**UNIT 1**

**1️⃣ A.I. Representation**

**Description:**

A.I. Representation refers to the ways we encode information and knowledge into a form that computers can understand and reason with to solve problems.

**Types of Representations:**

* **Logic:** Uses formal rules (propositional and first-order logic) to express facts and rules.
* **Semantic Networks:** Graph structure where nodes represent concepts and links represent relationships.
* **Frames:** Data structures representing stereotypical situations (objects with attributes and values).
* **Scripts:** Describe sequences of events/actions in routine situations.
* **State Space Representation:** Describes a problem by its states, actions, and transitions.

**Sample Viva Q&A**

**Q:** What is knowledge representation?  
**A:** It is a formal way of encoding knowledge, facts, and rules so that a machine can process and use them to solve problems.

**Q:** Difference between frames and scripts?  
**A:** Frames describe objects and their properties (static knowledge); scripts describe sequences of events (procedural knowledge).

**2️⃣ Non-AI & AI Techniques**

**Description:**

These are two broad classes of methods to solve computational problems.

* **Non-AI Techniques:**  
  Use predefined, deterministic algorithms (exact, rule-based, no learning).  
  Example: Sorting algorithms, mathematical computations.
* **AI Techniques:**  
  Handle uncertainty, incomplete data, and adapt over time.  
  Example: Expert systems, machine learning, natural language processing.

**Sample Viva Q&A**

**Q:** Can non-AI solve chess?  
**A:** No, because chess involves complex search and reasoning under uncertainty, suited for AI methods.

**Q:** Key difference between AI and Non-AI?  
**A:** AI adapts, learns, and handles uncertainty. Non-AI works on predefined, exact rules.

**3️⃣ Representation of Knowledge**

**Description:**

Refers to how knowledge is encoded for an AI system to process.

* **Declarative Knowledge:**  
  Facts and statements (e.g., "Paris is the capital of France")
* **Procedural Knowledge:**  
  How-to instructions (e.g., solving an equation)
* **Inference Mechanisms:**  
  Use rules of logic to deduce new knowledge from existing facts.
  + **Deduction:** General to specific
  + **Induction:** Specific to general

**Sample Viva Q&A**

**Q:** What is procedural knowledge? Example?  
**A:** Procedural knowledge defines steps to complete a task, e.g., how to solve a quadratic equation.

**Q:** What is inference?  
**A:** Deriving new knowledge logically from existing facts.

**4️⃣ Knowledge Base Systems**

**Description:**

Computer systems that store expert knowledge to solve complex problems.

* **Components:**
  + **Knowledge Base:** Stores facts and rules.
  + **Inference Engine:** Applies reasoning to derive new conclusions.
* **Example:**  
  MYCIN (diagnoses bacterial infections)

**Sample Viva Q&A**

**Q:** What is backward chaining?  
**A:** A reasoning approach that starts with a goal and works backward to find supporting evidence.

**Q:** Example of a knowledge-based system?  
**A:** MYCIN, used in medical diagnosis.

**5️⃣ State Space Search**

**Description:**

A method of representing and solving problems as a set of states and transitions.

* **Components:**
  + **Initial state**
  + **Goal state**
  + **Operators:** Actions to move between states
  + **State Space:** All possible states
  + **Solution Path:** Sequence of actions from initial to goal

**Sample Viva Q&A**

**Q:** What is state space?  
**A:** The complete set of all possible states and transitions in a problem.

**Q:** Example of state space problem?  
**A:** 8-puzzle or route-finding between cities.

**6️⃣ Production Systems**

**Description:**

A rule-based system where knowledge is represented using IF-THEN rules.

* **Components:**
  + **Production Rules (IF-THEN)**
  + **Working Memory:** Current facts
  + **Control Strategy:** Decides which rule to apply

**Example:**

IF fever THEN take medicine

**Sample Viva Q&A**

**Q:** What is control strategy?  
**A:** It selects the most appropriate rule to fire when multiple rules are applicable.

**Q:** Example of production rule?  
**A:** IF temperature > 100 THEN fan = ON

**7️⃣ Problem Characteristics**

**Description:**

Features that define and classify AI problems:

* **Single vs Multiple Solutions**
* **Deterministic (**predictable**) vs Stochastic** uncertainty
* **Static vs Dynamic**
* **Fully Observable(access to all relevant information) vs Partially Observable(decisions with incomplete knowledge)**
* **Discrete vs Continuous**

**Sample Viva Q&A**

**Q:** What is deterministic problem? Example?  
**A:** One where actions always produce predictable results. Example: Puzzle solving.

