LEXICAL SEMANTICS
&
COMPUTATIONAL SEMANTICS

LEXICAL SEMANTICS

Semantics is the study of meanings in language. It can be applied to entire texts or to single words.

Example:

The simple word "on" can have many meanings, such as: on call, on the roof, on cloud nine, on edge, on fire, on purpose, on demand, on top, or on the phone.

Lexical Semantics: Lexical Semantics deconstruct words and phrases within a line of text to understand the meaning in terms of context.

Lexicon is mental dictionary which stores lexemes or senses or both.

Example: The word Bank can mean the side of river or can mean at money vaults.

One word bank has multiple senses.

Example: The word Dad and father may have same sense or same meaning.

RELATIONS AMONG LEXEMES AND SENSES:

<u>Homonyms</u> that share a form but have unrelated distinct meanings.

Homonyms are generally defined as words different in meaning but either identical both in sound and spelling or identical only in sound or spelling.

Example:

i. Our house is on the West bank of the river

ii. I want to save my money in the bank.

<u>Homophones</u> are senses that are linked to lemmas with the same pronunciation but different spellings.

A homophone is a word that is pronounced the same as another word but differs in meaning and may differ in spelling.

Example: would/ wood

too/two/to

<u>Homographs</u> are distinct senses linked to lemmas with the same orthographic form but different pronunciations.

Example: Articulate (Adj) Articulate (Verb)

Relations among Lexemes and their Senses

Polysemy:

Two sense of Homonym words are related.

Example:

1. The bank was constructed in 1875.
2. I with drew the money from the bank.

Are those senses are same?

Sense2: “A Financial institution

Sense1: “The building belonging to the financial institution”.

In many cases of polysemy the semantic relation between the senses is systematic and structured.

Relations among lexemes and their senses

<u>Synonymy</u>:

Words that have same meaning in one or other contexts.

<u>Example:</u>

- Cough/sofa
- Water/H20

Two lemmas are said to be synonymy if they have same proportional meanings. They can be substituted for each other in all situations.

Synonymy is the relationship between two words that have the same sense.

- For example

i. The student speaks with a broad British accent.

ii. The student speaks with a wide British accent.

Hence they have the same or nearly the same sense but not in the other contexts to mean the same. However this kind of sense relation means “word of the same meaning”

RELATIONS AMONG LEXEMES AND THEIR SENSES

Antonyms

Senses which are opposite with respect to one feature of meaning.

Example: dark/light

short/long

Two kinds of Antonyms

1. Binary opposition/opposite ends of scale(long/short)
2. Reversive (rise/fall, up/down)

1. Binary antonyms come in pairs and between them exhaust all the relevant possibilities. If one is applicable, the other cannot be, and vice versa.

Example:

- true – false
- dead – alive

2. Reverse antonyms are two words that have a reverse relationship.

Example:

Take the words *buy* and *sell*. These words are related and have a reverse relationship. If one person buys something, another person sells something.

RELATIONS AMONG LEXEMES AND THEIR SENSES

Hypernym and Hyponymy

Word that are subset of another word are called Hyponymy.

Words that are superset of another are called Hypernym.

Sense is said to be Hyponym if the first sense is more specific, denoting a subclass of the other.

For example

- Rose is hyponym of flower
- Spoon is hyponym of utensil

Conversely, we say that

- *Flower* is a hypernym of *rose*
- *Utensil* is a hypernym of *spoon*

WORDNET

WordNet is the lexical database i.e. dictionary for the English language, specifically designed for natural language processing.

Properties:

- Synonyms are grouped together in something called **Synset**
- A **synset** contains lemmas, which are the base form of a word

WORDNET

Synset is a special kind of a simple interface that is present in NLTK to look up words in WordNet. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets).

Part of speech	No. synsets
noun	82115
verb	13767
adjective	18156
adverb	3621

Relation	Also called	Definition	Example
Hypernym	Superordinate	From concepts to superordinates	*breakfast*^{1} $\longrightarrow$ *meal*^{1}
Hyponym	Subordinate	From concepts to subtypes	*meal*^{1} $\longrightarrow$ *lunch*^{1}
Member Meronym	Has-Member	From groups to their members	*faculty*^{2} $\longrightarrow$ *professor*^{1}
Has-Instance		From concepts to instances of the concept	*composer*^{1} $\longrightarrow$ *Bach*^{1}
Instance		From instances to their concepts	*Austen*^{1} $\longrightarrow$ *author*^{1}
Member Holonym	Member-Of	From members to their groups	*copilot*^{1} $\longrightarrow$ *crew*^{1}
Part Meronym	Has-Part	From wholes to parts	*table*^{2} $\longrightarrow$ *leg*^{3}
Part Holonym	Part-Of	From parts to wholes	*course*^{7} $\longrightarrow$ *meal*^{1}
Antonym		Opposites	*leader*^{1} $\longrightarrow$ *follower*^{1}

The noun "bass" has 8 senses in WordNet.

