R18

Code No: 155CK

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY
HYDERABAD B. Tech III Year I Semester Examinations, August - 2022
NATURAL LANGUAGE PROCESSING
(Computer Science and Engineering)

Time: 3 hours **Max. Marks: 75**

Answer any five questions
All questions carry equal marks

1.a) Discuss the structure and components of words.
b) Explain any one Morphological model. [7+8]

2.a) Discuss the Complexity of the approaches in NLP.
b) Explain the Performance of various methods in NLP. [7+8]

3.a) Explain about data driven approach to syntax.
b) Give an overview of parsing algorithms. [8+7]

4.a) Give an overview of various approach for syntactic representation.
b) What are the issues in multilingual syntactic analysis? [8+7]

5.a) Given there is a train on platform 6.
Its Destination is Hyderabad.
There is another train is in platform 7.
Its destination is Delhi.
Write Procedure for Anaphora Resolution.
b) Explain Word Sense Disambiguation. [8+7]

6.a) Explain Morphological structure.
b) What kind of softwares are available for semantics in NLP. [7+8]

7.a) Explain Predicate Argument Structure.
b) How many kinds of predicates are used to handle a basic statement? Give an example. [7+8]

8.a) How Parameter Estimation supports Language Modelling?
b) Differentiate Bilingual and Cross lingual Language Models. [7+8]

---ooOoo—

ANSWER KEY

1. a) Discuss the structure and components of words.

1. Root or Base: The core component of a word, often carrying the primary meaning. It's the element to which prefixes and suffixes are added to form derived or inflected forms. For example, in the word "playful," "play" is the root.
2. Prefixes: Preceding the root, prefixes are affixes added to the beginning of a word to modify its meaning. Examples include "un-" in "unhappy" and "re-" in "rewrite."
3. Suffixes: Following the root, suffixes are affixes added to the end of a word to alter its meaning or grammatical category. Examples include "-ing" in "running" and "-able" in "comfortable."
4. Stems: The main part of a word to which inflectional morphemes are added. While similar to roots, stems can include derivational affixes. For instance, in "worker," "work" is the stem.
5. Morphemes: The smallest units of meaning in a language. Words can consist of one or more morphemes. For example, "unhappiness" consists of three morphemes: "un-" (prefix), "happy" (root), and "-ness" (suffix).
6. Derivational Morphology: The process of forming new words by adding suffixes or modifying the base form, often changing the word's grammatical category or meaning. For instance, "teach" (verb) becomes "teacher" (noun) with the addition of the "-er" suffix.
7. Inflectional Morphology: The process of adding inflectional endings to words to indicate grammatical relationships, such as tense, number, case, or gender. For example, "walk" (present tense) becomes "walked" (past tense).

8. Compound Words: Words formed by combining two or more independent words. For example, "blackboard" is formed by combining "black" and "board."
9. Root Words: Basic lexical units from which words are derived. While some words consist solely of a root, others may have additional affixes. For example, "play" is a root in "playful," but "play" alone can also function as a word.
10. Lexical Categories: Words are categorized into different classes based on their grammatical properties and functions in sentences. The main lexical categories include nouns, verbs, adjectives, adverbs, pronouns, prepositions, conjunctions, and interjections. Each category has its own structural and semantic characteristics.

b) Explain any one Morphological model.

1. Basic Concept: FSMA is a rule-based morphological analysis approach that operates on the principle of finite-state automata. It involves breaking down words into their constituent morphemes using a series of finite-state machines.
2. Morpheme Recognition: FSMA recognizes morphemes by traversing through a series of states, each representing a particular morpheme or combination of morphemes. The model progresses through the states based on input symbols (e.g., characters) in the word.
3. Transducer Composition: FSMA employs transducer composition, where individual finite-state machines representing different morphemes are combined to form a single transducer capable of analyzing complex words.
4. Lexical Knowledge: The model relies on lexical knowledge encoded in the form of morphological rules, affixes, and their combinations. These rules define the valid morphological structures of words in a language.

5. Efficiency: FSMA is known for its computational efficiency due to the use of finite-state machines, which can efficiently process large volumes of text with minimal memory and time requirements.
6. Ambiguity Handling: The model can handle ambiguity in morphological analysis by utilizing prioritized rules or disambiguation strategies based on contextual information or statistical probabilities.
7. Language Independence: FSMA can be adapted to different languages by customizing the set of morphological rules and affixes according to the specific characteristics of each language's morphology.
8. Applications: FSMA is widely used in various natural language processing tasks, including stemming, lemmatization, part-of-speech tagging, information retrieval, and machine translation.
9. Error Handling: The model incorporates error-handling mechanisms to deal with input words that do not conform to the expected morphological patterns. This may involve fallback strategies or generating suggestions for correction.
10. Integration with Other Components: FSMA can be integrated with other NLP components, such as syntactic parsers or semantic analyzers, to provide morphologically informed input for downstream processing tasks, enhancing overall linguistic analysis and understanding.

2. a) Discuss the Complexity of the approaches in NLP.

1. Linguistic Complexity: NLP approaches must grapple with the inherent complexity of language itself, including ambiguity, figurative language, idiomatic expressions, and syntactic intricacies. Handling linguistic nuances poses a significant challenge in tasks such as semantic analysis and language generation.
2. Computational Complexity: Many NLP tasks involve processing large volumes of text data, which can lead to computational challenges such as

high memory consumption and long processing times. Techniques for optimizing algorithms and scaling to big data environments are necessary to address computational complexity.

3. Ambiguity Resolution: Ambiguity is pervasive in natural language, requiring NLP systems to disambiguate between multiple possible interpretations of a given input. This ambiguity may arise at various levels, including lexical ambiguity (homonymy, polysemy), syntactic ambiguity (parsing ambiguities), and semantic ambiguity (word sense disambiguation).
4. Domain-Specific Complexity: NLP tasks often encounter domain-specific complexities, where language usage and vocabulary vary across different domains or industries. Developing NLP models that generalize well across diverse domains while capturing domain-specific nuances poses a significant challenge.
5. Multimodal Complexity: With the rise of multimodal data (combining text, images, audio, etc.), NLP approaches must contend with the complexities of processing and integrating information from multiple modalities. This involves developing models that can effectively fuse and interpret information from diverse sources.
6. Cultural and Linguistic Diversity: NLP approaches need to accommodate the cultural and linguistic diversity of human languages worldwide. Building models that are robust across languages, dialects, and socio-cultural contexts presents a formidable challenge due to variations in grammar, vocabulary, and language usage.
7. Ethical and Societal Complexity: NLP systems raise ethical considerations related to bias, fairness, privacy, and accountability. Addressing these complexities requires careful design and evaluation to ensure that NLP technologies are developed and deployed responsibly, without perpetuating harmful biases or infringing on users' rights.

8. Interpretability and Explainability: As NLP models become increasingly complex (e.g., deep learning models), there is a growing need for interpretability and explainability. Understanding how these models arrive at their predictions is crucial for building trust and ensuring transparency in decision-making processes.
9. Resource Constraints: Limited availability of labeled data, computational resources, and domain expertise can pose significant challenges in developing and deploying NLP solutions, particularly in resource-constrained environments or low-resource languages.
10. Dynamic Nature of Language: Language is constantly evolving, with new words, expressions, and linguistic conventions emerging over time. NLP approaches must adapt to these changes and remain up-to-date to maintain their effectiveness and relevance in evolving linguistic landscapes.

b) Explain the Performance of various methods in NLP.

1. Accuracy: One of the primary metrics used to evaluate NLP methods is accuracy, which measures the proportion of correctly predicted outcomes (e.g., correct classification of text sentiment, accurate named entity recognition).
2. Precision and Recall: Precision measures the ratio of correctly predicted positive outcomes to all predicted positive outcomes, while recall measures the ratio of correctly predicted positive outcomes to all actual positive outcomes. These metrics are important for tasks like information retrieval and text classification.
3. F1 Score: The F1 score combines precision and recall into a single metric, providing a balanced measure of a model's performance, particularly in situations where there is an imbalance between positive and negative classes.

4. Perplexity: Perplexity is commonly used to evaluate language models, measuring how well a language model predicts a given sample of text. Lower perplexity values indicate better model performance.
5. BLEU Score: The Bilingual Evaluation Understudy (BLEU) score is used to evaluate the quality of machine-translated text by comparing it to human-generated reference translations. It measures the overlap between machine-generated and reference translations in terms of n-gram precision.
6. ROUGE Score: The Recall-Oriented Understudy for Gisting Evaluation (ROUGE) score assesses the quality of automatic summarization systems by comparing machine-generated summaries to human-generated reference summaries. It measures the overlap between the generated and reference summaries in terms of n-gram recall.
7. Mean Average Precision (MAP): MAP is commonly used to evaluate the performance of information retrieval systems, particularly in tasks such as document retrieval and question answering. It computes the average precision across multiple queries.
8. Word Error Rate (WER): WER is used to evaluate the performance of speech recognition systems by measuring the proportion of incorrectly recognized words in the transcribed output compared to the reference transcript.
9. Named Entity Recognition (NER) F1 Score: NER F1 score evaluates the performance of named entity recognition systems by considering precision, recall, and F1 score specifically for recognizing named entities such as persons, organizations, and locations in text.
10. Cross-Validation: Cross-validation techniques, such as k-fold cross-validation, are used to assess the generalization performance of NLP methods across different subsets of data. This helps in estimating the robustness and stability of the methods across diverse datasets.

3) a) Explain about data driven approach to syntax.

1. Data-Driven Approach: In syntax, a data-driven approach relies on empirical data, such as large corpora of annotated sentences, to derive syntactic structures and patterns, rather than relying solely on linguistic theories or rules.
2. Corpus-Based Analysis: Data-driven syntax involves analyzing syntactic structures and patterns within corpora of text, which may include parsing sentences, extracting syntactic features, and identifying recurring patterns.
3. Statistical Methods: Statistical techniques, such as probabilistic models and machine learning algorithms, are employed to analyze the frequency and distribution of syntactic structures within the corpus, allowing for the identification of common syntactic patterns.
4. Dependency Parsing: Dependency parsing is a common data-driven approach to syntax, where syntactic relationships between words are represented as directed edges in a dependency tree. Dependency parsers learn to predict these relationships based on observed patterns in annotated data.
5. Constituency Parsing: Constituency parsing is another data-driven technique where sentences are parsed into hierarchical structures known as constituents, such as phrases and clauses. Constituency parsers learn to predict these structures from annotated corpora using statistical models.
6. Supervised Learning: In supervised learning approaches, syntactic parsers are trained on labeled data, where sentences are annotated with syntactic structures (e.g., dependency trees or constituency trees). The parser learns to predict syntactic structures based on features extracted from the input text.

7. Unsupervised Learning: Unsupervised learning techniques involve discovering syntactic patterns and structures from unlabeled text data, without explicit annotations. These approaches may involve clustering sentences based on similarity or learning latent syntactic representations using techniques such as autoencoders.
8. Hybrid Approaches: Some data-driven syntax methods combine both supervised and unsupervised learning techniques to leverage the strengths of each approach. For example, a parser may be pre-trained on a large unlabeled corpus using unsupervised learning and fine-tuned on smaller labeled datasets using supervised learning.
9. Evaluation Metrics: Data-driven syntax models are evaluated using metrics such as parsing accuracy, precision, recall, and F1 score, which measure the agreement between predicted and gold-standard syntactic structures.
10. Applications: Data-driven syntax approaches find applications in various natural language processing tasks, including machine translation, information extraction, sentiment analysis, and question answering, where accurate syntactic analysis is crucial for understanding and processing text.

b) Give an overview of parsing algorithms.

1. Definition: Parsing algorithms are computational methods used in natural language processing to analyze the grammatical structure of sentences, typically represented as trees or graphs, based on formal grammar rules.
2. Top-Down vs. Bottom-Up: Parsing algorithms can be broadly classified into top-down and bottom-up approaches. Top-down parsers start from the root of the parse tree and recursively expand nodes to match the input sentence, while bottom-up parsers start from the input sentence and build

upward to form the parse tree.

3. Deterministic vs. Probabilistic: Parsing algorithms can also be categorized as deterministic or probabilistic. Deterministic parsers generate a single parse tree for a given sentence based on a predefined grammar, while probabilistic parsers assign probabilities to different parse trees based on statistical models trained on annotated data.
4. Chart Parsing: Chart parsing is a dynamic programming technique used in many parsing algorithms, such as Earley parser and CYK parser. It involves constructing a chart data structure to efficiently store and update partial parse results during the parsing process, facilitating reusability and optimization.
5. Earley Algorithm: The Earley algorithm is a top-down chart parsing algorithm that operates in linear time for unambiguous grammars and cubic time for ambiguous grammars. It uses dynamic programming to efficiently handle ambiguity and recursion in context-free grammars.
6. CYK Algorithm: The Cocke-Younger-Kasami (CYK) algorithm is a bottom-up chart parsing algorithm that operates in cubic time for unambiguous grammars and potentially exponential time for ambiguous grammars. It employs dynamic programming to construct a parse table based on the grammar's Chomsky normal form.
7. Shift-Reduce Parsing: Shift-reduce parsing is a type of bottom-up parsing algorithm commonly used in transition-based dependency parsers. It involves shifting input tokens onto a stack and then reducing them using predefined transition rules until a parse tree is formed.
8. Transition-Based Parsing: Transition-based parsing is a class of parsing algorithms that operate by applying a sequence of transition actions to construct a parse tree incrementally. Examples include shift-reduce parsers and arc-eager dependency parsers, which efficiently handle the dependency relations between words in a sentence.

9. Constituency Parsing: Constituency parsing is the task of parsing sentences into hierarchical structures known as constituents, such as phrases and clauses. Constituency parsers employ algorithms such as recursive descent, CYK, and probabilistic context-free grammar (PCFG) parsing to generate constituency parse trees.
10. Dependency Parsing: Dependency parsing is the task of parsing sentences into directed graphs representing syntactic dependencies between words. Dependency parsers use algorithms such as transition-based parsing, graph-based parsing, and neural dependency parsing to generate dependency parse trees.

4) a) Give an overview of various approach for syntactic representation.

1. Constituency-Based Representation: Constituency-based approaches represent sentence structure in terms of hierarchical constituents, such as phrases and clauses. This representation, often depicted as a parse tree, captures the hierarchical organization of linguistic units within a sentence.
2. Dependency-Based Representation: Dependency-based approaches represent sentence structure in terms of directed dependencies between words, where each word is linked to its syntactic head or governing word. This representation captures the relationships and dependencies among words in a sentence.
3. Phrase Structure Grammar (PSG): PSG is a formal grammar framework used for constituency-based syntactic representation. It defines rules for generating hierarchical phrase structures, where each rule specifies how constituents can be combined to form larger constituents.
4. Dependency Grammar (DG): DG is a formal grammar framework used for dependency-based syntactic representation. It defines rules for generating directed dependencies between words, where each rule

specifies a syntactic relation between a dependent word and its head word.

5. Lexical Functional Grammar (LFG): LFG is a linguistic theory that combines both constituency-based and dependency-based approaches to syntactic representation. It represents sentence structure in terms of functional relationships between words and phrases, capturing both hierarchical and dependency relations.
6. Head-Driven Phrase Structure Grammar (HPSG): HPSG is a type of PSG that focuses on the notion of head words, which govern the structure and properties of phrases. It represents sentence structure in terms of hierarchical phrase structures where each phrase is headed by a particular word.
7. Tree Adjoining Grammar (TAG): TAG is a formal grammar framework that represents sentence structure in terms of elementary trees, which can be combined through tree adjoining operations to form larger trees. It captures the local and structural relationships between constituents in a sentence.
8. Dependency Parsing: Dependency parsing is a computational approach to syntactic representation that involves automatically analyzing sentences and generating dependency parse trees representing syntactic relationships between words. It is widely used in natural language processing tasks such as parsing and information extraction.
9. Constituency Parsing: Constituency parsing is a computational approach to syntactic representation that involves automatically analyzing sentences and generating constituency parse trees representing hierarchical phrase structures. It is used in tasks such as parsing, machine translation, and grammar checking.
10. Graph-Based Representation: Graph-based approaches represent sentence structure as a graph, where nodes correspond to words or constituents and

edges represent syntactic relationships between them. This representation allows for a flexible and versatile encoding of sentence structure, suitable for various linguistic analyses and applications.

b) What are the issues in multilingual syntactic analysis?

1. Cross-Linguistic Variation: Multilingual syntactic analysis encounters challenges due to the diverse syntactic structures and rules across languages. Each language may exhibit unique word order, morphological features, and syntactic phenomena, complicating the development of universal parsing models.
2. Word Order Differences: Languages vary in their word order preferences, with some languages exhibiting strict word order constraints (e.g., SVO in English) while others allow more flexibility (e.g., free word order in languages like Latin). Accommodating these differences poses challenges for syntactic parsers.
3. Morphological Complexity: Languages differ in their morphological complexity, with some languages having rich inflectional morphology (e.g., case markings, verb conjugations) while others rely more on word order or auxiliary verbs to convey grammatical information. Parsing morphologically rich languages requires handling complex morphological variations.
4. Parsing Ambiguity: Multilingual parsing faces increased parsing ambiguity due to the potential for structural ambiguity across languages. Ambiguous sentences may have multiple valid interpretations, making it challenging for parsers to select the correct syntactic analysis.
5. Resource Scarcity: Availability of annotated data and linguistic resources varies across languages, with some languages having limited or no annotated corpora for training syntactic parsers. Resource scarcity hinders the development of accurate multilingual parsing models, particularly for

low-resource languages.

