How Thesaurus-Based Expansion Works:

1. **Identify Key Terms:**
   - Start with the original query terms.
   - Example Query: `"climate change"`
2. **Find Synonyms and Related Terms:**
   - Use a thesaurus or lexical resource to identify synonyms and related terms for each key term in the query.
   - For "climate change," relevant synonyms might include "global warming," "environmental change," "climate crisis," and "climate disruption."
3. **Expand the Query:**
   - Incorporate the identified synonyms and related terms into the original query.
   - Expanded Query: `"climate change" OR "global warming" OR "environmental change" OR "climate crisis" OR "climate disruption"`

Example Using a Thesaurus-Based Approach:

1. **Original Query:**
   - `"economic growth"`

2. **Find Synonyms and Related Terms:**
    - **Thesaurus Lookup:**
        - "Economic growth" might have synonyms like "economic development," "economic expansion," "growth rate," and related terms such as "GDP growth," "prosperity," and "economic progress."
3. **Expanded Query:**
    - **Expanded Query:** `"economic growth" OR "economic development" OR "economic expansion" OR "growth rate" OR "GDP growth" OR "prosperity" OR "economic progress"`

Benefits of Thesaurus-Based Expansion:

- **Increased Recall:** Captures a broader range of documents by including synonyms and related terms that might not be in the original query.
- **Enhanced Coverage:** Addresses variations in terminology that different authors or sources might use.

Challenges:

- **Relevance Control:** Including too many synonyms can sometimes retrieve irrelevant documents if the terms are not contextually appropriate.
- **Resource Limitation:** The effectiveness depends on the quality and comprehensiveness of the thesaurus or lexical resource used.

Continue generating

If you're using a search engine or an information retrieval system, you might manually or programmatically expand your query using a thesaurus or a similar resource. In academic databases or advanced search tools, thesaurus-based query expansion is often automated to improve search results and retrieval efficiency.
The Permuterm Index is an indexing technique used in information retrieval systems, particularly for supporting wildcard queries. Wildcard queries are those where the user may not know the exact term they want to search for and may use a wildcard symbol (e.g., * or ?) to replace unknown characters.

How Permuterm Index Works:
Permutations of Terms: For each term in the vocabulary, all possible rotations (or permutations) of the term are generated and stored in the index.A special symbol, often $, is appended to the end of each term to handle the rotation properly. Example:
Consider the term "apple".
The term with the special symbol would be "apple$".
All possible rotations of "apple$" would be:
"apple$"
"pple$a"
"ple$ap"
"le$app"

"e$appl"
"$apple"
Indexing: Each of these rotations is stored in the index along with a pointer to the original term "apple".
The index might look something like this:
"apple$" → apple
"pple$a" → apple
"ple$ap" → apple
"le$app" → apple
"e$appl" → apple
"$apple" → apple
Querying with Wildcards: When a user submits a query with a wildcard, the system rotates the query and matches it against the permuterm index.
Example Query: *ple
Rotate the query by appending the wildcard to the end: "ple*$"
The system searches for all terms in the index that start with "ple$".
In our index, "ple$ap" would match, leading to the original term "apple" being retrieved.
Handling Different Wildcard Queries:
Prefix Wildcard: *ple → search for ple*$
Suffix Wildcard: app* → search for $app*
Infix Wildcard: a*e → rotate and search for e$a*
Advantages of Permuterm Index:
Efficient Wildcard Querying: Allows for fast and efficient searching for terms with leading, trailing, or embedded wildcards.
Comprehensive Matching: All permutations ensure that every possible wildcard pattern can be matched.
Disadvantages:
Index Size: The permuterm index can be quite large, as it stores multiple rotations for each term, leading to increased storage requirements.
Complexity: Generating and maintaining the permuterm index can be computationally intensive, especially for large vocabularies.
Example Use Case:
Imagine a user is searching for documents containing any word that starts with "bio" (e.g., "biology", "biochemistry"). With a permuterm index:
The user queries with "bio*"
This is rotated to "*bio$"
The system searches for all terms in the permuterm index that start with "bio$", retrieving terms like "biology", "biochemistry", etc.
The Permuterm Index is a powerful technique in information retrieval systems, particularly when supporting complex query patterns that involve wildcards.

Introduction to Sentiment Analysis
Sentiment Analysis, also known as opinion mining, is a natural language processing (NLP) technique used to determine the sentiment expressed in a piece of text. It involves analyzing text data to classify it into categories such as positive, negative, or neutral. Sentiment analysis is widely applied in various fields, including social media monitoring, customer feedback analysis, market research, and more.
Key Concepts in Sentiment Analysis
Sentiment Classification:

The primary goal of sentiment analysis is to classify the sentiment expressed in a text. This is typically done at different levels:

Document-Level Sentiment Analysis: Determines the overall sentiment of an entire document.

Sentence-Level Sentiment Analysis: Determines the sentiment of individual sentences within a document.

Aspect-Level Sentiment Analysis: Focuses on specific aspects or entities mentioned in the text and determines the sentiment related to them.

Polarity:

Sentiment is often expressed in terms of polarity:

Positive Sentiment: Indicates a favorable or optimistic opinion.

