### 1. Lexical Semantics:

#### - ****Definition (1 Mark):****

* **Lexical Semantics** deals with the meaning of words and how their meanings relate to each other in a language. It explores the meanings of individual words and the relationships between them.

#### - ****Key Concepts (1 Mark):****

* **Sense Relations:** Lexical semantics examines relationships between the senses of words, such as synonymy (similar meanings), antonymy (opposite meanings), hyponymy (subtype relationships), and hypernymy (supertype relationships).
* **Word Polysemy:** Polysemy refers to the phenomenon where a single word has multiple related meanings.

#### - ****Applications (1 Mark):****

* **Information Retrieval:** Understanding the nuances of word meanings is crucial for improving the accuracy of search engines.
* **Sentiment Analysis:** Lexical semantics aids in determining the sentiment of a piece of text by analyzing the meanings of words used.

#### - ****Recent Trends (1 Mark):****

* **Embeddings and Word Vectors:** Recent trends involve the use of word embeddings and vectors to capture semantic relationships, with techniques like Word2Vec and GloVe.
* **Cross-lingual Lexical Semantics:** With the growth of global applications, there is an increased focus on understanding and representing word meanings across different languages.

#### - ****Tools (1 Mark):****

* **NLTK (Natural Language Toolkit):** Python library providing tools for working with human language data, including lexical semantic analysis.
* **WordNet:** Lexical database of the English language that relates words to one another in terms of synonyms, hypernyms, and hyponyms.

### 2. Word-Sense Disambiguation (WSD):

#### - ****Definition (1 Mark):****

* **Word-Sense Disambiguation (WSD)** is the task of determining the correct sense of a word in a given context, particularly when a word has multiple meanings.

#### - ****Key Concepts (1 Mark):****

* **Ambiguity:** WSD addresses the inherent ambiguity of language where a word can have different meanings in different contexts.
* **Supervised and Unsupervised Approaches:** WSD methods can be supervised (using labeled training data) or unsupervised (making use of context and distributional information).

#### - ****Algorithmic Steps (1 Mark):****

* **Preprocessing:** Tokenize and process the input text.
* **Contextual Analysis:** Examine the surrounding words and context to determine the appropriate sense.
* **Machine Learning Approaches:** Use supervised learning algorithms like Naive Bayes or decision trees for WSD.

#### - ****Applications (1 Mark):****

* **Machine Translation:** WSD improves the accuracy of machine translation systems by selecting the correct meaning of words.
* **Information Retrieval:** Enhance search engine performance by understanding and disambiguating user queries.

#### - ****Recent Trends (1 Mark):****

* **Deep Learning for WSD:** Recent trends involve the application of deep learning techniques, including neural networks and transformers, for more accurate WSD.
* **Domain-Specific WSD:** Customizing WSD models for specific domains to improve accuracy in specialized contexts.

#### - ****Tools (1 Mark):****

* **Lesk Algorithm:** A traditional algorithm for WSD based on the overlap of word senses in dictionary definitions.
* **WordNet Interface in NLTK:** NLTK provides tools for WSD using WordNet as a lexical resource.

### 1. Semantic Role Labeling:

#### - ****Definition (1 Mark):****

* **Semantic Role Labeling (SRL)** is a natural language processing task that involves identifying and classifying the roles of words in a sentence, such as the predicate, agent, theme, etc.

#### - ****Key Concepts (1 Mark):****

* **Roles:** SRL assigns roles to words, indicating their functions in a sentence, such as who is doing the action (agent) and what is being affected (theme).
* **FrameNet and PropBank:** SRL often uses resources like FrameNet and PropBank for annotated examples of frames and roles.

#### - ****Algorithmic Steps (1 Mark):****

* **Dependency Parsing:** Obtain the syntactic structure of the sentence using techniques like dependency parsing.
* **Frame Identification:** Identify the frames (predicates) in the sentence.
* **Role Labeling:** Assign roles to the words based on their syntactic and semantic relationships.

#### - ****Applications (1 Mark):****

* **Question Answering:** SRL assists in understanding the relationships between entities in a sentence, contributing to more accurate question answering.
* **Information Extraction:** Enhance the extraction of structured information from unstructured text.

#### - ****Recent Trends (1 Mark):****

* **Neural Networks for SRL:** Recent trends involve the use of deep learning models, such as recurrent neural networks (RNNs) and transformers, to improve SRL accuracy.
* **Multilingual SRL:** Advances in making SRL models effective across multiple languages.

