Lecture 23: Formal Semantics, Semantic Roles

Ling 1330/2330 Intro to Computational Linguistics
Na-Rae Han, 11/16/2023

Finally, **meaning**

Computational semantics: key areas

- Formal semantics: Logic, model-theoretic semantics
  - NLTK Book ch.10 Analyzing the meaning of sentences
- Word sense: lexical semantics
  - J&M Ch.23: Word senses and WordNet
  - NLTK Book 2.5 WordNet
- Word sense: vector semantics
  - J&M Ch.6: Vector semantics and embeddings
- Predicate-argument semantics, semantic roles
  - J&M Ch.24: Semantic role labeling
  - NLTK how to, PropBank

Vast landscape,
so little time…

WordNet

- Project home: https://wordnet.princeton.edu/
- A hierarchical semantic database ("ontology") for English and many other languages (The Open Multilingual Wordnet).
- Beyond definitions, encodes *relations* between senses.
- Available via NLTK as `nltk.corpus.wordnet`
  - NLTK book https://www.nltk.org/book/ch02.html#wordnet
- **A single unique meaning** is designated as something like 'car.n.01': first noun meaning of 'car'. This is referred to as a **synset**: a "synonym set"
  - The idea: a *unique meaning* is represented by *a set of synonyms* that share the meaning.
  - 'car.n.01' is the most generic meaning of 'car', which can be seen through .definition()

```
>>> from nltk.corpus import wordnet as wn
>>> wn.synsets('motorcar')
    [Synset('car.n.01')]
>>> wn.synset('car.n.01').lemma_names()
    ['car', 'auto', 'automobile', 'machine', 'motorcar']
>>> wn.synset('car.n.01').definition()
    'a motor vehicle with four wheels; usually propelled by an internal combustion
    engine'

>>> wn.synsets('car')
    [Synset('car.n.01'), Synset('car.n.02'), Synset('car.n.03'),
    Synset('car.n.04'), Synset('cable_car.n.01')]

>>> for syn in wn.synsets('car'):
>>>     print(syn, syn.lemma_names())

    Synset('car.n.01') ['car', 'auto', 'automobile', 'machine', 'motorcar']
    Synset('car.n.02') ['car', 'railcar', 'railway_car', 'railroad_car']
    Synset('car.n.03') ['car', 'gondola']
    Synset('car.n.04') ['car', 'elevator_car']
    Synset('cable_car.n.01') ['cable_car', 'car']
>>>
```

'car.n.01' represents a **synset** (="synonym set"), a single unique sense.