Negative Sentiment: Indicates an unfavorable or critical opinion.

Neutral Sentiment: Indicates an objective or impartial opinion.

Emotion Detection:

Beyond simple polarity, some sentiment analysis systems also detect specific emotions, such as joy, anger, sadness, or surprise.

Opinion Holder and Target Identification:

Opinion Holder: The person or entity expressing the sentiment.

Target: The object or entity about which the sentiment is expressed.

Applications of Sentiment Analysis

Social Media Monitoring:

Example: Brands use sentiment analysis to monitor mentions of their products or services on social media platforms like Twitter, Facebook, or Instagram. This helps them gauge public opinion, track brand reputation, and respond to customer feedback in real-time.

Problem Solved: Provides insights into how customers feel about a brand, allowing for timely interventions and reputation management.

Customer Feedback Analysis:

Example: Companies analyze customer reviews, survey responses, and support tickets to understand customer satisfaction and identify areas for improvement.

Problem Solved: Helps businesses improve their products and services by addressing customer concerns and capitalizing on positive feedback.

Market Research:

Example: Businesses use sentiment analysis to understand public opinion on market trends, competitor products, or new product launches.

Problem Solved: Provides valuable insights into market dynamics, helping companies make informed strategic decisions.

Political Sentiment Analysis:

Example: Sentiment analysis is used to analyze public opinion during elections by analyzing social media posts, news articles, and comments on political events.

Problem Solved: Helps political analysts, parties, and candidates understand voter sentiment and adjust their campaigns accordingly.

Financial Market Analysis:

Example: Investors use sentiment analysis to predict stock market movements by analyzing news articles, tweets, and other financial data sources.

Problem Solved: Aids in making investment decisions by identifying trends in market sentiment that could affect stock prices.

Product Development:

Example: Sentiment analysis is used during product development to understand customer needs and preferences by analyzing feedback from beta testers, forums, and user reviews.

Problem Solved: Ensures that products are developed in line with customer expectations, leading to higher customer satisfaction and successful product launches.

Challenges in Sentiment Analysis

Sarcasm and Irony:

Sentiment analysis systems often struggle with detecting sarcasm and irony, which can lead to incorrect sentiment classification.

Context Understanding:

The meaning of a sentence can depend heavily on the context, making it challenging for sentiment analysis systems to accurately interpret the sentiment.

Ambiguity:

Words or phrases can have different meanings in different contexts, leading to ambiguity in sentiment detection.

Multilingual Sentiment Analysis:

Analyzing sentiment in different languages or across multilingual datasets adds complexity due to language-specific nuances and cultural differences.

Conclusion

Sentiment analysis is a powerful tool for extracting insights from text data. It enables businesses, researchers, and analysts to understand public opinion, customer satisfaction, and market trends, driving better decision-making. Despite the challenges, advances in NLP and machine learning continue to enhance the accuracy and applicability of sentiment analysis across various domains.

1. Term Frequency-Inverse Document Frequency (TF-IDF) Calculation

Problem:

Given the following set of documents, compute the TF-IDF value for the term "data" in Document 1.

- **Document 1:** "data mining is fun"
- **Document 2:** "data science is the future"
- **Document 3:** "mining data is important for data science"

Steps to Solve:

1. **Calculate Term Frequency (TF):**
   - TF measures how frequently a term occurs in a document.
   - Formula: $\text{TF}(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$

   For Document 1, the term "data" appears 1 time, and the total number of terms in Document 1 is 4.

$$\text{TF}(\text{"data"}, \text{Document 1}) = \frac{1}{4} = 0.25$$

2. **Calculate Inverse Document Frequency (IDF):**
   - IDF measures the importance of a term across the corpus.
   - Formula: $\text{IDF}(t) = \log \frac{N}{\text{DF}(t)}$, where N is the total number of documents and $\text{DF}(t)$ is the number of documents containing the term t.

   Here, the term "data" appears in all 3 documents.

$$\text{IDF}(\text{"data"}) = \log \frac{3}{3} = \log 1 = 0$$

   Since $\log 1 = 0$, the IDF for "data" is 0, indicating that the term is very common across all documents.

3. **Calculate TF-IDF:**
   - Formula: $\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$

$$\text{TF-IDF}(\text{"data"}, \text{Document 1}) = 0.25 \times 0 = 0$$

   The TF-IDF for "data" in Document 1 is 0, which implies that "data" is not very informative as it appears in all documents.

3. **Calculate TF-IDF:**
   - Formula: $\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$

$$\text{TF-IDF}(\text{"data"}, \text{Document 1}) = 0.25 \times 0 = 0$$

The TF-IDF for "data" in Document 1 is 0, which implies that "data" is not very informative as it appears in all documents.

