UNIT III - TEXT CLASSIFICATION AND INFORMATION RETRIEVAL

Naïve Bayes Classifier for Text Classification, Training and Optimization for Sentiment Analysis, Information Retrieval: Basic Concepts and Design Features, Information Retrieval Models: Classical, Non-Classical, and Alternative Models, Cluster Model, Fuzzy Model, and LSTM-Based Information, Retrieval, Word Sense Disambiguation (WSD) Methods: Supervised and Dictionary-Based Approaches.

TEXT CLASSIFICATION AND INFORMATION RETRIEVAL

1. Text Classification

Definition:

Text Classification is the task of **assigning predefined categories/labels** to a given piece of text (document, sentence, or word).

- T**ext Classification** → "What type of text is this?"
- **Information Retrieval** → "Which documents are relevant to my query?"
- Example:
    - Email → Spam / Not Spam
    - News Articles → Politics / Sports / Technology

Approaches:

1. **Rule-based**
    - Uses manually crafted rules, keywords, and regular expressions.
2. **Machine Learning-based**
    - Represent text as features (Bag-of-Words, TF-IDF, Word Embeddings).
    - Train classifiers like Naïve Bayes, SVM, Decision Trees, Neural Networks.
3. **Deep Learning-based**
    - CNN, RNN, LSTM, Transformers (BERT, GPT).
    - Automatically learn hierarchical features from raw text.

Applications:

- Spam filtering (Gmail)
- Sentiment analysis (positive/negative reviews)
- Document categorization (legal, medical, academic)
- Chatbots & Customer Support

2. Information Retrieval (IR)

Definition:

Information Retrieval is the process of **finding relevant documents** from a large collection in response to a user query.

- Example: Google search engine retrieving documents/webpages.

Components of IR System:

1. **Document Collection** (corpus)
2. **Indexing** (inverted index for fast retrieval)
3. **Query Processing** (user input text is tokenized, parsed, expanded)
4. **Retrieval Models** (ranking algorithms)
5. **Evaluation** (precision, recall, F1-score, MAP).

Popular IR Models:

- **Boolean Model** → exact matching using AND, OR, NOT.
- **Vector Space Model (VSM)** → uses cosine similarity with TF-IDF.
- **Probabilistic Models** → BM25, Language Models.
- **Neural IR** → BERT-based semantic search.

Applications:

- Web Search (Google, Bing)
- Recommender Systems (Netflix, Amazon)
- Question Answering (Quora, StackOverflow)
- Legal & Healthcare document retrieval

3. Relationship Between Text Classification & IR

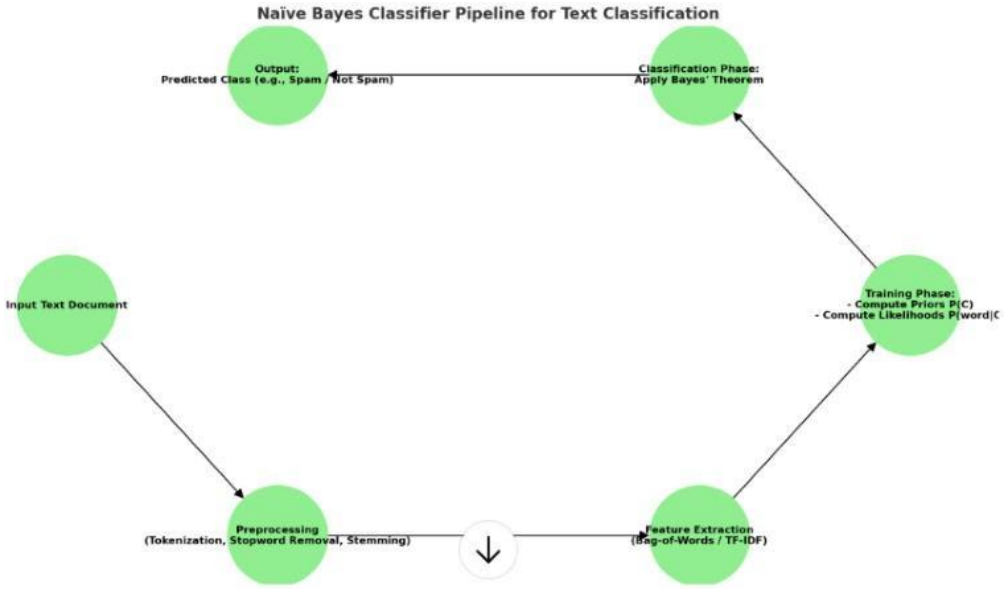
Text Classification	Information Retrieval
Assigns a label/category to text	Finds & ranks documents relevant to a query
Supervised learning task	Search & ranking problem
Example: Classify an email as spam	Example: Retrieve top 10 webpages about spam detection

NAÏVE BAYES CLASSIFIER FOR TEXT CLASSIFICATION

1. What is Naïve Bayes?

Naïve Bayes is a baseline classifier for text (spam filtering, sentiment analysis, topic categorization) — simple yet surprisingly effective in practice.

