Define and explain Probabilistic Retrieval.

Probabilistic retrieval is an information retrieval approach that uses probabilistic models to rank and retrieve documents based on their relevance to a user's query.

Key Concepts and Explanation

1. **Relevance as Probability**
   - Documents are retrieved and ranked based on the probability of relevance to a query.
   - The goal is to maximize the probability $P(R|Q, D)$, where:
     - R: Relevance of the document.
     - Q: User query.
     - D: Document.
2. **Binary Independence Model (BIM)**
   - Assumes:
     - A document is either relevant or non-relevant.
     - Terms in documents are independent.
   - Uses Bayes' Theorem:

$$P(R|Q, D) = \frac{P(Q, D|R) \cdot P(R)}{P(Q, D)}$$

   - Since $P(Q, D)$ is constant for ranking, only $P(Q, D|R) \cdot P(R)$ needs to be computed.
3. **Term Weighting**
   - Computes weights for terms in documents [redacted]
   - For a term t, the weight depends on:
     - $P(t|R)$: Probability that t occurs in relevant documents.
     - $P(t|\neg R)$: Probability that t occurs in non-relevant documents.
   - Term weights are calculated using:

$$w_t = \log \frac{P(t|R)}{P(t|\neg R)}$$

4. **Ranking of Documents**
   - Documents are ranked based on the sum of weights of query terms present in the document:

$$\text{Score}(D) = \sum_{t \in Q} w_t$$

5. **Feedback Mechanism**
   - **Relevance Feedback**: User feedback is used to refine $P(t|R)$ and $P(t|\neg R)$, improving retrieval effectiveness.

Example

Query: "Artificial Intelligence"

- Document A contains "Artificial" and "Intelligence" multiple times.
- Document B contains only "Artificial."
- Using probabilistic weights:
  - If $P(\text{"Artificial"}|R)$ and $P(\text{"Intelligence"}|R)$ are high, Document A will score higher than Document B.
- This ranking reflects the probability of relevance.

Advantages

1. Theoretically sound and grounded in probability theory.
2. Supports relevance feedback for iterative improvement.
3. Effective for ranking and filtering.

Limitations

1. Independence assumption between terms is often unrealistic.
2. Requires labeled data (relevant/non-relevant) for precise probability estimates.
3. Computationally intensive for large-scale datasets.

What is the binary independent retrieval model?

Binary Independence Model (BIM)

The **Binary Independence Model (BIM)** is a simplified probabilistic model used in information retrieval to estimate the relevance of documents to a user query. It assumes that the terms in a document are independent of one another and represent relevance as a binary variable: a document is either relevant or not.

Key Assumptions of BIM

1. **Binary Relevance:**
   - A document D is either relevant ($R = 1$) or not relevant ($R = 0$) to the query Q.
2. **Term Independence:**
   - The presence or absence of a term in a document is independent of the presence or absence of other terms.
3. **Binary Term Representation:**
   - Each term in a document is represented as a binary variable:
     - $t_i = 1$ if term t_i is present in the document.
     - $t_i = 0$ if term t_i is absent.

Probability Estimation

The model ranks documents based on the probability $P(R|D, Q)$, where:

- R: Relevance of a document.
- D: A document.
- Q: A query.

Using **Bayes' Theorem**:

$$P(R|D, Q) = \frac{P(D|R, Q) \cdot P(R|Q)}{P(D|Q)}$$

- Since $P(D|Q)$ is constant for all documents, it can be ignored for ranking purposes.
- Therefore, the model focuses on $P(D|R, Q) \cdot P(R|Q)$.

Document Scoring

The **scoring function** for a document D simplifies to:

$$\text{Score}(D) = \sum_{t \in Q} \log \frac{P(t|R)}{P(t|\neg R)}$$

Where:

- $P(t|R)$: Probability of the term t appearing in relevant documents.
- $P(t|\neg R)$: Probability of the term t appearing in non-relevant documents.

Term Contribution:

- A term contributes positively if $P(t|R) > P(t|\neg R)$ and negatively otherwise.

Example

Query: "Machine Learning"

- **Document A** contains both "Machine" and "Learning."
- **Document B** contains only "Machine."