**Q:** What is static vs dynamic?  
**A:** Static: World does not change while agent thinks.  
Dynamic: World changes (e.g., self-driving cars).

**8️⃣ Types of Production Systems**

**Description:**

Variants of production systems based on how rules and facts evolve.

* **Monotonic:** Derived facts are never invalidated.
* **Non-Monotonic:** Facts can be retracted/invalidated.
* **Partially Commutative:** Order of rule application doesn’t affect final result.

**Sample Viva Q&A**

**Q:** What is monotonic production system?  
**A:** One where new knowledge does not contradict old facts.

**Q:** Why non-monotonic systems?  
**A:** To model real-life scenarios where new evidence may invalidate earlier conclusions.

**9️⃣ Turing Test**

**Description:**

A test to check if a machine can exhibit intelligent behavior indistinguishable from a human.

* **Proposed by:** Alan Turing, 1950
* **Setup:** Human judge converses with human and machine via text. If judge cannot reliably tell which is machine, the machine passes.

**Sample Viva Q&A**

**Q:** Goal of Turing Test?  
**A:** To determine if a machine can mimic human conversational abilities.

**Q:** Limitation of Turing Test?  
**A:** It measures imitation, not genuine understanding or reasoning.

**🔟 Intelligent Agents**

**Agents and Environments**

**Description:**

An agent is an entity that perceives and acts upon an environment.

* **Environment types:**  
  Fully/Partially observable, Deterministic/Stochastic, Static/Dynamic

**Sample Viva Q&A**

**Q:** Example of partially observable environment?  
**A:** Driving in fog (limited visibility).

**Concept of Rationality**

**Description:**

A rational agent selects actions that maximize expected performance, based on knowledge and percepts.

**Sample Viva Q&A**

**Q:** Can rational agent make mistakes?  
**A:** Yes, if it has incomplete or wrong information.

**Structure of Agents**

**Description:**

Agent types based on complexity:

* **Simple reflex agent**: Reacts to current percept (no memory)
* **Model-based reflex**: Uses internal state (memory)
* **Goal-based agent**: Uses goals to choose actions
* **Utility-based agent**: Maximizes utility (preferences)
* **Learning agent**: Improves over time using feedback

**Sample Viva Q&A**

**Q:** Goal-based vs Utility-based agent?  
**A:** Goal-based seeks any solution; Utility-based picks best among alternatives.

**Problem Solving Agents**

**Description:**

Agents that systematically plan and solve goals using search.

* **Steps:**  
  Goal formulation → Problem formulation → Search → Execution

**Sample Viva Q&A**

**Q:** Why is transition model needed?  
**A:** To predict results of actions.

**1️⃣1️⃣ Problem Formulation (Examples)**

**Vacuum World**

**Description:**

A simple agent world with two rooms and dirt.

* **States:** Agent location, dirt status
* **Actions:** Move left, move right, suck dirt
* **Goal:** Clean both rooms

**Sample Viva Q&A**

**Q:** Task environment (PEAS) of vacuum world?  
**A:** Performance = Clean rooms  
Environment = 2 rooms  
Actuators = Move, Suck  
Sensors = Dirt detection

**8 Queens**

**Description:**

Place 8 queens on chessboard such that none attack each other.

* **States:** Configurations of queens
* **Constraints:** No row, column, diagonal threats

**Sample Viva Q&A**

**Q:** What is constraint satisfaction in 8 queens?  
**A:** Ensuring no two queens threaten each other on board.

**Route Finding**

**Description:**

Find path between locations on a map.

* **States:** Cities/locations
* **Actions:** Travel to neighboring cities
* **Goal:** Reach destination with minimal cost

**Sample Viva Q&A**

**Q:** What is path cost?  
**A:** Total cost of traversed edges between cities.

**Robot Navigation**

**Description:**

A robot navigates in a map with obstacles.

* **States:** Robot location, environment map
* **Actions:** Move, turn
* **Goal:** Reach target location

**Sample Viva Q&A**

**Q:** Why is robot navigation dynamic and partially observable?  
**A:** Environment changes (moving obstacles), and sensors are limited.

**UNIT 2**

**🔍 UNINFORMED SEARCH METHODS**

**1. Depth First Search (DFS)**

* Explores as far as possible along each branch before backtracking.
* Uses a stack (LIFO) structure – either implicitly through recursion or explicitly.
* May not find the shortest path.
* Can go into infinite loops in the absence of a cycle-checking mechanism.

