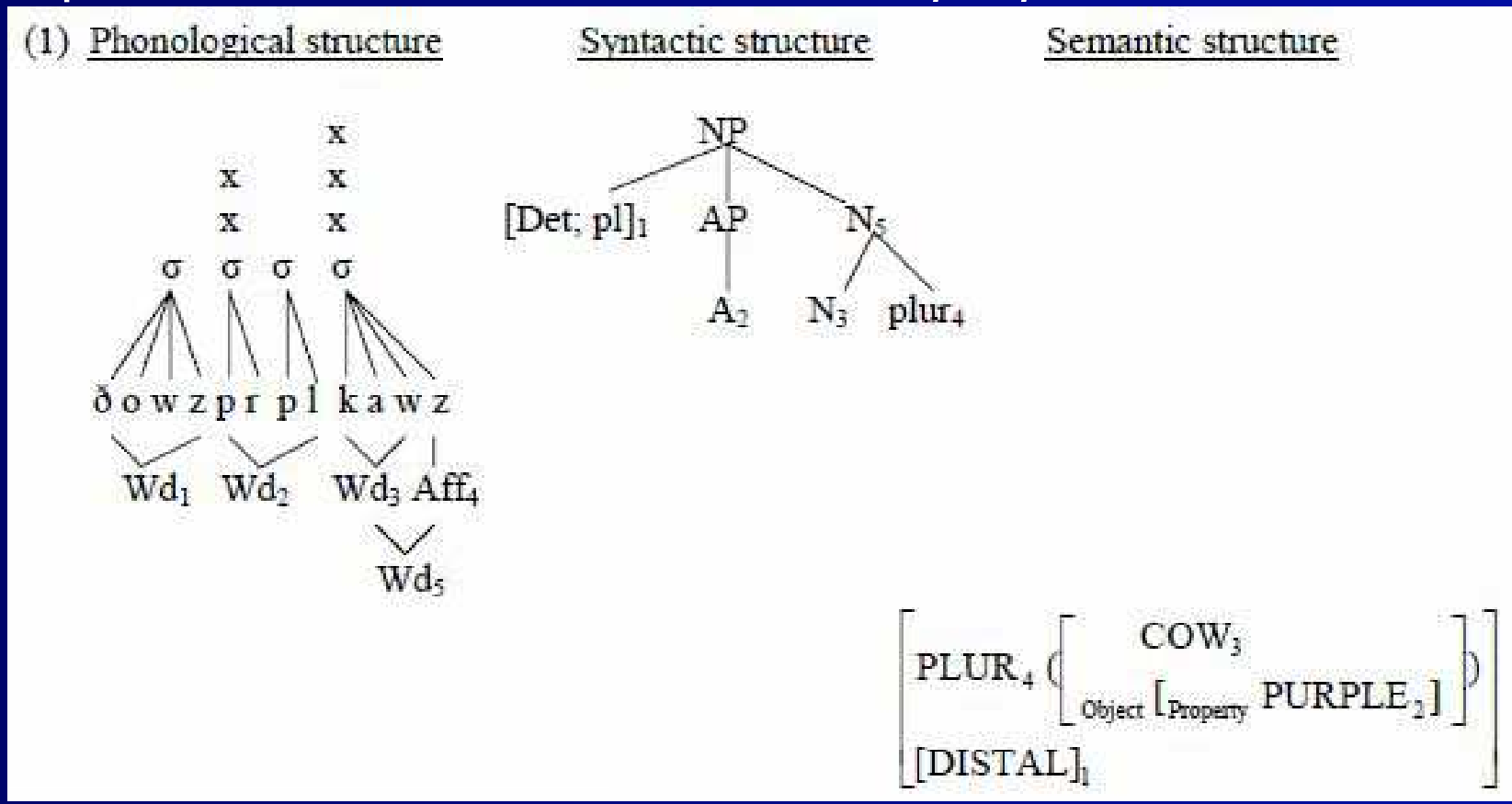
The listed representation levels do not cover all important aspects of linguistic signs:

- Semantics, at least traditionally, does not deal with the non-literal, context-dependent elements of meaning, which are left to pragmatics, while
- the study of relationships within units larger than a sentence (paragraphs, entire dialogues etc.) is the subject of discourse analysis.

Representation levels cont.

Linguistic structure
Representation levels
Grammars
Parsing and generation
Parsing and pipeline
Pipeline tasks

A simple example: a possible analysis of the representational structure of *those purple cows*.[2]



[2] From R. Jackendoff: The Parallel Architecture and its Lexicon. (Morphological structure is covered by "phonological structure" in the figure.)

Grammars

Linguistic structure
Representation levels
Grammars
Parsing and generation
Parsing and pipeline
Pipeline tasks

Relying on the notion of linguistic sign (l. sign for short) we can define further important concepts:

- A language is a set of l. signs.
- A grammar is a couple consisting of a
  1. a set of l. signs, the language's lexicon, and
  2. a finite set of operations that map one or more l. signs to a l. sign.
- A $\mathcal{G}$ grammar generates an $\mathcal{L}$ language iff $\mathcal{L}$ contains exactly those l. signs that are in $\mathcal{G}$'s lexicon or can be produced from $\mathcal{G}$'s lexicon by applying $\mathcal{G}$'s operations a finite number of times.

Grammars cont.