2. Cosine Similarity Calculation

Problem:
Calculate the cosine similarity between the following two documents:

- **Document A:** "text mining data"
- **Document B:** "data mining techniques"

Steps to Solve:

1. **Vector Representation:**
   - First, create a vocabulary of unique terms from both documents: $\text{"text"}, \text{"mining"}, \text{"data"}, \text{"techniques"}$
   - Represent each document as a vector:
     - **Document A:** $(1, 1, 1, 0)$ (contains "text", "mining", "data", but not "techniques")
     - **Document B:** $(0, 1, 1, 1)$ (contains "mining", "data", "technic

Continue generating

2. **Calculate Dot Product:**

$$\text{Dot Product} = 1 \times 0 + 1 \times 1 + 1 \times 1 + 0 \times 1 = 0 + 1 + 1 + 0 = 2$$

3. **Calculate Magnitude of Vectors:**

   - Magnitude of Document A:

$$\|A\| = \sqrt{1^2 + 1^2 + 1^2 + 0^2} = \sqrt{3}$$

   - Magnitude of Document B:

$$\|B\| = \sqrt{0^2 + 1^2 + 1^2 + 1^2} = \sqrt{3}$$

4. **Calculate Cosine Similarity:**

$$\text{Cosine Similarity} = \frac{\text{Dot Product}}{\|A\| \times \|B\|} = \frac{2}{\sqrt{3} \times \sqrt{3}} = \frac{2}{3} \approx 0.67$$

   The cosine similarity between Document A and Document B is approximately 0.67, indicating a moderate degree of similarity.

3. Jaccard Similarity Calculation

Problem:

Calculate the Jaccard similarity between the following two sets of terms from two documents:

- **Document A:** "information retrieval text mining"
- **Document B:** "text mining information extraction"

Steps to Solve:

Steps to Solve:

1. **Identify Unique Terms:**
   - **Set A:** { "information", "retrieval", "text", "mining" }
   - **Set B:** { "text", "mining", "information", "extraction" }
2. **Calculate Intersection and Union:**
   - **Intersection (common terms):** { "text", "mining", "information" }
   - **Union (all unique terms):** { "information", "retrieval", "text", "mining", "extraction" }
3. **Calculate Jaccard Similarity:**

$$\text{Jaccard Similarity} = \frac{\text{Number of Intersection Terms}}{\text{Number of Union Terms}} = \frac{3}{5} = 0.6$$

   The Jaccard similarity between the two documents is 0.6, indicating they share 60% of their unique terms.

4. Precision and Recall Calculation

Problem:
Given a set of documents retrieved by a search engine for the query "text mining," calculate precision and recall based on the following information:

- **Total documents retrieved:** 10
- **Relevant documents retrieved:** 6

4. Precision and Recall Calculation

Problem:

Given a set of documents retrieved by a search engine for the query "text mining," calculate precision and recall based on the following information:

- **Total documents retrieved:** 10
- **Relevant documents retrieved:** 6
- **Total relevant documents in the corpus:** 8

Steps to Solve:

1. **Calculate Precision:**
   - Precision measures the proportion of retrieved documents that are relevant.
   - Formula: $\text{Precision} = \frac{\text{Number of Relevant Documents Retrieved}}{\text{Total Number of Documents Retrieved}}$

$$\text{Precision} = \frac{6}{10} = 0.6$$

2. **Calculate Recall:**
   - Recall measures the proportion of relevant documents that were retrieved.
   - Formula: $\text{Recall} = \frac{\text{Number of Relevant Documents Retrieved}}{\text{Total Number of Relevant Documents}}$

$$\text{Recall} = \frac{6}{8} = 0.75$$

Continue generating

Precision is 0.6, and recall is 0.75.

5. Evaluation Metrics: F1 Score

Problem:

Given the precision and recall values calculated above (Precision = 0.6, Recall = 0.75), compute the F1 Score.

Steps to Solve:

1. **Calculate F1 Score:**
   - F1 Score is the harmonic mean of precision and recall.
   - Formula: $\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

$$\text{F1 Score} = 2 \times \frac{0.6 \times 0.75}{0.6 + 0.75} = 2 \times \frac{0.45}{1.35} \approx 0.67$$

The F1 Score is approximately 0.67, representing a balance between precision and recall.

Let's walk through the solution for the **Latent Dirichlet Allocation (LDA) Topic Probability** problem step by step.

Problem Recap:

You are given the following word-topic distribution from an LDA model for two topics:

- **Topic 1:** { "data": 0.4, "mining": 0.3, "learning": 0.3 }
- **Topic 2:** { "science": 0.5, "mining": 0.2, "AI": 0.3 }

Given a document with the words "data mining learning," calculate the probability that this document belongs to each of the two topics.

Solution Steps:

1. **Calculate the Probability of the Document Under Each Topic:**
   - For each topic, multiply the probabilities of the words "data," "mining," and "learning" occurring together.
2. **Assume Equal Prior Probabilities for Each Topic:**
   - Since the prior probabilities are equal, you can focus on the product of the word probabilities under each topic.
3. **Normalize the Probabilities:**
   - Convert the raw probabilities into a normalized probability distribu
     topics.

4. **Determine the Most Likely Topic:**
    - Identify which topic the document is most likely to belong to.