6. Code-Switching and Multilingualism: Multilingual text often involves code-switching, where speakers switch between languages within the same sentence or discourse. Syntactic parsers need to handle code-switched text and account for the syntactic structures of multiple languages within a single analysis.
7. Domain Adaptation: Multilingual syntactic parsers may struggle with domain adaptation, as syntactic structures and language usage may vary across different domains and genres. Models trained on one domain may not generalize well to other domains, requiring domain-specific adaptation techniques.
8. Syntactic Typology: Syntactic typological differences between languages, such as head-initial vs. head-final structures, verb-second (V2) phenomena, or case marking systems, pose challenges for developing unified parsing models that can handle syntactic diversity effectively.
9. Transfer Learning: Leveraging knowledge from high-resource languages to improve parsing accuracy for low-resource languages through transfer learning poses challenges, as syntactic structures and linguistic features may not directly transfer across languages.
10. Evaluation and Benchmarking: Establishing consistent evaluation metrics and benchmark datasets for multilingual syntactic analysis is challenging due to linguistic diversity and the need for language-specific evaluation protocols. Ensuring fair and reliable evaluation across languages is crucial for assessing the performance of multilingual parsing models.

5.a) Given there is a train on platform 6.

Its Destination is Hyderabad.

There is another train is in platform 7.

Its destination is Delhi.

Write Procedure for Anaphora Resolution.

1. Identify Anaphoric Expressions: Scan the text to identify anaphoric expressions, which are words or phrases that refer back to previously mentioned entities. In this example, "Its" and "Its destination" are potential anaphoric expressions.
2. Determine Antecedents: Identify the potential antecedents of the anaphoric expressions, which are the entities being referred to. In this case, the antecedents are "train on platform 6" and "train in platform 7."
3. Analyze Context: Consider the context surrounding the anaphoric expressions and antecedents to understand the relationship between them. Look for clues such as proximity, syntactic structure, and semantic compatibility.
4. Resolve Ambiguities: If there are multiple potential antecedents for an anaphoric expression, resolve any ambiguities by considering factors such as grammatical agreement, discourse coherence, and plausibility.
5. Establish Coreference Chains: Create coreference chains to track the relationships between anaphoric expressions and their antecedents throughout the text. Each coreference chain represents a series of references to the same entity.
6. Apply Grammatical Constraints: Apply grammatical constraints to ensure that the anaphoric expression and its antecedent agree in gender, number, and person, where applicable. This helps in narrowing down the possible antecedents.
7. Consider Semantic Constraints: Consider semantic constraints to ensure that the anaphoric expression and its antecedent are semantically compatible. This involves assessing whether the entities being referred to make sense in the given context.
8. Evaluate Discourse Coherence: Evaluate discourse coherence to ensure that the resolution of anaphoric expressions maintains the flow and

coherence of the text. Ensure that the resolved references align with the overall narrative or discourse structure.

9. Check for Referential Ambiguities: Check for referential ambiguities where anaphoric expressions could refer to multiple entities or interpretations. Resolve these ambiguities based on contextual cues and linguistic constraints.
10. Verify Resolution Accuracy: Finally, verify the accuracy of the anaphora resolution by reviewing the resolved references in context and ensuring that they accurately reflect the intended meaning and interpretation of the text.

b) Explain Word Sense Disambiguation.

1. Definition: Word Sense Disambiguation (WSD) is the task of determining the correct meaning of a word with multiple possible interpretations, known as senses, within a given context.
2. Ambiguity: WSD addresses the inherent ambiguity present in natural language, where words can have different meanings depending on the context in which they are used. This ambiguity can arise due to polysemy (multiple meanings of a word) or homonymy (same form, different meanings).
3. Importance: WSD is crucial for various natural language processing tasks, including machine translation, information retrieval, question answering, and text summarization. It helps in improving the accuracy and precision of these systems by ensuring that the correct sense of a word is used in context.
4. Approaches: WSD can be approached using knowledge-based methods, supervised learning, unsupervised learning, or hybrid techniques. Knowledge-based methods rely on lexical resources such as dictionaries and thesauri, while supervised and unsupervised learning methods

leverage labeled training data or corpus statistics, respectively.

5. Lexical Resources: Lexical resources such as WordNet, a large lexical database of English, are commonly used in WSD. WordNet organizes words into synsets (sets of synonymous words) and provides semantic relations between them, facilitating the disambiguation process.
6. Supervised Learning: Supervised WSD models are trained on annotated datasets where each word occurrence is labeled with its correct sense. These models learn to predict the sense of a word based on features extracted from the context, such as surrounding words, syntactic patterns, and semantic relations.
7. Unsupervised Learning: Unsupervised WSD methods do not require labeled training data and instead rely on unsupervised clustering or distributional similarity techniques to group word occurrences into clusters representing different senses. These methods are based on the assumption that words with similar contexts tend to have similar meanings.
8. Evaluation Metrics: WSD systems are evaluated using metrics such as accuracy, precision, recall, and F1 score, which measure the agreement between predicted senses and gold-standard annotations. Evaluation datasets typically contain manually annotated instances of ambiguous words.
9. Challenges: WSD faces several challenges, including word sense granularity (fine-grained vs. coarse-grained senses), data sparsity, sense ambiguity, and domain adaptation. Resolving these challenges requires developing robust models that generalize well across different contexts and domains.
10. Applications: WSD has applications in various NLP tasks, including disambiguating search queries, improving machine translation quality, enhancing information retrieval systems, and aiding in text

summarization by ensuring the accurate interpretation of words in context.

6. a) Explain Morphological structure.

1. Definition: Morphological structure refers to the internal organization of words, including their constituent morphemes and their arrangement.
2. Morphemes: Morphemes are the smallest meaningful units of language, representing individual words or meaningful parts of words. They can be free morphemes, which can stand alone as words (e.g., "book"), or bound morphemes, which must be attached to other morphemes to convey meaning (e.g., "-s" for plural).
3. Roots and Affixes: Words are typically composed of one or more morphemes, with a root morpheme serving as the core lexical element and affixes modifying or extending its meaning. Affixes can be prefixes (attached at the beginning of a word), suffixes (attached at the end), or infixes (inserted within the word).
4. Derivational Morphology: Derivational morphology involves the creation of new words through the addition of affixes to existing roots. These affixes can change the word's part of speech, meaning, or grammatical function (e.g., "teach" becomes "teacher" with the addition of the "-er" suffix).
5. Inflectional Morphology: Inflectional morphology involves the modification of words to indicate grammatical relationships such as tense, aspect, mood, number, case, or gender. Inflectional affixes are typically bound morphemes that do not change the word's basic meaning (e.g., "walk" becomes "walked" to indicate past tense).
6. Morphological Processes: Morphological structure encompasses various processes such as compounding (joining two or more words to create a new word, e.g., "blackboard"), blending (combining parts of two words to

form a new word, e.g., "brunch"), and reduplication (repeating all or part of a word to convey emphasis or plurality, e.g., "bye-bye").

7. Analyzing Morphological Structure: Morphological analysis involves breaking down words into their constituent morphemes and identifying their individual meanings and grammatical functions. This process aids in understanding word formation, lexical meaning, and grammatical relationships within a language.
8. Morphological Typology: Morphological structure varies across languages, leading to different morphological typologies. Languages may exhibit agglutinative morphology (where affixes are added in a linear sequence), fusional morphology (where multiple grammatical features are fused into single affixes), or isolating morphology (where words are composed of single morphemes).
9. Productivity: Morphological structure reflects the productivity of morphological processes within a language, indicating the extent to which speakers can create new words and inflections using existing morphemes and patterns.
10. Importance: Understanding morphological structure is crucial for various aspects of language processing, including vocabulary acquisition, language learning, morphological analysis, and computational linguistics tasks such as morphological parsing and generation.

b) What kind of softwares are available for semantics in NLP.

1. Word Embedding Models: Software tools such as Word2Vec, GloVe, and FastText provide pre-trained word embeddings, which are dense vector representations of words in a semantic space. These embeddings capture semantic relationships between words and are widely used in NLP tasks.
2. Semantic Role Labeling (SRL) Systems: SRL software tools like AllenNLP and PropBank perform semantic analysis by identifying and

labeling the roles played by words in sentences, such as the agent, patient, or instrument. These systems help in understanding the underlying semantics of sentences.

3. Named Entity Recognition (NER) Tools: NER software, including spaCy and Stanford NER, identify and classify named entities such as persons, organizations, and locations in text. Understanding named entities is crucial for extracting semantic information from documents.
4. Semantic Parsing Systems: Semantic parsing software like Semantic Machines and AllenNLP's Semantic Role Labeler converts natural language queries or commands into formal representations such as logical forms or semantic graphs. These representations capture the meaning of the input text and enable systems to execute tasks based on the inferred semantics.
5. Ontology and Knowledge Graph Tools: Software tools like Protege and Neo4j enable the creation and management of ontologies and knowledge graphs, which organize semantic information about concepts and their relationships. These tools facilitate semantic reasoning and inference in NLP applications.
6. Semantic Similarity Measures: Software libraries such as Gensim and TensorFlow provide functions for computing semantic similarity between words, sentences, or documents based on word embeddings or other semantic representations. These measures quantify the degree of semantic relatedness between linguistic units.
7. Sentiment Analysis Tools: Sentiment analysis software like VADER and TextBlob analyze text to determine the sentiment expressed, such as positive, negative, or neutral. Understanding sentiment is a crucial aspect of semantic analysis in NLP, especially for applications like opinion mining and social media analysis.
8. Semantic Search Engines: Semantic search software, including

Elasticsearch with the Semantic Search Plugin and Apache Solr, enhance traditional keyword-based search by incorporating semantic understanding of queries and documents. These engines improve search accuracy by considering semantic relationships between terms.

9. Semantic Web Tools: Tools such as RDFLib and Apache Jena enable the creation, querying, and manipulation of Semantic Web data represented using RDF (Resource Description Framework) and SPARQL (SPARQL Protocol and RDF Query Language). These tools facilitate semantic data integration and querying in NLP applications.
10. Dialog Systems and Chatbots: Dialog systems and chatbot platforms like Dialogflow and Rasa utilize semantic understanding to interpret user intents and contextually generate responses. These systems employ natural language understanding (NLU) techniques to extract semantic meaning from user inputs and carry out meaningful interactions.

7. a) Explain Predicate Argument Structure.

1. Predicate Argument Structure (PAS) delineates the relationships between a predicate (typically a verb) and its associated arguments (noun phrases or other syntactic elements) within a sentence.
2. It elucidates how these arguments contribute semantically to the meaning conveyed by the predicate, detailing who or what performs the action, what is affected by it, and additional contextual information.
3. PAS analysis aids in identifying thematic or semantic roles assigned to each argument, such as agent, patient, experiencer, location, instrument, etc., shedding light on their functional roles within the predicate's context.
4. This structural framework offers insights into the underlying semantic structure of sentences, abstracting away from surface-level syntactic variations to reveal deeper semantic relationships.
5. PAS can be represented through various formalisms, including logical

forms, lambda calculus, or semantic graphs, facilitating the mapping of syntactic structures to their corresponding semantic representations.

6. The composition and nature of arguments associated with a predicate are contingent upon its lexical semantics and syntactic frame, resulting in diverse PAS configurations across different predicates.
7. Thorough PAS analysis is pivotal for numerous natural language understanding tasks, such as semantic parsing, information extraction, question answering, and machine translation, enhancing accuracy and depth of comprehension.
8. It plays a crucial role in disambiguating sentence interpretations by clarifying the roles and relationships of various arguments with respect to the predicate, resolving semantic ambiguities.
9. Common thematic roles identified in PAS analysis include the agent (entity performing the action), theme (entity undergoing the action), experiencer (entity perceiving or experiencing the action), location (site of the action), and instrument (means employed for the action).
10. Overall, Predicate Argument Structure provides a structured framework for deciphering the semantic organization of sentences, offering a deeper understanding of the interplay between predicates and their associated arguments in conveying meaning.

b) How many kinds of predicates are used to handle a basic statement? Give example.

1. Verbal Predicates: These predicates express actions or states performed by the subject. Example: "She runs every morning." In this sentence, "runs" is a verbal predicate indicating the action performed by "She."
2. Nominal Predicates: These predicates equate the subject with a noun or noun phrase, attributing a property or identity to the subject. Example: "She is a doctor." Here, "is" is a nominal predicate equating "She" with

the noun phrase "a doctor."

3. Adjectival Predicates: These predicates describe the subject by attributing an adjective or adjectival phrase to it. Example: "The sky is blue." In this sentence, "is blue" is an adjectival predicate describing the sky.
4. Adverbial Predicates: These predicates modify the action described by the subject with an adverb or adverbial phrase. Example: "She sings beautifully." Here, "sings beautifully" is an adverbial predicate modifying the action of "She sings."
5. Prepositional Predicates: These predicates consist of a preposition followed by a noun phrase, indicating a relationship between the subject and an object. Example: "He is in the room." In this sentence, "is in the room" is a prepositional predicate indicating the location of "He."
6. Existential Predicates: These predicates assert the existence of the subject or the presence of something. Example: "There is a cat on the roof." Here, "is" serves as the existential predicate indicating the existence of "a cat on the roof."
7. Copular Predicates: These predicates link the subject to its complement, indicating a relationship or identity between them. Example: "The book seems interesting." In this sentence, "seems interesting" is a copular predicate linking "The book" to its complement "interesting."
8. Modal Predicates: These predicates express modality, indicating possibility, necessity, or permission regarding the action described by the subject. Example: "You should study." Here, "should study" is a modal predicate expressing necessity.
9. Aspectual Predicates: These predicates specify the aspect of the action described by the subject, such as its duration, completion, or repetition. Example: "She has been working all day." In this sentence, "has been working" is an aspectual predicate indicating continuous action.
10. Attributive Predicates: These predicates attribute a quality or

characteristic to the subject, often with the help of a linking verb. Example: "The flowers smell sweet." Here, "smell sweet" is an attributive predicate describing the quality of the flowers.

8. a) How Parameter Estimation supports Language Modelling?

1. Parameter estimation in language modeling involves determining the optimal values for model parameters, such as probabilities or weights, based on observed data, which is crucial for building accurate and effective language models.
2. It allows language models to capture the statistical properties of natural language, such as word frequencies, co-occurrence patterns, and syntactic structures, by estimating the likelihood of different linguistic events or sequences.
3. By optimizing model parameters using observed data, parameter estimation helps in training language models to predict the next word in a sequence or generate coherent text that resembles natural language.
4. Parameter estimation techniques, such as maximum likelihood estimation (MLE) or maximum a posteriori (MAP) estimation, provide principled approaches for estimating model parameters from training data, ensuring that language models generalize well to unseen text.
5. It facilitates the adaptation of language models to different domains or contexts by re-estimating model parameters using domain-specific or contextual data, thereby improving model performance in specialized tasks or environments.
6. Parameter estimation supports the development of sophisticated language models, including n-gram models, neural language models (e.g., recurrent neural networks, transformers), and probabilistic graphical models, by enabling the fine-tuning of model parameters to enhance predictive accuracy and linguistic coherence.

7. It enables the incorporation of linguistic knowledge, domain expertise, or prior information into language models through parameter regularization techniques, which help prevent overfitting and improve model robustness.
8. Parameter estimation aids in evaluating and comparing different language modeling approaches by quantifying the fit of each model to the training data and assessing their performance on held-out or test data using metrics such as perplexity or accuracy.
9. It supports the optimization of language model architectures and hyperparameters, such as network topology, learning rates, or regularization parameters, by iteratively adjusting model parameters based on feedback from the training process.
10. Parameter estimation is essential for continuous learning or online updating of language models, allowing them to adapt to evolving linguistic patterns, user preferences, or changes in the underlying data distribution over time.

b) Differentiate Bilingual and Cross lingual Language Models.

1. Training Data: Bilingual language models are trained on data containing text from two languages, whereas cross-lingual language models are trained on multilingual data containing text from multiple languages.
2. Language Representation: Bilingual language models typically represent each language separately, with distinct parameters for each language, while cross-lingual language models aim to learn shared representations that capture language-agnostic features across multiple languages.
3. Language Independence: Bilingual language models are designed to handle specific language pairs and may not generalize well to languages outside those pairs, whereas cross-lingual language models aim to be language-independent, capable of processing text in any supported language.

4. Alignment Mechanism: Bilingual language models often rely on alignment techniques, such as parallel corpora or translation pairs, to learn correspondences between languages, while cross-lingual language models leverage techniques like multilingual pretraining or adversarial training to learn language-agnostic representations.
5. Applications: Bilingual language models are commonly used for tasks such as machine translation, where the goal is to translate text between two languages, while cross-lingual language models are employed for tasks like cross-lingual document classification, information retrieval, or zero-shot translation, where the model must operate across multiple languages.
6. Fine-tuning: Bilingual language models may require fine-tuning on specific language pairs to achieve optimal performance for translation tasks, whereas cross-lingual language models can often be directly applied to various languages without extensive fine-tuning.
7. Resource Requirements: Bilingual language models may require language-specific resources, such as parallel corpora or bilingual dictionaries, for training and evaluation, whereas cross-lingual language models can leverage multilingual data and resources, potentially reducing the need for language-specific resources.
8. Transfer Learning: Bilingual language models may not readily transfer knowledge between languages beyond their trained language pairs, whereas cross-lingual language models are explicitly designed for transfer learning across languages, enabling knowledge transfer and adaptation across diverse linguistic contexts.
9. Evaluation: Bilingual language models are typically evaluated on language-specific tasks or benchmarks, such as translation quality for machine translation, while cross-lingual language models are evaluated on tasks that require understanding or processing text across multiple

languages.

10. Scalability: Cross-lingual language models offer scalability advantages over bilingual models by allowing for the inclusion of additional languages without significant modifications to the model architecture or training procedure, making them more adaptable to multilingual environments and applications.

Code No: 155CK **R18**

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY
HYDERABAD B. Tech III Year I Semester Examinations, February - 2022
NATURAL LANGUAGE PROCESSING
(Computer Science and Engineering)

Time: 3 hours **Max. Marks: 75**

Answer any five questions

All questions carry equal marks

\- - -

1.a) Explain in detail about some early NLP Systems.

b) List and explain the issues and challenges of NLP System in detail. [8+7]

2.a) Discuss in detail about Complexity approaches of NLP Systems. [8+7]

b) What are the measures used to find the performances of the NLP Methods.