**Viva Question:**  
*What is the major drawback of DFS?*  
**Answer:** DFS can enter an infinite loop in graphs with cycles and may not find the optimal solution.

**2. Breadth First Search (BFS)**

* Explores all neighbors at the current depth before moving to the next level.
* Uses a queue (FIFO) structure.
* Guarantees shortest path in an unweighted graph.
* Memory-intensive due to queue size.

**Viva Question:**  
*Why is BFS considered complete and optimal?*  
**Answer:** Because it explores all nodes at a level before going deeper, ensuring the shortest path is found in unweighted graphs.

**3. Depth Limited Search (DLS)**

* DFS with a depth limit.
* Prevents infinite descent into deep or cyclic paths.
* Risk of missing solution if it lies beyond depth limit.

**Viva Question:**  
*When would you use DLS instead of DFS?*  
**Answer:** When there's a risk of infinite paths, and we want to limit how deep the search goes.

**4. Iterative Deepening DFS (IDDFS)**

* Combines the space-efficiency of DFS with the optimality of BFS.
* Repeatedly applies DFS with increasing depth limits.
* Complete and optimal for unweighted graphs.

**Viva Question:**  
*Why is IDDFS preferred over DFS and BFS in large graphs?*  
**Answer:** Because it uses less memory like DFS and guarantees optimal solution like BFS.

**5. Bidirectional Search**

* Starts from both initial and goal nodes and works towards meeting in the middle.
* Time and space complexity reduced from O(b^d) to O(b^(d/2)).
* Requires both forward and reverse branching factors.

**Viva Question:**  
*What is the biggest challenge in implementing bidirectional search?*  
**Answer:** Reversing the goal state search and managing duplicate paths.

**6. Comparison of Uninformed Search Strategies**

| **Algorithm** | **Complete** | **Optimal** | **Time Complexity** | **Space Complexity** |
| --- | --- | --- | --- | --- |
| BFS | Yes | Yes | O(b^d) | O(b^d) |
| DFS | No | No | O(b^m) | O(bm) |
| DLS | Yes (if depth is enough) | No | O(b^l) | O(bl) |
| IDDFS | Yes | Yes | O(b^d) | O(bd) |
| Bidirectional | Yes | Yes | O(b^(d/2)) | O(b^(d/2)) |

**🎯 INFORMED SEARCH METHODS**

**1. Generate and Test**

* Generates potential solutions and tests them.
* Can be random or systematic.
* Systematic versions use heuristics for efficiency.
* Example: Traveling Salesman Problem (TSP).

**Viva Question:**  
*What is the British Museum approach?*  
**Answer:** A random version of generate-and-test where the algorithm explores blindly, like wandering in a museum to find a specific item.

**2. Hill Climbing**

* Starts from an arbitrary solution and iteratively makes changes to improve it.
* Moves in the direction of increasing value (or decreasing cost).
* May get stuck in local maxima.
* Imp points: memory efficient discards the pervious steps
* solutions

Backtracking

Jump and start from new area

**Viva Question:**  
*What are the limitations of Hill Climbing?*  
**Answer:** Local maxima, plateaus, and ridges can trap the algorithm and prevent reaching the global optimum.

**3. Best First Search**

* Uses a priority queue ordered by a heuristic function.
* Chooses the most promising node first.
* Greedy in nature (focuses on h(n)).

**Working:**open list: not visited  
close list: expanded

Explores the node with least heuristic value

**Viva Question:**  
*How does Best First Search differ from A*?\*  
**Answer:** Best First Search uses only the heuristic (h(n)) to decide the next node, while A\* uses both cost-so-far and heuristic (f(n) (evaluationfunc)= g(n)(distance) + h(n))(hueristic cost).

**4. *A Algorithm*\***

* Combines the benefits of BFS and heuristic search.
* f(n) = g(n) + h(n)
* Optimal and complete if h(n) is admissible (never overestimates).
* Less value after computation
* Completeness optimality
* Backtracking
* Keeping the unselected as well
* Time,space B^m
* **Question:**  
  *What makes A* optimal?\*  
  **Answer:** An admissible heuristic ensures that the lowest cost path is always found.

**5. *AO Algorithm*\***

**Key Features:**

* Based on AND-OR graph – represents problems where multiple sub-goals need to be solved together.
* Works like Best First Search.
* Uses heuristics and cost function f(n) = g(n) + h(n)
* Efficient in decomposable problems (e.g., planning, game trees, NLP).
* Uses back-propagation of cost to update parent nodes.