-Pseudo code:

# Pseudo code for a simple SRL using spaCy

import spacy

# Load English model with SRL capabilities

nlp = spacy.load("en\_core\_web\_sm")

# Process text

text = "The cat chased the mouse."

doc = nlp(text)

# Extract semantic roles

for token in doc:

print(f"{token.text}: {token.dep\_}, {token.head.text}")

#### - ****Tools:****

* **spaCy:** An open-source NLP library in Python that provides SRL capabilities.
* **AllenNLP:** A powerful library for NLP tasks, including pre-trained models for SRL.

### 2. Semantic Parsing:

#### - ****Definition (1 Mark):****

* **Semantic Parsing** is the process of converting natural language expressions into formal representations, such as logical forms or executable queries.

#### - ****Key Concepts (1 Mark):****

* **Formal Representations:** Semantic parsing aims to generate structured representations that capture the meaning of natural language expressions.
* **Grammar-based Approaches:** Utilize grammatical rules to map natural language to logical forms.

#### - ****Algorithmic Steps (1 Mark):****

* **Syntactic Analysis:** Parse the input sentence to understand its grammatical structure.
* **Semantic Role Labeling:** Identify the semantic roles of words to establish relationships.
* **Formal Representation Generation:** Map the syntactic and semantic information into a structured representation.

#### - ****Applications (1 Mark):****

* **Virtual Assistants:** Semantic parsing is crucial for understanding and executing user queries in virtual assistants.
* **Database Querying:** Convert natural language queries into SQL queries for database retrieval.

#### - ****Recent Trends (1 Mark):****

* **Neural Semantic Parsing:** The integration of neural network models, such as transformers, to improve the accuracy of semantic parsing.
* **Zero-shot Learning:** Advances in enabling semantic parsers to understand and generate queries for new or unseen domains.

#### - ****Pseudo Code:****

# Pseudo code for a simple semantic parsing using NLTK

import nltk

# Define a grammar for semantic parsing

grammar = nltk.CFG.fromstring("""

S -> NP VP

VP -> V NP

NP -> Det N

Det -> 'the' | 'a'

N -> 'cat' | 'mouse'

V -> 'chased'

""")

# Create a parser

parser = nltk.ChartParser(grammar)

# Parse a sentence

sentence = "the cat chased the mouse"

for tree in parser.parse(sentence.split()):

print(tree)

#### - ****Tools:****

* **NLTK (Natural Language Toolkit):** A comprehensive library for natural language processing tasks, including semantic parsing tools.
* **PyTorch and TensorFlow:** Deep learning frameworks that can be used to implement neural models for semantic parsing.

### 1. Reference Resolution:

#### - ****Definition (1 Mark):****

* **Reference Resolution** is a natural language processing task that involves determining the entities or concepts to which pronouns and other referring expressions refer within a given context.

#### - ****Key Concepts (1 Mark):****

* **Anaphora and Cataphora:** Reference resolution addresses anaphoric references (backward, e.g., pronouns) and cataphoric references (forward, e.g., demonstratives).
* **Coreference Chains:** Establishing links between expressions that refer to the same entity forms coreference chains.

#### - ****Algorithmic Steps (1 Mark):****

* **Mention Detection:** Identify mentions of entities or concepts in the text.
* **Antecedent Resolution:** Determine the antecedent or the entity to which a pronoun or expression refers.
* **Coreference Resolution:** Form coreference chains by linking related mentions.

#### - ****Applications (1 Mark):****

* **Text Summarization:** Improving the coherence and clarity of generated summaries by resolving references.
* **Question Answering:** Enhancing the accuracy of answers by understanding and resolving references.

#### - ****Recent Trends (1 Mark):****

* **Neural Approaches:** Recent trends involve the use of neural network-based models, such as neural coreference resolution, to capture complex relationships.
* **Pre-trained Language Models:** Integration of pre-trained language models like BERT and GPT for improved reference resolution.

#### - ****Pseudo Code:****

# Pseudo code for a simple anaphora resolution using spaCy

import spacy

# Load English model with coreference resolution capabilities

nlp = spacy.load("en\_coref\_lg")

# Process text

text = "John met Mary. He gave her a gift."

doc = nlp(text)

# Access coreference resolution

for ent in doc.ents:

print(f"{ent.text}: {ent.\_.coref\_clusters}")

#### - ****Tools:****

* **spaCy:** An open-source NLP library that provides reference resolution capabilities.
* **Stanford CoreNLP:** A suite of natural language processing tools that includes coreference resolution.