1. bass^{1} - (the lowest part of the musical range)
2. bass^{2}, bass part^{1} - (the lowest part in polyphonic music)
3. bass^{3}, basso^{1} - (an adult male singer with the lowest voice)
4. sea bass^{1}, bass^{4} - (the lean flesh of a saltwater fish of the family Serranidae)
5. freshwater bass^{1}, bass^{5} - (any of various North American freshwater fish with lean flesh (especially of the genus Micropterus))
6. bass^{6}, bass voice^{1}, basso^{2} - (the lowest adult male singing voice)
7. bass^{7} - (the member with the lowest range of a family of musical instruments)
8. bass^{8} - (nontechnical name for any of numerous edible marine and freshwater spiny-finned fishes)

The adjective "bass" has 1 sense in WordNet.

1. bass^{1}, deep^{6} - (having or denoting a low vocal or instrumental range) *"a deep voice"; "a bass voice is lower than a baritone voice"; "a bass clarinet"*

SEMANTIC ANALYSIS

Semantic analysis is the task of ensuring that the declarations and statements of a program are semantically correct i.e, that their meaning is clear and consistent with the way in which control structures and data types are supposed to be used.

Semantic Analyzer:

it can find semantic errors that occur because of the following mistakes:

- Names that aren't declared
- Operands of the wrong type for the operator they're used with

COMPOSITIONAL SEMANTICS:

- The meaning of the whole is made up of the meaning of its parts

  -George Cooks, Dan eats, Dan is Sick

  Cook(George) Eat(Dan) Sick (Dan)

  -If George cooks and Dan eats. Dan will get sick

  (Cook(George)^ eat (Dan)) -> Sick(Dan) Sick(Dan) -> (Cook(George)

- To incorporate semantics into grammar we must have to figure out following like right representation for a single constituent based on the part of that constituent and for a category constituent based on other grammar rules making use of that constituent

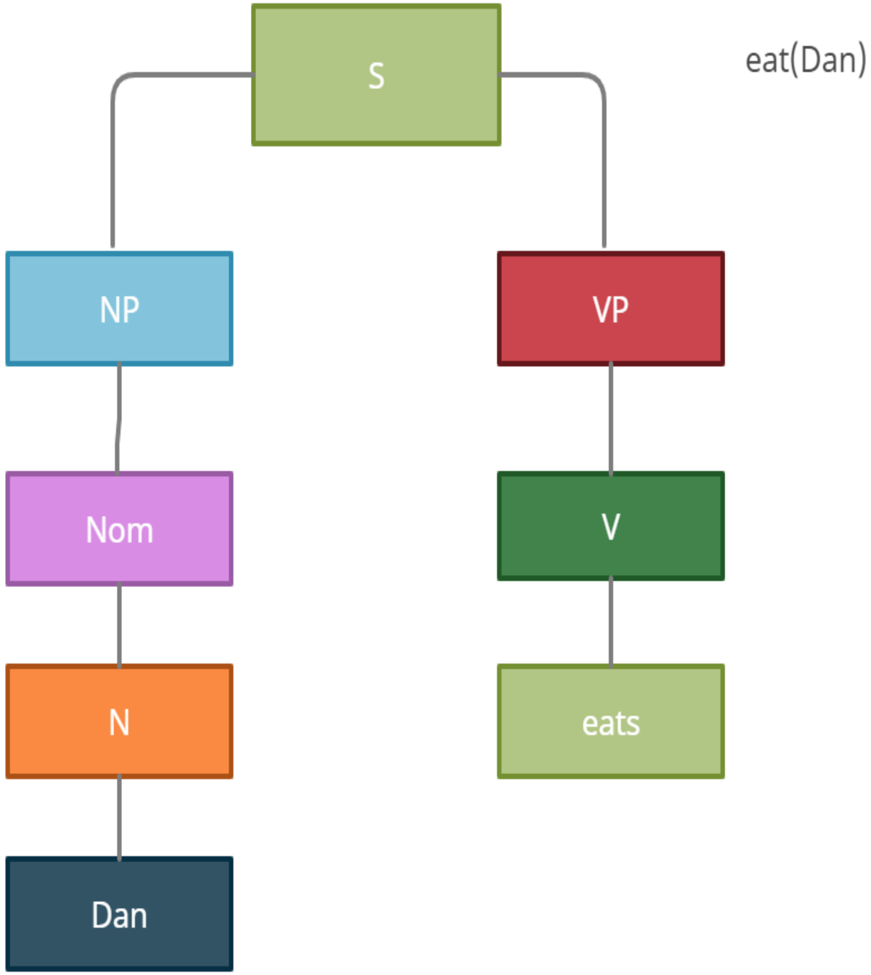
  e.g. Nom→ Adj Nom

- And this give us a function like semantics attachments incorporated into our CFG

  E.g.