Linguistic structure
Representation levels
Grammars
Parsing and generation
Parsing and pipeline
Pipeline tasks

Grammatical operations are typically decomposed into phonological, morphological, syntactical and semantic operations working simultaneously.

E.g., for a two-argument f grammatical operation on linguistic signs there are corresponding phonological, morphological etc. operations for which

$$f\left(\begin{bmatrix} ph_1 \\ mor_1 \\ syn_1 \\ sem_1 \end{bmatrix}, \begin{bmatrix} ph_2 \\ mor_2 \\ syn_2 \\ sem_2 \end{bmatrix}\right) = \begin{bmatrix} f_{\text{ph}}(ph_1, ph_2) \\ f_{\text{mor}}(mor_1, mor_2) \\ f_{\text{syn}}(syn_1, syn_2) \\ f_{\text{sem}}(sem_1, sem_2) \end{bmatrix}$$

for all possible arguments of f.

Grammars cont.

Linguistic structure
Representation levels
Grammars
Parsing and generation
Parsing and pipeline
Pipeline tasks

Terminological warning

According to our definition, a grammar covers the phonology, morphology and semantics of a language in addition to its syntax.

A more limited conception of grammar is also frequently used in the literature, which is restricted to morphology and syntax, or syntax only.

Also, language is frequently defined more narrowly as a set containing only the *sentences* (as sound or written symbol sequences) generated by the grammar, without their morphological, syntactic and semantic structure.

Describing grammars

Linguistic structure
Representation levels
Grammars
Parsing and generation
Parsing and pipeline
Pipeline tasks

One of the central goals of linguistics is to describe grammars generating (fragments of) natural languages: English grammars, Spanish grammars etc.

Grammars are commonly described either

- *explicitly*, by describing a lexicon and defining the operations that generate the elements of the language from it, or
- *implicitly*, by providing a representative set of examples from the language generated by the grammar, i.e., speech or text samples annotated with morphological, syntactic etc. analyses.

Parsing and generation

Linguistic structure
Representation levels
Grammars
Parsing and generation
Parsing and pipeline
Pipeline tasks

Some grammar-related tasks are of paricular importance in NLP:

Parsing

- Decide whether a sequence of written symbols belongs to the language generated by a given grammar: whether it consists of the words of the language, is syntactically well-formed, and meaningful.
- Determine the morphological, syntactic and semantic structure(s) corresponding to a sequence of written symbols in the language generated by a given grammar.

Parsing and generation cont.

Linguistic structure
Representation levels
Grammars
Parsing and generation
Parsing and pipeline
Pipeline tasks

Generation

- *Unconditional*: generate elements of the grammar's language.
- *Conditional*: generate elements of the grammar's language that satisfy certain conditions. The conditions are often semantic, i.e., generate elements of the language whose semantic structure (meaning) satisfies certain conditions.

Parsing and the traditional NLP pipeline

Linguistic structure
Representation levels
Grammars
Parsing and generation
Parsing and pipeline
Pipeline tasks

The task of parsing had a central place in traditional NLP because it was assumed that most NLP tasks can be solved by

- parsing text input and producing their representational structure according to specific grammar(s),
- using the resulting analyses as features for further processing.

The traditional NLP pipeline, accordingly, is a parsing pipeline for one or several grammars, in which each component produces a part of the input's representational structure.

Processing tasks in the traditional pipeline

Linguistic structure
Representation levels
Grammars
Parsing and generation
Parsing and pipeline
Pipeline tasks

- *Morphology and syntax*
  - tokenization
  - sentence splitting
  - morphological analysis
  - part-of-speech tagging
  - (shallow or deep) syntactic parsing
- *Semantics*
  - named entity recognition
  - word sense disambiguation
  - coreference resolution / entity linking
  - semantic role labeling (shallow semantic parsing)
  - (deep) semantic parsing

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

Pipeline tasks

Tokenization

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

The task is to segment an input character sequence into small meaningful units called *tokens*, typically words and punctuation:

'This is a sentence.' $\Rightarrow$ ['This', 'is', 'a', 'sentence', '.']

Speaking of tokens instead of words has two important advantages:

- allows more flexibility: punctuation, emoticons etc. are not words but still useful units for segmentation;
- implies that these segments are instances of certain *types* that collectively constitute a *vocabulary*.

What should count as a token?

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

The answer is task and model-dependent: e.g., for some purposes punctuation is irrelevant, while for others sentence boundaries and, therefore, punctuation are important.

Some influential tokenization styles nonetheless achieved “quasi-standard” status. For English, the “Penn Treebank rules” are the most common, with the following key features:

- punctuation marks are split from words and treated as separate tokens,
- verb contractions (like the ”’s” in ”she’s”) and clitics (like the ”n’t” in ”don’t”) are split.

Type assignment and normalization

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

Tokenization can also involve determining which *type* tokens belong to. E.g., if 'apple' and 'Apple' are instances of the same type, then our tokenizer standardizes or *normalizes* these to a common type disregarding capitalization. Typical normalization practices include

- "correcting" spelling variations and typos by tokenizing all variants as instances of the same type,
- standardizing numerical or date-type expressions
- and punctuation (e.g, treat "!!" as "!").

More radical strategies include assigning all numeric expressions or all words not in a predefined vocabulary to a single type.

Tokenization challenges

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

The challenges are task and approach-dependent, but also depend on the input's

- writing/alphabet (e.g., writings without spaces!),
- language,
- domain,
- amount of noise (e.g., number of typos).

