**NATURAL LANGUAGE PROCESSING ASSIGNMENT**

**Compositional Semantics**

**GROUP 4**

**Name Id**

**1, aweke adisie -------------------1306207**

**2, misgan wedajie-----------------1306388**

**3, masresha ayele-----------------1201332**

**4, mekuria werede----------------1302051**

**5, Tsedenya ysihak----------------1302931**

Submission date :- 5/31/2025

**Compositional Semantics**

# Theory and Practical Implementation

Compositional semantics is a theory in linguistics and natural language processing that proposes that the meaning of a sentence or phrase is derived by combining the meanings of its individual parts, such as words, phrases, and clauses, according to specific rules. Essentially, the meaning of a complex expression is built up from the meanings of its simpler components.

### Principle of Compositionality in Semantics

The principle of compositionality, often attributed to Gottlob Frege, states that the meaning of a complex expression (such as a sentence) is determined by the meanings of its constituent parts and the rules used to combine them. In other words, the meaning of a whole is a function of the meanings of its parts and their syntactic structure. This principle is foundational in formal semantics and linguistics, as it provides a systematic way to understand how language conveys meaning.

The principle of compositionality states that the meaning of a whole is determined by the meanings of its parts and the way they are combined.

Key aspects of the principle include:

* **Modularity**: Each word or phrase contributes a specific meaning to the sentence.
* **Modularity**: Each word or phrase contributes a specific meaning to the sentence.
* **Syntactic Structure**: The way words are combined (e.g., subject-verb-object order) influences the overall meaning.
* **Predictability**: By knowing the meanings of individual words and the rules of combination, one can predict the meaning of new sentences

For example, in the sentence "The cat chased the dog," the meaning is derived from:

* The meaning of "cat" (a feline animal).
* The meaning of "chased" (an action of pursuit).
* The meaning of "dog" (a canine animal).
* The syntactic structure, where "cat" is the subject, "chased" is the verb, and "dog" is the object, indicating that the cat is performing the action of chasing directed at the dog.

Example2, Consider the sentence "John ate a ripe apple." Compositional semantics suggests that the meaning of this sentence is derived by combining the meanings of "John," "ate," "a," "ripe," and "apple," along with their grammatical relationships within the sentence.

### Deriving Sentence Meaning from Its Parts

The meaning of a sentence is constructed by combining the meanings of its parts according to syntactic rules. This process can be broken down as follows:

**1, Lexical Semantics**: Each word in the sentence has a lexical meaning. For instance, nouns like "cat" and "dog" refer to entities, while verbs like "chased" describe actions or relations between entities.

Compositional semantics is often contrasted with lexical semantics, which focuses on the individual meanings of words. While lexical semantics deals with the meanings of individual words, compositional semantics examines how those meanings interact to form the meaning of larger units like phrases and sentences.

**2, Syntactic Combination**: The grammar of the language dictates how words are combined. In English, a common structure is Subject-Verb-Object (SVO). The syntactic structure determines the roles each word plays (e.g., who is performing the action and who is receiving it).

**3, Semantic Composition**: The meanings of individual words are combined based on their syntactic roles. For example, in "The cat chased the dog":

* "The cat" identifies the agent (the doer of the action).
* "Chased" describes the action.
* "The dog" identifies the patient (the entity affected by the action).
* Together, these form a proposition: an event where the cat pursues the dog.

**4 , Context and Pragmatics**: While compositionality focuses on the literal meaning derived from syntax and lexical items, context can add nuances (e.g., whether "chased" implies a playful or aggressive act).

Compositional semantics is often contrasted with lexical semantics, which focuses on the individual meanings of words. While lexical semantics deals with the meanings of individual words, compositional semantics examines how those meanings interact to form the meaning of larger units like phrases and sentences.

In formal semantics, this process is often represented using logical forms or lambda calculus, where meanings are composed systematically. For instance, the verb "chased" might be represented as a function that takes a subject and object to produce a truth value (e.g., chase(cat, dog) = True if the event occurred).

### Challenges and Limitations

While the principle of compositionality is powerful, it has limitations:

* **Idioms**: Phrases like "kick the bucket" have meanings that cannot be derived from the literal meanings of their parts.
* **Ambiguity**: Sentences like "I saw the man with the telescope" can have multiple meanings based on syntactic structure.
* **Context Dependency**: Words like "bank" can have different meanings depending on context (e.g., riverbank vs. financial institution).

Despite these challenges, compositionality remains a cornerstone of semantic analysis, enabling systematic interpretation of language.

Providing fully compositional treatments of language, especially complex constructions like idioms, can be challenging, as some meanings seem to go beyond a simple combination of parts.

**Applications in NLP:**

Compositional semantics is a core concept in natural language processing, used in tasks like sentence representation, document representation, and relational path representation.

## Practical Task: Compositional Meaning Extractor in Python

### Goal

The goal is to build a Python system that demonstrates compositional semantics by extracting the subject, verb, and object from a simple sentence and generating a logical representation of its meaning.

### Features

**Extracts Subject, Verb, and Object**: Identifies the key components of a sentence using natural language processing (NLP).

