	Information Retrieval	Data Retrieval
Data	Free text, unstructured	Database tables, structured
Queries	Keywords, Natural language	SQL, Relational algebras
Results	Approximate matches	Exact matches
Results	Ordered by relevance	Unordered
Accessibility	Non-expert humans	Knowledgeable users or automatic processes

Information Retrieval	Data Retrieval
The software program that deals with the organization, storage, retrieval, and evaluation of information from document repositories particularly textual information.	Data retrieval deals with obtaining data from a database management system such as ODBMS. It is A process of identifying and retrieving the data from the database, based on the query provided by user or application.
Retrieves information about a subject.	Determines the keywords in the user query and retrieves the data.
Small errors are likely to go unnoticed.	A single error object means total failure.
Not always well structured and is semantically ambiguous.	Has a well-defined structure and semantics.
Does not provide a solution to the user of the database system.	Provides solutions to the user of the database system.
It is a probabilistic model.	It is a deterministic model.

Information retrieval (IR)

is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

- To get the data by relevance of the query from large repository of the data.
- Unstructured /semi structured data
- Information retrieval (IR) is subfield of computer science that deals with **automated retrieval of information (especially text)** based on their **content and context**.
- Given a **set of documents**, **clustering** is the task of coming up with a good grouping of the documents based on their contents
- Given a **set of topics**, categories **classification** is the task of deciding which class(es), if any, each of a set of documents belongs to.

Key components of information retrieval include indexing, search, ranking, and evaluation of relevance, all aimed at efficiently connecting users with the information they need.

Applications of Information Retrieval

- **Web Search Engines**: Indexing and retrieving web pages.
- **Digital Libraries**: Accessing academic papers, books, and other scholarly resources.
- **Recommender Systems**: Suggesting products, movies, or content based on user preferences.
- **Medical Information Retrieval**: Retrieving relevant medical documents or research papers for healthcare professionals.

IR is foundational for various applications in data science, machine learning, and artificial intelligence, playing a crucial role in how we interact with vast amounts of information today.

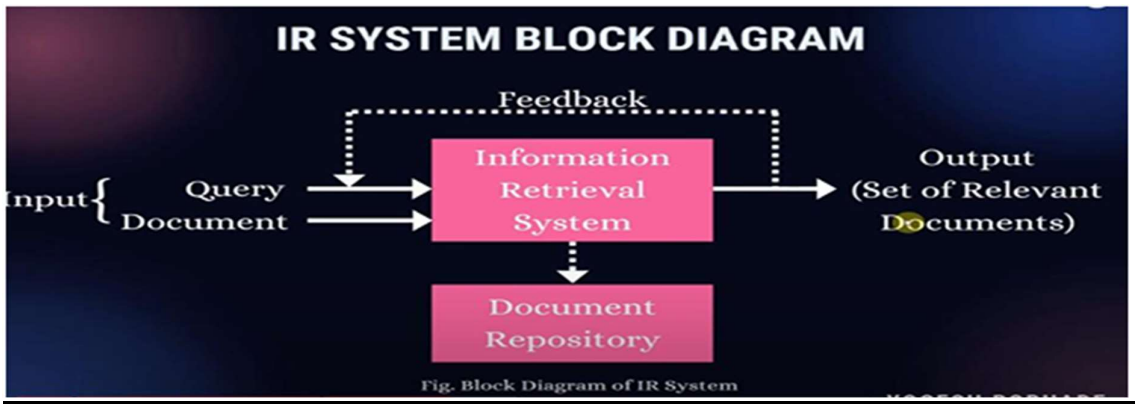


Fig. Block Diagram of IR System

1. **Query**: This is the input from the user, which specifies what information they are seeking. It can be a text-based query, a keyword, or a more complex search request.
2. **Information Retrieval System**:
   - **Query Processing**: Parses and processes the query to understand its intent and requirements.
   - **Indexing**: Creates and maintains an index of documents in the repository to facilitate efficient retrieval.
3. **Document Repository**:
   - **Storage**: Stores all the documents and data that can be retrieved. This may include text documents, images, videos, etc.
   - **Metadata**: Stores additional information about documents, such as keywords, summaries, and other relevant attributes.
4. **Output**: The results returned by the IR system based on the processed query and index. This typically includes a list of relevant documents or data items.
5. **Feedback**: Allows users to provide feedback on the relevance and quality of the results. This feedback can be used to improve future queries, enhance the indexing process, and refine the overall system performance.