For European languages and writing systems, special challenges are posed by

- abbreviations (frequently ending with a period),
- number expressions (possibly containing white spaces, commas and periods),
- "multiword expressions" (MWEs) like "New York".

Sentence splitting

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

The task is to segment the (typically pretokenized) input character sequence into sentences:

['John', 'entered', 'the', 'room', '.', 'It', 'was', 'empty', '.'] $\Rightarrow$
[['John', 'entered', 'the', 'room', '.'], ['It', 'was', 'empty', '.']]

The main challenges are

- the interdependence of sentence and token segmentation, e.g. segmenting fragments of the form 'xxx yyy. Zzzz' (is 'yyy.' a sentence ending or an abbreviation?);
- incorrect or lacking punctuation.

Morphology

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

Morphemes are the smallest meaningful units in a language. Words can consist of several morphemes, e.g. *unbearables* = *un* + *bear* + *able* + *s*

Useful distinctions among morphemes:

- *Bound* vs *free*: free morphemes (e.g. *bear*) can stand alone as independent words, bound morphemes (e.g. *-un*, *-s*) can only constitute words together with other morphemes.
- *Affixes* vs *roots*: roots are the main parts of the word with the most specific semantic content (*bear* in the example), around which other morphemes, affixes can be placed. Most roots are free.

Affix types

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

Affixes can be further grouped by their (typically positional) relation to other affixes:

Affix type	Relation	Example
prefix	precedes	*un-*, *anti-*
suffix	follows	*-s*, *-ing*
infix	between	*Singabloodypore*
circumfix	around	*ge...t* in German (e.g. *gespielt*)
stem change	changes	Arabic *kitaab* ('book') $\rightarrow$ *kutub* ('books')

Far from being the full list, other affix types include duplication, tone/pitch change etc.

Affix types cont.

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

A crucial distinction is between inflectional and derivational affixes:

- *inflectional* affixes create new forms of the *same word*, and can represent grammatical aspects such as person, tense etc., English examples include the plural *-(e)s* and the progressive *-ing*.
- *derivational affixes*, on the other hand, *form new words*, e.g., the *-able* in *bearable* forms an adjective from a verb.

Stem and lemma

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

- The *stem* of a word consists of the *base part* of the word that is common in all inflected forms. Consequently, the stem is often not a meaningful word, e.g., the stem of *produced* is *produc* (because of *producing* etc.).[3]
- A *lemma*, in contrast, is always a complete word, namely, the uninflected base form of inflected forms. Continuing the example, the lemma of *produced* is *produce*.

[3] The example is from Wikipedia's Lemma entry.

Morphological analysis tasks

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

- Decide whether a string is a well-formed word.
- *Stemming*: determine the stem of a word.
- *Lemmatization*: determine the lemma of a word.
- *Morphological tagging*: tag input words according to the grammatical information their inflections etc. express.
- *Morphological segmentation*: segment the input words into morphemes.
- *Full morphological analysis*: segment the words into morphemes and categorize each morpheme according to type, and grammatical information they convey. Often includes lemmatization too.

Morphological analysis challenges

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

- *Ambiguity / context dependence*: many words have multiple analyses that can only be disambiguated based on context, e.g. the *-s* in *chairs*. Cf.

  The president chairs the meeting.
  There were hundreds of chairs in the room.
- *Compounds*: many languages have compound words built from two or more simpler words: e.g., the German *Schadenfreude* from *Schaden* ('damage, harm') and *Freude* ('joy').
- *Morphologically rich languages*: English has a relatively simple morphology. Other languages, e.g., Hungarian and Turkish are way more complex...

Morphological analysis challenges cont.

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

Extreme Turkish example:

Turkish	English
Muvaffak	Successful
Muvaffakiyet	Success
Muvaffakiyet**siz**	**Un**successful ('**without** success')
Muvaffakiyetsiz**leş***(-mek)*	*(To)* **become** unsuccessful
Muvaffakiyetsizleş**tir***(-mek)*	*(To)* **make one** unsuccessful
Muvaffakiyetsizleştiri**ci**	**Maker of** unsuccessful ones
Muvaffakiyetsizleştirici**leş***(-mek)*	*(To)* **become** a maker of unsuccessful ones
Muvaffakiyetsizleştiricileş**tir***(-mek)*	*(To)* **make one** a maker of unsuccessful ones
Muvaffakiyetsizleştiricileştiri**ver***(-mek)*	*(To)* **easily/quickly** make one a maker of unsuccessful ones
Muvaffakiyetsizleştiricileştirivere**bil***(-mek)*	*(To)* **be able to make** one easily/quickly a maker of unsuccessful ones
Muvaffakiyetsizleştiricileştirivere**meye**bil*(-mek)*	**Not** *(to)* be able to make one easily/quickly a maker of unsuccessful ones
Muvaffakiyetsizleştiricileştiriveremeyebil**ecek**	**One who is** not able to make one easily/quickly a maker of unsuccessful ones
Muvaffakiyetsizleştiricileştiriveremeyebilecek**ler**	**Those** who are not able to make one easily/quickly a maker of unsuccessful ones
Muvaffakiyetsizleştiricileştiriveremeyebilecekleri**miz**	Those who **we** cannot make easily/quickly a maker unsuccessful ones
Muvaffakiyetsizleştiricileştiriveremeyebileceklerimiz**den**	**From** those we can not easily/quickly make a maker of unsuccessful ones
Muvaffakiyetsizleştiricileştiriveremeyebileceklerimizden**miş**	**(Would be)** from those we can not easily/quickly make a maker of unsuccessful ones
Muvaffakiyetsizleştiricileştiriveremeyebileceklerimizdenmiş**siniz**	**You** would be from those we can not easily/quickly make a maker of unsuccessful ones
Muvaffakiyetsizleştiricileştiriveremeyebileceklerimizdenmişsiniz**cesine**	**Like** you would be from those we can not easily/quickly make a maker of unsuccessful ones

Part-of-speech tagging

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

Part-of-speech (POS for short) categories correspond to basic syntactic roles words can play in sentences. E.g., in the sentence

Peter ate the apple.