Step-by-Step Calculation:

Step 1: Calculate the Probability of the Document Under Each Topic

- **Topic 1:**

$$P(\text{Document}|\text{Topic 1}) = P(\text{"data"}|\text{Topic 1}) \times P(\text{"mining"}|\text{Topic 1}) \times P(\text{"learnin}$$

$$P(\text{Document}|\text{Topic 1}) = 0.4 \times 0.3 \times 0.3 = 0.036$$

- **Topic 2:**

$$P(\text{Document}|\text{Topic 2}) = P(\text{"data"}|\text{Topic 2}) \times P(\text{"mining"}|\text{Topic 2}) \times P(\text{"learnin}$$

    - However, the word "learning" does not exist in Topic 2's distribution. For simplicity, we can assume its probability is very close to 0.
    - As a result, $P(\text{Document}|\text{Topic 2})$ will be very small or 0, making Topic 2 highly unlikely.

    For now, let's calculate the product with the known terms:

$$P(\text{Document}|\text{Topic 2}) = 0.0 \times 0.2 \times 0.3 = 0.0$$

Step 2: Equal Prior Probabilities

Assume that both topics have an equal prior probability, so no further multiplication is needed.

Step 3: Normalize the Probabilities

The raw probabilities are:

- $P(\text{Document}|\text{Topic 1}) = 0.036$
- $P(\text{Document}|\text{Topic 2}) = 0.0$

To normalize, we calculate the total probability:

$$P(\text{Document}) = 0.036 + 0.0 = 0.036$$

The normalized probabilities are:

- **Topic 1:**

$$P(\text{Topic 1}|\text{Document}) = \frac{0.036}{0.036} = 1.0$$

- **Topic 2:**

$$P(\text{Topic 2}|\text{Document}) = \frac{0.0}{0.036} = 0.0$$

1. Part of Speech (POS) Tagging Basics

Problem Recap:
Given the sentence: "The quick brown fox jumps over the lazy dog," manually assign the most likely POS tags to each word.

Solution:

1. **The (DT)** - "The" is a determiner (DT), used to specify a noun.
2. **quick (JJ)** - "quick" is an adjective (JJ) describing the noun "fox."
3. **brown (JJ)** - "brown" is also an adjective (JJ) describing the noun "fox."
4. **fox (NN)** - "fox" is a noun (NN), the subject of the sentence.
5. **jumps (VBZ)** - "jumps" is a verb (VBZ) in the third person singular present tense.
6. **over (IN)** - "over" is a preposition (IN) introducing the prepositional phrase.
7. **the (DT)** - "the" is again a determiner (DT).
8. **lazy (JJ)** - "lazy" is an adjective (JJ) describing the noun "dog."
9. **dog (NN)** - "dog" is a noun (NN), the object of the preposition "over."

POS Tagged Sentence: The/DT quick/JJ brown/JJ fox/NN jumps/VBZ over/IN the/DT lazy/JJ dog/NN

2. Stochastic Language Models

Problem Recap:
Given bigram probabilities, calculate the probability of the sentence "The dog jumps over the cat."

Solution:

1. **Calculate the Probability Expression:**

   The probability of the sentence is calculated using the chain rule:

   $$P(\text{"The dog jumps over the cat"}) = P(\text{"The"}) \times P(\text{"dog"}|\text{"The"}) \times P(\text{"jumps"}|\text{"}$$

2. **Insert the Bigram Probabilities:**

   $$P(\text{"The dog jumps over the cat"}) = 1 \times 0.5 \times 0.3 \times 0.2 \times 0.6 \times 0.3$$

3. **Calculate the Overall Probability:**

   $$P(\text{"The dog jumps over the cat"}) = 0.5 \times 0.3 \times 0.2 \times 0.6 \times 0.3 = 0.0054$$

Conclusion: The probability of the sentence "The dog jumps over the cat" is 0.0054.

3. Generative Models for POS Tagging

Problem Recap:
Given emission and transition probabilities, calculate the probability of the POS tag sequence DT -> NN -> VB for the sentence "The dog runs."

Solution:

1. **Calculate the Probability of the Tag Sequence:**

   The probability of the tag sequence is calculated using both the transition and emission probabilities:

$$P(\text{DT} \rightarrow \text{NN} \rightarrow \text{VB}) = P(\text{DT}|\text{START}) \times P(\text{NN}|\text{DT}) \times P(\text{VB}|\text{NN})$$

$$\times P(\text{"the"}|\text{DT}) \times P(\text{"dog"}|\text{NN}) \times P(\text{"runs"}|\text{VB})$$

2. **Insert the Given Probabilities:**

$$P(\text{DT} \rightarrow \text{NN} \rightarrow \text{VB}) = 0.6 \times 0.4 \times 0.3 \times 0.8 \times 0.7 \times 0.5$$

3. **Calculate the Overall Probability:**

$$P(\text{DT} \rightarrow \text{NN} \rightarrow \text{VB}) = 0.6 \times 0.4 \times 0.3 \times 0.8 \times 0.7 \times 0.5 = 0.02016$$

Conclusion: The probability of the POS tag sequence DT -> NN -> VB for the sentence "The dog runs" is 0.02016.

4. Parts of Speech Tagging using Hidden Markov Model (HMM)

Problem Recap:

Given state transition and emission probabilities in an HMM, calculate the most likely sequence of POS tags for the sentence "The dog barks."