Major challengers in IR: (ir qu ms pd ts led)

- Information Overload: The exponential growth of digital content on the internet and other sources has led to information overload. Users are often overwhelmed by the sheer volume of search results, making it difficult to find the most relevant information quickly.
- Relevance and Precision: Ensuring that the retrieved documents are relevant to the user's query is a critical challenge. IR systems need to strike a balance between providing comprehensive results and avoiding irrelevant or low-quality documents.
- Query Ambiguity: Queries from users can often be ambiguous or imprecise, making it challenging for IR systems to accurately understand the user's intent. This problem is particularly acute when dealing with natural language queries.
- User Intent Understanding: Understanding the user's underlying intent behind the query is crucial for providing relevant results. However, accurately capturing user intent remains challenging, especially for complex or long-tail queries.
- Multilingual Information Retrieval: As the internet connects people from different linguistic backgrounds, IR systems must handle queries and content in multiple languages. Translating queries and retrieving relevant documents across languages presents unique challenges.
- Scalability and Efficiency: With the exponential growth of data, IR systems need to handle large-scale collections efficiently. Real-time retrieval and quick response times are essential to meet user expectations.
- Personalization: Users have become accustomed to personalized experiences on the web. IR systems need to consider user preferences, behavior, and context to provide personalized search results and recommendations.
- Diversity in Information Sources: Modern IR systems must retrieve information from various sources, such as web pages, social media, images, videos, and more. Integrating and ranking information from diverse sources pose additional challenges.
- Trust and Credibility: With the abundance of misinformation and fake news on the internet, users seek reliable and credible information. Ensuring the trustworthiness of retrieved content is a growing concern.
- Semantic Gap: The semantic gap refers to the mismatch between the user's query and the representation of content in the documents. Bridging this gap to better match user intent is a continuing challenge.
- Long-Tail Queries: Handling infrequent or specific queries, known as long-tail queries, is challenging because they may have limited data for learning relevant patterns.
- Evaluation and Metrics: Developing appropriate evaluation metrics that reflect the real-world utility of IR systems is an ongoing challenge. Existing metrics, such as relevance, may not fully capture user satisfaction or the quality of the retrieved information.
- Dynamic Content: Content on the web is continuously changing and evolving. Keeping IR systems up-to-date with the latest information poses challenges in maintaining freshness and accuracy.

key features:

1. **Relevance**:

- The IR system's ability to retrieve documents that are relevant to the user's query. This is the primary metric of an IR system's effectiveness.

2. **Ranking**:

- IR systems rank the retrieved documents based on their relevance to the user's query, presenting the most relevant documents at the top.

3. **Query Processing**:

- The system processes user queries through techniques like tokenization, normalization (e.g., stemming, lemmatization), and query expansion to improve search accuracy.

4. **Document Indexing**:

- Documents are indexed to enable efficient search and retrieval. Indexing structures such as inverted indexes allow the system to quickly locate documents that contain query terms.

5. **Precision and Recall**:

- **Precision**: The proportion of retrieved documents that are relevant.

- **Recall**: The proportion of all relevant documents that are retrieved.

- A good IR system aims to balance precision and recall.

6. **User Feedback and Interaction**:

- The system may incorporate user feedback, such as relevance feedback or click-through data, to improve retrieval performance over time.

7. **Scalability**:

- The IR system should efficiently handle large volumes of data and scale well as the number of documents or users increases.

8. **Support for Different Types of Data**:

- Modern IR systems often support multiple data types (text, images, videos, audio) and formats.

9. **Speed and Efficiency**:

- IR systems are designed to retrieve relevant documents quickly, even from large data repositories, ensuring low latency in search responses.

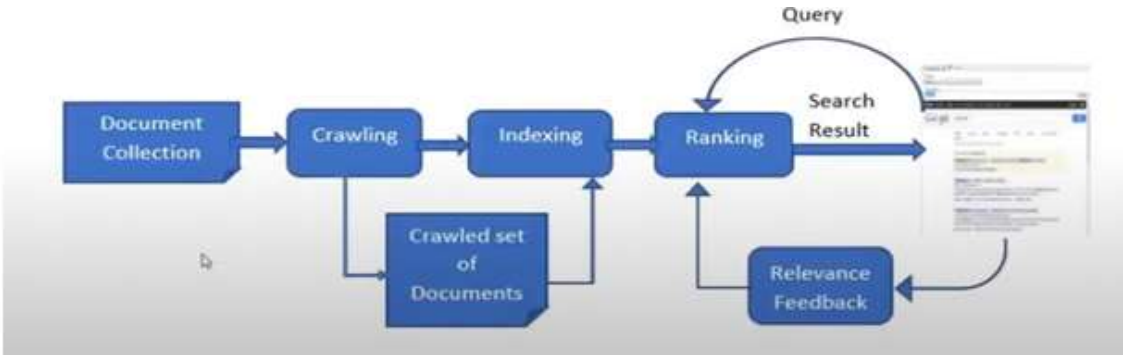
10. **Personalization**:

- Some IR systems offer personalized search results by considering user preferences, past searches, and behavior.