  Nom→ Adj Nom{λx Nom.sem(x)^Isa(x,Adj.sem)}

SYNTAX- DRIVEN SEMANTICS



WHAT DO WE DO WITH THEM?

- As we did with feature structure:

  alter an early-style parser so when

  constituents are completed the attached

  semantic function applied and meaning

  representation created and stored with state
- And let parser run to completion and then walk through resulting tree running semantic attachments from bottom-up

EXAMPLE

AyCaramba serves meat

- Associating Constants with constituents
  - ProperNoun -> AyCaramba { AyCaramba}
  - MassNoun -> meat {Meat}
- Defining function to produce this from input
  - NP -> ProperNoun {properNoun.sem}
  - NP -> MassNoun{MassNoun.sem}

  -Assumption: meaning reps of children are passed up to the parents for non branching constuents
- Verb here are action

- V-> serves {E(e,x,y) Isa(e Serving) ^ Server(e,x) ^ Served(e,y)}
- Will every verb have its own distinct representation?
- Predicate(Agent, Patient)

- How do we combine these Pieces?
  - VP -> V NP
  - Goal : E(e,x) Isa(e, Serving) ^Server(e,x) ^ Served(e, Meat)

-VP semantics must tell us

- Which vars to be replaced by which args
- How this replacement is done

LAMBDA NOTATION

- Extension to FOPC

  λ x P(x)

  λ + variable(s) + FOPC expression in those variables

- Lambda binding
  - Apply lambda-expression to logical terms to bind lambda-expression’s parameters to terms (**lambda reduction**)
  - Simple process: substitute terms for variables in lambda expression

  λxP(x)(car)

  P(car)

CONT…

- Lambda notation provides requisite verb semantics

  - Formal parameter list makes variables within the body of the logical expression available for binding to external arguments.

  - Lambda reduction implements the replacement

- Semantic attachment for

  -V→ serves{V.sem(NP.sem)}

  {E(e,x,y) Isa(e,Serving)^Server(e, y) ^Served(e,x)}

  becomes

  {λx E(e,y) Isa(e,Serving)^Server(e, y) ^Served(e,x)}

  - Now ‘x’ is available to be bound when V.sem is applied to NP.sem

CONT…

- λ-application binds x to value of NP.sem(Meat)
- λ-reduction replaces x within λ-expression to Meat
- Value of VP.Sem becomes:

{E(e,y) Isa(e,Serving)^Server(e, y) ^Served(e,Meat)}

Similarly we need a semantic attachment for S →NP VP

{VP.sem(NP.sem)} to add the subject NP to our semantic representation of

AyCaramba serves meat

- We need another λ-expression in the value of VP.sem
- But currently V.sem does not give us one- So, we change V.sem to include another λ-expression
- V →serves

{λx λy E(e) Isa(e,Serving)^Server(e, y) ^Served(e,x)}

CONT…

- VP semantics(V.sem(NP.sem) binds the outer λ-expression to the object NP (Meat) but leaves the inner λ-expression for subsequent binding to the subject NP when the semantics of S is determined :

{E(e)Isa(e,Serving)^Server(e, AyCaramba)^Served(e,Meat)}

Option1 (Integrated Semantics Analysis)

S -> NP VP {VP.sem(NP.sem)}

- VP. Sem has been stored in stats representing VP
- Np.sem stored with the state for NP
- When rule completed, go get value of VP.sem, go get NP.sem, and apply VP.sem to NP. Sem
- Store result in S.sem

- As fragments of input parsed, Semantic fragmented created
- Can be used to block ambiguous representation

NON- COMPOSITIONAL LANGUAGE

- What we do with language whose meaning is not derive from the meaning of its parts

  -Metaphor: You are the cream in my Coffee

  - She’s the cream in George’s coffee

  - The break-in was just the tip of the iceberg

  - This was only the tip of Shirley’s iceberg

  - **Idioms:** The old man Finally kicked the bucket.

  -The old man finally kicked the proverbial bucket

- Solution?

  - Mix lexical item with special grammar rules?

SEMANTIC ATTACHMENTS

- The attachments can be thought of as instructions that specify way to compute the meaning representation of a construction from the meanings of its constituents parts.