### 2. Discourse Phenomena:

#### - ****Definition (1 Mark):****

* **Discourse Phenomena** refer to patterns and structures in language beyond the sentence level, including how sentences connect, relate, and form coherent texts.

#### - ****Key Concepts (1 Mark):****

* **Coherence and Cohesion:** Ensuring that sentences within a discourse are logically connected and cohesive.
* **Discourse Markers:** Words or phrases that signal relationships between sentences or parts of text.

#### - ****Algorithmic Steps (1 Mark):****

* **Rhetorical Structure Theory (RST):** RST models the hierarchical structure of discourse, representing relationships between text spans.
* **Connective Detection:** Identify discourse markers and connectives that signal relationships.

#### - ****Applications (1 Mark):****

* **Text Summarization:** Improving the quality of generated summaries by considering discourse structure.
* **Dialogue Systems:** Enhancing the coherence of responses in conversational agents.

#### - ****Recent Trends (1 Mark):****

* **Deep Learning for Discourse Parsing:** Recent trends involve the application of deep learning techniques, such as transformers, for more accurate discourse parsing.
* **Genre-Specific Discourse Analysis:** Tailoring discourse analysis models for specific genres, such as scientific papers or news articles.

#### - ****Pseudo Code:****

# Pseudo code for discourse marker detection using NLTK

import nltk

# Tokenize and tag the input text

text = "However, the results indicate that..."

tokens = nltk.word\_tokenize(text)

tags = nltk.pos\_tag(tokens)

# Identify discourse markers

discourse\_markers = [word for word, tag in tags if tag == 'RB']

print(discourse\_markers)

#### - ****Tools:****

* **NLTK (Natural Language Toolkit):** A comprehensive library for natural language processing tasks, including tools for discourse analysis.
* **RST Discourse Parser:** A parser based on Rhetorical Structure Theory for discourse analysis.

### 1. Syntactic Constraints on Coreference:

#### - ****Definition (1 Mark):****

* **Syntactic Constraints on Coreference** refer to the grammatical and structural rules that dictate how pronouns or noun phrases refer to entities within a text.

#### - ****Key Concepts (1 Mark):****

* **Binding Theory:** In linguistics, Binding Theory provides rules for understanding how pronouns are bound to antecedents.
* **Grammatical Number and Gender:** Matching syntactic features such as number (singular/plural) and gender between anaphors and their antecedents.

#### - ****Diagram:****

* Syntactic Tree: Illustrate a syntactic tree representing the hierarchical structure of a sentence, emphasizing the positions of potential antecedents and anaphors.

#### - ****Algorithmic Steps (1 Mark):****

* **Parsing:** Use syntactic parsers to generate the tree structures that represent the grammatical relationships in a sentence.
* **Agreement Checking:** Implement rules to check for agreement in features such as number, gender, and person between potential antecedents and anaphors.

#### - ****Applications (1 Mark):****

* **Machine Translation:** Ensuring correct translation by considering syntactic constraints for accurate pronoun resolution.
* **Text Summarization:** Enhancing the coherence of generated summaries by adhering to syntactic rules in coreference.

#### - ****Recent Trends (1 Mark):****

* **Neural Syntax Models:** Incorporating neural syntax models to capture complex syntactic structures and improve coreference resolution.
* **Cross-Lingual Coreference:** Recent advancements in handling coreference across multiple languages, considering diverse syntactic structures.

#### - ****Pseudo Code:****

python

# Pseudo code for syntactic coreference resolution using spaCy

import spacy

# Load English model with coreference resolution capabilities

nlp = spacy.load("en\_coref\_lg")

# Process text

text = "John saw Mary. He greeted her."

doc = nlp(text)

# Access syntactic coreference resolution

for ent in doc.ents:

print(f"{ent.text}: {ent.\_.coref\_clusters}")

#### - ****Tools:****

* **spaCy:** An open-source NLP library with syntactic parsing and coreference resolution capabilities.
* **Stanford CoreNLP:** A suite of natural language processing tools that includes syntactic parsers.

### 2. Semantic Constraints on Coreference:

#### - ****Definition (1 Mark):****

* **Semantic Constraints on Coreference** involve the rules and principles governing how pronouns or noun phrases refer to entities based on their semantic meaning.

#### - ****Key Concepts (1 Mark):****

* **Ontological Categories:** Matching entities based on their semantic categories, such as person, organization, or location.
* **Word Sense Disambiguation:** Resolving ambiguous references by considering the specific senses of words.