3.a) Consider the following training set and implement Bi-gram model to calculate the probability of given test sentence.

Training Set: I am a human

I am not a robot

I I live in Hyderabad

Test Sentence: I I am not

b) What are the Limitations in syntax parsing? [8+7]

4.a) Give all possible parse trees for the sentence, Stolen painting found by tree.

b) Explain any one Parsing Algorithm with the help of an example. [8+7]

5.a) State the types of references used in discourse analysis.

b) Describe in detail about word sense disambiguation with an example. [8+7]

6.a) Give example to explain the concept of predicate argument structure.

b) What are System paradigms in predicates? Explain. [8+7]

7.a) Explain Anaphora Resolution with the help of an example.

b) What kind of predicate structure are used for language modelling? [8+7]

8.a) Discuss in detail about the concept of Cohesion Structure.

b) List out popular lexical resources used in the processing of natural language text. [8+7]

---ooOoo—

ANSWER KEY

1. a) Explain in detail about some early NLP Systems.

1. SHRDLU: Developed in the late 1960s by Terry Winograd, SHRDLU was one of the earliest NLP systems that demonstrated natural language understanding and interaction capabilities. It operated in a block world environment where users could communicate with the system to manipulate blocks and build structures using natural language commands.
2. ELIZA: Created by Joseph Weizenbaum in the mid-1960s, ELIZA was an early example of a chatbot designed to simulate a Rogerian psychotherapist. It used simple pattern matching and substitution rules to engage users in conversation, demonstrating the potential of computers to simulate human-like interaction.
3. LUNAR: Developed in the 1960s at the Stanford Artificial Intelligence Laboratory, LUNAR was a natural language understanding system designed to analyze geological data from the Apollo moon missions. It parsed and interpreted textual descriptions of lunar rock samples to extract relevant information for scientific analysis.
4. XCON: Developed by John McDermott at Carnegie Mellon University in the 1970s, XCON was an early expert system designed for configuring

computer systems. It used a rule-based approach to analyze user requirements and generate customized configurations, showcasing the application of NLP techniques in knowledge-based systems.

5. STRIPS: Developed in the late 1960s at Stanford Research Institute (SRI) by Richard Fikes and Nils Nilsson, STRIPS (Stanford Research Institute Problem Solver) was an early planning system that used natural language input to specify goals and actions in a simulated robot world. It demonstrated automated planning and problem-solving capabilities using symbolic representation.
6. TAUM-METEO: Developed in the early 1970s at the University of Montreal, TAUM-METEO was a machine translation system designed to translate weather forecasts from English to French. It employed rule-based and statistical approaches to analyze and generate translations, pioneering research in machine translation.
7. SYSTRAN: Founded in the late 1960s by Peter Toma, SYSTRAN was one of the earliest commercial machine translation systems. It initially focused on translating technical documents for the European Space Agency and later expanded to provide translation services for various languages, demonstrating the practical applications of NLP technology.
8. SHRDLU: Developed in the late 1960s by Terry Winograd, SHRDLU was one of the earliest NLP systems that demonstrated natural language understanding and interaction capabilities. It operated in a block world environment where users could communicate with the system to manipulate blocks and build structures using natural language commands.
9. ELIZA: Created by Joseph Weizenbaum in the mid-1960s, ELIZA was an early example of a chatbot designed to simulate a Rogerian psychotherapist. It used simple pattern matching and substitution rules to engage users in conversation, demonstrating the potential of computers to simulate human-like interaction.

10. LUNAR: Developed in the 1960s at the Stanford Artificial Intelligence Laboratory, LUNAR was a natural language understanding system designed to analyze geological data from the Apollo moon missions. It parsed and interpreted textual descriptions of lunar rock samples to extract relevant information for scientific analysis.

b) List and explain the issues and challenges of the NLP System in detail.

1. Ambiguity: Natural language is inherently ambiguous, with words, phrases, and sentences often having multiple meanings or interpretations. Resolving ambiguity is a significant challenge for NLP systems, as they must accurately understand and interpret context-dependent language nuances.
2. Syntax Complexity: Syntax encompasses the rules governing sentence structure and grammar, which can vary widely across languages and dialects. NLP systems must contend with the complexity of syntax, including parsing sentences, identifying grammatical structures, and handling syntactic ambiguities.
3. Semantic Understanding: Understanding the meaning of text beyond its literal interpretation is a complex task for NLP systems. They must infer semantics, context, and intent from linguistic cues, such as word semantics, word co-occurrences, and discourse structure, to accurately comprehend natural language input.
4. Lack of Context: Context is crucial for interpreting language accurately, but NLP systems often struggle to capture and incorporate contextual information effectively. Understanding context-dependent meanings, references, and implications poses challenges, particularly in ambiguous or conversational contexts.
5. Data Limitations: NLP performance heavily relies on the availability and quality of annotated training data. Data scarcity, especially for

under-resourced languages or specialized domains, hinders the development of robust models and exacerbates biases and generalization issues.

6. Domain Adaptation: NLP systems often face challenges in adapting to specific domains or industries with distinct language patterns, terminology, and conventions. Domain adaptation requires specialized models, fine-tuning strategies, and domain-specific resources to achieve accurate performance.
7. Multilinguality: Handling multilingual text introduces complexities due to language diversity, structural variations, and translation challenges. Cross-lingual understanding, transfer learning, and resource constraints pose hurdles for developing effective multilingual NLP solutions.
8. Bias and Fairness: NLP systems can perpetuate biases present in training data, leading to unfair or discriminatory outcomes. Addressing bias, fairness, and ethical considerations is crucial for building inclusive, trustworthy, and socially responsible NLP technologies.
9. Interpretability and Transparency: The opaque nature of many NLP models raises concerns about interpretability and transparency. Ensuring that models are interpretable and transparent enables users to understand model behavior, identify errors, and trust NLP outputs in critical applications.
10. Real-world Deployment: Deploying NLP systems in real-world applications poses challenges such as scalability, computational efficiency, integration with existing systems, user interface design, and performance in dynamic and uncontrolled environments. Overcoming these challenges is essential for successful adoption and practical use of NLP technologies across diverse domains and industries.

2. a) Discuss in detail about Complexity approaches of NLP Systems.

1. Rule-Based Systems: Traditional NLP approaches often rely on rule-based systems, where linguistic rules and patterns are manually crafted to perform tasks such as part-of-speech tagging, parsing, and information extraction. These systems are characterized by their explicit representation of linguistic knowledge and rules, which can be labor-intensive to develop and maintain.
2. Statistical Methods: Statistical approaches in NLP leverage probabilistic models and machine learning algorithms to analyze and generate natural language. These methods involve learning patterns and relationships from large datasets, allowing NLP systems to make predictions based on statistical inference. While statistical methods can capture complex linguistic phenomena and adapt to diverse domains, they may struggle with data sparsity and require substantial amounts of annotated training data.
3. Machine Learning: Machine learning techniques, including supervised, unsupervised, and reinforcement learning, have gained prominence in NLP for tasks such as text classification, sentiment analysis, and machine translation. These approaches enable NLP systems to automatically learn patterns and representations from data, offering flexibility and scalability in modeling language complexity.
4. Deep Learning: Deep learning, particularly neural network architectures like recurrent neural networks (RNNs), convolutional neural networks (CNNs), and transformers, has revolutionized NLP by enabling the development of sophisticated models capable of handling large-scale language data. Deep learning approaches excel at capturing hierarchical and compositional representations of language, allowing NLP systems to learn complex patterns and dependencies.
5. Neural Language Models: Neural language models, such as word embeddings (e.g., Word2Vec, GloVe) and contextualized word

representations (e.g., ELMo, BERT), encode semantic and syntactic information into distributed vector representations. These models capture the contextual nuances of language and have become foundational for various NLP tasks, including text classification, named entity recognition, and question answering.

6. Transformer-Based Models: Transformer architectures, introduced by the Transformer model, have emerged as state-of-the-art approaches for many NLP tasks. Models like BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer) leverage self-attention mechanisms to capture long-range dependencies and contextual information, achieving remarkable performance across diverse language tasks.
7. Transfer Learning: Transfer learning techniques have become prevalent in NLP, allowing models pretrained on large-scale corpora (e.g., BERT, GPT) to be fine-tuned for downstream tasks with limited annotated data. Transfer learning facilitates knowledge transfer across tasks and domains, reducing the need for extensive task-specific training data and accelerating model development.
8. Multimodal Approaches: NLP systems increasingly incorporate multimodal information, such as text, images, and audio, to enhance language understanding and generation. Multimodal approaches leverage techniques from computer vision and speech processing to analyze and generate text in conjunction with other modalities, enabling richer and more contextually relevant interactions.
9. Hybrid Approaches: Hybrid approaches combine multiple NLP techniques, such as rule-based systems, statistical methods, and deep learning models, to leverage the strengths of each approach and mitigate their respective limitations. Hybrid systems aim to achieve robustness, accuracy, and efficiency by integrating complementary methods for

different aspects of language processing.

10. Ethical and Social Considerations: As NLP systems become increasingly sophisticated and ubiquitous, addressing ethical and social considerations, such as bias mitigation, fairness, transparency, and privacy protection, has become paramount. Complexities in NLP approaches extend beyond technical challenges to encompass broader societal impacts and ethical responsibilities associated with deploying language technologies.

b) What are the measures used to find the performances of the NLP Methods.

1. Accuracy: Accuracy measures the overall correctness of predictions made by an NLP method. It is commonly used for classification tasks, where the proportion of correctly classified instances is calculated.
2. Precision and Recall: Precision measures the proportion of correctly predicted positive instances among all instances predicted as positive, while recall measures the proportion of correctly predicted positive instances among all actual positive instances. These measures are particularly useful for tasks with imbalanced class distributions.
3. F1 Score: The F1 score is the harmonic mean of precision and recall and provides a balance between the two metrics. It is often used as a single summary metric for evaluating the performance of classifiers, especially when precision and recall are both important.
4. Perplexity: Perplexity is a measure of how well a language model predicts a sample of text. It quantifies the level of uncertainty or "surprise" associated with the model's predictions and is commonly used for evaluating language models, with lower perplexity values indicating better performance.
5. BLEU Score: The Bilingual Evaluation Understudy (BLEU) score is a metric used to evaluate the quality of machine-translated text by

comparing it to one or more reference translations. It measures the degree of overlap between n-grams (typically up to four) in the candidate translation and the reference translations.

6. ROUGE Score: The Recall-Oriented Understudy for Gisting Evaluation (ROUGE) score is a set of metrics used to evaluate the quality of summaries produced by text summarization systems. It measures the overlap between n-grams in the generated summary and n-grams in the reference summaries.
7. Mean Absolute Error (MAE): MAE measures the average absolute difference between predicted and actual values in regression tasks. It provides a straightforward measure of prediction accuracy, with lower MAE values indicating better performance.
8. Mean Squared Error (MSE): MSE measures the average squared difference between predicted and actual values in regression tasks. It penalizes larger errors more heavily than MAE and is commonly used as a loss function for training regression models.
9. Area Under the Receiver Operating Characteristic Curve (AUC-ROC): AUC-ROC measures the performance of binary classifiers across different thresholds by plotting the true positive rate against the false positive rate. It provides a single scalar value representing the classifier's ability to discriminate between positive and negative instances.
10. Mean Average Precision (MAP): MAP measures the average precision across multiple recall levels for information retrieval tasks. It quantifies the overall quality of the ranked list of retrieved documents, with higher values indicating better performance in returning relevant documents.

3. a) Consider the following training set and implement Bi-gram model to calculate the probability of given test sentence.

Training Set: I am a human

I am not a robot

I I live in Hyderabad

Test Sentence: I I am not

1. Tokenization: The first step is to tokenize both the training set and the test sentence into individual tokens or words. In this case, the training set consists of three sentences: "I am a human", "I am not a robot", and "I live in Hyderabad", while the test sentence is "I I am not".
2. Bi-gram Generation: Next, we generate bi-grams from the training set. Bi-grams are sequences of two adjacent tokens. For example, from the first sentence "I am a human", the bi-grams are ["I am", "am a", "a human"].
3. Frequency Counting: We count the frequency of each bi-gram in the training set. For instance, in the given training set, the bi-gram "I am" occurs twice.
4. Probability Calculation: Using the frequency counts, we calculate the probability of each bi-gram. The probability of a bi-gram is computed as the count of the bi-gram divided by the count of its preceding token. For example, the probability of "am" given the preceding token "I" is 2/2 = 1.
5. Smoothing: To handle unseen bi-grams in the test sentence, we apply smoothing techniques such as add-one smoothing or add-k smoothing. This ensures that all possible bi-grams have non-zero probabilities, even if they do not occur in the training set.
6. Applying the Bi-gram Model: Using the calculated probabilities, we compute the probability of the test sentence "I I am not" according to the bi-gram model. We multiply the probabilities of consecutive bi-grams in the test sentence. For instance, the probability of "I I am not" is the product of the probabilities of "I I", "I am", and "am not".
7. Result Interpretation: The final probability obtained from the bi-gram

model represents the likelihood of the test sentence occurring according to the language patterns observed in the training set.

8. Evaluation and Comparison: We can evaluate the performance of the bi-gram model by comparing its probability estimates with other language models or by assessing its performance on held-out test data.
9. Limitations: Bi-gram models have limitations, such as their inability to capture long-range dependencies in language and their reliance on local context. These limitations can affect the accuracy of probability estimates, particularly for complex or context-sensitive language.
10. Extensions: To improve the accuracy of language modeling, more sophisticated models such as n-gram models, neural language models, or transformer-based models can be explored. These models can capture higher-order dependencies in language and offer better performance on diverse text datasets.

b) What are the Limitations in syntax parsing?

1. Ambiguity: Syntax parsing often encounters ambiguity in natural language sentences, where a single sentence structure can have multiple valid parse trees. Resolving this ambiguity accurately is challenging, especially for complex sentences with nested clauses and dependencies.
2. Structural Complexity: Natural language sentences exhibit diverse syntactic structures, ranging from simple subject-verb-object constructions to complex, nested phrases and clauses. Parsing such varied structures accurately requires sophisticated algorithms capable of handling structural complexity effectively.
3. Cross-linguistic Variation: Syntax varies significantly across languages, with each language having its own set of syntactic rules, word order preferences, and grammatical structures. Developing syntax parsers that generalize well across different languages poses a significant challenge

due to these cross-linguistic variations.

4. Syntactic Ambiguity: Certain linguistic phenomena, such as prepositional phrase attachment ambiguity and coordination ambiguity, pose challenges for syntax parsers. Resolving these ambiguities requires context-sensitive analysis and disambiguation techniques, which may not always be accurate.
5. Parsing Errors: Syntax parsers may produce errors, including incomplete parses, incorrect tree structures, or mislabeled dependencies, especially for sentences with unusual or ungrammatical constructions. Handling parsing errors effectively is crucial for maintaining the reliability and robustness of NLP systems.
6. Long-range Dependencies: Syntax parsing often involves capturing long-range dependencies between distant words or phrases in a sentence. Traditional syntactic parsers may struggle to model such dependencies accurately, leading to parsing inaccuracies, particularly in sentences with complex syntactic structures.
7. Domain-specific Syntax: Syntax parsing in domain-specific texts, such as scientific literature or technical documents, presents additional challenges due to domain-specific terminology, syntax, and conventions. Adapting syntax parsers to handle domain-specific syntax effectively requires specialized training data and domain-specific knowledge.
8. Parsing Efficiency: Syntax parsing algorithms must be efficient enough to handle large volumes of text data quickly and accurately. Developing parsing algorithms that strike a balance between parsing speed and accuracy is essential for real-world applications of syntax parsing.
9. Parsing Ambiguity Resolution: Resolving syntactic ambiguities often involves making probabilistic decisions based on contextual cues and linguistic knowledge. However, accurately resolving ambiguities without additional context or disambiguating information can be challenging,

leading to parsing errors or inaccuracies.

10. Evaluation Challenges: Evaluating the performance of syntax parsers poses challenges due to the lack of standardized evaluation metrics and the subjective nature of syntactic annotations. Establishing reliable evaluation methodologies and benchmark datasets is crucial for comparing the performance of different syntax parsing approaches effectively.

4. a) Give all possible parse trees for the sentence, Stolen painting found by tree.

Creating all possible parse trees for the sentence "Stolen painting found by tree" involves considering different interpretations of the syntactic structure. Here are ten possible parse trees for the given sentence:

1. [Stolen [painting [found [by tree]]]]: In this parse tree, "Stolen" is the subject, "painting" is the object, and "found by tree" is the predicate.
2. [Stolen [painting [found by [tree]]]]: Here, "Stolen" is the subject, "painting" is the object, and "found by tree" is a phrasal verb.
3. [[Stolen painting] [found [by tree]]]: This structure treats "Stolen painting" as a noun phrase acting as the subject, with "found by tree" as the predicate.
4. [[Stolen [painting found]] [by tree]]: In this interpretation, "Stolen" modifies "painting found," and "by tree" functions as an adverbial phrase.
5. [[Stolen painting] [found by [tree]]]: Here, "Stolen painting" acts as the subject, while "found by tree" is the predicate.
6. [[Stolen [painting found by]] tree]: This structure treats "Stolen painting found by" as a noun phrase modified by "tree."
7. [[Stolen painting found] [by tree]]: In this parse tree, "Stolen painting found" functions as a compound verb, and "by tree" acts as an adverbial phrase.

8. [[Stolen painting] [found [by [tree]]]]: Here, "Stolen painting" serves as the subject, while "found by tree" is the predicate.
9. [[Stolen [painting found]] [by [tree]]]: This interpretation considers "Stolen painting found" as a compound verb, modified by "by tree."
10. [[Stolen [painting [found by]]] tree]: In this structure, "Stolen painting found by" functions as an adjective modifying "tree."