11. **Query Suggestions and Auto-completion**:

- The system may suggest related queries or auto-complete search phrases based on common patterns and user intent.

Components of IR Model



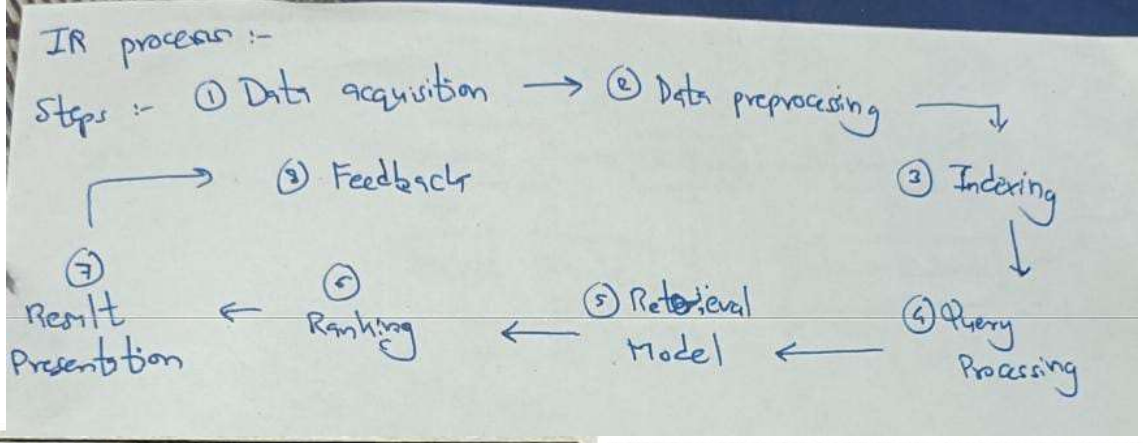
1. **Document Collection**: This is the set of documents (e.g., web pages, articles, books) that the IR system can search through. The quality and comprehensiveness of this collection directly impact the effectiveness of the system.
2. **Crawling**: Crawling is the process of automatically navigating through the web or other data sources to collect and update documents for the document collection. Web crawlers or spiders are often used for this task.
3. **Indexing**: Indexing involves processing the crawled documents to create an index, which is a structured representation that allows for fast retrieval. The index typically includes information like the terms present in each document and their locations.
4. **Crawled Set of Documents**: This refers to the subset of documents that have been collected through crawling and are stored in the document collection. These documents are then used for indexing.
5. **Ranking**: Ranking is the process of ordering the documents in the index based on their relevance to a given query. The ranking is often determined using algorithms like TF-IDF, BM25, or more advanced machine learning models.

6. **Query**: The query is the input provided by the user to search through the document collection. It can be a string of keywords, a question, or even a natural language sentence.
7. **Search Result**: The search result is the list of documents or snippets returned by the IR system in response to the user's query. These results are typically ranked by relevance.
8. **Relevance Feedback**: Relevance feedback is the process of using user feedback on the search results to improve the system's performance. For example, if the user marks certain results as relevant or irrelevant, the system can adjust its ranking algorithm accordingly.

Also the components of IR extra information:

1. **Documents and Queries:**
   - **Documents:** These are units of information, such as text files, web pages, or multimedia files.
   - **Queries:** A query is a user's request for information, often in the form of keywords or phrases.
2. **Relevance:**
   - Relevance refers to how well a document matches a query. The challenge in IR is to rank documents by their relevance to the user's query.
3. **Indexing:**
   - Indexing is the process of organizing data to make retrieval efficient. It involves creating data structures (like inverted indexes) that map keywords to the documents where they appear.
4. **Retrieval Models:**
   - Retrieval models determine how to rank documents based on a query. Some common models include:
     - **Boolean Model**: Uses logical operators (AND, OR, NOT) to match documents.
     - **Vector Space Model**: Represents documents and queries as vectors and ranks documents based on their cosine similarity.
     - **Probabilistic Models**: Rank documents based on the probability that they are relevant to the query.

5. **Evaluation Metrics:**
   - IR systems are evaluated based on how effectively they retrieve relevant information. Common metrics include:
     - **Precision**: The ratio of relevant documents retrieved to the total number of documents retrieved.
     - **Recall**: The ratio of relevant documents retrieved to the total number of relevant documents in the collection.
     - **F1 Score**: The harmonic mean of precision and recall.
6. **Search Engines:**
   - A practical application of IR is search engines like Google. These systems crawl the web, index content, and allow users to search and retrieve information.