Solution:

1. **Possible Tag Sequences:**
   - Sequence 1: DT -> NN -> VB
   - Sequence 2: DT -> NN -> NN
   - Sequence 3: DT -> VB -> NN
2. **Calculate the Probability for Each Sequence:**
   - **Sequence 1 (DT -> NN -> VB):**

$$P(\text{DT} \rightarrow \text{NN} \rightarrow \text{VB}) = P(\text{DT}|\text{START}) \times P(\text{NN}|\text{DT}) \times P(\text{VB}|\text{NN}) \times P(\text{"the}$$

$$= 0.6 \times 0.4 \times 0.3 \times 0.7 \times 0.5 \times 0.6 = 0.0252$$

   - **Sequence 2 (DT -> NN -> NN):**

$$P(\text{DT} \rightarrow \text{NN} \rightarrow \text{NN}) = 0.6 \times 0.4 \times 0.4 \times 0.7 \times 0.5 \times 0.5 = 0.0168$$

- **Sequence 3 (DT -> VB -> NN):**

$$P(\text{DT} \rightarrow \text{VB} \rightarrow \text{NN}) = 0.6 \times 0.3 \times 0.4 \times 0.7 \times 0.5 \times 0.5 = 0.0126$$

3. **Conclusion:**
   The most likely tag sequence for "The dog barks" is **DT -> NN -> VB** with a probability of 0.0252.

5. Viterbi Algorithm

Problem Recap:
Using the HMM parameters, apply the Viterbi algorithm to find the most likely sequence of POS tags for the sentence "She eats fish."

Solution:

1. **Initialization:**
   - For the first word "She":
     - PRP: $V_1(\text{PRP}) = P(\text{"she"}|\text{PRP}) \times P(\text{PRP}|\text{START}) = 0.9 \times 1 = 0.9$
   - NN and VB are not possible at the first position.
2. **Recursion:**
   - For the second word "eats":
     - VB: $V_2(\text{VB}) = \max\left(V_1(\text{PRP}) \times P(\text{VB}|\text{PRP})\right) \times P(\text{"eats"}|\text{VB}) = 0.9 \times 0.5 \times 0.8 = 0.36$
   - NN is not likely since "eats" is more likely a verb.
   - NN is not likely since "eats" is more likely a verb.
   - For the third word "fish":
     - NN: $V_3(\text{NN}) = \max\left(V_2(\text{VB}) \times P(\text{NN}|\text{VB})\right) \times P(\text{"fish"}|\text{NN}) = 0.36 \times 0.6 \times 0.5 = 0.108$
3. **Termination and Backtrace:**
   - The most likely sequence is PRP -> VB -> NN.
   - Backtrace: "She" is PRP, "eats" is VB, "fish" is NN.

Conclusion: The most likely POS tag sequence is **PRP -> VB -> NN**.

6. Implementing POS Tagging in Python

Problem Recap:
Write a Python program to implement a simple POS tagger using the NLTK library.

Solution:

1. Introduction to Topic Modelling

Problem: Explain the concept of topic modeling. Given a collection of documents, how would topic modeling help in understanding the underlying themes?

Solution:

Concept: Topic modeling is an unsupervised machine learning technique used to identify abstract topics within a collection of documents. It assumes that each document is a mixture of topics and that each topic is a mixture of words.

Application: Given a collection of documents, topic modeling helps in:

Identifying key themes (topics) that are prevalent across the documents.

Clustering words with similar meanings together under a common topic.

Organizing large sets of textual data into interpretable structures for better understanding and analysis.

For instance, if we have a collection of news articles, topic modeling might identify topics such as "Politics," "Sports," "Technology," etc., and associate each article with these topics based on the words it contains.

---

2. Topic Modelling using LDA

Problem: Describe how Latent Dirichlet Allocation (LDA) works for topic modeling. What are the key components involved in LDA?

Solution:

Components of LDA:

Document-Topic Distribution: Each document is represented as a distribution over topics.

Topic-Word Distribution: Each topic is represented as a distribution over words.

Dirichlet Priors: LDA uses Dirichlet distributions as priors for the document-topic and topic-word distributions.

How LDA Works:

Initialize Random Topic Assignments: Start with random assignment of topics to words in the document.

Iterative Process:

For each word in each document, reassign the topic based on the probability that the word belongs to a certain topic given the other words in the document and the words assigned to the same topic across all documents.

Convergence: The process continues until the topic assignments stabilize, meaning the distributions over topics for each document and words for each topic no longer change significantly.

LDA is generative, meaning it assumes documents are generated through a mixture of topics and works backward to uncover the topics based on the observed words.

---

3. Mathematical Foundations for LDA: Multinomial and Dirichlet Distributions (Part 1)

Problem: Explain the role of the multinomial distribution in LDA. How is it used in the context of topic modeling?

Solution:

Multinomial Distribution: The multinomial distribution is a generalization of the binomial distribution. It represents the probability of obtaining a certain combination of outcomes from multiple categories in a single trial.

Role in LDA:

Topic-Word Distribution: LDA uses a multinomial distribution to model the probability of words given a topic. Each topic is represented as a multinomial distribution over the vocabulary.

Document-Topic Distribution: Similarly, LDA models the probability of topics given a document as a multinomial distribution.
In LDA, the multinomial distribution helps in modeling the process of selecting words based on the topic distribution and topics based on the document distribution.