Peter and *apple* are nouns, *ate* is a verb and *the* is a determiner. The POS-tagging task is to determine the POS-category of each word in an already tokenized (and possibly sentence-split) input text, e.g.,:

['Peter', 'ate', 'the', 'apple', '.'] $\Rightarrow$
[('Peter', 'NOUN'), ('ate', 'VERB'), ('the', 'DET'), ('apple', 'NOUN'), ('.', 'PUNCT')]

Part-of-speech tagging cont.

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

Similarly to the morphological analysis of a word, POS-category is context-dependent. E.g., compare

John hit the ball.
His first song was a huge hit in Europe.

The concrete list of POS categories and their delineation also depends on the language and specific grammatical theory used, although some are quite universal, e.g., the categories NOUN, VERB, ADJECTIVE and ADVERB are almost always present.

Open vs. closed POS-categories

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

- *Closed* POS categories, e.g., determiners in English, consist of relatively small sets of words, and these sets do not change easily: it's a rare phenomenon that a new determiner is added to a language.
- *Open* POS categories, like that of English verbs, contain a large number of words and new members are added on daily basis.

A related distinction is between *function words* and *content words*. Words belonging to open POS categories are typically content words: they have a well characterizable semantic content on their own. Closed POS categories contain function words not having much independent content.

POS tag sets

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

In NLP, POS categories are typically encoded with shorthands, so called POS tags. Several tag sets are in use even for English; a very important, language agnostic set was developed for the Universal Dependencies project:

Open class tags

Tag	Description	Examples
ADJ	adjective	big, old, green, African, first
ADV	adverb	very, well, exactly, tomorrow
INTJ	interjection	psst, ouch, bravo, hello
NOUN	noun	girl, cat, tree, air
PROPN	proper noun	Mary, John, London, NATO
VERB	verb	run, eat, runs, ate

POS tag sets cont.

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

Closed class tags

Tag	Description	Examples
ADP	adposition	in, to, during
AUX	auxiliary	has, is, should, was, must
CCONJ	coordinating conjunction	and, or, but
DET	determiner	a, an, the, this, which, any, no
NUM	numeral	0, 1, 2, one, two
PART	particle	not, ’s (as in ”Andrew’s table”)
PRON	pronoun	I, myself, who
SCONJ	subordinating conjunction	that, if

Other tags

PUNCT	punctuation	. , ;
SYM	symbol	$, ¶, ©
X	other	for unanalyzable elements

Syntactic parsing

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

Syntactic theories aim to characterize “the set of rules or principles that govern how words are put together to form phrases, well formed sequences of words.”[4]

The most important “well formed sequences” in this context are *sentences*: the central goal of syntactic theories for a given language is to find structural rules or principles that characterize/delineate well formed sentences of the language in question.

[4] Koopman et al.: An Introduction to Syntactic Analysis and Theory, p. 1.

Syntactic parsing cont.

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

A sentence is well formed if it has a *structural description* or *syntactic parse* which satisfies the syntactic constraints of the theory in question. Syntactic well formedness doesn't guarantee coherence or meaningfulness. To use Chomsky's famous example:

Colorless green ideas sleep furiously.

is syntactically well formed but nonsensical, while

Furiously sleep ideas green colorless.

is not even well formed.

Syntactic parsing cont.

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

Constituency (aka *phrase structure*) and *dependency* based syntactic theories have been especially important for NLP.

A *constituent* is a single word or a group of consecutive words that form a "natural unit". E.g., the phrase *a nice little city*

- can be put into various sentence frames like *I wanted to visit ...*, *Budapest is ...*
- can be an answer to a question: *What did you visit?*
- can be substituted by pronouns: *I have visited a nice little city.* $\Rightarrow$ *I have visited it.*[5]

[5] See, e.g., the 'Constituent' entry of Wikipedia for further tests.

Constituency-based syntax

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

Constituency-based syntactic theories

- categorize constituents, and
- formulate rules according to which constituents can be put together to build larger ones, eventually building up a whole sentence.