9 IR PROCESSES AND FIELDS

Information Retrieval Process: The Information Retrieval process involves several interconnected steps, each contributing to the overall effectiveness of the retrieval system. The key steps in the IR process are:

- **Document Acquisition:** This step involves selecting and gathering documents and other objects from various sources such as the web, databases, digital libraries, and more. Web crawlers play a crucial role in collecting and indexing web pages for web search engines.
- **Data Preprocessing:** Before indexing and retrieval, raw data undergoes preprocessing to clean and transform it into a structured format. Tasks like tokenization, stop word removal, stemming, and lemmatization are performed to prepare the data for indexing.
- **Indexing:** Indexing creates data structures that efficiently represent the documents and their contents. It involves generating inverted indexes that map terms to the documents in which they occur. Indexing speeds up the retrieval process by reducing the search space.
- **Query Processing:** In this step, user queries are processed and analyzed to understand their information needs. Queries are transformed into a form suitable for retrieval, and additional processing like query expansion may be applied to improve retrieval accuracy.
- **Retrieval Model:** The retrieval model is the heart of the information retrieval system. It determines how documents are ranked and retrieved based on their relevance to the query. Common retrieval models include Boolean model, Vector Space Model, and Probabilistic Model.
- **Ranking:** Ranking is the process of assigning scores or ranks to documents based on their relevance to the query. Higher-ranked documents are presented at the top of search results, making them more accessible to users.
- **Results Presentation:** The retrieved documents and relevant information are presented to users in a user-friendly format, such as a list of search results, summaries, or snippets. Presentation plays a crucial role in enhancing the user experience.
- **User Feedback:** User feedback is valuable for refining and improving the retrieval process. Feedback can be used for relevance assessment, query expansion, or personalization, enhancing the system's performance over time.

Future Trends in Information Retrieval: As technology continues to advance, several trends are shaping the future of information retrieval:

- **Personalization:** IR systems are becoming more personalized, tailoring search results and recommendations based on user preferences, behavior, and context.
- **Semantic Search:** Advances in natural language processing and semantic understanding are enabling more sophisticated and context-aware search capabilities.
- **Deep Learning for IR:** Deep learning models, such as convolutional neural networks and transformers, are revolutionizing IR by capturing complex patterns and relationships in text data.
- **Multimodal Information Retrieval:** IR systems are increasingly incorporating multiple modalities like text, images, and audio to provide more comprehensive search results.
- **Explainable AI:** The interpretability of IR models is gaining importance, as users demand explanations for the retrieval process and ranking decisions.

	Information Retrieval	Information Extraction
6.	Used in many search engines – Google is the best IR system for the web.	Used in database systems to enter extracted features automatically.
7.	Typically uses a bag of words model of the source text.	Typically based on some form of semantic analysis of the source text.
8.	Mostly use the theory of information, probability, and statistics.	Emerged from research into rule-based systems.

1.8 TEXT CATEGORIZATION

Text categorization, also known as text classification or text tagging, is a crucial task in information retrieval. It involves automatically assigning predefined categories or labels to text documents based on their content. The goal of text categorization is to organize and classify large volumes of unstructured text data, making it easier for users to access relevant information and facilitating various information retrieval applications. Text categorization is widely used in web search engines, email filtering, sentiment analysis, document management systems, and more.

Key Steps in Text Categorization for Information Retrieval:

1. **Data Preprocessing:** The first step in text categorization is data preprocessing. It involves cleaning and transforming raw text data into a structured format suitable for further processing. Common preprocessing steps include removing punctuation, stop words, and special characters, converting text to lowercase, and tokenizing the text into individual words or tokens.
2. **Feature Extraction:** After preprocessing, features are extracted from the text data. Features are the representative characteristics of the documents that the classification model will use to make predictions. Common feature extraction methods include:
   - **Bag-of-Words:** Representing documents as a set of words and their frequencies in the document.
   - **TF-IDF (Term Frequency-Inverse Document Frequency):** Assigning weights to words based on their frequency in the document and rarity across the entire corpus.
   - **Word Embeddings:** Representing words as dense vectors in a continuous vector space, capturing semantic relationships between words.
3. **Training Data Preparation:** To build a text categorization model, a labeled dataset is required. This dataset consists of text documents along with their corresponding category labels. The data is divided into a training set, used to train the model, and a test set, used to evaluate the model's performance.
4. **Model Training:** Various machine learning algorithms are used to train text categorization models. Common algorithms include:
   - **Naive Bayes:** A probabilistic algorithm that calculates the probability of a document belonging to a specific category based on the frequencies of words in the document.
   - **Support Vector Machines (SVM):** A supervised learning algorithm that finds the optimal hyperplane to separate different categories.
   - **Deep Learning Models:** Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have been successfully used for text categorization tasks.
5. **Model Evaluation:** The trained model is evaluated using the test dataset to assess its performance. Common evaluation metrics include accuracy, precision, recall, F1-score, and confusion matrix. The model's ability to correctly predict the correct category for unseen documents is crucial for its effectiveness in information retrieval.
6. **Model Deployment and Application:** Once the model is trained and evaluated, it can be deployed for use in information retrieval applications. New, unseen documents can be fed into the model, and the model will predict the appropriate category label for each document.

