Module 3

Semantics-Representing Meaning-canonical forms-FOPC-ambiguity resolution-scoping phenomena

What is Semantics?

Semantics is the study of meaning. In computational linguistics, it's about how to represent and reason with the meaning of words, phrases, and sentences in a way a computer can process.

- **Syntax vs. Semantics**
  - **Syntax**: The grammatical structure of a sentence. Is it well-formed?
    - `Colorless green ideas sleep furiously.` (Syntactically PERFECT, semantically NONSENSE).
  - **Semantics**: The meaning of a sentence. Is it sensible? What does it entail?
    - `The car hit the tree.` (Semantically valid, describes an event).

The ultimate goal is to build a representation of meaning from the text.

The Need for a Meaning Representation

We can't just use the raw sentences. We need a formal, unambiguous structure.

Requirements for a Meaning Representation:

1. **Verifiability**: The ability to check if the representation is true or false given a Knowledge Base (KB).
2. **Unambiguous Representation**: A single, agreed-upon structure for a specific meaning.
3. **Canonical Form**: Different sentences with the same meaning should map to the same representation.
4. **Inference and Reasoning**: The ability to derive new knowledge from what's already known

Canonical Forms: One Meaning, One Form

Different sentences can have the same core meaning. A good representation captures this.

Canonical Form is the standardized representation of a specific meaning.

- **Example**:
    1. "Maria serves pasta."
    2. "Pasta is served by Maria."
    3. "It is Maria who serves the pasta."

All three should map to the **same logical form** in our system.A plausible FOPC canonical form could be:

$\exists e, Serving(e) \land Server(e, Maria) \land Served(e, Pasta)$

Representing Meaning: First-Order Predicate Calculus(FOPC)

First-Order Predicate Calculus (FOPC) is a powerful and expressive formal language for representing meaning. It gives us the tools to model objects, properties, and relations.

Key Idea: We can model the world as:

- **Objects**: Entities like people, places, things (e.g., *John, London, book*).
- **Properties**: Characteristics of objects (e.g., *red, tall, prime*).
- **Relations**: Connections between objects (e.g., *likes, located_in, older_than*).

FOPC: The Building Blocks

Constants: Specific objects in the world.

- `John`, `Mary`, `Table1`

Predicates: Properties of objects or relations between them.

- `Tall(John)` -> John has the property of being tall.
- `Likes(John, Mary)` -> There is a 'likes' relation between John and Mary.

Variables: Stand-ins for objects.

- `x`, `y`, `z`

Logical Connectives: $\land$ (and), $\lor$ (or), $\neg$ (not), $\Rightarrow$ (implies).

FOPC: The Building Blocks

Quantifiers:

- ∀ **Universal Quantifier** ("for all"): $\forall x,\ Human(x) \Rightarrow Mortal(x)$
  - "All humans are mortal."
- ∃ **Existential Quantifier** ("there exists"): $\exists y,\ Cat(y) \land Black(y)$
  - "There exists a cat that is black." or "Some cat is black."

Ambiguity: The Core Problem in NLP

A single surface form (sentence) can have multiple possible meanings. Resolving this is a primary goal of semantic analysis.

- **Lexical Ambiguity**: A single word has multiple meanings.
  - "The fishermen went to the **bank**." (River bank or financial institution?)
- **Syntactic (Structural) Ambiguity**: A sentence has multiple valid parse trees.
  - "I saw the man with the telescope." (Who has the telescope? Me or the man?)
- **Scope Ambiguity**: The scope of quantifiers like 'every', 'a', 'some' is unclear.

Scoping Phenomena

Quantifier Scope Ambiguity is one of the most famous and difficult semantic challenges.

- **Example Sentence**: "Every student took a test."

This sentence has two distinct logical interpretations (readings):

1. **Reading 1**: `∀s (Student(s) ⇒ ∃t (Test(t) ∧ Took(s, t)))`
   - "For every student, there exists some test that they took."
   - This is the most common interpretation. Each student took a test, but they could all be *different* tests.
2. **Reading 2**: `∃t (Test(t) ∧ ∀s (Student(s) ⇒ Took(s, t)))`
   - "There exists one specific test that every student took."
   - This implies all students took the *same* test.