Using the BIM scoring function:

- If $P(\text{"Machine"}|R)$ and $P(\text{"Learning"}|R)$ are high, Document A gets a higher score than Document B.
- Document relevance is computed based on the terms' probabilities and their binary occurrence in the documents.

Strengths

1. Theoretical foundation grounded in probability theory.
2. Simplicity and interpretability.
3. Effective for small-scale or structured datasets.

Limitations

1. The **independence assumption** is unrealistic for correlated terms.
2. Binary term representation loses information about term frequency.
3. Requires labeled data to estimate probabilities accurately.

What is the probability ranking principle for interactive information retrieval?

Probability Ranking Principle (PRP) in Interactive Information Retrieval

The **Probability Ranking Principle (PRP)** is a fundamental concept in information retrieval (IR). It states that documents should be ranked in order of their probability of relevance to a user's query. In **interactive information retrieval (IIR)**, the principle is adapted to account for user interaction and iterative refinement of search results.

At each stage of interaction, the system should present documents in order of their estimated probability of relevance, based on both the query and the history of the user's interactions (e.g., feedback, clicks, or queries).

Definition of PRP

The PRP asserts that:

- For a given query, documents should be ranked in decreasing order of the probability of being relevant to the query.
- This ranking maximizes the overall effectiveness of the retrieval system, assuming independent evaluation of relevance for individual documents.

Mathematically, a document D is ranked based on:

$$P(R|D,Q)$$

Where:

- $P(R|D,Q)$: Probability of document D being relevant to query Q.

Steps in Applying PRP for Interactive IR

1. **Initial Ranking:**
   - Rank documents based on initial probabilities $P(R|D,Q)$ derived from the query.
2. **User Interaction:**
   - Collect user actions, such as clicks, ratings, or explicit relevance feedback.
3. **Model Update:**
   - Adjust the probability estimates $P(R|D,Q)$ using the feedback to better reflect the user's intent.
4. **Re-ranking:**
   - Generate a new ranking of documents based on the updated probabilities.
5. **Iterative Refinement:**
   - Repeat the process until the user finds the desired information or terminates the interaction.

Mathematical Representation

In IIR, the system ranks documents based on an updated probability estimate:

$$P(R|D,Q,H)$$

Where:

- R: Relevance of the document.
- D: Document being considered.
- Q: Original query.
- H: History of interactions (feedback, click-through rates, etc.).

The ranking score $S(D)$ is proportional to the updated relevance probability:

$$S(D) \propto P(R|D,Q,H)$$

Define the following with respect to Information Retrieval:

i) Vector Space Model
ii) Term Frequency
iii) Inverse Document Frequency

i) Vector Space Model (VSM)

The **Vector Space Model** is an algebraic model used in Information Retrieval where documents and queries are represented as vectors in a multidimensional space. Each dimension corresponds to a unique term in the corpus.

- **Key Idea:** The similarity between a document and a query is computed using measures like **cosine similarity**, based on the angle or distance between their respective vectors.
- **Example:** A document that shares more terms with the query will have a higher similarity score and be ranked higher.

---

ii) Term Frequency (TF)

Term Frequency (TF) is the measure of how often a specific term appears in a document, normalized to account for document length.

$$\text{TF}_{t,d} = \frac{\text{Number of occurrences of term } t \text{ in document } d}{\text{Total number of terms in document } d}$$

- **Key Idea:** Frequent terms are considered more important within the context of a single document.
- **Example:** If "data" appears 10 times in a document of 100 words, $\text{TF} = 0.1$.

iii) Inverse Document Frequency (IDF)

Inverse Document Frequency (IDF) measures how unique or rare a term is across the entire collection of documents. It is defined as:

$$\text{IDF}_t = \log \frac{N}{n_t}$$

Where:

- N: Total number of documents.
- n_t: Number of documents containing the term t.
- **Key Idea:** Rare terms (with high IDF) are more valuable for distinguishing documents.
- **Example:** If "algorithm" appears in 2 out of 100 documents, $\text{IDF} = \log(100/2)$.

State and explain different types of Bayesian networks?