Advantages of Text Categorization in Information Retrieval

- **Efficient Information Organization:** Text categorization helps organize vast amounts of unstructured text data into predefined categories, making it easier for users to find relevant information quickly and efficiently.
- **Automation and Scalability:** Text categorization allows for automation, reducing the need for manual categorization of documents. This scalability is essential when dealing with large document collections.
- **Personalization:** Text categorization can be used to personalize information retrieval, tailoring search

results and recommendations based on users' preferences and interests.

- **Enhanced User Experience:** By categorizing documents, users can navigate through information more effectively, leading to an improved user experience and increased satisfaction.

1.8.1 Naive Bayes (NB) Classifier

Naive Bayes is a probabilistic algorithm based on Bayes' theorem and assumes that the features (words) are conditionally independent given the class label. Despite its naive assumption, Naive Bayes often performs surprisingly well in text categorization tasks due to its simplicity and efficiency. The algorithm calculates the probability of a document belonging to each category and assigns it to the category with the highest probability.

Advantages

- Fast training and classification times, making it suitable for large datasets.
- Works well with high-dimensional data, like text, despite the independence assumption.
- Robust to irrelevant features and can handle missing data well.

Disadvantages

- The independence assumption might not hold true for some text categorization problems, leading to suboptimal performance.
- Cannot learn complex relationships between features, limiting its ability to capture subtle dependencies in the data.

1.8.2 Support Vector Machines (SVM)

SVM is a powerful supervised learning algorithm used for both classification and regression tasks. SVM finds the optimal hyperplane that best separates different categories in the feature space. In text categorization, SVM maps the features (words) to a high-dimensional space and maximizes the margin between the categories, effectively creating a decision boundary.

Advantages

- SVM can handle high-dimensional data like text efficiently.
- Effective in capturing complex decision boundaries and can work well in non-linearly separable cases using kernel functions.
- Less sensitive to irrelevant features compared to other algorithms like k-Nearest Neighbors.

Disadvantages

- SVM can be computationally expensive, especially for large datasets.
- Choosing the appropriate kernel function can be challenging and may require fine-tuning.

1.8.3 Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN)

CNNs and RNNs are deep learning models that have shown remarkable success in various natural language processing tasks, including text categorization.

1. **Convolutional Neural Networks (CNN):** CNNs are known for their ability to capture local patterns in the data using convolutional layers. In text categorization, CNNs use filters to scan across sequences of words, capturing important n-gram features.

Advantages

- CNNs can capture local patterns and relationships effectively, making them suitable for tasks like text categorization.
- Requires fewer training data compared to RNNs, making them more suitable for small and medium-sized datasets.

Disadvantages

- CNNs may not capture long-range dependencies in the text due to their local scanning approach.
- May require substantial computational resources for training and inference.

2. **Recurrent Neural Networks (RNN):** RNNs are designed to process sequences of data and can maintain memory of past information. In text categorization, RNNs process sentences or documents word-by-word, capturing contextual information.

Advantages

- RNNs can effectively capture long-range dependencies in text data due to their sequential nature.
- Suitable for tasks where the context of each word is essential, such as sentiment analysis.

Disadvantages

- RNNs can suffer from vanishing or exploding gradient problems during training, affecting learning over long sequences.
- Long training times and higher computational resources are required for large datasets.

1.8.4 Decision Trees (DTs)

- Decision trees are non-parametric models that use a tree-like structure to make decisions based on features' values. In text categorization, decision trees split the data based on the most informative words at each node, leading to a hierarchy of decisions.

Advantages

- Easy to interpret and visualize, allowing users to understand the decision-making process.
- Handles both categorical and numerical features effectively.
- Robust to irrelevant features and missing data.

Disadvantages

- Prone to overfitting, especially for deep trees, and may not generalize well to unseen data.
- May not capture complex relationships between features as effectively as other algorithms like SVM or deep learning models.