**Logs a Logical Representation**: Outputs a structured representation of the sentence’s meaning (e.g., verb(subject, object)).

**Implementation**

import spacy

import logging

from typing import Dict, Tuple, Optional

# Set up logging

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

logger = logging.getLogger(\_\_name\_\_)

class CompositionalMeaningExtractor:

    def \_\_init\_\_(self, model\_name: str = "en\_core\_web\_sm"):

        """Initialize the NLP model with a specified model name."""

        try:

            self.nlp = spacy.load(model\_name)

            logger.info(f"Successfully loaded spaCy model: {model\_name}")

        except Exception as e:

            logger.error(f"Failed to load spaCy model: {e}")

            raise

    def extract\_svo(self, sentence: str) -> Tuple[Optional[str], Optional[str], Optional[str]]:

        """Extract subject, verb, and object from a sentence with enhanced dependency parsing."""

        try:

            doc = self.nlp(sentence)

            subject = None

            verb = None

            obj = None

            for token in doc:

                # Identify main verb (ROOT, VERB)

                if token.dep\_ == "ROOT" and token.pos\_ == "VERB":

                    verb = token.lemma\_  # Use lemma for normalized verb form

                    logger.info(f"Identified verb: {verb}")

                    # Find subject (nsubj or nsubjpass for passive sentences)

                    for child in token.children:

                        if child.dep\_ in ("nsubj", "nsubjpass"):

                            # Include compound nouns (e.g., "The big dog")

                            subject = self.\_get\_compound\_phrase(child)

                            logger.info(f"Identified subject: {subject}")

                        # Find object (dobj or pobj for prepositional objects)

                        if child.dep\_ in ("dobj", "pobj"):

                            obj = self.\_get\_compound\_phrase(child)

                            logger.info(f"Identified object: {obj}")

            if not all([subject, verb, obj]):

                logger.warning(f"Incomplete SVO structure in sentence: {sentence}")

                return None, None, None

            return subject, verb, obj

        except Exception as e:

            logger.error(f"Error processing sentence '{sentence}': {e}")

            return None, None, None

    def \_get\_compound\_phrase(self, token: spacy.tokens.Token) -> str:

        """Extract compound phrases (e.g., 'The big dog') for a token."""

        compound = []

        for t in token.subtree:

            if t.dep\_ in ("compound", "amod", "det") or t == token:

                compound.append(t.text)

        return " ".join(compound) if compound else token.text

    def get\_logical\_representation(self, subject: Optional[str], verb: Optional[str], object\_: Optional[str]) -> Optional[str]:

        """Generate a logical representation with additional semantic roles."""

        if all([subject, verb, object\_]):

            logical\_form = f"{verb}({subject}, {object\_})"

            logger.info(f"Logical representation: {logical\_form}")

            return logical\_form

        else:

            logger.warning("Cannot generate logical representation due to missing components.")

            return None

    def extract\_entities(self, sentence: str) -> Dict[str, list]:

        """Extract named entities from the sentence."""

        try:

            doc = self.nlp(sentence)

            entities = {"PERSON": [], "ORG": [], "GPE": [], "OTHER": []}

            for ent in doc.ents:

                if ent.label\_ in entities:

                    entities[ent.label\_].append(ent.text)

                else:

                    entities["OTHER"].append(ent.text)

            logger.info(f"Extracted entities: {entities}")

            return entities

        except Exception as e:

            logger.error(f"Error extracting entities from '{sentence}': {e}")

            return {}

    def process\_sentence(self, sentence: str) -> Dict:

        """Process a sentence to extract SVO, logical representation, and entities."""

        logger.info(f"Processing sentence: {sentence}")

        subject, verb, obj = self.extract\_svo(sentence)

        logical\_form = self.get\_logical\_representation(subject, verb, obj)

        entities = self.extract\_entities(sentence)

        return {

            "sentence": sentence,

            "subject": subject,

            "verb": verb,

            "object": obj,

            "logical\_form": logical\_form,

            "entities": entities

        }

    def batch\_process(self, sentences: list) -> list:

        """Process multiple sentences and return results."""

        return [self.process\_sentence(sentence) for sentence in sentences]

def main():

    """Main function to demonstrate the Compositional Meaning Extractor."""

    extractor = CompositionalMeaningExtractor()

    # Extended test sentences with varied structures

    sentences = [

        "The big cat chased the small dog in the park.",

        "Mary quickly ate a juicy apple.",

        "The young boy kicked the red ball.",

        "This is not a valid sentence.",  # Non-SVO sentence

        "Elon Musk founded SpaceX in 2002.",

        "The book was read by Alice.",  # Passive voice

        "A group of tourists visited New York City."

    ]

    results = extractor.batch\_process(sentences)

    for result in results:

        print("\nResult:")

        print(f"Sentence: {result['sentence']}")

        print(f"Subject: {result['subject']}")

        print(f"Verb: {result['verb']}")

        print(f"Object: {result['object']}")

        print(f"Logical Form: {result['logical\_form']}")

        print(f"Entities: {result['entities']}")

if \_\_name\_\_ == "\_\_main\_\_":

    main()