**Algorithm Steps:**

1. Initialize graph at start node.
2. Traverse current path, expand unexplored nodes.
3. Compute f’ values for successors.
4. If f’ = 0, mark as SOLVED.
5. Back-propagate to update parent costs.
6. Repeat until goal is solved or deemed futile.

**Viva Questions:**

1. *What is an AND-OR graph?*  
   **Answer:** A structure where nodes may require solving multiple sub-nodes together (AND) or choosing one of many alternatives (OR).
2. *What makes AO* different from A\*?\*  
   **Answer:** AO\* handles problems with multiple simultaneous subgoals (AND) using an AND-OR graph, while A\* finds a single best path in OR-graphs.

**6. Constraint Satisfaction**

* Involves variables, domains, and constraints.
* Solves by assigning values satisfying all constraints.
* Uses techniques like backtracking, forward checking, and constraint propagation.

**Viva Question:**  
*What is constraint propagation?*  
**Answer:** Reducing domains of variables by applying constraints before actual search begins.

**🎮 GAME PLAYING**

**1. Minimax Search**

* Used in two-player games.
* Maximizer aims to maximize score, minimizer aims to reduce it.
* Works on depth-limited game trees.
* Complete and optimal(if min and max are optimal)

**Viva Question:**  
*What’s the main assumption of Minimax?*  
**Answer:** That both players play optimally.

**2. Alpha-Beta Cutoffs**

* Optimization to Minimax that prunes unneeded branches.
* α: best option for maximizer; β: best for minimizer.
* Prunes when α ≥ β.

**Viva Question:**  
*How does Alpha-Beta improve Minimax?*  
**Answer:** It avoids exploring parts of the tree that cannot influence the final decision, saving time.

**3. Waiting for Quiescence**

* Delays evaluation until the position is “quiet” (i.e., no imminent dramatic changes).
* Prevents misleading evaluations during volatile positions in games like chess.

**Viva Question:**  
*Why is quiescence important in game-playing AI?*  
**Answer:** To avoid unstable evaluations caused by immediate, drastic changes (like captures in chess).

UNIT 3

**Logical Agents**

**Notes**

* **Definition**: Agents that use explicit knowledge representation and logical inference to make decisions.
* **Knowledge-Based Agents Components**:
  + **Knowledge Base (KB)**: Stores facts and rules.
  + **Inference Engine**: Derives new knowledge.
* **Agent cycle**:
  + Perceive the environment.
  + Update KB.
  + Infer new knowledge.
  + Decide and act.
* **Wumpus World**:
  + A grid world with hazards (Wumpus, pits) and rewards (gold).
  + Percepts: Stench (Wumpus), Breeze (pit), Glitter (gold), Bump (wall), Scream (Wumpus killed).
  + Agent uses percepts + logical reasoning to plan safe moves and achieve goals.

**Possible Viva Questions**

* **What is a logical agent?**  
  → An agent that makes decisions based on explicit facts and inference, not just condition-action rules.
* **What are the components of a knowledge-based agent?**  
  → Knowledge Base, Inference Engine, Perceptual inputs, and Action mechanism.
* **Explain the Wumpus World environment.**  
  → A grid-based world with hazards and percepts; the agent uses logic to navigate safely, find gold, and exit.

**Propositional Logic**

**Notes**

* **Representation**:
  + Propositions: Statements (e.g., P = "It is raining").
  + Connectives: AND (∧), OR (∨), NOT (¬), IMPLIES (→), IFF (↔).
* **Inference**: Deriving new propositions that logically follow from known facts.
* **Reasoning Patterns**:
  + **Modus Ponens**: If P → Q and P, infer Q.
  + **Modus Tollens**: If P → Q and ¬Q, infer ¬P.
  + **And-Elimination**: From P ∧ Q, infer P.
* **Resolution**:
  + Rule of inference to combine clauses with complementary literals.
  + Requires Conjunctive Normal Form (CNF).
* **Forward Chaining**: Starts with facts → applies rules → derives new facts.
* **Backward Chaining**: Starts with a goal → works backwards to find supporting facts.