Boolean retrieval

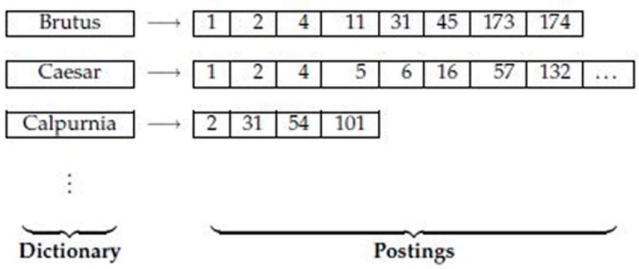
Boolean retrieval is a basic model for information retrieval that uses Boolean logic to match documents with user queries. In this model, documents are represented as sets of keywords, and queries are expressed using Boolean operators: AND, OR, and NOT.

Here's a quick overview of how it works:

1. **AND Operator**: Returns documents that contain all the keywords in the query. For example, "apple AND orange" will return documents that mention both "apple" and "orange."
2. **OR Operator**: Returns documents that contain at least one of the keywords in the query. For example, "apple OR orange" will return documents that mention either "apple" or "orange" or both.
3. **NOT Operator**: Excludes documents that contain the keyword following the NOT operator. For example, "apple NOT orange" will return documents that mention "apple" but not "orange."

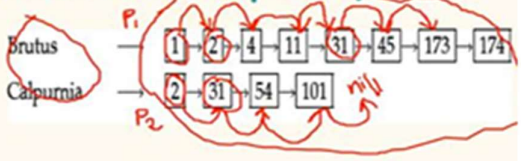
Boolean retrieval is simple and straightforward but can be limited in handling more complex queries or providing ranked results. More advanced models, like vector space models and probabilistic retrieval models, often build on or extend Boolean retrieval.

```
INTERSECT(p1, p2)
 1  answer ← ⟨ ⟩
 2  while p1 ≠ NIL and p2 ≠ NIL
 3  do if docID(p1) = docID(p2)
 4        then ADD(answer, docID(p1))
 5             p1 ← next(p1)
 6             p2 ← next(p2)
 7        else if docID(p1) < docID(p2)
 8                then p1 ← next(p1)
 9                else p2 ← next(p2)
10  return answer
```



Implementation(Brutus AND Calpurnia)

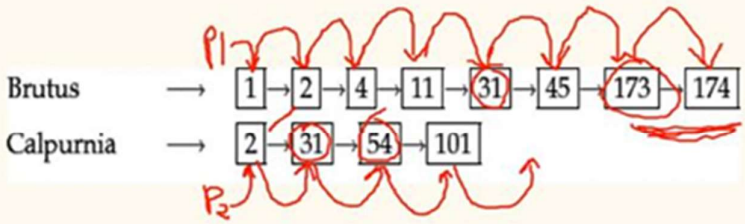
```
INTERSECT(p1, p2)
1  answer ← ⟨ ⟩
2  while p1 ≠ NIL and p2 ≠ NIL
3  do if docID(p1) = docID(p2)
4        then ADD(answer, docID(p1))
5             p1 ← next(p1)
6             p2 ← next(p2)
7        else if docID(p1) < docID(p2)
8               then p1 ← next(p1)
9               else p2 ← next(p2)
10 return answer
```



2	31			

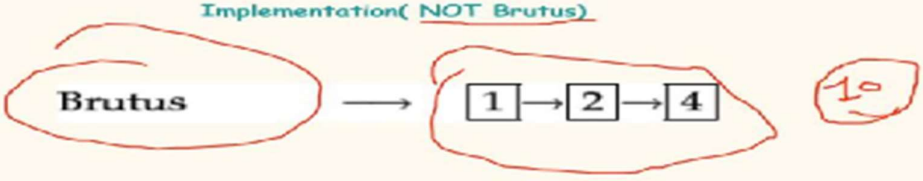
Answer : based on Intersection

Implementation(Brutus OR Calpurnia)



1	2	4	11	31	45	54	101	173	174

Implementation(NOT Brutus)



3	5	6	7	8	9	10		

Answer : based on NOT

Vector Space Model in IR

The Vector Space Model (VSM) is a popular model in Information Retrieval (IR) used to represent text documents and queries as vectors in a multi-dimensional space. This model is fundamental to many search engines and IR systems because it allows for the ranking of documents based on their relevance to a query.

Vector Model

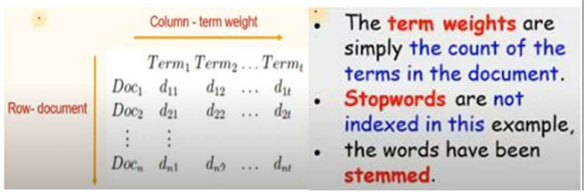
Documents and queries are assumed to be a part of a t-dimensional vector space, t is the no. if index terms (words, stems, phrases..etc)

Document Di I represented by a vector of index terms: Di=(di1, di2,..,dit)
dij represents the weight of the jth term

Query Q is represented by a vector of t weights: Q=(q1,q2,…,qt)
qj is the weight of jth term in the query

Vector Model

A document collection containing n documents can be represented as a matrix of term weights,



- The term weights are simply the count of the terms in the document.
- Stopwords are not indexed in this example,
- the words have been stemmed.