- A **probabilistic classifier** based on **Bayes' Theorem**.
- Assumes **conditional independence** between features (hence "naïve").
- Widely used for **text classification** because it works well with **high-dimensional data** (e.g., thousands of words).



🔹 2. Bayes' Theorem

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

Where:

- $P(C|X)$ → Probability of class C given text X (posterior).
- $P(X|C)$ → Likelihood of seeing text X given class C.
- $P(C)$ → Prior probability of class C.
- $P(X)$ → Evidence (constant for all classes).

Classifier chooses the class with the **highest posterior probability**.

$$\hat{C} = \arg\max_{C} P(C) \prod_{i=1}^{n} P(x_i|C)$$

🔹 3. Types of Naïve Bayes

1. **Multinomial Naïve Bayes** → best for word counts, Bag-of-Words, TF-IDF.
2. **Bernoulli Naïve Bayes** → binary features (word present/absent).
3. **Gaussian Naïve Bayes** → continuous features (rare in text).

🔹 4. Example: Spam Detection

Suppose we classify emails into **Spam (S)** or **Not Spam (N)**.

- Training data:
  - Spam emails often contain words like "free", "offer", "win".
  - Not spam emails often contain words like "meeting", "project", "report".

📥 New email: `"free offer today"`

Steps:

1. Compute **priors**:

$$P(Spam),\ P(NotSpam)$$

2. Compute **likelihoods** for each word:
   - $P(free|Spam), P(offer|Spam), P(today|Spam)$
   - $P(free|NotSpam), P(offer|NotSpam), P(today|NotSpam)$
3. Apply Bayes rule:

$$P(Spam|email) \propto P(Spam) \times P(free|Spam) \times P(offer|Spam) \times P(today|Spam)$$

4. Compare with Not Spam probability → Choose higher one.

5. Advantages

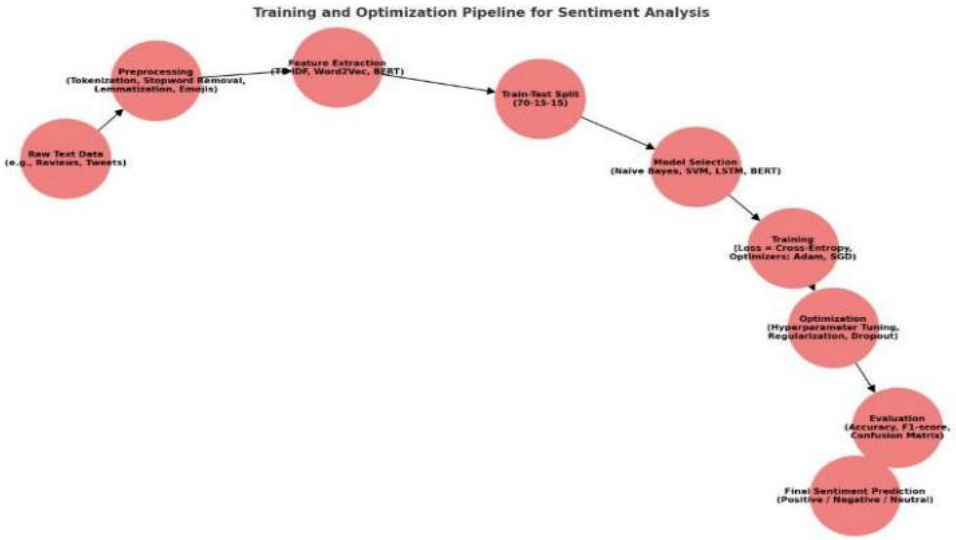
- Simple and fast.
- Works well for **large vocabulary, sparse data**.
- Requires less training data.

6. Limitations

- Assumes independence of words (not always true).
- Can perform poorly if features are highly correlated.
- Doesn't capture semantic meaning of words.

TRAINING AND OPTIMIZATION FOR SENTIMENT ANALYSIS

Training sentiment analysis involves preparing clean data, selecting an appropriate model, and optimizing it with proper hyperparameter tuning, feature engineering, and regularization to achieve strong generalization.



1. Training Pipeline

Step 1: Data Preparation

- Collect labeled dataset → e.g., tweets, reviews, or news headlines with *Positive, Negative, Neutral* tags.
- Preprocess:
  - Tokenization
  - Stopword removal
  - Lemmatization/Stemming
  - Handling emojis/emoticons (e.g., 👍 = positive)
  - Converting text → numerical features (Bag-of-Words, TF-IDF, Word2Vec, BERT embeddings).