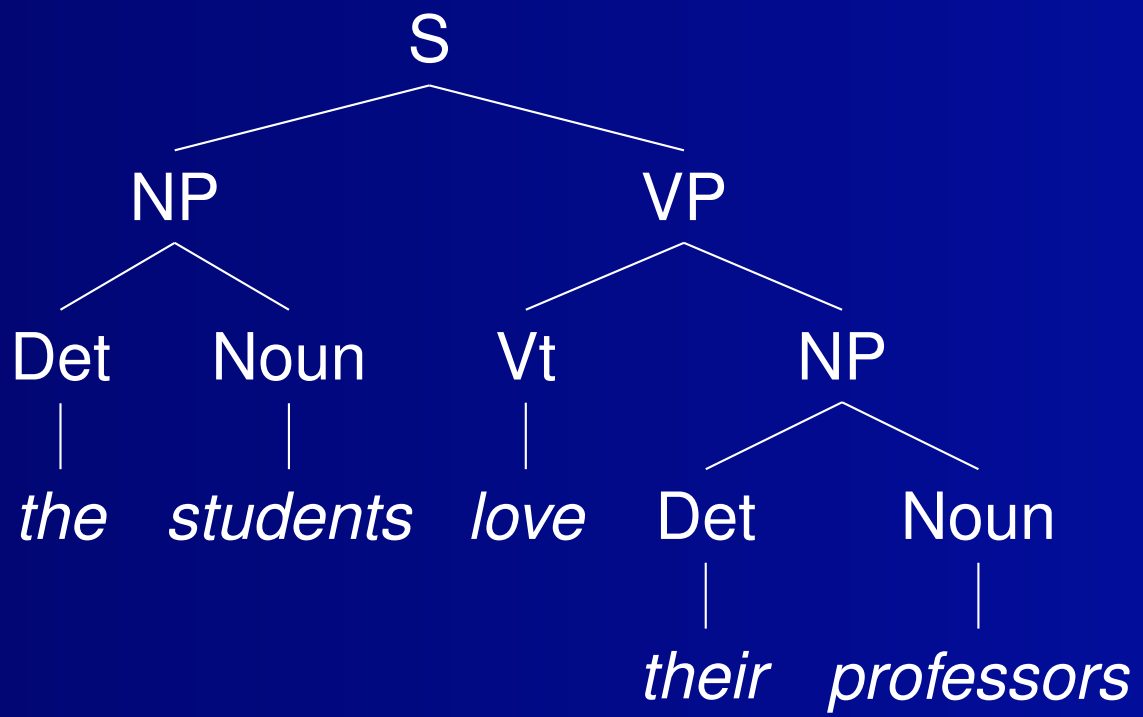
The syntactic structure of a well formed sentence, is, accordingly, its constituency structure, e.g., for the sentence *The students love their professors*:

$[[\text{The}_{\mathrm{D}}\ \text{students}_{\mathrm{N}}]_{\mathrm{NP}}[\text{love}_{\mathrm{Vt}}\ [\text{their}_{\mathrm{D}}\ \text{professors}_{\mathrm{N}}]_{\mathrm{NP}}]_{\mathrm{VP}}]_{\mathrm{S}}$

Constituency-based syntax cont.

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

In a more transparent constituency tree form:



This is the structure constituency-based parsers output.

Dependency-based syntax

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

Dependency grammars, in contrast, treat the *dependency relation* between words as fundamental. The precise criteria vary from theory to theory, but typically a d word depends on a h word (equivalently, h heads d) in a sentence if

- d modifies the meaning of h, makes it more specific, e.g. *eats* $\Rightarrow$ *eats bread*, *eats slowly* etc.
- and there is an asymmetric relationship of omissibility between them: d can be omitted from the sentence keeping h but not vice versa.

Dependency-based syntax cont.

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

Dependency grammars impose important global constraints on the dependency relations within a well formed sentence:

- There is exactly one independent word (the root of the sentence).
- All other words depend directly on exactly one word.

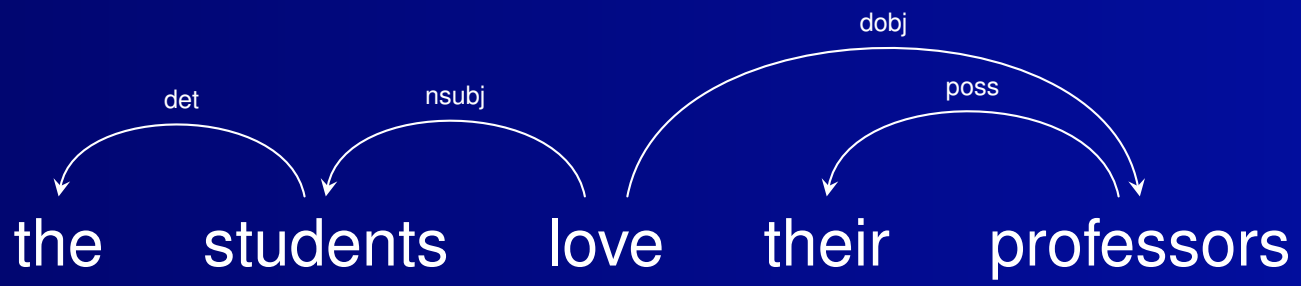
As a consequence, the direct dependency graph of a sentence is a tree.

Most dependency grammars work with *typed direct dependencies*: there is finite list of direct dependency types with specific constraints on when they can hold.

Dependency-based syntax cont.