---

4. Mathematical Foundations for LDA: Multinomial and Dirichlet Distributions (Part 2)
Problem: What is a Dirichlet distribution, and how is it used as a prior in LDA? Explain with an example.
Solution:
Dirichlet Distribution: The Dirichlet distribution is a family of continuous multivariate probability distributions parameterized by a vector of positive reals. It is often used as a prior for multinomial distributions in Bayesian statistics.
Role in LDA:
Document-Topic Distribution: A Dirichlet prior is placed on the document-topic distribution, which means that the distribution of topics in each document is governed by a Dirichlet distribution.
Topic-Word Distribution: Similarly, a Dirichlet prior is placed on the topic-word distribution, governing the distribution of words in each topic.
Example: Suppose we have three topics and a document. The Dirichlet prior might indicate that it's likely for a document to be heavily focused on one topic but still have some contribution from the others. If the Dirichlet parameter is [2, 2, 2], it suggests that each topic is equally likely in the document. If the parameter is [5, 1, 1], it indicates a preference for the first topic.

---

5. Gibbs Sampling for LDA (Part 1)
Problem: Describe the concept of Gibbs sampling in the context of LDA. Why is it used?
Solution:
Gibbs Sampling: Gibbs sampling is a Markov Chain Monte Carlo (MCMC) algorithm used to sample from a high-dimensional probability distribution when direct sampling is difficult.
In the Context of LDA:
Purpose: LDA involves complex probability distributions over documents, topics, and words. Direct computation of these distributions is intractable due to their high dimensionality.
How It's Used: Gibbs sampling is used to approximate the distributions by iteratively sampling from the conditional distribution of each variable given the others. In LDA, this means iterating over each word in each document, sampling a new topic assignment based on the current assignments of all other words.
Why It's Used: Gibbs sampling helps LDA converge to a stable distribution of topics without having to compute the full joint distribution directly, making the algorithm computationally feasible.

---

6. Gibbs Sampling for LDA (Part 2)
Problem: Given an initial set of topic assignments, demonstrate how Gibbs sampling updates the topic assignment for a specific word in a document.
Solution:
Initial Setup: Suppose we have a document with words [w1, w2, w3], and initial topic assignments [T1, T2, T1].
Gibbs Sampling Update:
Remove the Word: Temporarily remove w2 and its current topic assignment T2.

Calculate the Conditional Probability:
Compute the probability of assigning each possible topic to w2, given the current topic assignments of the other words.
This probability is proportional to the product of:
The probability of w2 given the topic (how likely the word is to occur under each topic).
The probability of the topic given the document (how likely the topic is given the remaining topic assignments in the document).
Sample a New Topic: Based on these probabilities, sample a new topic for w2.
Update the Assignment: Assign the new topic to w2 and proceed to the next word.
Iteration: Repeat this process for all words in all documents until convergence.

Steps Explanation:
Preprocessing: Tokenize and clean the documents by removing stopwords.
Create Dictionary: Map each word to a unique ID.
Create Corpus: Convert the documents into a bag-of-words format.
Train LDA Model: Use the gensim library to train an LDA model on the corpus.
Display Topics: Print the top words associated with each topic.

1. Topic Modelling using LDA
Problem: You have a corpus of 10,000 documents, and you wish to identify 5 topics using LDA. Explain how LDA assigns topics to the documents and what output you can expect after running the algorithm.
Solution:
How LDA Assigns Topics:
Initialization: LDA starts by randomly assigning each word in each document to one of the 5 topics.
Iterative Process: For each word in each document, LDA reassigns a topic based on two factors:
The proportion of words in the document that are currently assigned to each topic.
The proportion of times the word is assigned to each topic across the entire corpus.
Convergence: After several iterations, the assignments stabilize, meaning the algorithm converges.
Expected Output:
Document-Topic Distribution: For each document, LDA provides a probability distribution over the 5 topics, indicating the strength of each topic within that document.
Topic-Word Distribution: For each topic, LDA provides a probability distribution over words, indicating which words are most strongly associated with each topic.
Top Words per Topic: LDA outputs the most significant words for each topic, allowing interpretation of the topics.
Example Output: For a document discussing technology trends:
Topic 1: { "AI": 0.3, "Machine Learning": 0.25, "Deep Learning": 0.2, "Data": 0.15, "Neural Networks": 0.1 }

---

2. Introduction to Topic Modelling
Problem: Given a collection of customer reviews, describe how topic modeling could help a business understand customer sentiments and identify areas for improvement.
Solution:
Application of Topic Modelling:
Identify Common Themes: Topic modeling can identify the most common themes across the reviews, such as "customer service," "product quality," or "pricing."

Sentiment Analysis: By associating topics with sentiment words, businesses can determine the overall sentiment (positive, negative, neutral) related to each theme.
Actionable Insights: For example, if a significant topic is "delivery time" and it's associated with negative sentiments, the business might focus on improving its delivery process.
Process:
Preprocessing: Tokenize and clean the reviews.
LDA Application: Apply LDA to extract topics.
Interpretation: Analyze the top words for each topic and associate them with specific themes.
Sentiment Analysis: Combine with sentiment analysis to understand the sentiment associated with each topic.
Conclusion: Topic modeling can reveal hidden patterns in customer reviews, helping businesses make data-driven decisions.