Bayesian Networks: An Overview

A **Bayesian Network (BN)**, also called a **Bayes Net**, is a probabilistic graphical model that represents a set of random variables and their conditional dependencies through a **Directed Acyclic Graph (DAG)**. It combines principles of graph theory and probability theory to model uncertainty and causal relationships in a system.

Key Components of Bayesian Networks

1. **Nodes:**
   Represent random variables (e.g., symptoms, diseases, weather conditions).
   - Variables can be discrete (e.g., Yes/No) or continuous (e.g., temperature).
2. **Edges:**
   Directed edges represent conditional dependencies between variables.
   - If there is an edge from A to B, A is a parent of B, and B is conditionally dependent on A.
3. **Conditional Probability Distributions (CPDs):**
   Each node is associated with a probability distribution that quantifies the relationship between the node and its parents.
   - If a node has no parents, its distribution is unconditional (prior probability).
   - For example, $P(B|A)$ describes the probability of B given A.

1. Static Bayesian Network

- **Definition**: Represents a fixed set of variables and their dependencies, assuming no temporal or sequential relationship.
- **Key Features**:
  - Variables do not change over time.
  - Often used in static scenarios like diagnostic systems or risk assessments.
- **Example**: A medical diagnosis network where nodes represent symptoms, diseases, and test results.

2. Dynamic Bayesian Network (DBN)

- **Definition**: Extends static BNs to model variables that evolve over time.
- **Key Features**:
  - Captures temporal dependencies between variables across time slices.
  - Each time slice is represented as a static BN, with additional edges to represent dependencies between successive time slices.
- **Example**: Weather prediction, where today's weather depends on yesterday's weather conditions.

3. Naive Bayesian Network

- **Definition**: A simplified Bayesian network where all predictor variables are assumed to be conditionally independent given the target variable.
- **Key Features**:
  - Easy to construct and computationally efficient.
  - Assumes independence among features, which may not hold in real-world scenarios.
- **Example**: Spam email classification, where the presence of specific words (features) independently affects the classification outcome (spam or not).

4. Markov Blanket Bayesian Network

- **Definition**: Focuses on identifying the Markov blanket of a node, which includes its parent, child, and the children's parents (co-parents).
- **Key Features**:
  - Provides the minimal set of nodes that completely shield a node from the rest of the network.
  - Useful for feature selection and causal inference.
- **Example**: In a medical diagnosis network, the Markov blanket of a disease node includes symptoms (children), causes (parents), and other factors influencing symptoms (co-parents).

hierarchical baysian network:: output is given as input to another layer

List and explain the challenges in Natural language processing

Natural Language Processing (NLP) is a subfield of AI that focuses on enabling machines to understand, interpret, and generate human language. Despite significant advancements, there are still several challenges in NLP:

Challenges in Natural Language Processing (NLP)

1. **Ambiguity**:
   Words and sentences can have multiple meanings depending on the context.
   - **Example:** "Bank" can mean a financial institution or the side of a river.
2. **Context Understanding**:
   NLP systems struggle to grasp the broader context in which words are used.
   - **Example:** The meaning of "apple" can vary in different contexts (fruit or tech company).
3. **Language Variations**:
   Different languages and dialects have unique structures, making it difficult to build universal models.
   - **Example:** Word order, grammar, and vocabulary vary across languages.
4. **Sarcasm and Irony**:
   Detecting sarcasm and irony in text can be challenging, as the literal meaning differs from the intended meaning.
   - **Example:** "Oh great, another Monday!" might imply frustration.
5. **Named Entity Recognition (NER)**:
   Identifying proper names (persons, places, organizations) in text can be difficult due to variations in spelling, abbreviation, or new terms.
   - **Example:** "Apple" can refer to a fruit or a tech company.
6. **Data Scarcity**:
   High-quality labeled data is often hard to obtain, especially for low-resource languages.
   - **Example:** NLP models for less spoken languages often lack sufficient training data.

Explain with suitable example different levels of NLP. lsscp

Natural Language Processing (NLP) is a complex field that involves multiple levels of language analysis, each with its own specific tasks and techniques. These levels can be broadly categorized into **lexical**, **syntactic**, **semantic**, **discourse**, and **pragmatic** levels. Below, we explain each level with examples.