**Possible Viva Questions**

* **What is propositional logic?**  
  → Logic that uses propositions and logical connectives to represent and reason about statements.
* **Explain Modus Ponens with an example.**  
  → If "If it rains (P) → The ground is wet (Q)" and "It rains (P)", infer "The ground is wet (Q)".
* **What is resolution in propositional logic?**  
  → A rule of inference that resolves two clauses containing complementary literals to deduce new clauses.
* **Differentiate forward and backward chaining.**  
  → Forward chaining derives facts from data; backward chaining proves a goal by working backward from it.

**First-Order Logic (FOL)**

**Notes**

* **Representation**:
  + **Predicates**: Represent properties or relations (e.g., Loves(John, Mary)).
  + **Quantifiers**:
    - Universal (∀): ∀x P(x) — "For all x, P(x)".
    - Existential (∃): ∃x P(x) — "There exists x such that P(x)".
* **Inference**:
  + Uses rules similar to propositional logic but involves **unification**.
* **Reasoning Patterns**:
  + **Universal Instantiation**: ∀x P(x) ⇒ P(c)
  + **Existential Instantiation**: ∃x P(x) ⇒ P(k) where k is new constant.
* **Resolution**: Combines unification + propositional resolution.
* **Forward/Backward Chaining**: Extended to handle predicates and variables.

**Possible Viva Questions**

* **What is First-Order Logic? How is it different from propositional logic?**  
  → FOL uses predicates and quantifiers to express relationships and properties of objects, while propositional logic does not handle internal structure of propositions.
* **What is the role of quantifiers in FOL?**  
  → Universal quantifier (∀): states something is true for all instances; Existential quantifier (∃): states that something is true for at least one instance.
* **What is unification in FOL?**  
  → A process of making two predicates identical by finding appropriate substitutions for variables.
* **Explain Universal Instantiation.**  
  → From ∀x P(x), infer P(a) for a specific constant 'a'.

**Expert System: Design and Implementation**

**Notes**

* **Definition**: A computer program that simulates the decision-making ability of a human expert.
* **Components**:
  + **Knowledge Base**: Contains domain knowledge as facts and rules.
  + **Inference Engine**: Applies rules to known facts to infer conclusions.
  + **User Interface**: Interaction between user and system.
  + **Explanation Facility**: Justifies the reasoning process.
  + **Knowledge Acquisition Module**: Adds/modifies knowledge base.
* **Design Steps**:
  + Identify the problem domain.
  + Acquire knowledge from experts.
  + Represent knowledge using rules, frames, etc.
  + Design system architecture.
  + Implement the expert system.
  + Test and validate.
  + Maintain and update.

**Possible Viva Questions**

* **What is an expert system?**  
  → A system that emulates expert-level decision-making using a knowledge base and inference engine.
* **What are the main components of an expert system?**  
  → Knowledge Base, Inference Engine, User Interface, Explanation Facility, Knowledge Acquisition Module.
* **What are the steps in designing an expert system?**  
  → Identify domain → Acquire knowledge → Represent knowledge → Design architecture → Implement → Test → Maintain.
* **How does an expert system reason?**  
  → The inference engine applies rules from the knowledge base to available facts to deduce new facts or decisions using forward/backward chaining.

**Unit 4**

**1. Introduction to Natural Language Processing (NLP)**

**Definition:**

Natural Language Processing (NLP) is a subfield of Artificial Intelligence (AI) and Computational Linguistics that enables computers to understand, interpret, generate, and interact using human languages. It bridges linguistics, computer science, and machine learning.

**Applications:**

* Machine Translation (Google Translate)
* Chatbots (Siri, Alexa)
* Sentiment Analysis (Social Media Monitoring)
* Information Extraction (NER, Event Extraction)

**Typical NLP Pipeline (Steps of NLP):**

1. **Text Preprocessing** (Tokenization, Cleaning)
2. **Syntactic Processing** (POS Tagging, Parsing)
3. **Semantic Analysis** (Word Sense Disambiguation, Relationship Extraction)
4. **Discourse Integration & Pragmatic Analysis** (Context Modeling)
5. **Applications & Interpretation** (Translation, Summarization)

**2. Lexical Processing**

**Key Concepts:**

1. **Tokenization:**  
   Breaking a text into words, phrases, symbols, or meaningful elements called "tokens".
2. **Stop Word Removal:**  
   Eliminating common words (e.g., “is”, “and”, “the”) that carry little semantic meaning.
3. **Stemming and Lemmatization:**

* **Stemming:** Cuts words to their root forms (e.g., "running" → "run")
* **Lemmatization:** Reduces words to base dictionary forms with context (e.g., "better" → "good")

1. **Handling Noise & Errors:**  
   Correcting typos, expanding abbreviations, and standardizing text.