    These parse trees illustrate the various syntactic interpretations possible for the sentence "Stolen painting found by tree," highlighting the complexity and ambiguity inherent in natural language parsing.

b) Explain any one Parsing Algorithm with the help of an example.

1. Bottom-up Approach: CYK is a bottom-up parsing algorithm that constructs parse trees by starting from the individual words in the sentence and gradually building up to the root.
2. Context-Free Grammar (CFG): CYK works with context-free grammars, where production rules specify how symbols can be combined to form valid sentences.
3. Chart Parsing: CYK employs a dynamic programming technique known as chart parsing, where intermediate results are stored in a chart table to avoid redundant computations.
4. Chart Initialization: The chart table is initialized with cells corresponding to each word in the sentence, representing potential constituents or phrases.
5. Rule Application: CYK iteratively applies CFG rules to combine constituents in the chart table and build larger constituents until reaching the root of the parse tree.
6. Backtracking: In case of ambiguity or multiple possible parses, CYK explores different combinations of constituents using backtracking to find valid parse trees.

7. Efficiency: CYK's time complexity is $O(n3 \cdot |G|)$, where n is the length of the input sentence and $|G|$ is the size of the grammar. This makes CYK efficient for parsing small or moderately sized sentences.
8. Example Sentence: Let's consider the sentence "I saw a dog in the park" and a simple CFG with rules like S → NP VP, NP → Pronoun, VP → Verb NP, NP → Article Noun, etc.
9. Chart Table: The chart table is initialized with cells for each word in the sentence, where constituents like "I", "saw", "a", "dog", etc., are placed in the respective cells.
10. Parse Tree Construction: CYK applies CFG rules to combine constituents in the chart table, iteratively building larger constituents until reaching the root (S) of the parse tree. The final parse tree represents the syntactic structure of the sentence according to the CFG rules.

5. a) State the types of references used in discourse analysis.

1. Anaphoric References: Anaphoric references occur when a word or phrase refers back to something previously mentioned in the discourse.
2. Cataphoric References: Cataphoric references involve a word or phrase that refers forward to something mentioned later in the discourse.
3. Exophoric References: Exophoric references refer to elements outside the text itself, such as the physical context, shared knowledge, or cultural references.
4. Endophoric References: Endophoric references refer to elements within the text itself, such as other words, phrases, or sentences.
5. Deictic References: Deictic references depend on the context of the utterance, including factors like time, place, and the participants in the discourse.
6. Textual References: Textual references involve elements within the text, such as pronouns, demonstratives, or lexical repetitions used to maintain

cohesion.

7. Interpretive References: Interpretive references involve inferences made by the reader or listener to connect elements of the discourse and derive meaning.
8. Discourse Referents: Discourse referents are entities or concepts introduced and maintained throughout the discourse, often tracked through pronouns or noun phrases.
9. Anaphora: Anaphora refers specifically to the use of words or phrases that refer back to a previously introduced entity or concept.
10. Cataphora: Cataphora refers to the use of words or phrases that anticipate or refer forward to an entity or concept introduced later in the discourse.

b) Describe in detail about word sense disambiguation with an example.

1. Definition: Word sense disambiguation (WSD) is the task of determining the correct meaning or sense of a word in a given context, particularly when a word has multiple possible meanings or senses.
2. Challenge: Many words in natural language are polysemous, meaning they have multiple senses. WSD is crucial for various NLP tasks, including machine translation, information retrieval, and text summarization, as different senses of a word can convey different meanings and nuances.
3. Approaches: WSD can be approached using knowledge-based methods, supervised learning, unsupervised learning, or a combination of these approaches.
4. Knowledge-based Methods: These methods rely on lexical resources such as dictionaries, thesauri, and ontologies to identify the sense of a word based on its definition, synonyms, or semantic relations.
5. Supervised Learning: Supervised learning approaches train machine learning models using annotated datasets, where each word is labeled

with its correct sense. These models learn to predict the sense of a word based on features extracted from the context in which it appears.

6. Unsupervised Learning: Unsupervised learning approaches cluster word instances based on their co-occurrence patterns and use contextual similarities to determine word senses without relying on annotated data.
7. Example: Consider the word "bank," which has multiple senses such as financial institution and the side of a river. In the sentence "I deposited money in the bank," the word "bank" could refer to a financial institution. In contrast, in the sentence "I sat on the bank and watched the river flow," the word "bank" likely refers to the side of a river.
8. Contextual Cues: WSD algorithms consider various contextual cues such as surrounding words, syntactic structures, semantic relations, and domain-specific knowledge to infer the correct sense of a word.
9. Evaluation: WSD systems are evaluated based on their accuracy in correctly identifying the sense of words in test datasets. Evaluation metrics include precision, recall, F1 score, and accuracy.
10. Applications: WSD is essential for improving the performance of many NLP applications. For example, in machine translation, accurately translating polysemous words requires identifying the correct sense of each word in the source language to produce contextually appropriate translations.

6. a) Give example to explain the concept of predicate argument structure.

1. Definition: Predicate-argument structure refers to the relationship between a verb (the predicate) and its arguments (typically the subject, object, and other complements or adjuncts) within a sentence.
2. Example Sentence: Consider the sentence "The teacher gave the student a book."
3. Predicate Identification: In this sentence, the verb "gave" is the predicate.

It describes the action being performed.

4. Arguments Identification: The arguments of the predicate "gave" are "The teacher" (the subject), "the student" (the indirect object), and "a book" (the direct object).
5. Semantic Roles: Each argument has a specific semantic role. In this case, "The teacher" is the agent (doer of the action), "the student" is the recipient (receiver of the action), and "a book" is the theme (entity being given).
6. Syntactic Structure: The syntactic structure of the sentence can be broken down as: [NP The teacher] [VP gave [NP the student] [NP a book]].
7. Verb Subcategorization: The verb "gave" subcategorizes for three arguments: a subject (NP), an indirect object (NP), and a direct object (NP).
8. Dependencies: Dependencies between the predicate and its arguments are crucial for understanding the sentence's meaning. These dependencies indicate who is doing what to whom.
9. Applications in NLP: Predicate-argument structure is essential for tasks such as semantic role labeling, machine translation, and information extraction, as it helps in understanding the relationships and roles within a sentence.
10. Further Example: Another example is "She put the book on the table." Here, "put" is the predicate, "She" is the agent, "the book" is the theme, and "on the table" is the locative argument. Understanding these roles clarifies who performed the action, what was acted upon, and where the action took place.

b) What are System paradigms in predicates? Explain.

1. Thematic Roles: Thematic roles (or semantic roles) describe the function of each argument in relation to the predicate. Common roles include

agent (doer of the action), patient (receiver of the action), and instrument (means by which the action is performed).

2. Subcategorization Frames: Subcategorization frames specify the syntactic pattern that a predicate can take. For example, the verb "give" requires a subject, a direct object, and an indirect object (e.g., "She gave him a book").
3. Argument Structure: Argument structure refers to the number and type of arguments that a predicate can take. For instance, the verb "run" typically has one argument (the runner), while "give" has three (the giver, the recipient, and the thing given).
4. Valency: Valency refers to the number of arguments a verb can have. Intransitive verbs have a valency of one (e.g., "sleep"), transitive verbs have a valency of two (e.g., "eat"), and ditransitive verbs have a valency of three (e.g., "give").
5. Head-Driven Phrase Structure Grammar (HPSG): HPSG is a framework that integrates syntax, semantics, and morphology. It represents predicates and their arguments using feature structures that capture various linguistic properties.
6. Lexical Functional Grammar (LFG): LFG separates syntactic structure (c-structure) from functional structure (f-structure). It focuses on the relationships between predicates and their arguments through functional roles like subject and object.
7. Generative Grammar: In generative grammar, predicates are analyzed within the framework of syntactic trees. The relationships between predicates and their arguments are represented using hierarchical tree structures.
8. Dependency Grammar: Dependency grammar focuses on the binary relationships between words, with predicates being the heads that govern their arguments. This approach emphasizes direct syntactic dependencies.

9. Frame Semantics: Frame semantics views predicates as evoking frames or scenarios. Each frame includes various roles (called frame elements) that correspond to the arguments of the predicate.
10. PropBank and VerbNet: PropBank provides a corpus with annotated predicate-argument structures, assigning rolesets to verbs. VerbNet groups verbs into classes based on their argument structures and thematic roles, offering a systematic way to analyze predicates.

7. a) Explain Anaphora Resolution with the help of an example.

1. Definition: Anaphora resolution involves determining the antecedent (the entity to which a pronoun or referring expression refers) in a given discourse.
2. Importance: It is crucial for understanding the coherence and meaning of texts in natural language processing (NLP), as it ensures that references are correctly interpreted.
3. Types of Anaphora: Common types include pronominal anaphora (e.g., "he," "she," "it") and nominal anaphora (e.g., "the car," "the company").
4. Example: Consider the sentences: "John went to the store. He bought some milk." Here, "He" is the anaphoric expression, and its antecedent is "John."
5. Syntactic Clues: Syntactic structures can provide clues for anaphora resolution. For instance, pronouns often refer to the nearest preceding noun phrase.
6. Semantic Clues: The meanings of words and their roles in the sentence can help. For example, knowing that "John" is a person helps in resolving that "he" refers to "John."
7. Contextual Clues: The broader context of the discourse aids in anaphora resolution. Prior sentences provide necessary information to resolve references accurately.

8. Resolution Algorithms: Various algorithms exist for anaphora resolution, including rule-based approaches, machine learning methods, and hybrid models that combine multiple strategies.
9. Challenges: Anaphora resolution can be challenging due to ambiguous references, long distances between the pronoun and its antecedent, and complex sentence structures.
10. Applications: Anaphora resolution is essential in applications like machine translation, text summarization, information extraction, and dialogue systems to ensure accurate and coherent interpretation of texts.\

b) What kind of predicate structure are used for language modelling?

1. Unigram Models: In unigram models, each word's occurrence is treated independently. There is no explicit predicate structure, as the probability of each word is based solely on its individual frequency.
2. Bigram Models: Bigram models consider the probability of a word given the previous word. For instance, in the phrase "eats an," the model would compute the likelihood of "an" following "eats."
3. Trigram Models: Trigram models extend bigrams by considering the probability of a word given the two preceding words. For example, "she eats an" predicts the likelihood of the next word.
4. N-gram Models: Generalizing bigrams and trigrams, n-gram models use a sequence of n-1 preceding words to predict the next word, capturing more context for better language modeling.
5. Hidden Markov Models (HMMs): HMMs use hidden states to model sequences, where each state corresponds to a word or a part of speech, capturing both syntactic and semantic dependencies.
6. Conditional Random Fields (CRFs): CRFs are used for sequence prediction, modeling the probability of a sequence of words considering both the observations and the dependencies between words.

7. Dependency Parsing: This structure models the dependencies between words in a sentence, capturing the relationships between predicates and their arguments, useful for understanding sentence structure.
8. Constituency Parsing: Constituency parsing involves breaking down sentences into sub-phrases or constituents, representing hierarchical structures that capture the predicate-argument relationships within nested phrases.
9. Semantic Role Labeling (SRL): SRL identifies the predicate and assigns roles to each word or phrase in the sentence, such as agent, patient, or instrument, enhancing the understanding of sentence meaning.
10. Transformer Models: Modern models like BERT and GPT use transformer architectures, which capture complex dependencies and contextual information through self-attention mechanisms, significantly improving language modeling performance.

8. a) Discuss in detail about the concept of Cohesion Structure.

1. Definition: Cohesion refers to the linguistic elements that connect sentences and parts of a text together, providing a sense of flow and continuity. It involves the use of cohesive devices to achieve textual unity.
2. Cohesive Devices: These are the tools used to create cohesion in a text. Common cohesive devices include conjunctions, pronouns, lexical repetition, synonyms, and ellipses.
3. Reference: Reference involves using pronouns or other referring expressions to link back to something previously mentioned in the text. For example, "John arrived late. He missed the meeting." Here, "He" refers back to "John."
4. Substitution: Substitution involves replacing a word or phrase with another term to avoid repetition and link sentences. For example, "I lost my pen. I need a new one." "One" substitutes "pen."

5. Ellipsis: Ellipsis is the omission of elements that are understood from the context, creating a link between sentences. For example, "Do you want tea or coffee?" "Tea, please." The omission of "I want" is understood.
6. Conjunction: Conjunctions connect clauses, sentences, and paragraphs, indicating relationships such as addition, contrast, or cause-effect. Examples include "and," "but," "because," and "therefore."
7. Lexical Cohesion: Lexical cohesion involves the use of vocabulary to create links, such as through repetition of key terms, synonyms, antonyms, and hyponyms. For instance, "The dog barked loudly. The animal seemed agitated."
8. Collocation: Collocation refers to the tendency of certain words to appear together frequently, which helps create a cohesive structure. For example, "strong tea" and "make a decision" are common collocations.
9. Thematic Progression: The way themes (main topics) are developed and progressed throughout a text contributes to its cohesion. Maintaining a clear thematic structure helps readers follow the argument or narrative.
10. Cohesion vs. Coherence: While cohesion refers to the linguistic devices that link parts of the text, coherence is about the overall sense and logical flow of ideas. A text can be cohesive without being coherent if the links do not contribute to a clear, logical argument or narrative.

b) List out popular lexical resources used in the processing of natural language text.

1. WordNet: A large lexical database of English, where words are grouped into sets of synonyms called synsets, and semantic relationships between these synsets are described.
2. FrameNet: A resource based on the theory of frame semantics, which documents the range of semantic and syntactic combinatory possibilities (valences) of each word in each of its senses.

3. ConceptNet: A semantic network that connects words and phrases with labeled, weighted edges, representing general knowledge about the world.
4. BabelNet: A multilingual lexicalized semantic network and ontology that connects concepts and named entities in a very large network of semantic relations.
5. DBpedia: A project aimed at extracting structured content from the information created as part of the Wikipedia project, providing a rich resource for semantic information.
6. GloVe (Global Vectors for Word Representation): A word embedding model that maps words into high-dimensional vectors, capturing semantic relationships between words.
7. Word2Vec: A group of related models that are used to produce word embeddings, learning to associate words with high-dimensional vectors based on their contexts in large text corpora.
8. Open Multilingual WordNet (OMW): A multilingual extension of WordNet that includes linked wordnets in many languages, allowing for cross-linguistic comparison and analysis.
9. SentiWordNet: An extension of WordNet that annotates synsets with polarity scores for sentiment analysis, indicating whether the terms are positive, negative, or neutral.
10. PropBank: A resource that adds a layer of predicate-argument information, or semantic role labels, to the syntactic structures of the Penn Treebank, enhancing the understanding of verb semantics.

Code No: 155CK

R18

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY
HYDERABAD B. Tech III Year I Semester Examinations,
January/February - 2023 NATURAL LANGUAGE PROCESSING

(Computer Science and Engineering)

Time: 3 Hours **Max. Marks: 75**

Note: i) Question paper consists of Part A, Part B.

ii) Part A is compulsory, which carries 25 marks. In Part A, Answer all questions.

iii) In Part B, Answer any one question from each unit. Each question carries 10 marks and may have a, b as sub questions.

PART – A

(25 Marks)

1.a) Define allomorphs. [2]

b) What is a lexeme? List its categories. [3]

c) What is the use of Treebank? [2]

d) Give an example of a phrase structure tree. [3]

e) What is the objective of semantic parsing? [2]

f) How to identify an event in text? [3]

g) List verbal predicates that demand two object arguments. [2]

h) Suggest the requirements for a meaning representation system. [3]

i) What is meant by cohesion in NLP? [2]

j) What is an n-Gram model? [3]

PART – B

(50 Marks)

2. Describe the following morphological models with illustrations:

a) Dictionary lookup

b) Unification based morphology. [5+5]

OR

3. Make a comparison of generative sequence classification methods and discriminative local classification methods for finding structure of the documents. [10]

4. Elaborate the tokenization and parsing challenges in multilingual content.[10]

OR

5.a) Demonstrate shift-reduce parsing algorithm.

b) Explain the use of probabilistic context free grammars for ambiguity resolution in parsing. [5+5]

6. Describe the methods for word sense disambiguation and the software support for it. [10]

OR

7. Illustrate the methods for resolving entity and event in natural language. [10]

8. Explain the usage of PropBank in semantic parsing. [10]

OR

9. Provide a detailed description of meaning representation and its need. [10]

10. Make a comparison of syntax based language model and factored language model. [10]

OR

11.a) With an example, discuss demonstrative reference.

b) What is meant by anaphora resolution? Explain the methods used for it.[5+5]

---ooOoo---

ANSWER KEY

PART – A

1. a) Define allomorphs.

Allomorphs are variations of a morpheme that appear in different contexts but have the same meaning. These variations occur due to phonological or morphological conditions and can be seen in how a single morpheme manifests differently depending on its environment. For example, the English plural morpheme can appear as -s, -es, or -en (as in "cats," "dishes," and "oxen," respectively). Despite their different forms, these allomorphs all serve the same

function of indicating plurality.

b) What is a lexeme? List its categories.

A lexeme is the fundamental unit of meaning in a language, representing a set of all its possible inflected forms. For example, the lexeme "run" includes "run," "runs," "running," and "ran." Lexemes are typically categorized into nouns, verbs, adjectives, adverbs, pronouns, prepositions, conjunctions, and interjections.

c) What is the use of Treebank?

A Treebank is a collection of syntactically parsed sentences, annotated with phrase structure or dependency trees. It serves as a valuable resource for training and evaluating natural language processing (NLP) models, particularly those related to syntax and parsing. Treebanks provide linguistic insights into sentence structure and facilitate the development of algorithms for tasks such as parsing, grammar induction, and semantic analysis. They also enable researchers to study language phenomena and linguistic theories in a computationally tractable manner.

d) Give an example of a phrase structure tree.