1. Lexical Level (Word Level)

- **Definition**: The lexical level deals with the analysis of individual words, their meanings, and their possible forms. It involves tasks like tokenization, stemming, and lemmatization.
- **Key Tasks**:
  - **Tokenization**: Splitting text into words or smaller meaningful units (tokens).
  - **Stemming**: Reducing words to their root form (e.g., "running" → "run").
  - **Lemmatization**: Converting words to their base or dictionary form, considering the context (e.g., "better" → "good").
- **Example**:
  Given the sentence:
  "She is running fast."
  Tokenization would split it into ["She", "is", "running", "fast"].
  Stemming would convert "running" to "run".
  Lemmatization would also convert "running" to "run" (but correctly handling context).

2. Syntactic Level (Sentence Level)

- **Definition**: The syntactic level focuses on the structure of sentences, identifying how words are arranged to form grammatical sentences. It deals with syntax trees, sentence parsing, and part-of-speech tagging.
- **Key Tasks**:
  - **Part-of-Speech (POS) Tagging**: Assigning each word in a sentence a part-of-speech tag (e.g., noun, verb, adjective).
  - **Parsing**: Identifying the grammatical structure of a sentence, often represented as a **syntax tree**.
- **Example**:
  Sentence: "The cat sat on the mat."
  - **POS tagging**:
    - "The" (Determiner)
    - "cat" (Noun)
    - "sat" (Verb)
    - "on" (Preposition)
    - "the" (Determiner)
    - "mat" (Noun)
  - **Syntactic Parsing**: A tree structure might show:

3. Semantic Level (Meaning Level)

- **Definition**: The semantic level is concerned with understanding the meaning of words, sentences, and larger chunks of text. It involves tasks like word sense disambiguation, named entity recognition, and semantic role labeling.
- **Key Tasks**:
  - **Word Sense Disambiguation (WSD)**: Determining the correct meaning of a word based on context.
  - **Named Entity Recognition (NER)**: Identifying entities such as names of people, places, organizations, dates, etc.
  - **Semantic Role Labeling**: Identifying the roles of words in a sentence (e.g., agent, action, object).
- **Example**:
  Consider the sentence:
  "He went to the bank to fish."
  - **Word Sense Disambiguation**: The word "bank" can mean a financial institution or the side of a river. The context suggests it means the side of a river.
  - **NER**: "He" might refer to a person, and "bank" could be recognized as a location, depending on the context.

4. Discourse Level (Context Level)

- **Definition**: The discourse level focuses on understanding how sentences relate to each other in a larger text, ensuring coherence and context. This includes **coreference resolution**, **anaphora resolution**, and **discourse parsing**.
- **Key Tasks**:
  - **Coreference Resolution**: Determining which words or phrases in a text refer to the same entity.
  - **Anaphora Resolution**: Resolving pronouns or other referring expressions to their antecedents.
  - **Discourse Parsing**: Understanding how sentences are related in terms of discourse structure (e.g., cause-effect, contrast).
- **Example**:
  In the text:
  "John went to the store. He bought some milk."
  Coreference Resolution would identify that "He" refers to "John".
  Discourse Parsing could identify that the two sentences are related causally.

5. Pragmatic Level (Use Level)

- **Definition**: The pragmatic level focuses on understanding the intended meaning in a specific context, beyond literal interpretations. It deals with speech acts, conversational implicature, and the interpretation of indirect or figurative meanings (e.g., sarcasm, irony).
- **Key Tasks**:
  - **Speech Act Recognition**: Understanding the intent behind a sentence (e.g., request, question, statement).
  - **Conversational Implicature**: Understanding implied meaning in conversations, such as "Can you pass the salt?" being interpreted as a request, not just a question.
  - **Irony/Sarcasm Detection**: Recognizing when something is said with an opposite or ironic intent.
- **Example**:
  If someone says, "Nice job!" after a person makes a mistake, **pragmatics** would help interpret the statement as sarcasm rather than a compliment.