**3. Syntactic Processing**

**Key Concepts:**

1. **Grammars and Parsers:**  
   Defines valid sentence structures. Parsers break sentences into their grammatical components.
2. **Part-of-Speech (POS) Tagging:**  
   Assigns grammatical tags (noun, verb, adjective) to each word.
3. **Tree Representation (Syntax Trees):**  
   Visualizes hierarchical syntactic structure of sentences.

Example:

(S

(NP Ram)

(VP (V runs)))

**4. Semantic Analysis**

**Key Concepts:**

1. **Lexical Semantics:**  
   Determines meaning of individual words.
2. **Compositional Semantics:**  
   Determines meaning of sentences based on the combination of words.
3. **Word Sense Disambiguation:**  
   Identifies correct meaning of words with multiple senses (e.g., "bank" = riverbank vs. financial institution)
4. **Semantic Role Labeling:**  
   Identifies subject, predicate, and object relationships.

**5. Discourse Integration & Pragmatic Analysis**

**Key Concepts:**

1. **Discourse Integration:**  
   Resolves references across sentences (e.g., pronouns like "he", "it") to maintain coherence.

Example:  
"Ram won the race. He was very happy." → "He" = Ram

1. **Pragmatic Analysis:**  
   Understands language based on context, goals, and social cues.  
   Example:  
   "Can you open the window?" (Not just ability, but a polite request)
2. **Modeling Beliefs, Goals, and Plans:**  
   Understanding user intentions and context beyond literal text — essential for dialogue systems.

**6. Case Study: NLP Techniques in HackerRank**

**Example Use Case:**  
HackerRank leverages NLP for:

* **Resume Parsing:** Extracts skills, experience, education from resumes
* **Coding Problem Recommendation:** Maps problem descriptions and user profiles using semantic matching
* **Chatbots and FAQ Assistants:** Auto-answering developer questions using knowledge bases

**Possible Viva Questions and Answers (Detailed)**

| **Question** | **Answer** |
| --- | --- |
| What is NLP? | NLP is a branch of AI that enables machines to understand, interpret, and generate human language for tasks like translation, summarization, and chatbots. |
| Define Tokenization. | Tokenization splits text into smaller units (words, phrases, symbols) called tokens for easier analysis. |
| Difference between stemming and lemmatization? | Stemming cuts words to root forms without context; lemmatization finds dictionary root forms with context. E.g., "better" → "good" (lemmatized). |
| What is POS tagging? | Assigning grammatical labels (noun, verb, adjective) to each word in a sentence. |
| What is Semantic Role Labeling? | Identifying subject, verb, and object in a sentence to understand "who did what to whom". |
| Explain word sense disambiguation with example. | Determining correct meaning of a word from context. E.g., "He sat on the bank" → "bank" = riverbank (not financial). |
| Define discourse integration. | Linking references (like pronouns) between sentences to maintain context and coherence. |
| What is pragmatic analysis? | Understanding meaning in context, social use of language, and user intentions. |
| Give real-life examples of NLP. | Google Translate (machine translation), Siri (chatbots), Grammarly (grammar check), HackerRank (resume parsing). |
| What is Named Entity Recognition? | Identifying proper nouns (people, places, organizations) in text. |
| Challenges in NLP? | Ambiguity, sarcasm detection, context dependency, multilingual processing, handling noisy text. |
| What is parsing in NLP? | Analyzing sentence structure according to grammatical rules and producing parse trees. |
| Difference between lexical and compositional semantics? | Lexical deals with meaning of individual words; compositional derives sentence meaning from word combination. |
| How does HackerRank use NLP? | For resume parsing, coding problem matching, and chatbot-based support. |
| Why is context important in NLP? | Context helps resolve ambiguities, disambiguate words, and understand user goals (especially in discourse and pragmatics). |

**UNIT 5**

**🧠 1. Text Generation and Chatbots**

**Notes:**

* Text generation is the ability of AI models to generate human-like text.
* Chatbots like **ChatGPT** and **Google Gemini** are powered by large language models (LLMs).
* They use **transformer architectures** to understand input context and generate coherent replies.
* Applications include customer support, virtual assistants, educational tutors, etc.

**Viva Questions:**

* **Q:** What is the core technology behind chatbots like ChatGPT?  
  **A:** Transformer-based deep learning models trained on large text corpora.
* **Q:** How do AI chatbots maintain context in a conversation?  
  **A:** By using attention mechanisms and storing conversation history (context window).