- The notation of semantic attachments states that the meaning representation assigned to a construction.

SENTENCES

This section expands our coverage to include the other sentence types. Let's start by considering the following examples:

Flight 487 serves lunch.

Serve lunch.

Does flight 207 serve lunch?

Which flights serve lunch?

Declarative sentences:

We can simply alter the basic sentence rule :

S-> NP VP { DCL(Vp.sem(NP.sem))} ∴ DCL is an operator

Imperative Sentences:

It begins with a verb phrase and lack an overt subject.

S-> VP {IMP(VP.sem(Dummy You))} ∴ IMP is an operator

SENTENCES(CONT…)

❖YES OR NO QUESTIONS:

- Yes-no-questions consist of a sentence initial auxiliary verb, followed by a subject noun phrase and verb phrase.
- The following asemantic attachment simply ignores the auxiliary and with the exception of the YNQ operator, constructs the same representation that would created for the corresponding declarative sentence.

S-> Aux NP VP { YNQ(VP.sem(NP.sem))}

❖WH-SUBJECT-QUESTIONS:

- Wh-subject-questions ask for specific information about the subject of the sentence rather than the sentence as a whole.

S-> Wh Word NP VP { WHQ(NP.sem,var,VP.sem(NP.sem)))}

- WHQ is an operator and we can add variable x tells that what type of question it is.

2. NOUN PHRASES

Compound Nominals: Compound Nominals also known as noun-noun sequences consist of simple sequences of nouns .

Example:

- Flight schedule
- Summer flight schedule

Both sentences contain only the nouns. The word schedule is taken as a head of the noun and remaining noun denoted as NN.

$\lambda x.Schedule(x) \wedge NN(x, Flight)$

$\lambda x.Schedule(x) \wedge NN(x, Flight) \wedge NN(x, Summer)$

2. NOUN PHRASES

Genitive Noun Phrases Genitive Noun Phrases make use of complex determiners that consist of noun phrases with possessive markers, as in ‘Atlanta’s airport’ and ‘Maharani’s menu’.

NP → ComplexDet Nominal

$\{<\exists x Nominal.sem(x) \land GN(x, ComplexDet.sem)>\}$

Applying these rules to Atlanta’s airport results in the following complex-term:

$< \exists x Isa(x, Airport) \land GN(x, Atlanta) >$

3.Verb Phrases

λ-expression attached to the verb is simply applied to the semantic attachments of the verb's arguments.

Infinitive Verb Phrases :

I told Harry to go to Maharani.

The meaning representation for this example should be something like the following:

$$\exists e,f,x\, Isa(e,Telling) \wedge Isa(f,Going) \wedge Teller(e,Speaker) \wedge Tellee(e,Harry) \wedge ToldThing(e, f) \wedge Goer(f,Harry) \wedge Destination(f,x)$$

IMPLEMENTATION

Meaning and Synonym

```
[27] from nltk.corpus import wordnet   #Import wordnet from the NLTK
     synset = wordnet.synsets("Best")
     print('Word and Type : ' + synset[0].name())
     print('Synonym of Best is: ' + synset[0].lemmas()[0].name())
     print('The meaning of the word : ' + synset[0].definition())
     print('Example of Journey : ' + str(synset[0].examples()))
```

```
Word and Type : best.n.01
Synonym of Best is: best
The meaning of the word : the supreme effort one can make
Example of Journey : ['they did their best']
```

SYNONYMS AND ANTONYMS

```
[28] import nltk
     from nltk.corpus import wordnet   #Import wordnet from the NLTK
     syn = list()
     ant = list()
     for synset in wordnet.synsets("Best"):
        for lemma in synset.lemmas():
           syn.append(lemma.name())    #add the synonyms
           if lemma.antonyms():    #When antonyms are available, add them into the list
              ant.append(lemma.antonyms()[0].name())
     print('Synonyms: ' + str(syn))
     print('Antonyms: ' + str(ant))
```

```
Synonyms: ['best', 'best', 'topper', 'Best', 'C._H._Best', 'Charles_Herbert_Best', 'outdo', 'outflank', 'trump', 'best', 'scoop', 'best', 'better', 'best', 'good'
Antonyms: ['worst', 'worst', 'bad', 'evil', 'ill', 'ill', 'badly', 'badly', 'disadvantageously', 'badly']
```

SUMMING UP

- Principle of Compositionality
- Semantics of NL sentences and phrases can be composed from the semantics of their subparts
- Rules can be derived which map syntactic analysis to semantic representation (Rule to Rule Hypothesis)
  - Lambda notation provides a way to extend FOPC to this end
  - But coming up with rule to rule mappings is hard
- Wordnet is a large database of lexical relation for English words