---

3. Mathematical Foundations for LDA: Multinomial and Dirichlet Distributions (Part 1)
Problem: Consider a simplified scenario where you have 3 documents and a vocabulary of 5 words. Assume there are 2 topics. Write the multinomial distribution for a single document, assuming equal probabilities for each word within a topic.
Solution:
Vocabulary: Vocab={"word1","word2","word3","word4","word5"}\text{Vocab} = \{ \text{"word1"}, \text{"word2"}, \text{"word3"}, \text{"word4"}, \text{"word5"} \}Vocab={"word1","word2","word3","word4","word5"}
Topics: Topic 1=Uniform distribution over vocab\text{Topic 1} = \text{Uniform distribution over vocab}Topic 1=Uniform distribution over vocab
Topic 2=Uniform distribution over vocab\text{Topic 2} = \text{Uniform distribution over vocab}Topic 2=Uniform distribution over vocab
Multinomial Distribution for a Single Document:
Assume a document is composed of 10 words.
Topic 1 and Topic 2 each have an equal probability (0.5) of generating a word.

Topic-Word Probabilities:

$$P(\text{"word1"}|\text{Topic 1}) = \frac{1}{5} = 0.2$$
$$P(\text{"word2"}|\text{Topic 1}) = \frac{1}{5} = 0.2$$
$$P(\text{"word1"}|\text{Topic 2}) = \frac{1}{5} = 0.2$$
... and similarly for other words ...

Multinomial Distribution Example:

For a document with word counts [2, 3, 1, 2, 2] corresponding to the words in the vocabulary:

$$P(\text{document}) = P(\text{"word1"})^2 \times P(\text{"word2"})^3 \times P(\text{"word3"})^1 \times P(\text{"word4"})^2 \times P(\text{"w}$$

If the document is generated by Topic 1:

$$P(\text{document}|\text{Topic 1}) = 0.2^2 \times 0.2^3 \times 0.2^1 \times 0.2^2 \times 0.2^2 = 0.2^{10}$$

Conclusion: The multinomial distribution represents the probability of observing a specific set of word frequencies given the topic's word distribution.

4. Mathematical Foundations for LDA: Multinomial and Dirichlet Distributions (Part 2)

Problem: Suppose the Dirichlet parameter for the topic distribution in a document is α = [2, 2]. Calculate the expected proportion of topics in the document.
Solution:
Dirichlet Distribution: The Dirichlet distribution is parameterized by α, which in this case is [2, 2].
Expected Proportions: For a Dirichlet distribution with parameters α = [α1, α2], the expected value for each topic is given by:
by:

$$E(\theta_i) = \frac{\alpha_i}{\sum_j \alpha_j}$$

Calculation:

For α = [2, 2]:

$$E(\theta_1) = \frac{2}{2+2} = \frac{2}{4} = 0.5$$

$$E(\theta_2) = \frac{2}{4} = 0.5$$

5. Gibbs Sampling for LDA (Part 1)
Problem: Explain how Gibbs sampling helps in inferring the topic distribution for each document in LDA. Use an example with a small corpus to illustrate the steps involved.
Solution:
Concept: Gibbs sampling is a way to approximate the posterior distribution of the model parameters in LDA, especially when direct computation is infeasible.
Steps in Gibbs Sampling for LDA:
Initialize Randomly: Start with a random assignment of topics to words in the corpus.
Iterate Over Words:
For each word wiw_iwi in document ddd, remove its current topic assignment.
Calculate the probability of assigning each possible topic kkk to the word based on:
The proportion of words in document ddd currently assigned to topic kkk.
The proportion of times word wiw_iwi is assigned to topic kkk across all documents.
Sample a new topic for word wiw_iwi based on this probability.
Update Assignments: Reassign the new topic to word wiw_iwi and proceed to the next word.
Repeat: Continue the process until convergence, i.e., when topic assignments stabilize.
Example: Given a small corpus of 3 documents:
Doc1: "apple orange banana"
Doc2: "banana apple"
Doc3: "orange banana"
After several iterations of Gibbs sampling:
Doc1 might have a high probability of Topic 1 (related to fruits).
Doc2 might be a mix of Topic 1 and another topic.
Doc3 might also have a high probability of Topic 1.
Conclusion: Gibbs sampling enables LDA to infer the topic distribution for each document by iteratively refining the topic assignments for words.

Gibbs Sampling for LDA (Part 2)

Problem: Suppose after running Gibbs sampling for several iterations, the following topic-word assignments for a word in a document are observed:
Topic 1: 3 times
Topic 2: 7 times
If the topic distribution is Dirichlet(α = [1, 1]), what is the probability of assigning the word to Topic 1 or Topic 2 in the next iteration?
Solution:
Gibbs Sampling Probability: The probability of assigning a word to a topic kkk during Gibbs sampling is proportional to:

Solution:

Gibbs Sampling Probability:

The probability of assigning a word to a topic k during Gibbs sampling is proportional to:

$$P(z_i = k|z_{-i}, w) \propto \frac{n_{d,k}^{-i} + \alpha_k}{\sum_k (n_{d,k}^{-i} + \alpha_k)} \times \frac{n_{k,w}^{-i} + \beta_w}{\sum_w (n_{k,w}^{-i} + \beta_w)}$$

where:

- $n_{d,k}^{-i}$ is the count of words in document d assigned to topic k, excluding the current word.
- $n_{k,w}^{-i}$ is the count of word w assigned to topic k across the entire corpus, excluding the current word.