**😊 2. Sentiment Analysis**

**Notes:**

* A Natural Language Processing (NLP) task that identifies emotional tone (positive, negative, neutral) in text.
* Commonly used in social media monitoring, customer feedback, and marketing.
* Techniques: Lexicon-based methods and machine learning models (Naive Bayes, LSTM, BERT).

**Viva Questions:**

* **Q:** What are two main approaches to sentiment analysis?  
  **A:** Lexicon-based (uses predefined word lists) and machine learning-based methods.
* **Q:** Why is sentiment analysis important in business?  
  **A:** It helps understand customer opinions and improve products or services accordingly.

**🤖 3. Introduction to ChatGPT**

**Notes:**

* Developed by **OpenAI**.
* Based on the **GPT (Generative Pretrained Transformer)** family.
* The latest version is **GPT-4 Turbo** with support for multimodal inputs (text, code, images, files).
* Uses RLHF (Reinforcement Learning from Human Feedback) to improve outputs.

**Viva Questions:**

* **Q:** What is GPT in ChatGPT?  
  **A:** Generative Pretrained Transformer – a model that generates human-like text from training on diverse data.
* **Q:** What is RLHF in ChatGPT?  
  **A:** Reinforcement Learning from Human Feedback – a fine-tuning method using human preferences.

**🌐 4. Introduction to Google Bard (Gemini)**

**Notes:**

* Bard is powered by **Gemini 1.5 Pro**, developed by Google DeepMind.
* Based on **Mixture-of-Experts (MoE)** transformer models.
* Supports multimodal inputs: **text, images, audio, video**.
* Strong integration with Google Search and Workspace (Docs, Sheets, Colab).

**Viva Questions:**

* **Q:** What architecture does Gemini use?  
  **A:** Mixture-of-Experts Transformer.
* **Q:** What is a key advantage of Gemini over ChatGPT?  
  **A:** Gemini supports video/audio input and has access to real-time web results via Google Search.

**🔍 5. Explainable AI (XAI)**

**Notes:**

* XAI refers to AI systems whose actions can be understood and trusted by humans.
* Important in domains like healthcare, finance, and law.
* Methods: LIME, SHAP, feature importance graphs.
* Continuous model evaluation ensures models remain accurate and fair over time.

**Viva Questions:**

* **Q:** Why is explainable AI necessary?  
  **A:** To ensure trust, transparency, and accountability in AI systems.
* **Q:** What are SHAP and LIME?  
  **A:** They are tools to interpret model predictions by explaining feature contributions.

**⚖️ 6. AI Ethics and Bias**

**Notes:**

* Ethics in AI refers to moral responsibilities in designing and deploying AI systems.
* AI models can reflect or amplify biases present in training data.
* Ethical concerns: fairness, accountability, transparency, bias mitigation, privacy.

**Viva Questions:**

* **Q:** How does bias occur in AI models?  
  **A:** Due to skewed or unbalanced training data that reflects societal biases.
* **Q:** What is algorithmic fairness?  
  **A:** Ensuring AI outcomes are equitable across different groups or individuals.

**🌍 7. Societal Impact of AI**

**Notes:**

* AI's wide adoption affects jobs, education, social interaction, and privacy.
* Positive: automation, efficiency, personalized services.
* Negative: job displacement, surveillance, misinformation, loss of human contact.

**Viva Questions:**

* **Q:** How can AI affect employment?  
  **A:** Through automation of repetitive tasks, potentially reducing the need for certain job roles.
* **Q:** What is the privacy concern with AI?  
  **A:** AI systems may collect and analyze personal data, risking misuse or breaches.

ASSIGNMENTS:

Certainly! Here's an **expanded and detailed description** for each experiment topic in your AI Lab assignment list, giving more theoretical depth and real-world relevance.

**Experiment 1: Non-AI Techniques (Tic Tac Toe, N-Queens, Magic Square)**

**Expanded Description:**

These are classical logic-based problems that are often the first step in algorithmic thinking. Solving them without AI techniques emphasizes control structures like loops, recursion, and brute force.

* **Tic Tac Toe**: This is a finite game with a small search space. It can be solved completely using exhaustive checking or simple rules.
* **N-Queens**: A combinatorial problem demonstrating backtracking. It serves as a gateway to understanding how constraints can drastically prune a search space.
* **Magic Square**: This puzzle requires generating numbers such that their arrangement in a square matrix maintains equal sums across rows, columns, and diagonals — a good example of using brute-force plus validation.