D_1 Tropical Freshwater Aquarium Fish.
D_2 Tropical Fish, Aquarium Care, Tank Setup.
D_3 Keeping Tropical Fish and Goldfish in Aquariums, and Fish Bowls.
D_4 The Tropical Tank Homepage - Tropical Fish and Aquariums.

Terms	Documents			
	D_1	D_2	D_3	D_4
aquarium	1	1	1	1
bowl	0	0	1	0
care	0	1	0	0
fish	1	0	2	1
freshwater	1	0	0	0
goldfish	0	0	1	0
homepage	0	0	0	1
keep	0	0	1	0
setup	0	1	0	0
tank	0	1	0	1
tropical	1	1	1	2

Fig. 7.1. Term-document matrix for a collection of four documents

Document D3, for example, is represented by the , vector (1, 1, 0, 2, 0, 1, 0, 1, 0, 0, 1).

Query : "tropical fish", vector representation of query be (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1).

Example : The Vector Space Model

Given this representation, documents could be ranked by computing the distance between the points representing the documents and the query.

a similarity measure is used (rather than a distance or dissimilarity measure

the documents with the highest scores are the most similar to the query

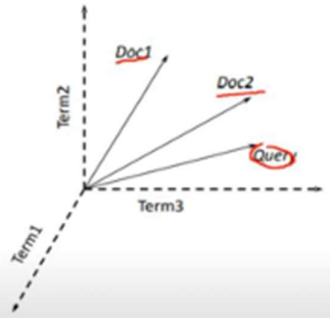


Fig. 7.2. Vector representation of documents and queries

The cosine correlation measures the cosine of the angle between the query and the document vectors.

When the vectors are normalized so that all documents and queries are represented by vectors of equal length, the cosine of the angle between two identical vectors will be 1 (the angle is zero), and for two vectors that do not share any non-zero terms, the cosine will be 0. The cosine measure is defined as,

$$Cosine(D_i, Q) = \frac{\sum_{j=1}^{t} d_{ij} \cdot q_j}{\sqrt{\sum_{j=1}^{t} d_{ij}^2 \cdot \sum_{j=1}^{t} q_j^2}}$$

As an example, consider two documents D1 = (0.5, 0.8, 0.3) and D2 = (0.9, 0.4, 0.2) indexed by three terms, where the numbers represent term **weights**.

Given the query Q = (1.5, 1.0, 0) indexed by the same terms, the cosine measures for the two documents are:

$$Cosine(D_1, Q) = \frac{(0.5 \times 1.5) + (0.8 \times 1.0)}{\sqrt{(0.5^2 + 0.8^2 + 0.3^2)(1.5^2 + 1.0^2)}} = \frac{1.55}{\sqrt{(0.98 \times 3.25)}} = 0.87$$

$$Cosine(D_2, Q) = \frac{(0.9 \times 1.5) + (0.4 \times 1.0)}{\sqrt{(0.9^2 + 0.4^2 + 0.2^2)(1.5^2 + 1.0^2)}} = \frac{1.75}{\sqrt{(1.01 \times 3.25)}} = 0.97$$

Advantages:

- Handles partial matching: documents do not need to exactly match the query terms.
- Provides a ranked list of documents based on their relevance.
- Easily extendable to handle large vocabularies and datasets.

Disadvantages:

- Assumes terms are independent, which may not always be the case.
- High-dimensional space can be computationally expensive.
- Not always effective for synonymy and polysemy (i.e., different words with similar meanings or the same word with different meanings).

Steps in Vector Space Model:

1. **Document Representation:**
   Each document and query are represented as vectors in a space where each dimension corresponds to a unique term in the corpus.
2. **Vector Creation:**
   For each document, a vector is created with components corresponding to the TF-IDF weights of terms in the document. Similarly, a query vector is created.
3. **Similarity Calculation:**
   The similarity between the query vector and each document vector is calculated using cosine similarity.
4. **Ranking:**
   Documents are ranked based on their similarity scores, and the most relevant documents are retrieved and presented to the user.

A probabilistic model in information retrieval is an approach that ranks documents based on the probability that a given document is relevant to a user's query. The fundamental idea is to treat the retrieval process as a problem of estimating the probability of relevance of a document to a query.