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

A dependency parse tree of the earlier example:

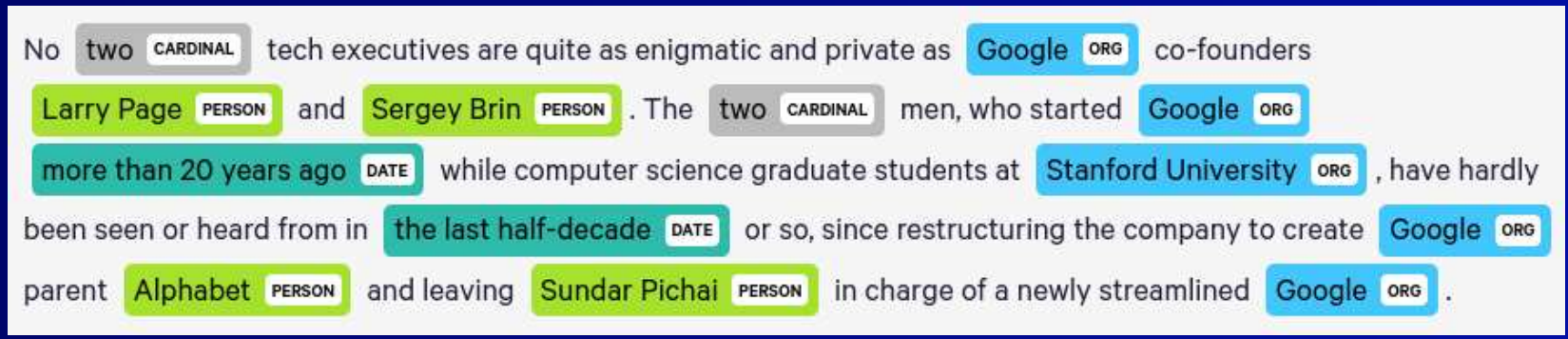


Compared to the constituency tree, it contains fewer nodes (one per word), but the edges are labeled with the corresponding dependency types.

Named entity recognition

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

Named entity recognition (NER) is the task of finding expressions in the input text that are *naming* entities and tagging them with the corresponding entity type:



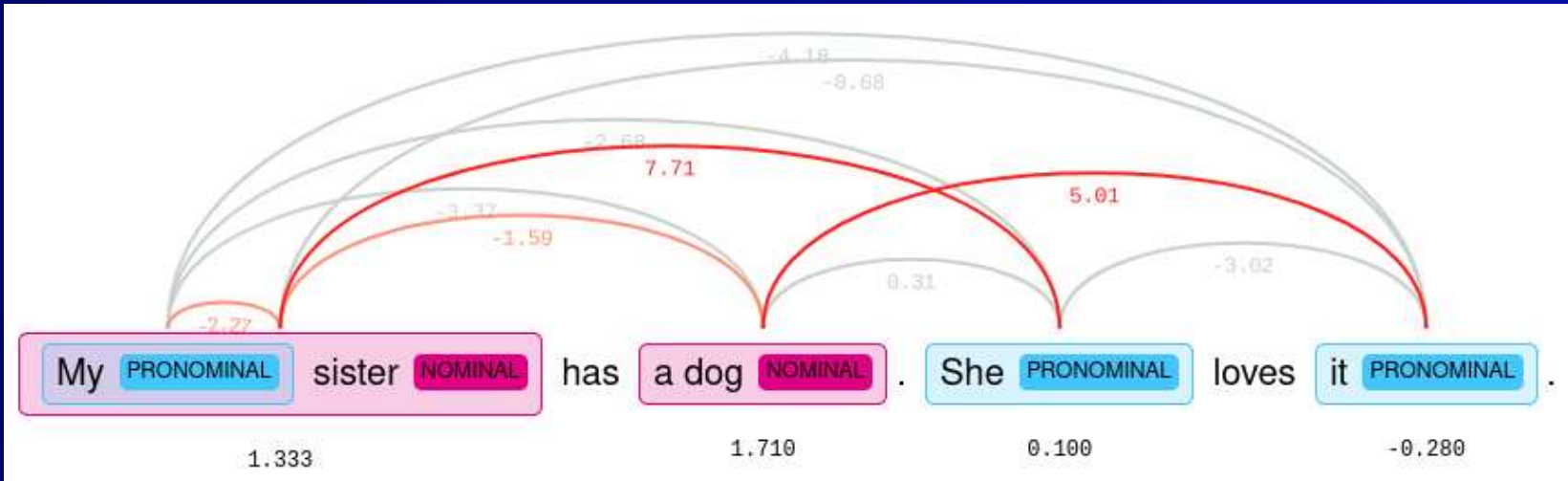
(The figure shows output from spaCy's NER demo page.)

Typically used entity types are *person*, *organization* and *location*, but many NER models cover additional types such as *date*, *event*, *work or art*, *law* etc.

Coreference resolution

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

NER determines the *type* of entities that names refer to – it doesn't decide whether they refer to the same or different entities. The *coreference resolution* task, in contrast, is to locate a broader range of referring expressions in the input including common nouns and pronouns, and cluster them into groups whose members all refer to the same entity:[6]



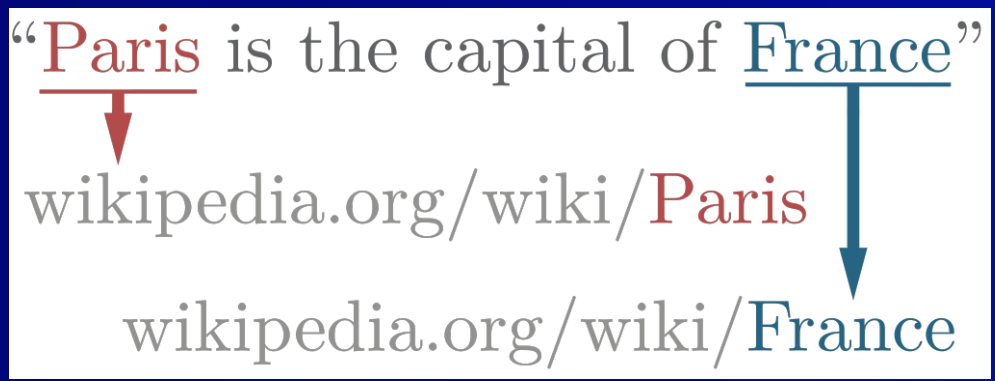
[6]The figure shows the output of the neuralcoref model built on spaCy.