'car' has 5 distinct senses

Each sense can be conveyed by a set of synonymous words

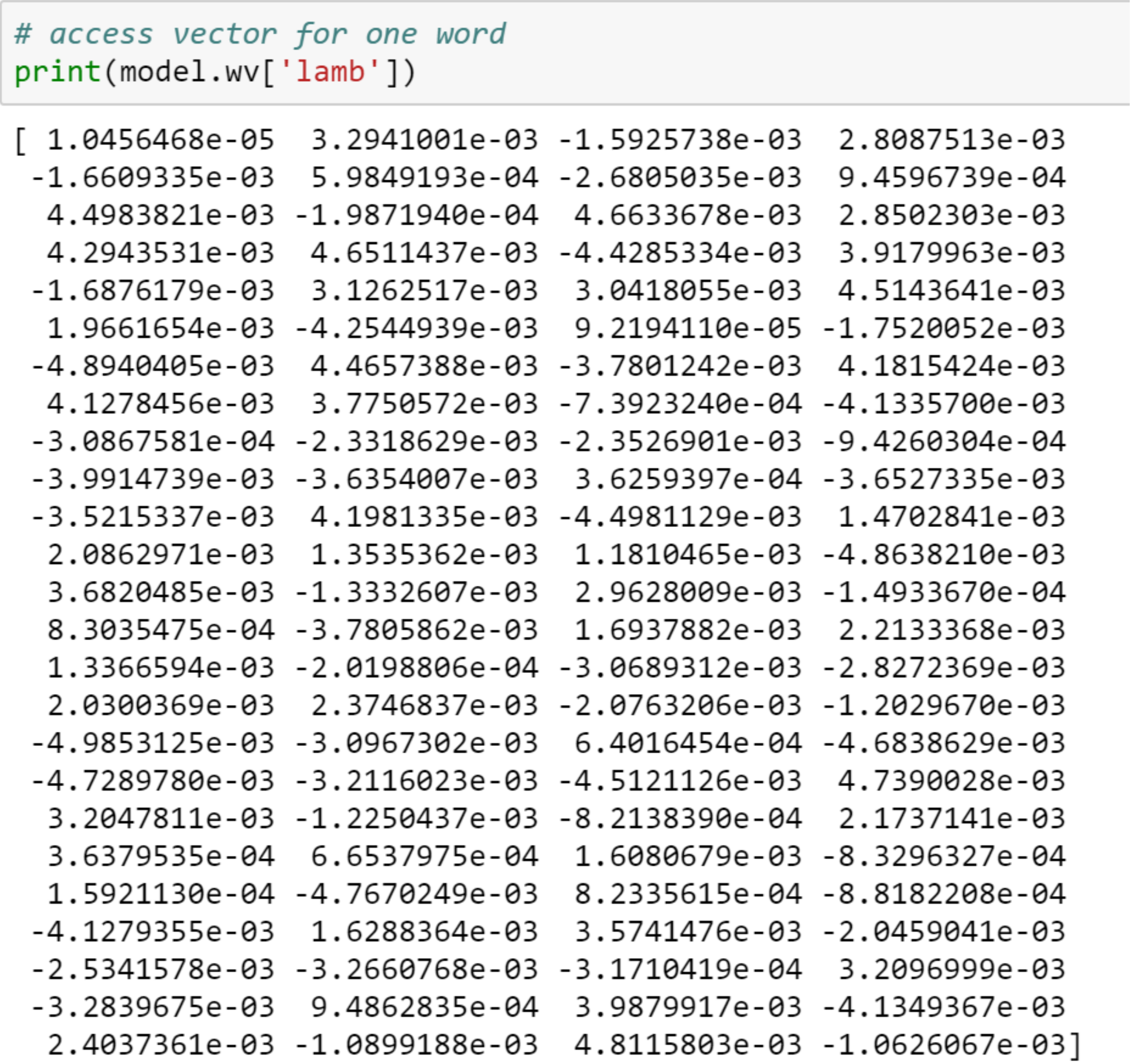
Word sense, the symbolic way

- WordNet:

```
>>> wn.synsets('lamb')
    [Synset('lamb.n.01'), Synset('lamb.n.02'), Synset('lamb.n.03'),
    Synset('lamb.n.04'), Synset('lamb.n.05'), Synset('lamb.v.01')]
>>> wn.synset('lamb.n.01').definition()
    'young sheep'
>>> wn.synset('lamb.n.04').definition()
    'a sweet innocent mild-mannered person (especially a child)'
>>> wn.synset('lamb.n.05').definition()
    'the flesh of a young domestic sheep eaten as food'
>>> wn.synset('lamb.n.01').hyponyms()
    [Synset('baa-lamb.n.01'), Synset('hog.n.02'), Synset('lambkin.n.01'),
    Synset('persian_lamb.n.02'), Synset('teg.n.01')]
>>>
```

Symbolic approach = "representational" approach

How is THIS for a word sense?



```python
# access vector for one word
print(model.wv['lamb'])
```

```
[ 1.0456468e-05  3.2941001e-03 -1.5925738e-03  2.8087513e-03
 -1.6609335e-03  5.9849193e-04 -2.6805035e-03  9.4596739e-04
  4.4983821e-03 -1.9871940e-04  4.6633678e-03  2.8502303e-03
  4.2943531e-03  4.6511437e-03 -4.4285334e-03  3.9179963e-03
 -1.6876179e-03  3.1262517e-03  3.0418055e-03  4.5143641e-03
  1.9661654e-03 -4.2544939e-03  9.2194110e-05 -1.7520052e-03
 -4.8940405e-03  4.4657388e-03 -3.7801242e-03  4.1815424e-03
  4.1278456e-03  3.7750572e-03 -7.3923240e-04 -4.1335700e-03
 -3.0867581e-04 -2.3318629e-03 -2.3526901e-03 -9.4260304e-04
 -3.9914739e-03 -3.6354007e-03  3.6259397e-04 -3.6527335e-03
 -3.5215337e-03  4.1981335e-03 -4.4981129e-03  1.4702841e-03
  2.0862971e-03  1.3535362e-03  1.1810465e-03 -4.8638210e-03
  3.6820485e-03 -1.3332607e-03  2.9628009e-03 -1.4933670e-04
  8.3035475e-04 -3.7805862e-03  1.6937882e-03  2.2133368e-03
  1.3366594e-03 -2.0198806e-04 -3.0689312e-03 -2.8272369e-03
  2.0300369e-03  2.3746837e-03 -2.0763206e-03 -1.2029670e-03
 -4.9853125e-03 -3.0967302e-03  6.4016454e-04 -4.6838629e-03
 -4.7289780e-03 -3.2116023e-03 -4.5121126e-03  4.7390028e-03
  3.2047811e-03 -1.2250437e-03 -8.2138390e-04  2.1737141e-03
  3.6379535e-04  6.6537975e-04  1.6080679e-03 -8.3296327e-04
  1.5921130e-04 -4.7670249e-03  8.2335615e-04 -8.8182208e-04
 -4.1279355e-03  1.6288364e-03  3.5741476e-03 -2.0459041e-03
 -2.5341578e-03 -3.2660768e-03 -3.1710419e-04  3.2096999e-03
 -3.2839675e-03  9.4862835e-04  3.9879917e-03 -4.1349367e-03
  2.4037361e-03 -1.0899188e-03  4.8115803e-03 -1.0626067e-03]
```

This is a **word vector**.
Next week!