Probability Ranking Principle doesn't tell us how to calculate or estimate the probability of relevance.

simple probabilistic model based on treating information retrieval as a classification problem.

assumes relevance is binary, there will be two sets of documents, the relevant documents and the non-relevant documents.

the system should classify the document as relevant or non-relevant, and retrieve it if it is relevant.

P(R|D) is a conditional probability representing the probability of relevance given the representation of that document

P(NR|D) is the conditional probability of non-relevance

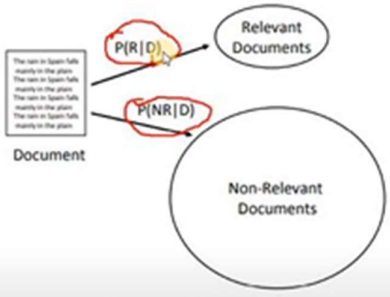


Fig. 7.3. Classifying a document as relevant or non-relevant

let's focus on P(R|D)..... P(R|D) ⟶ P(D|R)

if we had information about how often specific words occurred in the relevant set, then, given a new document, it would be relatively straightforward to calculate how likely it would be to see the combination of words in the document occurring in the relevant set.

Let's assume that the probability of the word "president" in the relevant set is 0.02, and the probability of "lincoln" is 0.03. If a new document contains the words "president" and "lincoln", we could say that the probability of observing that combination of words in the relevant set is 0.02 × 0.03 = 0.0006, assuming that the two words occur independently

So how does calculating P(D|R) get us to the probability of relevance? It turns out there is a relationship between P(R|D)and P(D|R) that is expressed by **Bayes' Rule** :

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

where P(R) is the a priori probability of relevance (in other words, how likely any document is to be relevant), and P(D)acts as a normalizing constant.

classify a document as relevant if P(D|R)P(R) > P(D|NR)P(NR). This is the same as classifying a document as relevant if:

likelihood ratio

$$\frac{P(D|R)}{P(D|NR)} > \frac{P(NR)}{P(R)}$$

If we use the **likelihood ratio** as a score, the highly ranked documents will be those that have a high likelihood of belonging to the relevant set.

Latent Semantic Indexing Model

Latent Semantic Indexing (LSI), also known as Latent Semantic Analysis (LSA), is a technique in natural language processing for analyzing relationships between a set of documents and the terms they contain. It is particularly useful for information retrieval and textual data analysis. Here's a breakdown of how it works:

Key Concepts:

1. **Term-Document Matrix**: LSI starts by constructing a term-document matrix, where each row represents a term, each column represents a document, and each cell contains the frequency of a term in a document.
2. **Singular Value Decomposition (SVD)**: LSI applies Singular Value Decomposition (SVD) to the term-document matrix. SVD is a mathematical technique that factorizes the matrix into three matrices:
   - U: Term matrix
   - Σ: Diagonal matrix of singular values
   - V^T: Document matrix

   This decomposition reduces the dimensions of the original matrix by retaining only the most significant singular values, capturing the underlying structure of the data while reducing noise.
3. **Latent Semantics**: After SVD, the reduced matrices reveal the latent structure (or "semantics") in the data. The terms and documents are now represented in a new space where similar terms and documents are closer together. This helps in identifying patterns and relationships that were not immediately apparent in the original high-dimensional space.
4. **Applications**:
   - **Information Retrieval**: LSI improves search accuracy by considering the context in which terms appear, rather than just matching keywords. This is particularly useful in search engines and recommendation systems.
   - **Document Clustering**: LSI can cluster similar documents together based on the latent topics discovered during the SVD process.
   - **Dimensionality Reduction**: It helps in reducing the dimensionality of the data while preserving the relationships between terms and documents.

How LSI Works:

- **Step 1**: Construct a term-document matrix from a collection of documents.
- **Step 2**: Apply SVD to this matrix to obtain the U, Σ, and V matrices.
- **Step 3**: Reduce the dimensionality by keeping only the top k singular values (k is a chosen number representing the latent topics).
- **Step 4**: Use the reduced matrices to compare and analyze documents and terms in the new latent space.

Advantages:

- LSI captures the underlying structure of the data, leading to better semantic understanding.
- It can handle synonymy and polysemy (multiple meanings of a word) effectively.

Disadvantages:

- LSI requires significant computational resources, especially for large datasets.
- The choice of the number of dimensions (k) can be tricky and requires domain knowledge or experimentation.
- It may not perform well with very large corpora compared to more modern techniques like word embeddings (e.g., Word2Vec, BERT).

Use Cases:

- **Search Engines**: To improve the relevance of search results by understanding the context of queries.
- **Document Summarization**: To extract the most relevant content by identifying the main topics in the text.
- **Topic Modeling**: To discover hidden topics in a collection of documents.