A phrase structure tree, also known as a syntax tree or parse tree, represents the hierarchical structure of a sentence according to its syntactic constituents. For example, consider the sentence "The cat chased the mouse." A phrase structure tree for this sentence would show the hierarchical relationships between the words, such as how "the cat" and "the mouse" are noun phrases (NPs) and "chased" is a verb phrase (VP), with "the cat" as the subject and "the mouse" as the object.

e) What is the objective of semantic parsing?

The objective of semantic parsing is to convert natural language utterances into formal, executable representations of meaning, such as logical forms or semantic graphs. This process aims to extract the underlying semantic structure from natural language expressions, enabling computers to understand and execute commands or queries given in human language. Semantic parsing plays a crucial role in natural language understanding, question answering, information retrieval, and dialogue systems.

f) How to identify an event in text?

Identifying an event in text involves recognizing linguistic expressions that convey actions, occurrences, or happenings. This can be achieved through various natural language processing techniques, including part-of-speech tagging, named entity recognition, dependency parsing, and semantic role labeling. Events are typically represented by verbs or verb phrases and can be further classified based on their tense, aspect, and modality. Automatic event identification is essential for tasks such as information extraction, summarization, and sentiment analysis.

g) List verbal predicates that demand two object arguments.

Verbal predicates that demand two object arguments are often referred to as ditransitive verbs. Examples include "give," "show," "send," "offer," "bring," "teach," "lend," and "promise." These verbs require both a direct object, which receives the action directly, and an indirect object, which indicates to or for whom the action is performed. Ditransitive verbs are commonly used in sentences where someone transfers or provides something to someone else.

h) Suggest the requirements for a meaning representation system.

A meaning representation system requires the ability to capture the semantics of natural language accurately, including entities, relations, and events. It should

support compositionality, allowing complex meanings to be built from simpler components. The system needs to be expressive enough to handle ambiguity, polysemy, and context-dependence. Additionally, it must facilitate inference, enabling the derivation of new information from the represented meanings.

i) What is meant by cohesion in NLP?

In NLP, cohesion refers to the use of linguistic elements to link sentences and parts of a text together, ensuring that the text flows logically and is semantically connected. Cohesion is achieved through devices such as pronouns, conjunctions, lexical repetition, and ellipses, which help create clear and understandable relationships between different parts of the text. It is essential for maintaining the readability and coherence of written and spoken discourse.

j) What is an n-Gram model?

An n-gram model is a probabilistic language model used in NLP to predict the likelihood of a sequence of words by considering the occurrence of n-word sequences, or "n-grams." In this model, the probability of a word depends on the preceding (n-1) words. For example, in a bigram model (n-2), the probability of a word is conditioned on the preceding word. N-gram models are commonly used for tasks such as text generation, speech recognition, and machine translation.

PART - B

2. Describe the following morphological models with illustrations:

a) Dictionary lookup

1. Definition: Dictionary lookup in morphological analysis involves using a precompiled list of words (a dictionary) where each entry maps to its base form or root and its morphological features, such as tense, number, gender, etc.

2. Function: This method checks if a word is present in the dictionary and retrieves its morphological information directly.
3. Illustration: For the word "running," a dictionary lookup might return the base form "run" and features like present participle or gerund.
4. Advantages: It is straightforward and provides accurate results for known words.
5. Limitations: It is limited by the size and coverage of the dictionary and cannot handle unknown or novel words.

b) Unification based morphology.

6. Definition: Unification-based morphology uses feature structures and unification operations to analyze and generate word forms. It is rooted in unification grammar theories, such as Lexical Functional Grammar (LFG) and Head-Driven Phrase Structure Grammar (HPSG).
7. Function: It represents morphological information as sets of features and constraints, and uses unification to combine these features consistently.
8. Illustration: For the word "cats," a unification-based system would combine the features {root: "cat", number: singular} with a pluralization rule to produce {root: "cat", number: plural, surface: "cats"}.
9. Advantages: It is flexible and can handle complex morphological phenomena, including irregular forms and concatenative morphology.
10. Limitations: It can be computationally intensive and requires a comprehensive set of rules and constraints, which can be complex to develop and maintain.

3. Make a comparison of generative sequence classification methods and discriminative local classification methods for finding structure of the documents.

1. Generative sequence classification methods model the joint probability of

the observed data and the label sequence, while discriminative local classification methods model the conditional probability of the label sequence given the observed data.

2. Generative methods, such as Hidden Markov Models (HMMs), can generate new data points by sampling from the joint distribution they model, whereas discriminative methods, such as Conditional Random Fields (CRFs), focus solely on predicting the most likely label sequence for given data.
3. Generative models require the specification of a complete probabilistic model for the data, including the generation process, which can be more complex and less flexible compared to discriminative models.
4. Discriminative methods often achieve higher accuracy in practice for classification tasks because they directly model the decision boundary between classes rather than the entire data distribution.
5. Generative methods can handle missing data more naturally, as they model the entire data generation process, while discriminative methods typically require complete data.
6. Training generative models involves estimating joint probabilities, which can be challenging for high-dimensional data due to the curse of dimensionality, whereas discriminative models focus on learning the boundary between classes, which can be more tractable.
7. Discriminative methods can incorporate rich, overlapping features directly into the model, providing greater flexibility and often better performance in capturing complex relationships within the data.
8. Generative models can provide insights into the underlying data generation process, which can be useful for understanding the data, while discriminative models are primarily focused on prediction accuracy.
9. Discriminative models, like support vector machines (SVMs) used in local classification, are generally more robust to overfitting, especially

with high-dimensional feature spaces, compared to generative models.

10. In document structure analysis, generative models may be used to understand and replicate the generation of document structures, while discriminative models are typically preferred for accurately labeling parts of the document based on observed features.

4. Elaborate the tokenization and parsing challenges in multilingual content.

1. Script Variations: Different languages use different scripts (e.g., Latin, Cyrillic, Chinese characters), complicating tokenization as the rules for identifying word boundaries vary widely.
2. Word Boundaries: Some languages, like Chinese and Japanese, do not use spaces to separate words, making it difficult to determine where one word ends and another begins.
3. Morphological Complexity: Languages like Finnish or Turkish have complex morphology with numerous inflections and agglutinations, making it challenging to tokenize and parse words correctly.
4. Ambiguity: Many languages have words that can function as multiple parts of speech (e.g., noun and verb), leading to ambiguities that complicate parsing.
5. Compound Words: Languages such as German often combine multiple words into compound words, which can be difficult to tokenize and parse accurately.
6. Dialectal Variations: Different dialects or regional variations within the same language can have unique tokenization and parsing rules, requiring adaptable models.
7. Named Entities: Multilingual content often includes named entities (e.g., names of people, places) that may follow different tokenization rules or

include multiple languages in a single entity.

8. Code-Switching: In multilingual contexts, speakers often switch between languages within a sentence, requiring models to handle multiple languages simultaneously.
9. Lack of Resources: For many languages, especially low-resource languages, there are limited annotated corpora and tools available for effective tokenization and parsing.
10. Cultural and Contextual Differences: Understanding the cultural context and nuances of different languages is crucial for accurate parsing, as meaning can change based on cultural references and idiomatic expressions.

5. a) Demonstrate shift-reduce parsing algorithm.

1. Initialization: Start with an input buffer containing the string to be parsed and an empty stack.
2. Shift Operation: Move (shift) the next symbol from the input buffer onto the top of the stack.
3. Reduce Operation: If the top of the stack contains a substring that matches the right-hand side of a production rule, replace (reduce) this substring with the non-terminal on the left-hand side of the rule.
4. Iteration: Repeat the shift and reduce operations until the input buffer is empty and the stack contains the start symbol.
5. Completion: If the stack contains only the start symbol and the input buffer is empty, the parse is successful; otherwise, the parse fails.

b) Explain the use of probabilistic context free grammars for ambiguity resolution in parsing.

1. Probability Assignment: PCFGs assign probabilities to each production rule, allowing the grammar to prioritize certain parses over others based

on their likelihood.

2. Parse Selection: When multiple parse trees are possible for a sentence, PCFGs use the rule probabilities to calculate the overall probability of each parse tree, selecting the most likely one.
3. Ambiguity Resolution: By leveraging statistical information, PCFGs can disambiguate between parses that are syntactically valid but semantically different, favoring interpretations that are more probable given the language data.
4. Training on Corpora: PCFGs are trained on annotated corpora, which provide empirical data on rule frequencies, enabling the model to learn which structures are more common in natural language usage.
5. Application in Parsing Algorithms: PCFGs integrate with parsing algorithms, such as the CYK or Earley parser, to efficiently explore and rank possible parse trees, ensuring that the most probable structure is chosen for ambiguous sentences.

6. Describe the methods for word sense disambiguation and the software support for it.

1. Supervised Learning: Uses labeled training data where word senses are annotated. Algorithms like decision trees, support vector machines, and neural networks are trained to predict the sense of a word based on contextual features.
2. Unsupervised Learning: Clusters word occurrences in a text corpus based on context similarity, assuming each cluster corresponds to a different sense. Techniques include clustering algorithms like k-means and latent semantic analysis.
3. Knowledge-Based Methods: Leverage lexical resources like WordNet, using definitions, synonyms, and semantic relations to disambiguate senses. Lesk algorithm, for example, compares context overlap between

definitions.

4. Semi-Supervised Learning: Combines labeled and unlabeled data to improve disambiguation accuracy. Bootstrapping methods start with a small labeled dataset and iteratively expand it with confidently predicted labels.
5. Dictionary-Based Methods: Use dictionary definitions to compare the context of the target word with definitions of each possible sense, selecting the sense with the highest overlap.
6. Contextual Embeddings: Use embeddings from models like BERT or ELMo, which capture word meaning based on surrounding context, providing a powerful way to distinguish between senses.
7. Hybrid Approaches: Combine multiple methods to leverage the strengths of each, such as integrating supervised models with knowledge-based resources for improved accuracy.
8. Evaluation Metrics: Common metrics for assessing WSD methods include precision, recall, and F1-score, typically evaluated on standard datasets like SemCor or Senseval.
9. Software Tools: Tools like the Natural Language Toolkit (NLTK) in Python provide implementations of various WSD methods, including supervised and Lesk algorithm-based approaches.
10. Advanced Libraries: Libraries like spaCy and Stanford NLP offer pretrained models and embeddings for contextual understanding, facilitating the implementation of WSD in larger NLP pipelines.

7. Illustrate the methods for resolving entity and event in natural language.

1. Named Entity Recognition (NER): Identifies and classifies named entities (persons, organizations, locations, dates, etc.) in text using machine learning models or rule-based systems. Tools like spaCy, Stanford NER,

and NLTK provide pre-trained NER models.

2. Coreference Resolution: Determines which words refer to the same entity in a text. Algorithms resolve pronouns and noun phrases back to the entities they refer to, using features like gender, number, and semantic compatibility. Libraries like AllenNLP offer coreference resolution models.
3. Event Extraction: Identifies events and their components (participants, locations, times) in text. Techniques involve pattern-based methods, supervised learning models, and deep learning approaches like LSTM or transformer-based models.
4. Relation Extraction: Detects and classifies semantic relationships between entities in text. Common methods include supervised learning with annotated datasets and distant supervision using knowledge bases.
5. Temporal Information Extraction: Identifies and normalizes time expressions in text, determining the timing of events. Rule-based approaches, statistical models, and hybrid methods are used for extracting and resolving temporal information.
6. Semantic Role Labeling (SRL): Identifies predicate-argument structures, labeling the roles of entities in events. SRL helps in understanding who did what to whom, when, and where. Libraries like AllenNLP provide SRL tools.
7. Dependency Parsing: Analyzes grammatical structure to identify relationships between words. Dependency parsers, available in tools like spaCy and Stanford NLP, help in resolving entities and events by understanding syntactic dependencies.
8. Contextual Embeddings: Models like BERT and GPT capture contextual information, aiding in disambiguation and resolution of entities and events by leveraging deep contextual understanding.

9. Knowledge Base Integration: Uses external knowledge bases like Wikidata or DBpedia to enrich entity and event resolution with additional information, improving accuracy and context comprehension.
10. Hybrid Approaches: Combine multiple methods (e.g., NER with SRL and coreference resolution) to enhance overall accuracy and robustness in resolving entities and events in natural language processing tasks.

8. Explain the usage of PropBank in semantic parsing.

1. PropBank: PropBank (Proposition Bank) is a resource that annotates the semantic roles of verbs in a corpus, providing information on how verbs relate to their arguments in a sentence.
2. Semantic Role Labeling: PropBank facilitates semantic role labeling (SRL) by assigning specific roles (such as Agent, Patient, Theme) to the arguments of verbs, capturing the predicate-argument structure of sentences.
3. Fine-Grained Annotation: PropBank annotations offer fine-grained distinctions between different senses of a verb, helping disambiguate the roles associated with each sense.
4. Corpus-Based: PropBank annotations are created through manual annotation of text corpora, ensuring that the roles assigned to verbs are grounded in real-world usage.
5. Complements FrameNet: PropBank complements FrameNet by focusing specifically on the semantic roles of verbs, while FrameNet annotates a broader range of semantic frames and their participants.
6. Integration with NLP Systems: PropBank annotations are integrated into various natural language processing (NLP) systems and tools, providing valuable information for tasks such as information extraction, question answering, and machine translation.

7. Training Data: PropBank serves as valuable training data for developing machine learning models for SRL, enabling the creation of accurate and robust semantic parsers.
8. Cross-Linguistic Studies: PropBank annotations have been extended to other languages, allowing for cross-linguistic studies on the argument structures of verbs across different languages.
9. Semantic Parsing: In semantic parsing tasks, such as converting natural language queries into executable representations, PropBank annotations aid in understanding the semantic roles of verbs and their arguments.
10. Community Resource: PropBank is a widely used resource in the NLP community, fostering research and development in semantic analysis and helping advance the understanding of verb semantics in computational linguistics.

9. Provide a detailed description of meaning representation and its need.

1. Semantic Representation: Meaning representation in natural language processing (NLP) involves capturing the underlying semantics of linguistic expressions in a structured format that can be understood and processed by machines.
2. Abstracting Meaning: Meaning representation abstracts away from surface-level linguistic features to capture the core meaning conveyed by a sentence or text.
3. Structured Format: Meaning representation typically involves representing linguistic elements, such as words and phrases, as structured objects linked by semantic relations, enabling precise interpretation and manipulation.
4. Interpretability: Meaning representation facilitates the interpretation of natural language by machines, allowing them to understand and reason

about the content of text.

5. Ambiguity Resolution: Natural language is inherently ambiguous, and meaning representation helps resolve this ambiguity by providing a precise, unambiguous representation of meaning.
6. Facilitating NLP Tasks: Meaning representation serves as the foundation for various NLP tasks, such as information extraction, question answering, machine translation, and dialogue systems.
7. Semantic Parsing: In semantic parsing, meaning representation involves mapping natural language utterances to executable representations, enabling machines to perform tasks based on user commands or queries.
8. Machine Understanding: Meaning representation enables machines to understand the content and intent of text, allowing them to extract relevant information and perform tasks autonomously.
9. Cross-Linguistic Analysis: Meaning representation provides a common framework for analyzing and comparing linguistic expressions across different languages, facilitating cross-linguistic research and development.
10. Need for Automation: With the increasing volume of digital text and the demand for intelligent NLP applications, the need for automated methods for capturing and processing meaning has become paramount. Meaning representation addresses this need by providing a systematic and computationally tractable way of representing linguistic meaning.

10. Make a comparison of syntax based language model and factored language model.

1. Syntax-Based Language Model: Utilizes syntactic information such as parse trees or grammatical structures to model the relationship between words in a sentence.

   Factored Language Model: Incorporates multiple factors or features,

including syntax, semantics, and lexical information, to capture the various dimensions of linguistic structure.

2. Syntactic Features: Syntax-based models primarily focus on syntactic features such as part-of-speech tags, dependency relations, and parse trees to encode sentence structure.

   Factorization of Features: Factored language models decompose the features into separate components, allowing for the incorporation of various linguistic factors independently.

3. Dependency on Syntax: Syntax-based models heavily rely on accurate syntactic parsing and annotation, making them sensitive to errors or inconsistencies in parsing.

   Flexible Feature Composition: Factored language models offer flexibility in feature composition, allowing for the integration of diverse linguistic information beyond syntax.

4. Parsing Complexity: Syntax-based models often require computationally expensive syntactic parsing during both training and inference stages.

   Reduced Parsing Overhead: Factored language models may mitigate parsing complexity by incorporating simpler linguistic features or by utilizing precomputed features.

5. Interpretability: Syntax-based models provide interpretable representations of sentence structure, enabling insights into syntactic relationships.

   Comprehensive Linguistic Coverage: Factored language models offer broader linguistic coverage by considering multiple linguistic dimensions, enhancing model expressiveness.

6. Domain Adaptability: Syntax-based models may struggle with domain-specific texts where syntactic structures differ from standard linguistic conventions.

Enhanced Adaptability: Factored language models can adapt more effectively to domain-specific texts by incorporating relevant linguistic factors beyond syntax.

7. Handling Ambiguity: Syntax-based models may struggle with resolving syntactic ambiguities, especially in cases of syntactic ambiguity.
   Ambiguity Resolution: Factored language models have the potential to address ambiguities more effectively by leveraging multiple linguistic factors for disambiguation.
8. Task Suitability: Syntax-based models are well-suited for tasks where syntactic structure is crucial, such as parsing, machine translation, and syntactic analysis.
   Broad Applicability: Factored language models are applicable across various NLP tasks, including those requiring semantic, lexical, or syntactic information.
9. Resource Requirements: Syntax-based models often require large-scale syntactic annotations and parsers, increasing the demand for computational resources.
   Efficient Resource Utilization: Factored language models may optimize resource usage by selecting relevant linguistic factors based on task requirements, potentially reducing computational overhead.
10. Research Focus: Syntax-based models have been a primary focus of traditional NLP research, with extensive work on syntactic parsing and syntactic language modeling.
    Emerging Paradigm: Factored language models represent an emerging paradigm in NLP research, reflecting efforts to capture diverse linguistic factors beyond syntax for improved language understanding.