Formal semantics

- NLTK book ch.10 is all about formal semantics.
  - https://www.nltk.org/book/ch10.html
- Focus on **logic** programming.
  - Old-school, symbolic approach to computational semantics.
  - **Prolog**: "programming in logic". A taste here.

Every vegetarian likes a politician.

- Reading 1: Vegetarians might like different politicians (or could be same)

$$\forall x(Vegetarian(x) \rightarrow \exists y(Politician(y) \land Like(x,y)))$$

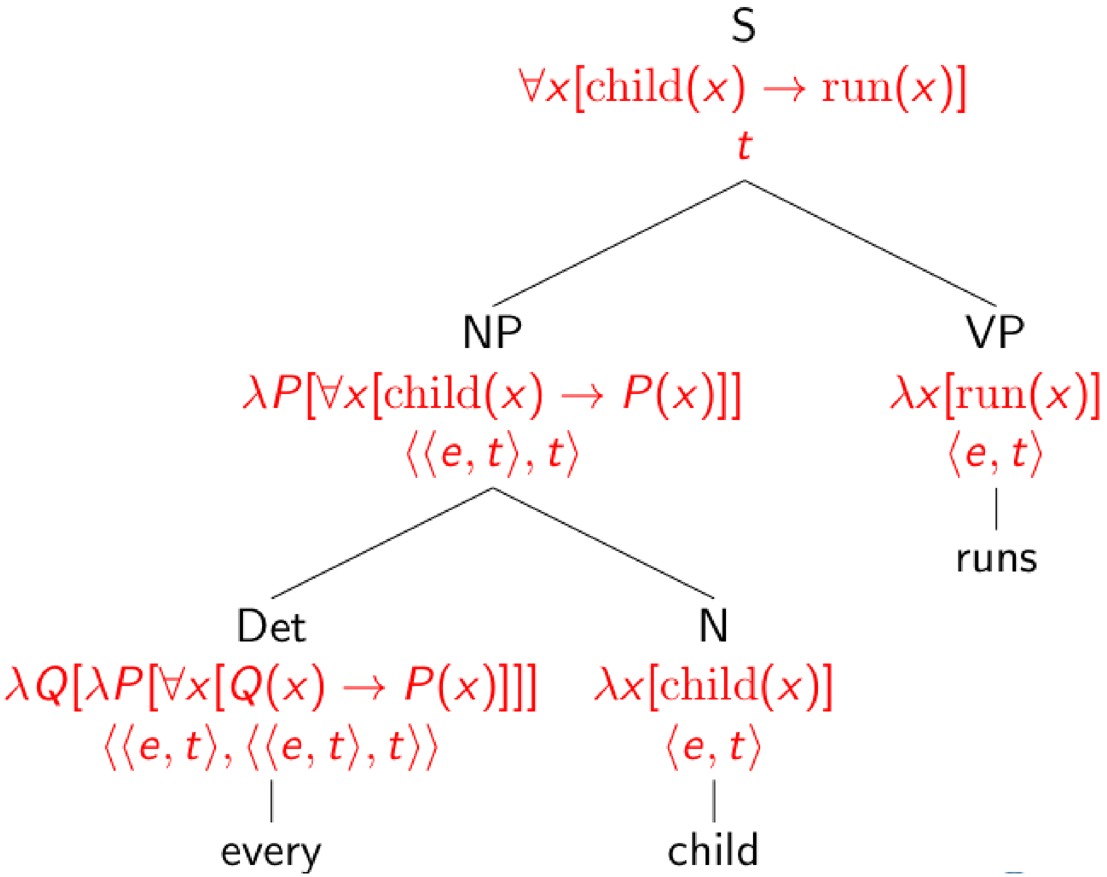
- Reading 2: There's one very popular politician universally liked by vegetarians

$$\exists y(Politician(y) \land \forall x(Vegetarian(x) \rightarrow Like(x,y)))$$

More in LING 1682 "Intro to Semantic Theory"

Combining semantics + syntax

- A sentence's meaning is derived **compositionally**:

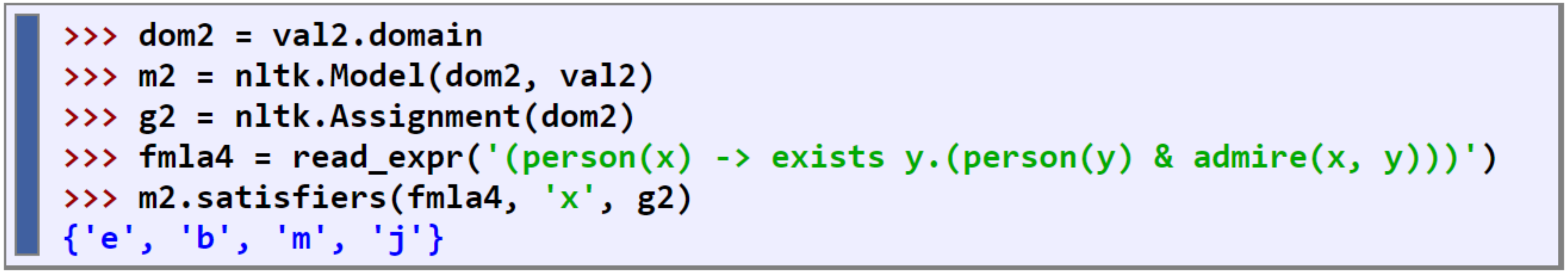

S
$\forall x[\mathrm{child}(x) \rightarrow \mathrm{run}(x)]$
t
NP
$\lambda P[\forall x[\mathrm{child}(x) \rightarrow P(x)]]$
$\langle\langle e, t\rangle, t\rangle$
VP
$\lambda x[\mathrm{run}(x)]$
$\langle e, t\rangle$
runs
Det
$\lambda Q[\lambda P[\forall x[Q(x) \rightarrow P(x)]]]$
$\langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$
every
N
$\lambda x[\mathrm{child}(x)]$
$\langle e, t\rangle$
child

Lambda calculus

*Image credit: https://canvas.gu.se/courses/20331

Formal logic in NLTK

- NLTK ch.10 Analyzing the Meaning of Sentences
  - https://www.nltk.org/book/ch10.html
  - Let's review it.



```
>>> dom2 = val2.domain
>>> m2 = nltk.Model(dom2, val2)
>>> g2 = nltk.Assignment(dom2)
>>> fmla4 = read_expr('(person(x) -> exists y.(person(y) & admire(x, y)))')
>>> m2.satisfiers(fmla4, 'x', g2)
{'e', 'b', 'm', 'j'}
```

Predicate-argument semantics

- Focuses on **verb meaning** as expressed via its **arguments**

Homer ate the donut.
AGENT PATIENT

- **Semantic roles** express the role that arguments of a predicate take in the event
- Databases like **PropBank** and **FrameNet** augment Treebanks with detailed semantic role information
- **Semantic role labeling**: the task of assigning roles to spans in sentences

Thematic roles

- **Thematic roles**: assigned by the speaker to entities that are involved in a situation (also called **theta roles**)
  - *Bart tightened the screw with a wrench.*
- Why are thematic roles important?
  - Thematic roles are a core part of the **verb meaning**
  - They interact closely with the **verb syntax**: are mapped to grammatical relations/roles (= subject, object, indirect object, object of preposition)
  - **Voice** is the device that alters the mapping between thematic roles and grammatical roles. Passive voice:
    - *The screw was tightened by Bart with a wrench.*

Theta roles and verb semantics

- *Charlie raised the car with a jack.*
  - **raise**^{1} V: <AGENT, THEME, INSTRUMENT> ← Theta-grid
- *The jack raised the car.*
  - **raise**^{2} V: <INSTRUMENT, THEME>
- *The car rose.* (cf. **The car raised.*)
  - **rise** V: <THEME>

Predicate-argument semantics

- J&M ch.24 Semantic Role Labeling
  - https://web.stanford.edu/~jurafsky/slp3/24.pdf
  - Let's review it.

Proposition Bank (PropBank)

- The **Proposition Bank** (aka PropBank): a resource of semantic role annotations
  - Augments Penn Treebank corpora (English, Chinese…)
  - Shies away from using universally defined thematic role labels (AGENT, THEME, LOCATION…)
  - Instead, uses numberings (Arg0, Arg1, Arg2) whose exact roles are verb-specific
  - Some rule of thumb however: Arg0 represents PROTO-AGENT, Arg1 represents PROTO-PATIENT
  - Also marks modifier elements as ArgMs

PropBank in NLTK

- NLTK has a how-to page:
  - https://www.nltk.org/howto/propbank.html
- Demo in Jupyter Notebook.

Wrapping up

This Photo by Unknown Author is licensed under CC BY

- Have a great Thanksgiving!
- After the break:
  - Continue computational semantics: vector semantics

- Homework 9 out
  - Last Python homework…
  - Due 11/30 (Thu), start PART 1 now!