SIMPLE WAY TO REMEMBER

Summary of NLP Levels

Level	Focus	Key Tasks & Techniques	Example
Lexical Level	Word-level analysis	Tokenization, Lemmatization, Stemming	Breaking "running" into "run" (stemming)
Syntactic Level	Sentence structure and grammar	POS tagging, Parsing, Syntax Trees	Parsing "The cat sat on the mat"
Semantic Level	Meaning of words and sentences	Word Sense Disambiguation, Named Entity Recognition	Resolving "bank" as a river bank
Discourse Level	Understanding relationships between sentences	Coreference, Anaphora Resolution	Resolving "He" to "John"
Pragmatic Level	Context and intended meaning	Speech Acts, Implicature, Irony Detection	Interpreting "Nice job!" as sarcasm

List the problems associated with n-gram. Explain how these problems are handled

N-gram Language Model

An **N-gram language model** is a statistical approach used in natural language processing (NLP) to predict the likelihood of a sequence of words. It models the probability of a word based on the previous $N-1$ words in the sequence.

1. Sparsity Problem

- **Explanation:** As "n" increases, the number of possible N-grams grows exponentially, leading to many N-grams that do not appear in the training data. This results in a **sparse dataset**, making it difficult to estimate probabilities accurately for unseen N-grams.
- **Solution:**
  - **Smoothing Techniques:** Methods like **Laplace Smoothing** or **Good-Turing Smoothing** are used to assign non-zero probabilities to unseen N-grams by redistributing probability mass from observed N-grams.
  - **Lowering N (N-gram Size):** Using smaller N-grams (e.g., bigrams or unigrams) reduces sparsity because there are fewer possible combinations.

2. High Dimensionality

- **Explanation:** The larger the value of "n", the more possible N-grams there are. This results in **high-dimensional feature spaces**, which can lead to computational inefficiency and require large amounts of memory and storage.
- **Solution:**
  - **Dimensionality Reduction:** Techniques like **Principal Component Analysis (PCA)** or **Latent Semantic Analysis (LSA)** can be applied to reduce the dimensionality of the N-gram model
  - **Using Subsampling:** Reducing the frequency of less common N-grams through methods like **frequency thresholding** can help reduce dimensionality.

3. Lack of Long-term Dependencies

- **Explanation:** N-grams are based on fixed-length sequences of words, which can capture only short-term dependencies between adjacent words. They struggle to model **long-range dependencies** (e.g., relationships between words in distant parts of the text).
- **Solution:**
  - **Higher-order N-grams:** Increasing the value of "n" can help capture longer dependencies, but this comes at the cost of sparsity and high dimensionality.
  - **Recurrent Neural Networks (RNNs):** Modern approaches like **LSTMs** (Long Short-Term Memory networks) and **Transformers** can capture long-term dependencies better than N-grams by maintaining a memory of past context.

4. Data-Dependence

- **Explanation**: N-gram models depend heavily on the quality and quantity of the training data. If the training corpus is small or not diverse enough, the model will fail to generalize well and produce poor results on unseen data.
- **Solution**:
  - **Use Larger Datasets**: Training the model on a larger and more diverse dataset helps to capture more context and variability.
  - **Pre-trained Language Models**: Using pre-trained models such as **BERT** or **GPT** that are trained on large, diverse datasets can help overcome this issue.

5. Computational Complexity

- **Explanation**: The larger the value of "n", the more computational resources are required to store, process, and calculate probabilities for N-grams. This can result in **slower performance** in real-time applications.
- **Solution**:
  - **Efficient Data Structures**: Using efficient data structures like **hash tables** or **trie** can speed up the computation and storage of N-grams.
  - **Pruning**: Limiting the N-gram model to only the most frequent N-grams reduces the computational burden.

6. Ambiguity and Contextual Limitations

- **Explanation**: N-gram models cannot handle words with multiple meanings (ambiguity) or situations where the meaning of a word depends on a broader context. They lack a deep understanding of the **semantics** and **pragmatics** of language.
- **Solution**:
  - **Contextual Embeddings**: Using models like **Word2Vec**, **GloVe**, or **BERT**, which learn dense word representations that capture semantic meaning and context, can help overcome ambiguity.
  - **Neural Language Models**: Transitioning to neural models like **Transformers**, which consider the entire context of a sentence rather than just local n-grams, can improve performance on tasks requiring nuanced understanding.