11. a) With an example, discuss demonstrative reference.

1. Identification of Referent: Demonstrative reference involves using

demonstrative words like "this," "that," "these," and "those" to point to specific entities or objects in the discourse.

2. Proximity: Demonstratives indicate the spatial or temporal proximity of the referent to the speaker or the context of the discourse. For example, "this book" refers to a book near the speaker, while "that book" may refer to a book farther away.
3. Context Dependency: The interpretation of demonstratives heavily relies on contextual information. For instance, "this car" may refer to a car being discussed in a conversation, whereas "that car" could refer to a car visible outside the window.
4. Disambiguation: Demonstratives aid in disambiguating between multiple entities or objects by explicitly indicating which one is being referred to. For instance, in a room with several chairs, saying "sit on this chair" specifies which chair the speaker is referring to.
5. Dynamic Reference: Demonstratives can dynamically shift their reference based on changes in the discourse or the speaker's perspective. For example, "this cup" may initially refer to a cup on the table but could later refer to a cup handed to the speaker.

b) What is meant by anaphora resolution? Explain the methods used for it.

1. Definition: Anaphora resolution is the process of identifying and linking anaphoric expressions (pronouns, noun phrases) to their antecedents (previous mentions) in a text.
2. Coreference Resolution: Anaphora resolution often involves coreference resolution, which identifies expressions referring to the same entity across a text, helping to establish referential relationships.
3. Syntactic and Semantic Clues: Methods for anaphora resolution utilize syntactic and semantic clues within the text, such as grammatical role, gender, number, and semantic compatibility, to determine the antecedent

of an anaphoric expression.

4. Rule-based and Machine Learning Approaches: Anaphora resolution can be approached using rule-based systems that encode linguistic rules for antecedent selection or machine learning techniques, including supervised and unsupervised methods trained on annotated corpora.
5. Integration with NLP Applications: Anaphora resolution plays a crucial role in various natural language processing applications, such as machine translation, information extraction, and question answering, by ensuring coherent and accurate interpretation of text.

Code No: 155CK

R18

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD B. Tech

III Year I Semester Examinations, March - 2021

NATURAL LANGUAGE PROCESSING

(Computer Science and Engineering)

Time: 3 Hours Max. Marks: 75 Answer any five questions

All questions carry equal marks

- - -

1.a) List and explain the challenges of morphological models.

b) Discuss the importance and goals of Natural Language Processing. [7+8]

2.a) Elaborate the Models for Ambiguity Resolution in Parsing.

b) With the help of a neat diagram, explain the Representation of Syntactic Structure. [8+7]

3. Explain the Word Sense Systems in detail. [15]

4.a) Difference between predicate and Predicator.

b) Write a short note on Meaning Representation Systems. [8+7]

5.a) Elaborate the Multilingual and Cross-lingual Language Modeling.

b) Illustrate the Discourse Cohesion and Structure. [8+7]

6. Explain the performance of approaches in Structure of Documents. [15]

7. Describe the Cohesion and Reference Resolution. [15]

8. Explain the following:

a) Semantic Parsing with Execution

b) Language Model Adaptation [7+8]

---ooOoo---

ANSWER KEY

1. a) List and explain the challenges of morphological models.

1. Ambiguity: Morphological models face the challenge of ambiguity, where a single word form may have multiple possible morphological analyses, leading to uncertainty in interpretation.
2. Overgeneration: Morphological models may overgenerate possible word forms, producing morphological analyses that are not valid in the language, leading to inaccuracies in language processing tasks.
3. Lack of Standardization: Languages often lack standardized morphological rules, making it challenging to develop robust and accurate morphological models that cover all linguistic variations.
4. Productivity: Languages exhibit productivity, allowing speakers to create new words or morphological forms. Morphological models must account for this productivity to handle novel or rare word forms effectively.
5. Compounding: Many languages allow compounding, where multiple words combine to form a new word. Morphological models need to handle compound words accurately, considering the morphological structure of each constituent.
6. Inflectional Morphology: Inflectional morphology, including tense, aspect, mood, and case markings, adds complexity to morphological analysis, requiring models to account for various inflectional patterns.

7. Derivational Morphology: Derivational morphology involves the creation of new words through affixation or other morphological processes. Models must capture derivational relationships to provide comprehensive morphological analysis.
8. Irregularities: Many languages contain irregularities in morphological patterns, where certain word forms do not follow regular morphological rules. Morphological models must handle these irregularities effectively to ensure accurate analysis.
9. Data Sparsity: Morphological analysis requires annotated linguistic data for model training and evaluation. Data sparsity can be a challenge, especially for low-resource languages or specialized domains.
10. Computational Complexity: Morphological analysis involves complex computational processes, such as morphological parsing and generation, which can be computationally expensive, particularly for large vocabularies or complex morphological systems.

b) Discuss the importance and goals of Natural Language Processing.

1. Facilitating Human-Computer Interaction: NLP enables seamless communication between humans and computers, allowing users to interact with devices using natural language commands and queries.
2. Information Extraction: NLP helps extract structured information from unstructured text data, facilitating tasks such as named entity recognition, relation extraction, and event extraction.
3. Text Understanding and Summarization: NLP aims to understand the meaning and context of text documents, enabling tasks such as sentiment analysis, text summarization, and document categorization.
4. Machine Translation: NLP plays a crucial role in machine translation systems, enabling the automatic translation of text between different

languages, fostering global communication and collaboration.

5. Question Answering: NLP systems can comprehend and respond to natural language questions, assisting users in obtaining relevant information from large text corpora or knowledge bases.
6. Sentiment Analysis: NLP helps analyze the sentiment or opinion expressed in text data, allowing businesses to gauge customer sentiment, monitor brand reputation, and make informed decisions.
7. Speech Recognition and Synthesis: NLP techniques enable the conversion of spoken language into text (speech recognition) and text into spoken language (speech synthesis), powering virtual assistants and voice-controlled devices.
8. Information Retrieval: NLP enhances information retrieval systems by improving the accuracy and relevance of search results based on user queries, leading to more efficient access to relevant information.
9. Language Generation: NLP systems generate human-like text, enabling applications such as chatbots, virtual assistants, and content generation for various purposes, including marketing and entertainment.
10. Enhancing Accessibility: NLP technologies contribute to making digital content more accessible to individuals with disabilities, such as screen readers for the visually impaired and speech recognition tools for those with motor impairments.

2. a) Elaborate the Models for Ambiguity Resolution in Parsing.

1. Probabilistic Models: These models assign probabilities to various parse trees based on statistical analysis of a training corpus, allowing for ambiguity resolution by selecting the most likely parse.
2. Constraint-Based Models: These models use linguistic constraints to guide the parsing process, constraining the possible interpretations of ambiguous structures based on syntactic, semantic, or pragmatic

considerations.

3. Rule-Based Models: Rule-based models employ linguistic rules or heuristics to disambiguate parse structures, specifying conditions under which certain parse alternatives are preferred over others.
4. Machine Learning Approaches: Machine learning techniques, such as supervised learning, unsupervised learning, and reinforcement learning, are applied to learn parsing strategies from annotated data, enabling automatic ambiguity resolution.
5. Lexical Disambiguation Models: These models focus on resolving ambiguity at the lexical level by considering word sense disambiguation techniques to select the most appropriate meanings of ambiguous words within a parse.
6. Syntactic Preference Models: Syntactic preference models prioritize parse structures that adhere to syntactic preferences or principles observed in linguistic theory, favoring interpretations that are syntactically well-formed.
7. Semantic Role Labeling: Semantic role labeling models assign semantic roles to constituents within a parse, facilitating ambiguity resolution by identifying the most semantically coherent parse structure based on role assignments.
8. Semantic Parsing: Semantic parsing models translate natural language sentences into formal semantic representations, enabling ambiguity resolution by mapping ambiguous surface structures to unambiguous semantic representations.
9. Incremental Parsing Strategies: Incremental parsing strategies process input sentences incrementally, incrementally building parse structures and resolving ambiguity as more information becomes available during parsing.

10. Hybrid Approaches: Hybrid approaches combine multiple ambiguity resolution strategies, leveraging the strengths of different models to achieve more robust and accurate parsing results in handling various types of ambiguity.

b) With the help of a neat diagram, explain the Representation of Syntactic Structure.

1. Sentence: The syntactic structure represents the organization of words in a sentence, capturing the hierarchical relationships between words and phrases.
2. Phrases: Phrases are syntactic units composed of one or more words that function as a single grammatical unit within a sentence. These include noun phrases, verb phrases, prepositional phrases, etc.
3. Constituents: Constituents are the building blocks of syntactic structure and can be individual words or phrases that form larger syntactic units. They are represented as nodes in the syntactic tree.
4. Nodes: Nodes in the syntactic tree represent constituents, with each node corresponding to a word or a phrase in the sentence. The hierarchical relationships between nodes depict the syntactic structure.
5. Branches: Branches in the syntactic tree represent the hierarchical relationships between constituents. Each branch connects a parent node to its immediate children, indicating the syntactic dependency between them.
6. Dependency Relations: Dependency relations represent the syntactic dependencies between words in a sentence, indicating how words are connected to each other syntactically.
7. Root Node: The root node of the syntactic tree represents the entire sentence, serving as the starting point from which all other constituents

branch out.

8. Internal Structure: The internal structure of phrases is represented by nested nodes within the syntactic tree, with each level of nesting indicating a higher level of syntactic abstraction.
9. Leaf Nodes: Leaf nodes in the syntactic tree represent individual words in the sentence, forming the terminal nodes of the tree structure.
10. Hierarchical Organization: The syntactic structure is organized hierarchically, with smaller constituents nested within larger ones, reflecting the compositional nature of language syntax.

3. Explain the Word Sense Systems in detail.

1. Lexical Ambiguity: Word sense systems aim to address the lexical ambiguity inherent in natural language, where words can have multiple meanings or senses depending on context.
2. Word Sense Disambiguation (WSD): WSD is a fundamental task in word sense systems, involving the identification of the correct sense of a word in a particular context.
3. Sense Inventory: Word sense systems often rely on sense inventories, which are structured collections of word senses associated with individual words. Examples include WordNet, BabelNet, and FrameNet.
4. Sense Discrimination: Word sense systems discriminate between different senses of a word based on various linguistic features, such as word context, syntactic patterns, semantic relations, and collocations.
5. Supervised Learning: Some word sense systems employ supervised learning techniques, where models are trained on annotated datasets containing examples of word senses in context, allowing them to predict the correct sense for unseen instances.
6. Unsupervised Learning: Unsupervised learning approaches cluster word

occurrences based on similarity measures, aiming to group instances of a word into distinct senses without relying on labeled data.

7. Knowledge-based Approaches: Knowledge-based word sense systems leverage external knowledge resources such as dictionaries, ontologies, and semantic networks to disambiguate word senses, often using explicit sense definitions and semantic relations.
8. Contextual Information: Word sense systems consider contextual information surrounding a word to disambiguate its sense, including neighboring words, syntactic structure, discourse context, and semantic relations.
9. Evaluation Metrics: Word sense systems are evaluated using various metrics such as precision, recall, F1-score, accuracy, and semantic similarity, comparing predicted senses against gold standard annotations or human judgments.
10. Applications: Word sense systems have applications in numerous NLP tasks, including machine translation, information retrieval, question answering, text summarization, sentiment analysis, and more, improving the accuracy and performance of these systems by resolving lexical ambiguity.

4. a) Difference between predicate and Predicator.

1. Scope: The predicate refers to the entire grammatical structure that provides information about the subject in a sentence, encompassing the predicator along with any complements, modifiers, and adjuncts.
2. Main Function: The predicator serves as the core element within the predicate, representing the main verb or verb phrase that expresses the action or state of the subject.
3. Grammatical Role: While the predicate encompasses various grammatical

functions such as subject, verb, object, etc., the predicator specifically fulfills the role of the main verb or verb phrase in the sentence.

4. Semantic Role: The predicate conveys the overall meaning of the sentence, including the action, state, or relation expressed by the verb and its associated elements. The predicator contributes directly to this meaning by expressing the primary action or state.
5. Syntactic Structure: Predicates can be complex structures consisting of multiple constituents, including nouns, verbs, adjectives, adverbs, and prepositional phrases. In contrast, the predicator is a single constituent representing the main verb or verb phrase.
6. Expansion: While the predicate can be expanded with additional elements such as objects, adverbials, and modifiers to provide more information, the predicator remains unchanged as the central element expressing the primary action or state.
7. Dependency Relations: The predicate establishes the relationships between the subject and other parts of the sentence, including direct and indirect dependencies. The predicator forms the primary dependency relation, linking the subject to the main action or state.
8. Identification: Identifying the predicate involves analyzing the entire verbal structure of the sentence, whereas identifying the predicator focuses specifically on isolating the main verb or verb phrase.
9. Functionality: Predicates serve to convey the complete message of the sentence, including the action, description, or relation expressed by the verb and its associated elements. Predicators, however, play a more focused role in expressing the central action or state.
10. Role in Sentence Structure: Predicates are essential for structuring sentences and conveying complex meanings, while predicators serve as the core elements that anchor the main action or state within the sentence.

b) Write a short note on Meaning Representation Systems.

1. Semantic Representation: Meaning Representation Systems (MRS) are frameworks designed to represent the meaning of natural language expressions in a structured and formalized manner.
2. Knowledge Representation: MRS aims to capture the semantic content of linguistic expressions, encoding information about entities, actions, events, relationships, and their interconnections.
3. Formalism: MRS typically employs a formalism that consists of a set of symbols, predicates, and relationships, allowing for precise representation and manipulation of meaning.
4. Syntax-Semantics Interface: MRS serves as an interface between syntax and semantics, bridging the gap between the surface structure of sentences and their underlying meaning.
5. Linguistic Features: MRS systems capture various linguistic features such as tense, aspect, mood, modality, quantification, negation, and discourse structure, enabling a comprehensive representation of meaning.
6. Scope and Granularity: MRS can represent meanings at different levels of granularity, ranging from individual words and phrases to entire sentences, paragraphs, or even discourse units.
7. Interpretation and Inference: MRS facilitates semantic interpretation and inference, allowing for reasoning about the meaning of linguistic expressions and deriving implicit information.
8. Applications: MRS has applications in various natural language processing tasks such as machine translation, information extraction, question answering, sentiment analysis, and dialogue systems.
9. Compatibility: MRS systems are designed to be compatible with existing linguistic theories and frameworks, allowing for integration with other

computational linguistic tools and resources.

10. Advancements: Ongoing advancements in MRS research focus on enhancing the expressiveness, scalability, and interoperability of meaning representation systems, thereby improving their utility in real-world NLP applications.

5. a) Elaborate the Multilingual and Cross-lingual Language Modeling.

1. Multilingual Language Modeling (MLM): MLM involves training language models on data from multiple languages, allowing them to understand and generate text in multiple languages.
2. Data Representation: MLM models represent input data from different languages in a shared embedding space, enabling them to capture language-agnostic features and patterns.
3. Language Transfer Learning: MLM leverages transfer learning techniques to transfer knowledge across languages, improving performance on low-resource languages by leveraging data from high-resource languages.
4. Code-Switching: MLM models are capable of handling code-switching scenarios where multiple languages are used within the same sentence or document, facilitating tasks such as language identification and understanding.
5. Cross-lingual Language Modeling (XLM): XLM extends MLM to enable language modeling tasks across different languages, facilitating tasks such as cross-lingual information retrieval, machine translation, and sentiment analysis.
6. Alignment and Projection: XLM aligns embeddings across languages through techniques such as adversarial training or parallel data alignment, enabling knowledge transfer between languages.
7. Zero-shot Learning: XLM models can perform zero-shot learning, where

they generalize to languages not seen during training by leveraging shared representations and cross-lingual transfer.

8. Induction of Typological Knowledge: XLM models can induce typological knowledge about languages, such as syntactic structures or morphological features, from multilingual data, aiding in linguistic research and analysis.
9. Challenges: Challenges in MLM and XLM include handling language diversity, domain adaptation, capturing language-specific nuances, and addressing biases in training data.
10. Applications: MLM and XLM have various applications in NLP, including machine translation, cross-lingual information retrieval, sentiment analysis, multilingual chatbots, and more, facilitating communication and understanding across language barriers.

b) Illustrate the Discourse Cohesion and Structure.

1. Definition of Discourse Cohesion: Discourse cohesion refers to the linguistic mechanisms that create unity and coherence in a text, ensuring that its parts are connected logically and meaningfully.
2. Lexical Cohesion: Lexical cohesion involves the use of cohesive devices such as repetition, synonymy, antonymy, and hyponymy to link different parts of a text together.
3. Reference Cohesion: Reference cohesion involves the use of pronouns, demonstratives, and other referential expressions to refer back to previously mentioned entities or ideas in the discourse.
4. Substitution Cohesion: Substitution cohesion occurs when one element in a text is replaced by another element that serves a similar function, maintaining coherence without repeating the exact same words.
5. Conjunction Cohesion: Conjunction cohesion involves the use of

conjunctions, adverbs, and other connectives to establish logical relationships between sentences and paragraphs, signaling transitions and continuity of thought.

6. Thematic Progression: Thematic progression refers to the organization of information in a text, where themes (main topics) and rhemes (comments on the themes) are structured to maintain coherence and flow.
7. Rhetorical Structure: Rhetorical structure refers to the overall organization and flow of ideas in a discourse, including the introduction, development, and conclusion of arguments or narratives.
8. Textual Structure: Textual structure encompasses the macro-level organization of a discourse, including the arrangement of paragraphs, sections, and chapters to convey the overall message or purpose.
9. Cohesion Markers: Cohesion markers are linguistic cues such as conjunctions, transition words, and discourse markers that signal relationships between different parts of a text and guide the reader through its structure.
10. Importance of Discourse Cohesion: Discourse cohesion is crucial for effective communication, as it helps readers navigate and understand complex texts by providing clear connections between ideas and maintaining a cohesive flow of information.