Given:
Topic 1: 3 times
Topic 2: 7 times
α=[1,1]

Probability Calculation:

$$P(z_i = 1) \propto \frac{3+1}{10+2} = \frac{4}{12} = \frac{1}{3}$$

$$P(z_i = 2) \propto \frac{7+1}{12} = \frac{8}{12} = \frac{2}{3}$$

Conclusion: The word is twice as likely to be assigned to Topic 2 as to Topic 1 in the next iteration.

1. Sentiment Scoring with Lexicons
Problem: You have the following sentences and a sentiment lexicon:
Sentence: "The movie was surprisingly good but a bit too long."
Lexicon:
"good" : +2
"surprisingly" : +1
"too" : -1
"long" : -2
Calculate the overall sentiment score of the sentence using the lexicon and explain what this score represents.
Solution:
Identify Words in the Sentence:
"good" → +2
"surprisingly" → +1

"too" → -1
"long" → -2
Calculate the Total Sentiment Score:
Total score = (+2) + (+1) + (-1) + (-2) = 0
Interpretation:
A sentiment score of 0 indicates that the sentence is neutral overall. The positive and negative sentiments cancel each other out.
Conclusion: Despite having both positive and negative elements, the sentence has a neutral sentiment when using this lexicon.

2. Probability Calculation in a Naive Bayes Sentiment Classifier
Problem: Consider a Naive Bayes classifier trained on the following sentiment-labeled dataset:
Positive Reviews: 200
Negative Reviews: 100
You encounter a new sentence with the following word probabilities:
P(word="great" | Positive) = 0.02
P(word="great" | Negative) = 0.01
Calculate the posterior probability that the sentence "The product is great" is positive using Naive Bayes, assuming the word "great" is the only feature used in the model. Assume uniform priors.
Solution: Prior Probabilities:
P(Positive)=200300=23,P(Negative)=100300=13P(Positive) = \frac{200}{300} = \frac{2}{3}, \quad P(Negative) = \frac{100}{300} = \frac{1}{3}P(Positive)=300200=32 ,P(Negative)=300100=31
Likelihood of "great":
P(word="great"|Positive)=0.02,P(word="great"|Negative)=0.01P(word="great" | Positive) = 0.02, \quad P(word="great" | Negative) = 0.01P(word="great"|Positive)=0.02,P(word="great"|Negative)=0.01
Posterior Probability: Using Bayes' Theorem:
P(Positive|"great")=P("great"|Positive)×P(Positive)P("great")P(Positive | "great") = \frac{P("great" | Positive) \times P(Positive)}{P("great")}P(Positive|"great")=P("great")P("great"|Positive)×P(Positive)
Where P("great")P("great")P("great") is:
P("great")=P("great"|Positive)×P(Positive)+P("great"|Negative)×P(Negative)P("great") = P("great" | Positive) \times P(Positive) + P("great" | Negative) \times P(Negative)P("great")=P("great"|Positive)×P(Positive)+P("great"|Negative)×P(Negative)
P("great")=(0.02×23)+(0.01×13)=0.0133+0.0033=0.0166P("great") = (0.02 \times \frac{2}{3}) + (0.01 \times \frac{1}{3}) = 0.0133 + 0.0033 = 0.0166P("great")=(0.02×32)+(0.01×31)=0.0133+0.0033=0.0166
Now, calculate the posterior probability:
P(Positive|"great")=0.02×230.0166=0.01330.0166≈0.80P(Positive | "great") = \frac{0.02 \times \frac{2}{3}}{0.0166} = \frac{0.0133}{0.0166} \approx 0.80P(Positive|"great")=0.01660.02×32=0.01660.0133≈0.80
Conclusion: The posterior probability that the sentence "The product is great" is positive is 0.80, indicating that it is highly likely to be positive.

3. Term Frequency-Inverse Document Frequency (TF-IDF) Calculation
Problem: Given the following two documents:

Document 1: "I love programming in Python." Document 2: "Python programming is fun."
Calculate the TF-IDF score for the word "Python" in both documents. Assume a corpus of these two documents.
Solution: Term Frequency (TF): TF("Python", Document 1) = 1/5 (Python appears once in a document of 5 words) = 0.2
TF("Python", Document 2) = 1/4 (Python appears once in a document of 4 words) = 0.25
Document Frequency (DF): DF("Python") = 2 (Python appears in both documents)

3. **Inverse Document Frequency (IDF):**

$$IDF(\text{"}Python\text{"}) = \log(\frac{TotalDocuments}{DF(\text{"}Python\text{"})}) = \log(\frac{2}{2}) = \log(1) = 0$$

TF-IDF Calculation: TF-IDF("Python", Document 1) = 0.2 × 0 = 0 TF-IDF("Python", Document 2) = 0.25 × 0 = 0