Explain language models?

A **Language Model (LM)** is a statistical model used in Natural Language Processing (NLP) to predict the likelihood of a sequence of words in a sentence. It assigns probabilities to sequences of words based on their occurrence in a given corpus, capturing the structure and patterns of natural language. Language models are foundational to many NLP tasks like speech recognition, machine translation, text generation, and more.

Definition

A language model assigns a probability to a sequence of words, $P(w_1, w_2, ..., w_T)$, or predicts the next word given a sequence of preceding words, $P(w_T|w_1, w_2, ..., w_{T-1})$.

Types of Language Models

Language models can be categorized based on their underlying techniques and approach:

1. Statistical Language Models

- Use statistical methods to estimate probabilities of word sequences.
- Examples:
  - **Unigram Model:** Each word's probability is independent of other words.
  - **Bigram and N-gram Models:** Use $N - 1$ preceding words to predict the next word.

2. Neural Language Models

- Use neural networks to model complex relationships between words.
- Examples:
  - **Feedforward Neural Networks:** Basic neural networks to learn word embeddings and predict word probabilities.
  - **Recurrent Neural Networks (RNNs):** Handle sequential data to capture context.
  - **Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs):** Address long-term dependencies.
- Advantages:
  - Capture richer contextual information.
  - Handle larger vocabularies using word embeddings.

3. Transformer-Based Language Models

- Use self-attention mechanisms for parallel processing and long-range dependency capture.
- Examples:
  - **BERT (Bidirectional Encoder Representations from Transformers):** Focuses on bidirectional context.
  - **GPT (Generative Pre-trained Transformer):** Specialized in text generation.
- Features:
  - Highly effective for tasks like machine translation, summarization, and text classification.

Applications of Language Models

1. **Text Prediction and Generation:**
   - Autocomplete systems, virtual assistants, and content creation.
2. **Machine Translation:**
   - Converting text from one language to another.
3. **Speech Recognition:**
   - Converting spoken language into text.
4. **Sentiment Analysis:**
   - Understanding emotions or opinions expressed in text.
5. **Information Retrieval:**
   - Improving search engine results by understanding query context.

What is language modelling approach to information retrieval?

Language Modeling Approach to Information Retrieval

The **Language Modeling (LM) approach** to information retrieval treats both documents and queries as probabilistic language models. It assumes that each document D is generated from a language model $P(D)$, and a query Q is generated by a query model $P(Q|D)$. The goal is to rank documents based on the likelihood that they generate the given query.

Key Points:

1. **Document Representation**: Each document is represented by a language model $P(D)$, which estimates the probability of observing a sequence of words (the document) based on word frequencies within the document.
2. **Query Generation**: Given a query, the language model estimates how likely it is that a document D could have generated that query. This is done using the probability $P(Q|D)$, which is calculated based on the words in the query and their occurrences in the document.
3. **Ranking**: Documents are ranked according to the likelihood $P(Q|D)$, i.e., the probability of generating the query given the document. The higher the likelihood, the more relevant the document is to the query.
4. **Smoothing**: Since not all query terms may appear in a document, smoothing techniques like **Laplace smoothing** are used to assign a non-zero probability to unseen terms.
5. **Retrieval Function**: The retrieval process involves ranking documents based on the probability of observing the query given each document, typically using the **maximum likelihood estimate (MLE)** or **Bayes' theorem**.

Okapi BM25 Model in Information Retrieval

Okapi BM25 is a probabilistic ranking function widely used in Information Retrieval (IR) to rank documents based on their relevance to a query. It is an evolution of the probabilistic model, incorporating term frequency and document length normalization to address practical challenges.

Core Idea

BM25 scores documents by considering:

1. **Term Frequency**: The frequency of query terms in the document.
2. **Inverse Document Frequency (IDF)**: The rarity of a term across all documents.
3. **Document Length Normalization**: Longer documents may have higher term frequencies; normalization mitigates this bias.

BM25 is not a single model but a family of scoring functions, with specific versions depending on parameter tuning.