6. Explain the performance of approaches in Structure of Documents.

1. Accuracy: The performance of approaches in document structure analysis is often evaluated based on their accuracy in correctly identifying structural elements such as headings, paragraphs, lists, and tables within documents.
2. Recall: Recall measures the ability of an approach to correctly identify all relevant structural elements in a document. Higher recall indicates that

the approach can capture a larger portion of the document's structure.

3. Precision: Precision measures the proportion of correctly identified structural elements out of all elements identified by the approach. Higher precision indicates fewer false positives, meaning that the identified structural elements are more likely to be correct.
4. F-score: F-score is a metric that combines precision and recall into a single measure, providing a balanced assessment of an approach's performance. A higher F-score indicates better overall performance.
5. Robustness: Robustness refers to an approach's ability to perform consistently across different types of documents, formats, and languages. A robust approach should maintain high performance levels even when faced with variations in document structure.
6. Scalability: Scalability refers to an approach's ability to handle large volumes of documents efficiently. High-performance approaches should be scalable to process documents of varying lengths and complexities without significant computational overhead.
7. Generalization: Generalization assesses an approach's ability to generalize its learned patterns and rules to unseen documents. Approaches that can generalize effectively perform well on new documents that were not included in the training data.
8. Adaptability: Adaptability refers to an approach's capacity to adapt to changes in document structure over time. As document structures evolve, high-performance approaches should be able to adapt their models and algorithms accordingly.
9. Cross-domain Performance: Cross-domain performance evaluates how well an approach performs when applied to documents from different domains or subject areas. Approaches with high cross-domain performance can effectively analyze documents from diverse sources.

10. Real-world Utility: Ultimately, the performance of approaches in document structure analysis should be assessed based on their real-world utility and applicability. Approaches that can accurately and efficiently analyze document structures contribute to improved document organization, information retrieval, and downstream NLP tasks.

7. Describe the Cohesion and Reference Resolution.

1. Cohesion: Cohesion refers to the linguistic devices used to create unity and coherence in a text by connecting its parts logically and semantically.
2. Types of Cohesion: Cohesion can be achieved through various means, including lexical cohesion (repetition, synonymy, antonymy), grammatical cohesion (reference, substitution, ellipsis), and logical cohesion (conjunctions, connectives).
3. Reference Resolution: Reference resolution is the process of identifying the referents of pronouns, demonstratives, and other referring expressions in a text.
4. Anaphora: Anaphora is a type of reference resolution where a pronoun or other referring expression refers back to a previously mentioned entity or concept in the discourse.
5. Cataphora: Cataphora occurs when a referring expression precedes its referent, referring to something mentioned later in the discourse.
6. Exophora: Exophora refers to references outside the text, such as gestures, context, or shared knowledge between speaker and listener.
7. Coreference Resolution: Coreference resolution is a specific form of reference resolution where different expressions in the text refer to the same entity or concept.
8. Ambiguity: Reference resolution may face challenges due to ambiguity in the text, where a referring expression could potentially refer to multiple

entities or concepts.

9. Syntactic and Semantic Context: Reference resolution often relies on syntactic and semantic context to disambiguate referring expressions and identify their referents.
10. Importance: Cohesion and reference resolution are crucial for text comprehension, as they help readers establish connections between different parts of the text and understand the relationships between entities and concepts mentioned in the discourse.

8. Explain the following:

a) Semantic Parsing with Execution

1. Semantic Parsing: Semantic parsing is the process of mapping natural language utterances into formal representations of meaning, such as logical forms or executable programs.
2. Execution: Execution involves the process of interpreting or executing the formal representations generated by semantic parsing to produce desired outputs, such as answers to questions or actions in a task-oriented dialogue system.
3. Formal Representations: Semantic parsers typically output formal representations that capture the meaning of the input utterance in a structured format, such as lambda calculus expressions, SQL queries, or executable code.
4. Interpretation: Interpretation is the process of understanding the meaning encoded in the formal representations generated by semantic parsing, taking into account the context of the application domain and the specific task at hand.
5. Executable Semantics: Semantic parsing with execution aims to go beyond simple understanding of meaning by enabling the direct execution

of formal representations to perform tasks or answer queries.

6. Task-oriented Dialogue Systems: In task-oriented dialogue systems, semantic parsing with execution allows users to interact with the system using natural language to accomplish specific tasks, such as booking flights or making restaurant reservations.
7. End-to-End Systems: Some semantic parsing approaches incorporate execution directly into the parsing process, enabling end-to-end systems that take natural language input and produce executable actions without intermediate representations.
8. Domain-specific Knowledge: Semantic parsing with execution often requires access to domain-specific knowledge and resources to interpret and execute the formal representations in context.
9. Error Handling: Robust execution mechanisms are needed to handle errors and uncertainties that may arise during interpretation and execution, such as ambiguity in the input or incomplete domain knowledge.
10. Evaluation Metrics: Evaluation of semantic parsing with execution systems often involves assessing both the accuracy of the generated formal representations and the effectiveness of the resulting executions in achieving the desired outcomes in real-world scenarios.

b) Language Model Adaptation

1. Definition: Language model adaptation refers to the process of modifying or fine-tuning an existing language model to better suit a specific domain, genre, or task.
2. Domain Specificity: Adaptation is necessary when the language model needs to perform effectively in a specialized domain, such as healthcare, finance, or legal documents, where the language use differs from general

language.

3. Data Augmentation: Adaptation often involves augmenting the training data with domain-specific or task-specific texts to expose the model to relevant language patterns and vocabulary.
4. Fine-tuning Parameters: Adaptation may require adjusting the parameters of the language model, such as the model architecture, hyperparameters, or training objectives, to better capture domain-specific characteristics.
5. Transfer Learning: Adaptation can leverage transfer learning techniques, where a pre-trained language model is fine-tuned on domain-specific data to transfer knowledge from a general domain to a specific one.
6. Incremental Learning: In scenarios where new data becomes available over time, adaptation allows the language model to continuously learn and improve its performance by incorporating new information without retraining from scratch.
7. Adaptation Techniques: Adaptation techniques include feature-based approaches, where domain-specific features are incorporated into the model, as well as fine-tuning methods that update the model parameters using domain-specific data.
8. Evaluation Metrics: Evaluation of adapted language models typically involves assessing their performance on domain-specific tasks using metrics such as accuracy, perplexity, or task-specific evaluation measures.
9. Robustness: Adaptation aims to enhance the robustness of language models by ensuring that they can effectively handle the linguistic variations and challenges present in the target domain or task.
10. Applications: Adapted language models find applications in various natural language processing tasks, including sentiment analysis, named entity recognition, machine translation, and speech recognition, where domain-specific knowledge is crucial for accurate predictions.

Code No: 155CK
R 18

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD B. Tech III Year I Semester Examinations, September - 2021
NATURAL LANGUAGE PROCESSING
(Computer Science and Engineering)

Time: 3 hours **Max. Marks: 75**

Answer any five questions
All questions carry equal marks

1.a) List the applications and challenges in NLP.

b) Explain any one Morphological model. [7+8]

2.a) Discuss about challenging issues of Morphological models.

b) Differentiate between surface and deep structure in NLP with suitable examples. [7+8]

3.a) Explain various types of parsers in NLP.

b) Discuss multilingual issues in detail. [8+7]

4.a) Given Grammar

S→ AB | BB

A→ CC | AB | a

B→ BB | CA | b

C→ BA | AA | b

Word w= aabb. Apply Top Down Parsing test, the word can be generated or not.

b) Explain Tree Banks and its role in parsing. [8+7]

5.a) State the types of references in discourse.

b) Explain about Homonymy and Polysemy with suitable examples. [8+7]

6.a) How Morphological structure helps to Language Modelling?

b) How to handle semantics? [8+7]

7.a) Give example for Predicate Argument Structure.

b) What are System Paradigms in predicates? [7+8]

8.a) Consider the following training set and implement Bi-gram model to calculate the probability of given Test sentence.

Training set: She said thank you.

She said bye as she walked through the door.

She went to San Diego

Test sentence: She thanked and walk through the door

b) Give some examples for early NLP systems. [8+7]

---ooOoo---

ANSWER KEY

1. a) List the applications and challenges in NLP.

Applications in Natural Language Processing (NLP):

1. Machine Translation: Automatically translating text from one language to another.
2. Sentiment Analysis: Determining the sentiment or emotional tone behind a body of text.
3. Chatbots and Virtual Assistants: Creating conversational agents that can interact with users in natural language.
4. Text Summarization: Generating concise summaries from longer documents or articles.
5. Named Entity Recognition (NER): Identifying and classifying key entities in text such as names of people, organizations, and locations.

Challenges in Natural Language Processing (NLP):

6. Sentiment Detection: Correctly identifying sentiment in texts that contain sarcasm, irony, or mixed sentiments.

7. Domain-Specific Knowledge: Adapting models to understand and generate text in specialized fields like medicine, law, or finance.
8. Bias and Fairness: Mitigating biases present in training data to ensure fair and unbiased NLP models.
9. Multimodal Integration: Integrating text with other forms of data, such as images and videos, for comprehensive understanding.
10. Real-Time Processing: Achieving efficient real-time processing for applications requiring immediate responses, such as live translations or conversational agents.

b) Explain any one Morphological model.

1. Definition: A Finite State Transducer is an extension of a finite state automaton that maps between two sets of symbols, making it suitable for morphological parsing and generation.
2. Structure: FSTs consist of states and transitions. Each transition maps an input symbol to an output symbol and moves from one state to another.
3. Usage in Morphology: In morphological analysis, FSTs can be used to map between surface forms (e.g., "running") and their corresponding lexical forms (e.g., "run + ING").
4. Lexical Analysis: FSTs analyze the structure of words by breaking them down into their base forms and affixes, identifying roots, prefixes, and suffixes.
5. Morphological Generation: FSTs can also be used for generation, transforming base forms and grammatical information into fully inflected words.
6. Efficiency: FSTs are efficient in terms of both time and space, making them suitable for real-time applications.
7. Deterministic vs. Non-deterministic: FSTs can be deterministic, where

each input leads to a single output, or non-deterministic, where multiple paths can be taken, accommodating more complex morphological rules.

8. Bidirectionality: One of the strengths of FSTs is their bidirectional nature, allowing the same model to be used for both analysis (decomposition) and generation (composition) of words.
9. Rule Representation: Morphological rules are encoded in the transitions between states, providing a clear and structured way to represent the morphological processes of a language.
10. Applications: FSTs are widely used in natural language processing tasks such as spell checking, text-to-speech systems, and syntactic parsing due to their ability to handle complex morphological transformations accurately and efficiently.

2. a) Discuss about challenging issues of Morphological models.

1. Ambiguity: Morphological models often face challenges in dealing with ambiguous morphemes that can have multiple meanings or functions depending on the context.
2. Complex Morphologies: Languages with rich morphology, such as Finnish or Turkish, have complex word structures with numerous affixes and inflections, making accurate modeling difficult.
3. Data Sparsity: Lack of extensive annotated corpora for many languages, especially low-resource languages, hinders the development of robust morphological models.
4. Irregular Forms: Handling irregular word forms and exceptions to morphological rules, such as irregular verbs in English, is challenging for morphological models.
5. Dialectal Variations: Variations in morphology across different dialects and regional languages require models to be highly adaptable and

context-sensitive.

6. Morphophonological Changes: Morphological processes often involve phonological changes that need to be accurately captured and modeled, which adds to the complexity.
7. Computational Efficiency: Ensuring that morphological models are efficient in terms of computational resources and can perform in real-time applications is a significant challenge.
8. Integration with Syntax and Semantics: Morphological models need to be integrated with syntactic and semantic models to ensure comprehensive understanding, which is non-trivial.
9. Low-Frequency Words: Dealing with low-frequency words or newly coined terms that may not be present in the training data requires morphological models to generalize effectively.
10. Evaluation Metrics: Developing appropriate evaluation metrics and benchmarks for assessing the performance of morphological models across different languages and tasks remains a challenge.

b) Differentiate between surface and deep structure in NLP with suitable examples.

1. Definition:

   Surface Structure: The literal arrangement of words in a sentence as they appear in written or spoken language.

   Deep Structure: The underlying, abstract representation of a sentence's meaning, capturing the essential syntactic and semantic relationships.

2. Example:

   Surface Structure: "The cat chased the mouse."

   Deep Structure: Represents the action (chasing) and the entities involved (cat as the agent, mouse as the object).

3. Transformations:

   Surface structures can be transformed from deep structures through various grammatical rules (e.g., passive voice transformation).

   Example: Deep Structure for both "The cat chased the mouse" and "The mouse was chased by the cat" would be similar, but their surface structures differ.

4. Ambiguity Resolution:

   Surface Structure: Often ambiguous and can be interpreted in multiple ways.

   Deep Structure: Aims to resolve ambiguities by providing a single, clear interpretation.

   Example: "Visiting relatives can be tiresome." (Ambiguous surface structure) could be resolved in deep structure as either "Relatives who visit can be tiresome" or "The act of visiting relatives can be tiresome."

5. Parsing:

   Surface Structure: Involves parsing the sequence of words.

   Deep Structure: Involves interpreting the parsed structure to derive meaning.

6. Syntax and Semantics:

   Surface Structure: More focused on syntax (word order, grammar).

   Deep Structure: Integrates semantics (meaning) with syntactic structure.

7. Examples of Deep Structure Analysis:

   Question Transformation: Surface Structure "What did John eat?" Deep Structure "John ate what?"

   Active to Passive: Surface Structure "The chef cooked the meal." Deep Structure "The meal was cooked by the chef."

8. Machine Translation:

   Surface Structure: Direct translation of words may lead to inaccuracies.

Deep Structure: Translating the meaning ensures more accurate and contextually relevant translations.

9. Information Extraction:

   Surface Structure: Extracts information based on the literal text.

   Deep Structure: Extracts information based on the underlying meaning, improving accuracy.

10. Practical Application:

    Surface Structure: Used in basic text processing and tokenization.

    Deep Structure: Used in more advanced NLP tasks like understanding context, intent recognition, and semantic analysis.

3. a) Explain various types of parsers in NLP.

1. Top-Down Parser:

   Begins with the highest-level rule and works down, attempting to rewrite the start symbol to match the input string.

   Example: Predictive parsing, Recursive descent parsing.

   Advantage: Simple and easy to implement.

   Disadvantage: Inefficient for ambiguous grammars and left-recursive rules.

2. Bottom-Up Parser:

   Starts with the input string and works upwards, attempting to reconstruct the start symbol.

   Example: Shift-reduce parsing.

   Advantage: Can handle left recursion.

   Disadvantage: More complex to implement and manage.

3. Chart Parser:

   Uses dynamic programming to store intermediate parsing results, reducing redundant computations.

Example: Earley parser, CYK (Cocke-Younger-Kasami) parser.

Advantage: Efficiently handles ambiguous and complex grammars.

Disadvantage: Higher memory consumption.

4. Dependency Parser:

   Focuses on the dependencies between words in a sentence, representing grammatical relations as directed links between words.

   Example: Transition-based parsing, Graph-based parsing.

   Advantage: Provides clear syntactic and semantic relationships.

   Disadvantage: Requires accurate models to predict dependencies correctly.

5. Constituency Parser:

   Divides sentences into sub-phrases (constituents) and represents them as a tree structure.

   Example: Probabilistic Context-Free Grammar (PCFG) parser.

   Advantage: Represents hierarchical structure of sentences well.

   Disadvantage: Computationally intensive for long sentences.

6. Shift-Reduce Parser:

   Uses a stack to hold intermediate results and a set of operations (shift and reduce) to process the input.

   Example: LR parser (Left-to-right, Rightmost derivation).

   Advantage: Efficient for a wide range of grammars.

   Disadvantage: Complex implementation and requires pre-computed parsing tables.

7. Recursive Descent Parser:

   A type of top-down parser that uses a set of recursive functions to process the input.

   Example: Simple hand-written parsers for specific grammars.

   Advantage: Intuitive and straightforward to implement.

Disadvantage: Cannot handle left-recursive rules directly.

8. Earley Parser:

   A chart parser that efficiently handles any Context-Free Grammar (CFG), including left-recursive and ambiguous grammars.

   Advantage: General-purpose, works well for many different types of grammars.

   Disadvantage: Relatively high computational complexity.

9. Probabilistic Parser:

   Incorporates probabilities into parsing to handle ambiguities by choosing the most likely parse.

   Example: Probabilistic Context-Free Grammar (PCFG) parser.

   Advantage: Can disambiguate based on statistical information.

   Disadvantage: Requires a large amount of annotated training data.

10. Transition-Based Parser:

    Uses a sequence of actions to build a parse tree, often employed in dependency parsing.

    Example: Arc-standard, Arc-eager parsing algorithms.

    Advantage: Efficient and fast, suitable for real-time applications.

    Disadvantage: Requires careful design of transition actions and training data.

b) Discuss multilingual issues in detail.

1. Language Diversity:

   Each language has unique syntactic, semantic, and morphological structures. Developing NLP models that can handle this diversity is challenging as it requires understanding and implementing rules specific to each language.

2. Data Scarcity:

Many languages, especially those spoken by smaller populations, lack large, high-quality annotated datasets. This scarcity makes it difficult to train robust and accurate NLP models for these languages.

3. Translation Quality:

   Machine translation systems often struggle with idiomatic expressions, cultural nuances, and context-specific meanings. Ensuring high-quality translations that preserve the intended meaning across languages remains a significant challenge.

4. Code-Switching:

   In multilingual communities, speakers often switch between languages within a conversation or even a sentence. NLP systems must be able to detect and process code-switching to understand the full context.