Entity linking

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

Similarly to coreference resolution, entity linking is also concerned with the identity of the references, but differs from it in two important respects:

- as NER, it is limited to name-like expressions,
- it determines the identity of the entities by connecting the names to entity records in an *external* knowledge base, e.g., Wikipedia:



(The figure is from Wikipedia’s Entity Linking entry.)

Word sense disambiguation

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

Word sense disambiguation also connects expressions with meanings/senses in an external inventory, but

- it is concerned with common nouns and other types of content words: verbs, adjectives and adverbs;
- the sense collections are typically purpose-built lexical resources – the quasi-standard is to use the WordNet lexical database.

Word sense disambiguation cont.

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

For instance, a WordNet-based WSD system should disambiguate the *mouse* noun in the sentence

The scroll wheel in my mouse has stopped working.

to the WordNet sense

MOUSE#4: 'a hand-operated electronic device that controls the coordinates of a cursor [...]'. Other possibilities in WordNet:

- MOUSE#1: 'any of numerous small rodents [...]'
- MOUSE#2: 'a swollen bruise [...]'
- MOUSE#3: 'person who is quiet or timid [...]'

Semantic role labeling

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

Semantic role labeling (SRL) is the task of identifying *predicate* and *argument* expressions in the input text, determining which argument belongs to which predicate, and what is their relationship. In this context,

- *predicates* are expressions referring to events/situations (e.g., verbs referring to actions),
- while *arguments* refer to the *participants* of these events/situations,
- and the *role labeling* part of the task is to determine what kind or role the participants corresponding to the arguments play in the situations referred to by the predicates.

Semantic role labeling cont.

Linguistic structure

Pipeline tasks

Tokenization

Sentence splitting

Morphology

POS tagging

Syntactic parsing

NER

Coreference resolution

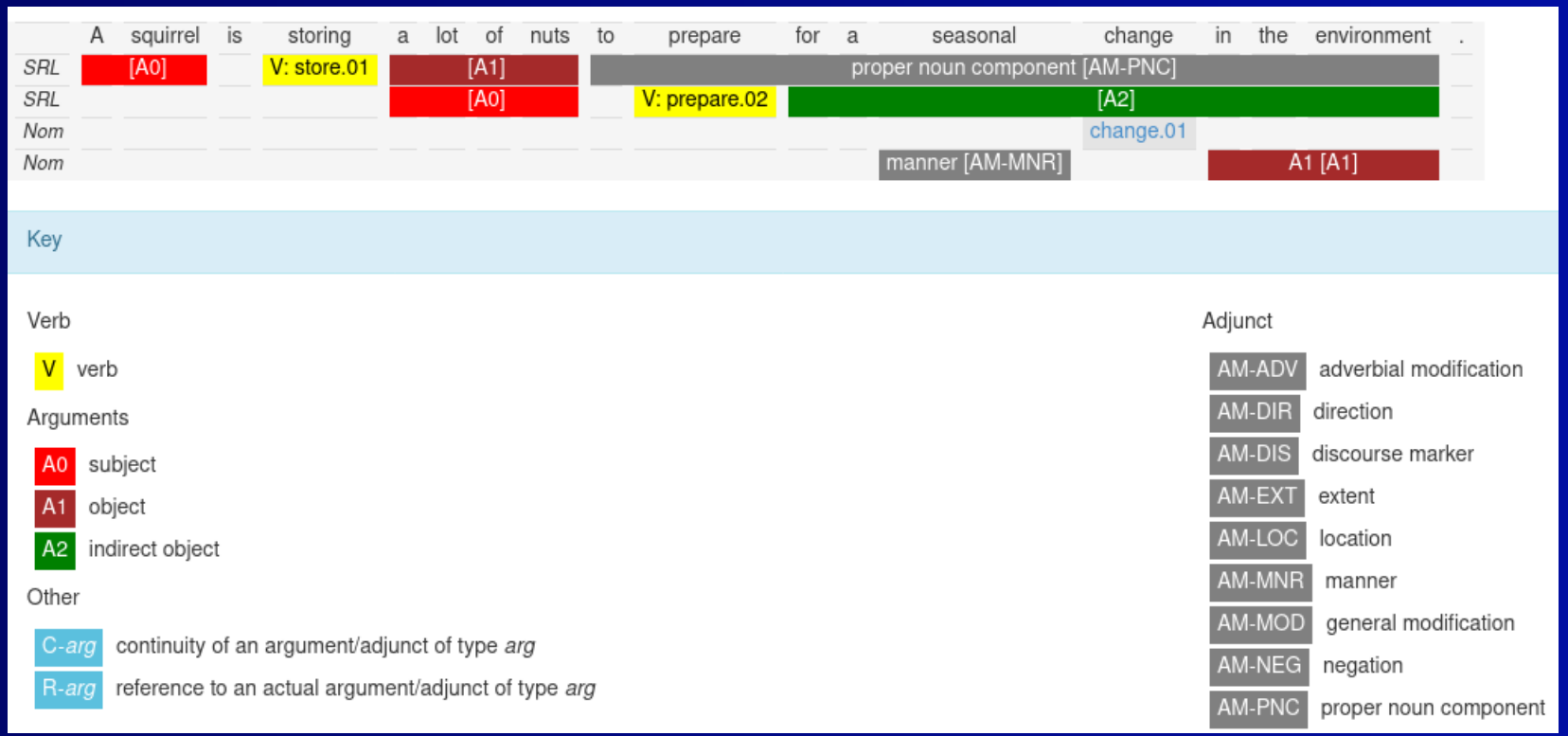
Entity linking

WSD

Semantic role labeling

Semantic parsing

A relatively simple example using the SLR demo of the U. Penn. Cognitive Computation Group:



Semantic parsing

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

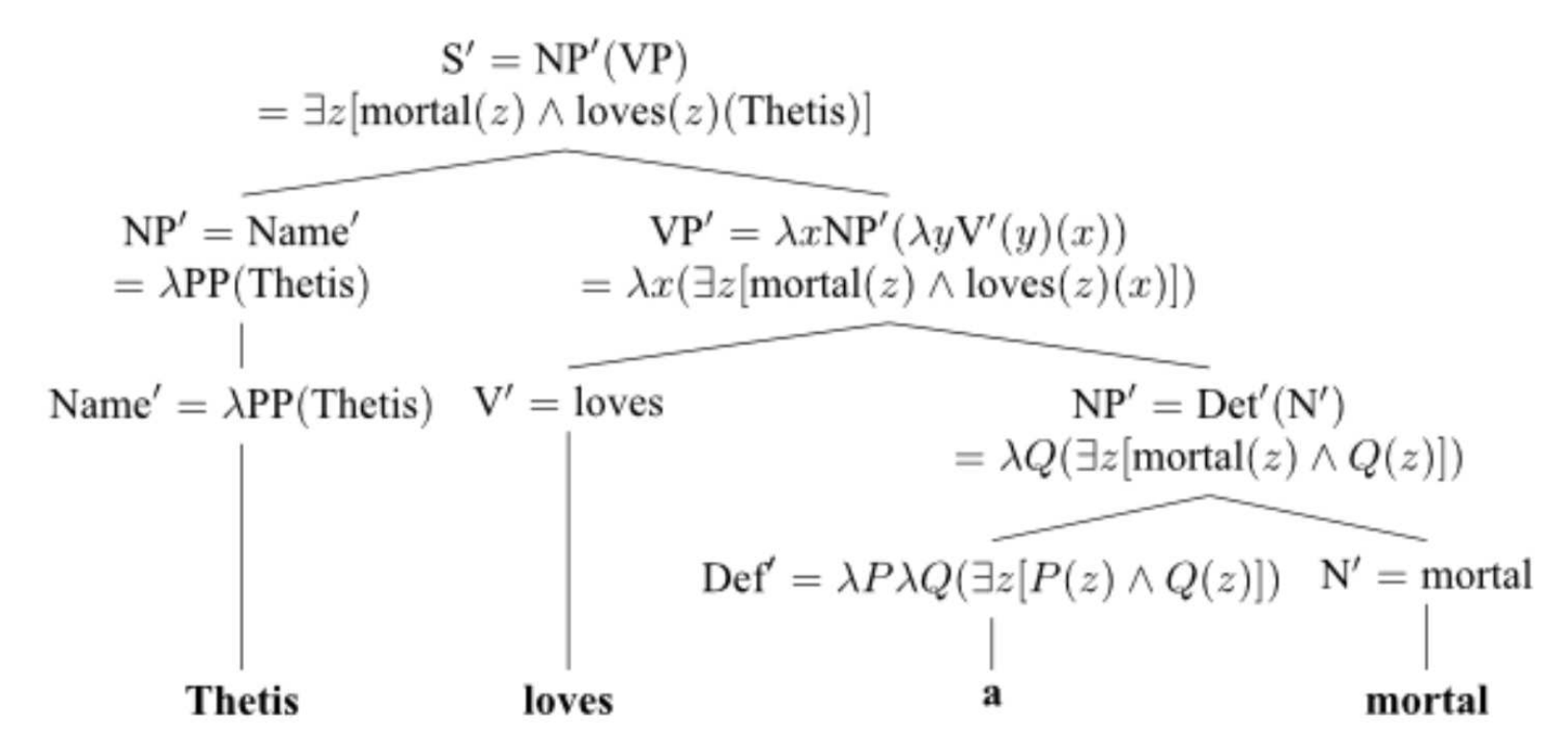
This is the the task of full or deep semantic parsing, which covers not “only” coreference resolution, WSD and predicate-argument structure, but aims at providing a complete formal *semantic representation*, which is

- a formal structure representing the meaning of the input text,
- represents literal meaning,
- disambiguated (as far as possible),
- canonical (as far as possible) in the sense that a text meaning has a unique representation,
- there are efficient algorithms to determine their logical and semantic relationship to other semantic and knowledge representations.

Semantic parsing cont.

Linguistic structure
Pipeline tasks
Tokenization
Sentence splitting
Morphology
POS tagging
Syntactic parsing
NER
Coreference resolution
Entity linking
WSD
Semantic role labeling
Semantic parsing

A first-order logic based semantic representation of the sentence *Thetis loves a mortal*:[7]



[7] From the Computational Linguistics entry of the Stanford Encyclopedia of Philosophy.