#### - ****Diagram:****

* Semantic Network: Illustrate a semantic network representing relationships between entities and their attributes, guiding coreference resolution.

#### - ****Algorithmic Steps (1 Mark):****

* **Entity Recognition:** Identify and classify entities in the text based on their semantic categories.
* **Semantic Similarity:** Compute semantic similarity scores between potential antecedents and anaphors, considering their ontological categories.

#### - ****Applications (1 Mark):****

* **Information Extraction:** Improving the accuracy of extracting structured information by resolving coreference based on semantic meaning.
* **Chatbots and Virtual Assistants:** Enhancing the naturalness of conversations by considering semantic constraints in coreference resolution.

#### - ****Recent Trends (1 Mark):****

* **Contextualized Word Embeddings:** Recent trends involve leveraging contextualized word embeddings (e.g., BERT) for better capturing semantic nuances in coreference.
* **Multimodal Coreference:** Integrating information from different modalities, such as images and text, to enhance semantic understanding in coreference resolution.

#### - ****Pseudo Code:****

python

# Pseudo code for semantic coreference resolution using spaCy

import spacy

# Load English model with coreference resolution capabilities

nlp = spacy.load("en\_coref\_lg")

# Process text

text = "Apple announced the new iPhone. It features a powerful camera."

doc = nlp(text)

# Access semantic coreference resolution

for ent in doc.ents:

print(f"{ent.text}: {ent.\_.coref\_clusters}")

#### - ****Tools:****

* **spaCy:** An open-source NLP library with semantic parsing capabilities for coreference resolution.
* **WordNet Interface in NLTK:** NLTK provides tools for semantic analysis, including semantic constraints in coreference.

### Pronoun Resolution Algorithm:

#### 1. ****Tokenization and Part-of-Speech Tagging:****

* Tokenize the input text into individual words or tokens.
* Perform part-of-speech tagging to assign grammatical categories to each token (e.g., noun, pronoun).

#### 2. ****Identification of Pronouns:****

* Identify pronouns in the text based on their part-of-speech tags.

#### 3. ****Mention Detection:****

* Identify all potential antecedents or entities in the text.
* Consider entities from the current sentence and preceding sentences as potential antecedents.

#### 4. ****Gender and Number Agreement:****

* Consider gender and number agreement between the pronoun and potential antecedents.
* Eliminate candidates that do not match the gender and number features of the pronoun.

#### 5. ****Syntactic Constraints:****

* Leverage syntactic information, such as sentence structure and grammatical relationships, to further narrow down potential antecedents.
* Apply syntactic rules, considering the hierarchical structure of the sentence.

#### 6. ****Semantic Constraints:****

* Utilize semantic information to refine antecedent selection.
* Consider semantic relationships and ontological categories to improve accuracy.

#### 7. ****Contextual Analysis:****

* Analyze the context surrounding the pronoun to capture contextual cues.
* Take into account nearby words, phrases, and discourse markers for a more comprehensive understanding.

#### 8. ****Scoring Mechanism:****

* Assign scores or weights to potential antecedents based on various features (e.g., syntactic, semantic, contextual).
* Rank potential antecedents according to their scores.

#### 9. ****Selection of Antecedent:****

* Choose the antecedent with the highest score as the resolved referent for the pronoun.

#### 10. ****Output:****

* Return the resolved pronoun-antecedent pair or indicate if resolution is not possible.

### Pseudo Code Example (using NLTK):

python

import nltk

def pronoun\_resolution(text):

tokens = nltk.word\_tokenize(text)

pos\_tags = nltk.pos\_tag(tokens)

pronouns = [word for word, tag in pos\_tags if tag == 'PRP']

resolved\_pairs = []

for pronoun in pronouns:

antecedents = find\_potential\_antecedents(pronoun, tokens, pos\_tags)

resolved\_antecedent = resolve\_pronoun(pronoun, antecedents)

resolved\_pairs.append((pronoun, resolved\_antecedent))

return resolved\_pairs

def find\_potential\_antecedents(pronoun, tokens, pos\_tags):

# Implement logic to find potential antecedents based on syntactic and semantic constraints

# Consider gender and number agreement, sentence structure, and semantic relationships

pass

def resolve\_pronoun(pronoun, antecedents):

# Implement scoring mechanism and select the antecedent with the highest score

pass

# Example Usage

text = "He saw Mary, and she greeted him."

resolved\_pairs = pronoun\_resolution(text)

print(resolved\_pairs)

### Note:

* The actual implementation details of functions like find\_potential\_antecedents and resolve\_pronoun will depend on the specific rules and features you want to incorporate.
* Advanced models, such as machine learning-based approaches using neural networks, can also be employed for pronoun resolution to capture more complex patterns and improve accuracy.