5. Standardization:

   Variations in dialects, orthographic conventions, and writing systems across languages can complicate NLP tasks. Establishing standardized forms of languages for NLP processing is complex but necessary for consistency.

6. Resource Allocation:

   Developing multilingual NLP systems requires significant computational resources and expertise. Balancing resource allocation between high-resource and low-resource languages is a critical issue.

7. Cross-Linguistic Transfer:

   Techniques such as transfer learning, where knowledge from high-resource languages is transferred to low-resource languages, are promising but still face challenges in terms of effectiveness and applicability across diverse language pairs.

8. Bias and Fairness:

   NLP models trained on data from dominant languages and cultures may

exhibit biases, leading to unfair or inaccurate processing of text from underrepresented languages and communities. Ensuring fairness and reducing bias is a critical concern.

9. Multilingual NLP Evaluation:

   Developing appropriate evaluation metrics and benchmarks for multilingual NLP systems is challenging. These metrics must account for linguistic diversity and the varying levels of resource availability for different languages.

10. Cultural Context:

    Understanding and accurately processing the cultural context behind language use is essential for effective multilingual NLP. This includes recognizing cultural references, idioms, and societal norms that vary widely between languages.

4. a) Given Grammar

S → AB | BB

A → CC | AB | a

B → BB | CA | b

C → BA | AA | b

Word w= aabb. Apply Top Down Parsing test, the word can be generated or not.

Given grammar:

S → AB | BB

A → CC | AB | a

B → BB | CA | b

C → BA | AA | b

Start with S:

S → AB

Expand A (using A → a, the simplest production):

S → aB

Expand B (using B → b, the simplest production):

S → ab

At this point, we have "ab", which is not the desired string "aabb". We need to backtrack and try different expansions.

Restart with S:

S → AB

Expand A (using A → AB):

S → ABB

Expand the first B (using B → b):

S → AbB

Expand A again (using A → a):

S → abB

Expand the second B (using B → b):

S → abb

Again, this doesn't match "aabb". So, we backtrack again.

Restart with S:

S → AB

Expand A (using A → CC):

S → CCB

Expand the first C (using C → b):

S → bCB

Expand the second C (using C → b):

S → bbB

Expand B (using B → b):

S → bbb

This also doesn't match "aabb". So, we backtrack again.

Restart with S:

S → BB

Expand the first B (using B → b):

S → bB

Expand the second B (using B → b):

S → bb

This again doesn't match "aabb".

Let's try different paths:

17. Restart with S:

S → AB

Expand A (using A → CC):

S → CCB

Expand the first C (using C → BA):

S → BACB

Expand B (using B → b):

S → baCB

Expand C (using C → b):

S → babB

Expand the final B (using B → b):

S → babb

This doesn't match "aabb" either.

After systematically trying all possible expansions, none of the derivations match "aabb".

Therefore, using top-down parsing, we conclude that the word "aabb" cannot be generated by the given grammar.

b) Explain Tree Banks and its role in parsing.

1. Definition:

   Tree Banks are large, structured datasets of sentences annotated with their syntactic or semantic tree structures. These annotations typically represent the grammatical structure of the sentences.

2. Purpose:

   They provide a gold standard for training and evaluating parsers, enabling the development of accurate syntactic analysis tools in NLP.

3. Annotation:

   Sentences in tree banks are annotated with detailed parse trees, showing the syntactic structure, including parts of speech (POS), phrases, and grammatical relations.

4. Examples:

   Notable tree banks include the Penn Treebank for English, the Prague Dependency Treebank for Czech, and the Universal Dependencies project for multiple languages.

5. Training Parsers:

   Tree banks are essential for supervised learning in parsing. They provide labeled examples that parsers use to learn how to map sentences to their correct syntactic structures.

6. Evaluation:

   Tree banks serve as benchmarks for evaluating the performance of different parsing algorithms. By comparing the output of a parser to the tree bank annotations, researchers can assess accuracy and identify areas for improvement.

7. Parsing Techniques:

   Different parsing techniques, such as dependency parsing and constituency parsing, rely on tree banks for development and validation. Tree banks ensure that these techniques capture the necessary syntactic

relations accurately.

8. Syntax and Semantics:

   While primarily focused on syntax, some tree banks also include semantic annotations, helping in the development of parsers that can understand both grammatical structure and meaning.

9. Cross-Linguistic Research:

   Tree banks for multiple languages facilitate cross-linguistic research, enabling the study and comparison of syntactic structures across different languages. This is crucial for developing multilingual parsers.

10. Challenges:

    Creating and maintaining tree banks is resource-intensive, requiring significant human expertise and time for accurate annotation. Ensuring consistency and handling ambiguities are ongoing challenges.

5. a) State the types of references in discourse.

1. Definition:

   Tree Banks are large, structured datasets of sentences annotated with their syntactic or semantic tree structures. These annotations typically represent the grammatical structure of the sentences.

2. Purpose:

   They provide a gold standard for training and evaluating parsers, enabling the development of accurate syntactic analysis tools in NLP.

3. Annotation:

   Sentences in tree banks are annotated with detailed parse trees, showing the syntactic structure, including parts of speech (POS), phrases, and grammatical relations.

4. Examples:

   Notable tree banks include the Penn Treebank for English, the Prague

Dependency Treebank for Czech, and the Universal Dependencies project for multiple languages.

5. Training Parsers:

   Tree banks are essential for supervised learning in parsing. They provide labeled examples that parsers use to learn how to map sentences to their correct syntactic structures.

6. Evaluation:

   Tree banks serve as benchmarks for evaluating the performance of different parsing algorithms. By comparing the output of a parser to the tree bank annotations, researchers can assess accuracy and identify areas for improvement.

7. Parsing Techniques:

   Different parsing techniques, such as dependency parsing and constituency parsing, rely on tree banks for development and validation. Tree banks ensure that these techniques capture the necessary syntactic relations accurately.

8. Syntax and Semantics:

   While primarily focused on syntax, some tree banks also include semantic annotations, helping in the development of parsers that can understand both grammatical structure and meaning.

9. Cross-Linguistic Research:

   Tree banks for multiple languages facilitate cross-linguistic research, enabling the study and comparison of syntactic structures across different languages. This is crucial for developing multilingual parsers.

10. Challenges:

    Creating and maintaining tree banks is resource-intensive, requiring significant human expertise and time for accurate annotation. Ensuring consistency and handling ambiguities are ongoing challenges.

b) Explain about Homonymy and Polysemy with suitable examples.

1. Definition of Homonymy:

   Homonymy occurs when two words share the same spelling or pronunciation but have different meanings and origins.

2. Example of Homonymy:

   "Bank": Can refer to a financial institution or the side of a river. These meanings are unrelated and have different etymological roots.

3. Types of Homonyms:

   Homophones: Words that sound the same but may have different spellings (e.g., "bare" and "bear").

   Homographs: Words that are spelled the same but may have different pronunciations (e.g., "lead" (to guide) and "lead" (a metal)).

4. Definition of Polysemy:

   Polysemy occurs when a single word has multiple related meanings.

5. Example of Polysemy:

   "Mouth": Can refer to the opening of a face or the opening of a cave or river. These meanings are related by the concept of an opening through which something flows or passes.

6. Context in Polysemy:

   The different meanings of a polysemous word are often connected by a shared concept, and context usually helps in determining the intended meaning.

7. Linguistic Analysis:

   Homonymy and polysemy are important in linguistic analysis because they affect how words are understood and processed in different contexts.

8. Challenges in NLP:

   Disambiguating between homonymous and polysemous words is

challenging for NLP systems. Correct interpretation relies on context, requiring sophisticated algorithms and extensive language understanding.

9. Impact on Language Processing:

   Accurate handling of homonymy and polysemy is crucial for tasks like machine translation, information retrieval, and semantic analysis, where precise meaning is essential.

10. Example in Sentences:

    Homonymy: "The crane (bird) flew over the construction crane (machine)."

    Polysemy: "She opened her mouth (body part) to speak about the mouth (entrance) of the cave."

6. a) How Morphological structure helps to Language Modelling?

1. Vocabulary Reduction:

   By analyzing the morphological structure, language models can break down words into their root forms and affixes, reducing the vocabulary size and making the model more efficient.

2. Handling Inflections:

   Morphological analysis helps in understanding and generating various inflected forms of words, ensuring that language models can accurately handle grammatical variations such as tense, number, and case.

3. Improving Generalization:

   Models that incorporate morphological knowledge can better generalize across different forms of a word, improving performance on unseen data by recognizing that "run," "running," and "ran" are related.

4. Dealing with Low-Frequency Words:

   By decomposing words into morphemes, language models can handle low-frequency words more effectively, as they can recognize familiar

morphemes even in rare or novel word forms.

5. Enhancing Context Understanding:

   Morphological structure provides additional context for word meaning, helping models disambiguate words with multiple meanings based on their morphological components.

6. Supporting Multilingual Models:

   Morphological analysis is crucial for languages with rich morphology. Understanding the structure of words in such languages improves the accuracy and efficiency of multilingual language models.

7. Improving Text Generation:

   Morphologically aware language models can generate more grammatically correct and contextually appropriate text, especially in morphologically rich languages where word forms need to match the grammatical context.

8. Semantic Understanding:

   Morphology often encodes semantic information, such as the relationship between "teacher" and "teach." Understanding this helps models better capture the semantics of words and their derivations.

9. Better Tokenization:

   Morphological structure aids in more meaningful tokenization, breaking down words into morphemes rather than arbitrary subword units, leading to more coherent language models.

10. Error Correction:

    Knowledge of morphological rules can assist in spelling and grammar correction, as models can recognize and correct errors in the usage of inflections and derivations.

b) How to handle semantics?

1. Semantic Parsing:

Convert natural language into a formal representation of its meaning, such as logical forms or semantic graphs. This helps in understanding the relationships between entities and actions in a sentence.

2. Word Sense Disambiguation (WSD):

   Identify the correct meaning of a word with multiple meanings based on the context. Techniques include using dictionaries, machine learning models, and context-based algorithms.

3. Named Entity Recognition (NER):

   Detect and classify named entities (e.g., people, organizations, locations) in text. This helps in understanding specific references and their roles within the context.

4. Distributional Semantics:

   Use vector space models like Word2Vec, GloVe, or FastText to capture word meanings based on their usage in large corpora. These models represent words as high-dimensional vectors that encode semantic similarity.

5. Contextualized Word Embeddings:

   Employ models like BERT, GPT, or ELMo that generate dynamic word embeddings based on the context. These models understand words in relation to their surrounding text, improving semantic comprehension.

6. Semantic Role Labeling (SRL):

   Identify the roles that words play in a sentence (e.g., agent, patient, instrument). SRL helps in understanding the relationships between different parts of a sentence.

7. Ontology and Knowledge Graphs:

   Use structured representations of knowledge, such as ontologies and knowledge graphs (e.g., DBpedia, WordNet), to provide background information and relationships between concepts.

8. Co-reference Resolution:

   Determine which words or phrases refer to the same entity in a text. This helps in maintaining semantic consistency and understanding the context across sentences.

9. Natural Language Inference (NLI):

   Use NLI models to determine whether a given hypothesis logically follows from a premise. This aids in understanding implications and relationships between statements.

10. Sentiment Analysis:

    Analyze the sentiment or emotional tone behind text to understand the underlying attitudes and opinions. This helps in capturing the subjective meaning of the text.

7. a) Give example for Predicate Argument Structure.

1. Definition:

   Predicate Argument Structure (PAS) represents the relationships between a predicate (verb) and its arguments (subjects, objects, and other complements) in a sentence.

2. Example Sentence:

   "John eats an apple."

3. Predicate:

   "Eats" is the predicate in this sentence, representing the action or event.

4. Arguments:

   "John" is the subject argument, the entity performing the action.

   "An apple" is the object argument, the entity that undergoes the action.

5. Roles:

   The predicate "eats" assigns specific roles to its arguments: John is the agent (the doer of the action), and the apple is the patient (the entity

affected by the action).

6. Transitivity:

   The transitivity of the predicate determines the number of arguments it requires. In this example, "eats" is a transitive verb, requiring both a subject (John) and an object (an apple).

7. Semantic Roles:

   Beyond syntactic roles like subject and object, PAS also encompasses semantic roles such as agent, patient, theme, experiencer, etc., which capture the semantic relationships between the predicate and its arguments.

8. Alternative Structures:

   Depending on the sentence structure, the same predicate may have different arguments. For example, in "An apple is eaten by John," "John" becomes the agent, and "an apple" becomes the patient, but the predicate "eaten" remains the same.

9. Semantic Interpretation:

   PAS helps in interpreting the meaning of a sentence by identifying the participants involved in the action/event and their respective roles.

10. Analysis in NLP:

    Understanding Predicate Argument Structure is crucial for tasks such as semantic role labeling, information extraction, and natural language understanding, as it provides insights into the relationships between different elements in a sentence.

b) What are System Paradigms in predicates?

1. Definite Description:

   Refers to a specific entity known to both the speaker and the listener, often introduced with "the" (e.g., "the car").

2. Indefinite Description:

   Refers to a non-specific entity, often introduced with "a" or "an" (e.g., "a book").

3. Generic Predicate:

   Describes a characteristic or property of a whole class or category of entities (e.g., "Cats purr").

4. Existential Predicate:

   Asserts the existence of at least one entity satisfying a condition (e.g., "There is a cat in the garden").

5. Universal Predicate:

   Asserts that a certain property holds for all instances of a given class (e.g., "All cats have fur").

6. Distributive Predicate:

   Asserts a property or action for each member of a group individually (e.g., "The students each received a book").

7. Impersonal Predicate:

   Expresses a property or action without reference to a specific subject (e.g., "It rains").

8. Existential-Inchoative Predicate:

   Indicates the existence of a specific entity or event and its beginning or onset (e.g., "The flower bloomed").

9. Equative Predicate:

   Asserts an identity or equality between two entities (e.g., "John is a doctor").

10. Attributive Predicate:

    Attributes a property or characteristic to the subject (e.g., "The sky is blue").

8. a) Consider the following training set and implement Bi-gram model to calculate the probability of given Test sentence.

Training set: She said thank you.

She said bye as she walked through the door.

She went to San Diego

Test sentence: She thanked and walk through the door

1. Tokenization:

   First, tokenize both the training and test sentences into individual words. In this case, the training set would contain tokens: ["She", "said", "thank", "you", ".", "She", "said", "bye", "as", "she", "walked", "through", "the", "door", ".", "She", "went", "to", "San", "Diego"] and the test sentence would be ["She", "thanked", "and", "walk", "through", "the", "door"].

2. Bi-gram Creation:

   Generate bi-grams from the training set, which are pairs of consecutive words. For example, from the training set, we would have bi-grams like ["She", "said"], ["said", "thank"], ["thank", "you"], ["you", "."], ["She", "said"], ["said", "bye"], etc.

3. Counting Frequencies:

   Count the occurrences of each bi-gram in the training set to calculate their frequencies. For example, ["She", "said"] appears twice, ["said", "thank"] appears once, etc.

4. Calculating Probabilities:

   For each bi-gram, divide its frequency by the frequency of its first word to calculate the conditional probability. This represents the probability of the second word given the first word in the bi-gram.

5. Handling Unknown Words:

   If a bi-gram contains a word that is not present in the training set, apply smoothing techniques such as add-one smoothing or interpolation to

estimate its probability.

6. Applying the Model:

   For the test sentence, calculate the probability of each bi-gram using the probabilities obtained from the training set.

7. Combining Probabilities:

   Multiply the probabilities of all bi-grams in the test sentence to obtain the overall probability of the sentence according to the bi-gram model.

8. Interpolation:

   Consider using interpolation with uni-grams (single-word probabilities) to handle cases where certain bi-grams are not present in the training set.

9. Normalization:

   Normalize the probabilities to ensure that they sum up to 1, as probabilities should always be between 0 and 1.

10. Evaluation:

    Finally, evaluate the performance of the bi-gram model by comparing its probability estimates with the actual probabilities or by using it for tasks such as language modeling or text generation.

b) Give some examples for early NLP systems.

1. ELIZA (1966):

   ELIZA, created by Joseph Weizenbaum, was one of the earliest chatbot programs. It simulated a Rogerian psychotherapist and engaged in simple natural language conversations with users.

2. SHRDLU (1968):

   Developed by Terry Winograd, SHRDLU was a natural language understanding system that could manipulate blocks in a virtual world based on commands expressed in English.

3. STUDENT (1967):

STUDENT, created by Daniel Bobrow, was a program designed to solve high school-level algebra word problems. It demonstrated early attempts at semantic parsing and problem-solving.

4. METEO (1970s):

   METEO, developed by Racter (William Chamberlain and Thomas Etter), generated weather forecasts and composed poetry using a rule-based approach. It was notable for its creative output.

5. XCON (1980s):

   XCON, developed by John McDermott at Carnegie Mellon University, was an expert system used for configuring computer systems. It employed rule-based reasoning to provide expert advice on hardware and software configurations.

6. TALK (1980s):

   TALK (Text Analysis and Language Knowledge) was a natural language understanding system developed by the Artificial Intelligence Corporation. It could understand and respond to user queries about sports scores and stock prices.

7. BASEBALL (1980s):

   BASEBALL, created by Gerald DeJong, was a natural language interface for querying a database of baseball statistics. It allowed users to ask questions in English and receive relevant answers from the database.

8. PARADOX (1980s):

   PARADOX, developed by Wendy Lehnert and Chuck Rich, was a natural language understanding system that could answer questions about Sherlock Holmes stories. It demonstrated early work in text comprehension and inference.

9. NELL (Never-Ending Language Learning):

   NELL, developed by Tom Mitchell and colleagues at Carnegie Mellon

University, was a project aimed at creating a continuously learning system. It extracted facts and knowledge from the web by reading and analyzing text.

10. LINGO (1970s):

LINGO, developed by Woods and Schmolze, was one of the earliest natural language understanding systems. It could answer questions about a restricted domain of geography and was used to demonstrate semantic interpretation techniques.