**Real-world relevance: Foundation for understanding search spaces, constraint logic, and backtracking.**

**Experiment 2: Water Jug Problem with DFS and BFS**

**Expanded Description:**

The Water Jug problem models real-life constraint-based state problems. It demonstrates the **unstructured, uninformed search space**, where the algorithm has no insight into how far the current state is from the goal.

* **DFS (Depth First Search)**: Goes deep into state transitions, risking long or infinite paths.
* **BFS (Breadth First Search)**: Finds the shortest path in an unweighted graph but uses more memory.

**Real-world relevance: Forms the basis for solving puzzles, robot path planning, and even parsing algorithms in compilers.**

**Experiment 3: Hill Climbing (8 Puzzle Problem)**

**Expanded Description:**

Hill Climbing is a **local search algorithm** that uses a heuristic to move toward better solutions. It does not consider the whole path, only the immediate improvement, making it fast but shortsighted.

* **8 Puzzle Problem**: Involves sliding tiles into the correct configuration. It has a large search space, making it ideal for heuristic-based approaches like Hill Climbing.

**Challenges: Local maxima, plateaus, and ridges may trap the algorithm without reaching the goal.**

**Real-world relevance: Used in operations research, machine learning hyperparameter tuning, and robotics.**

**Experiment 4: Best First Search**

**Expanded Description:**

This is a **heuristic-based greedy search** strategy. It always picks the most promising node according to a heuristic (like distance to goal), not the cost to reach that node.

* Common in **maze solving**, **shortest-path problems**, and basic **robot navigation**.
* Unlike A\*, it doesn't track total cost, so it may miss optimal solutions.

**Real-world relevance: GPS route planning, AI in games, robotic path planning (but less reliable than A\* in terms of optimality).**

***Experiment 5: A Algorithm*\***

**Expanded Description:**

A\* is one of the most important AI search algorithms. It guarantees finding the shortest path if the heuristic used is **admissible** and **consistent**.

* **g(n)**: Cost from the start node to current node.
* **h(n)**: Heuristic estimate of cost from current node to goal.
* **f(n) = g(n) + h(n)**: Total estimated cost.

**Benefits:**

* Finds optimal path.
* Balances between DFS (deep search) and Best First Search (greedy).

**Real-world relevance: Widely used in GPS systems, games (like Pac-Man, RTS games), robot motion planning, and network routing.**

**Experiment 6: Constraint Satisfaction Problems (CSP)**

**Expanded Description:**

CSPs involve **solving problems by assigning values to variables** within the limits of constraints. It's a backbone of **declarative problem solving** in AI.

* **Cryptarithmetic**: Letters are mapped to digits to satisfy arithmetic equations.
* **Crossword Puzzle**: Words must fit in a grid without violating length and meaning constraints.
* **Map Coloring**: Adjacent regions must not have the same color (graph coloring problem).

**Key Concepts:**

* Domains, variables, constraints
* Backtracking, forward checking, constraint propagation

**Real-world relevance: Timetabling, scheduling, sudoku solvers, circuit design, and AI planning systems.**

**Experiment 7: Minimax Algorithm (Tic Tac Toe)**

**Expanded Description:**

Minimax is a **decision-making algorithm for two-player games**. It assumes both players play optimally. The algorithm simulates all possible moves and selects the best one for the maximizing player, assuming the opponent is minimizing.

* Best suited for **zero-sum games** like Chess, Tic Tac Toe, and Connect Four.

**Extensions:**

* **Alpha-Beta pruning**: Optimizes Minimax by cutting off branches that cannot influence the final decision.

**Real-world relevance: Used in AI for competitive games and decision-making under adversarial conditions.**

**Experiment 8: NLP Problems (POS Tagging, Similarity Score, Spell Checker)**

**Expanded Description:**

These are core problems in **Natural Language Processing**, the field of AI that enables machines to understand and process human languages.

* **POS Tagging**: Assign grammatical categories (noun, verb, etc.) to each word. Done via rule-based systems or statistical models (like HMM or CRF).
* **Similarity Score**: Uses vector representations (TF-IDF, Word2Vec, BERT embeddings) to measure semantic or lexical closeness.
* **Spell Checker**: Detects misspelled words and suggests corrections based on dictionaries or phonetic algorithms (Soundex, edit distance).

**Real-world relevance: Search engines, virtual assistants (Siri, Alexa), grammar checkers, chatbots, and machine translation tools.**