### 1. ****Text Coherence:****

#### - ****Definition (1 Mark):****

* **Text Coherence** refers to the smooth and logical flow of ideas within a text, ensuring that sentences and paragraphs are connected and contribute to a unified meaning.

#### - ****Key Concepts (1 Mark):****

* **Logical Ordering:** Arranging sentences and paragraphs in a way that follows a clear sequence or logic.
* **Cohesive Devices:** Using transitional words, phrases, and coherence markers to signal relationships between ideas.

#### - ****Diagram:****

* Flowchart: Illustrate a flowchart representing the logical progression of ideas in a coherent text.

#### - ****Algorithmic Steps (1 Mark):****

* **Topic Sentences:** Begin paragraphs with clear topic sentences that convey the main idea.
* **Transitional Phrases:** Use words like "however," "therefore," or "moreover" to guide readers through the logical flow.
* **Pronoun Reference:** Ensure consistent and clear reference to entities through pronouns.

#### - ****Applications (1 Mark):****

* **Text Summarization:** Coherent texts are crucial for generating concise and meaningful summaries.
* **Machine Translation:** Improving the coherence of translated texts for better understanding.

#### - ****Recent Trends (1 Mark):****

* **Neural Models for Coherence:** Application of neural network models, such as transformers, for capturing and generating coherent text.
* **Cross-Modal Coherence:** Ensuring coherence in multimodal content, where text is combined with images or other modalities.

#### - ****Pseudo Code:****

python

# Pseudo code for coherence using NLTK

from nltk import sent\_tokenize

def check\_coherence(text):

sentences = sent\_tokenize(text)

for i in range(1, len(sentences)):

coherence\_score = measure\_coherence(sentences[i-1], sentences[i])

if coherence\_score < threshold:

print(f"Warning: Lack of coherence between '{sentences[i-1]}' and '{sentences[i]}'")

def measure\_coherence(sentence1, sentence2):

# Implement coherence measurement logic (e.g., similarity score)

pass

# Example Usage

text = "First sentence. Second sentence follows."

check\_coherence(text)

#### - ****Tools:****

* **NLTK (Natural Language Toolkit):** Provides tools for coherence measurement and analysis.
* **GPT (Generative Pre-trained Transformer):** Advanced language models like GPT-3 can generate coherent text based on input prompts.

### 2. ****Discourse Structure:****

#### - ****Definition (1 Mark):****

* **Discourse Structure** refers to the organization and arrangement of sentences and paragraphs to form a cohesive and meaningful discourse.

#### - ****Key Concepts (1 Mark):****

* **Rhetorical Structure Theory (RST):** A theory that models how texts are structured hierarchically, with different layers of relationships between text spans.
* **Coherence Relations:** The relationships between sentences, such as cause-effect, contrast, or elaboration.

#### - ****Diagram:****

* RST Tree: Illustrate a tree diagram representing the hierarchical structure of a discourse according to RST.

#### - ****Algorithmic Steps (1 Mark):****

* **Segmentation:** Identify and segment the text into coherent units, such as paragraphs or sections.
* **Connective Detection:** Identify and analyze connectives or discourse markers that signal relationships between segments.
* **RST Parsing:** Use RST-based parsers to generate a hierarchical representation of the discourse structure.

#### - ****Applications (1 Mark):****

* **Text Generation:** Understanding discourse structure is crucial for generating coherent and logically organized text.
* **Information Extraction:** Extracting structured information from a document by considering the discourse organization.

#### - ****Recent Trends (1 Mark):****

* **Neural Discourse Parsing:** Application of neural network models for more accurate and context-aware discourse parsing.
* **Genre-Specific Analysis:** Customizing discourse analysis models for specific genres, such as academic papers or news articles.

#### - ****Pseudo Code:****

python

# Pseudo code for RST parsing using NLTK

from nltk import RSTParser

def parse\_discourse\_structure(text):

parser = RSTParser()

discourse\_tree = parser.parse(text)

return discourse\_tree

# Example Usage

text = "Introduction. Methods. Results. Conclusion."

discourse\_structure = parse\_discourse\_structure(text)

#### - ****Tools:****

* **NLTK (Natural Language Toolkit):** Provides tools for RST parsing and discourse analysis.
* **Stanford CoreNLP:** A suite of NLP tools, including discourse parsers, for analyzing text structure.