Step 2: Model Selection

- **Classical ML Models:** Naïve Bayes, Logistic Regression, SVM.
- **Deep Learning Models:** CNNs, RNNs (LSTM, GRU), Transformers (BERT, RoBERTa).

Step 3: Training

- Split dataset → Train (70%), Validation (15%), Test (15%).
- Feed text features into the model.
- Model learns to map inputs → sentiment labels.
- Loss functions:

- **Cross-Entropy Loss** (for classification).
- Optimized using algorithms like **SGD, Adam, RMSprop**.

2. Optimization Techniques

A. Feature Engineering

- Use **TF-IDF** instead of raw word counts.
- Use **n-grams** (bigrams, trigrams) to capture context (e.g., "not good" vs "good").
- Pretrained embeddings (Word2Vec, GloVe, BERT) improve semantic understanding.

B. Hyperparameter Tuning

- Adjust **learning rate, batch size, regularization, dropout rate**.
- Use **Grid Search, Random Search, or Bayesian Optimization**.
- Early stopping to prevent overfitting.

C. Regularization

- **Dropout** in neural networks.
- **L2 regularization** (weight decay).
- Ensures model generalizes well on unseen data.

D. Data Augmentation

- Synonym replacement, back translation, paraphrasing.
- Helps when labeled data is small.

E. Class Imbalance Handling

- If dataset has more positives than negatives:
  - Oversample minority class (SMOTE).
  - Undersample majority class.
  - Class-weight adjustment in loss function.

3. Evaluation Metrics

- **Accuracy** (not enough if imbalanced).
- **Precision, Recall, F1-score** → better indicators.
- **Confusion Matrix** to analyze errors.

4. Example Workflow

1. Input: "This movie was amazing, I loved it!"
2. Preprocessing: tokens = [“movie”, “amazing”, “loved”]
3. Features: TF-IDF vector or BERT embedding.
4. Training: Model learns that “amazing”, “loved” → Positive class.
5. Optimization: Adjust learning rate + dropout until best validation F1-score.
6. Output: **Positive Sentiment** 🟩

INFORMATION RETRIEVAL: BASIC CONCEPTS AND DESIGN FEATURES

1. Basic Concepts of Information Retrieval

Information Retrieval is the process of **storing, searching, and retrieving information** from large collections of unstructured or semi-structured data, such as documents, text, or multimedia. Unlike databases (which work with structured data), IR deals with **natural language text**.

🔹 **Key Concepts:**

- **Document** – A unit of information (article, webpage, email, report, etc.).
- **Collection/Corpus** – The set of documents to be searched.
- **Query** – The information need expressed by the user (often in keywords or natural language).
- **Indexing** – Creating efficient data structures (e.g., inverted index) to enable fast retrieval.
- **Matching/Retrieval** – Comparing queries with documents using a ranking model.
- **Relevance** – The degree to which a retrieved document satisfies the user’s information need.

– Example: Searching *"best machine learning books"* in Google.

- The query = your keywords
- Corpus = billions of webpages
- Retrieval = Google’s ranking algorithms returning relevant links

2. IR vs. Database Retrieval

- **Database Retrieval** → Structured data, exact matching (SQL queries).
- **Information Retrieval** → Unstructured/semi-structured data, approximate matching, ranking-based results.

3. Design Features of an IR System

When building an IR system (like Google Search, PubMed, or a digital library), key design features include:

(a) Document Representation

- Documents must be represented in a way that computers can process.
- Common model → **Vector Space Model** (each document is a vector of terms/weights, often using **TF-IDF**).

(b) Indexing

- **Inverted Index**: Maps each term → list of documents containing it.
- Optimized for **fast keyword-based search**.

(c) Query Processing

- Handling synonyms, stemming, lemmatization, stop-word removal.
- Support for **Boolean queries (AND, OR, NOT)** or **natural language queries**.

(d) Retrieval Models

- **Boolean Model** → Exact matches (AND/OR logic).
- **Vector Space Model** → Measures similarity (e.g., cosine similarity).
- **Probabilistic Models** → Estimate probability of relevance (e.g., BM25).
- **Neural IR Models** → Use embeddings and deep learning (e.g., BERT).

(e) Ranking & Relevance Feedback

- Results ranked by a **scoring function**.
- **Relevance feedback**: Users mark documents as relevant/non-relevant → system refines ranking.

(f) User Interface & Interaction

- Query box, autocomplete, suggestions, snippets, highlighting of keywords, etc.

(g) Performance Metrics

- **Precision** = fraction of retrieved docs that are relevant.
- **Recall** = fraction of relevant docs that were retrieved.
- **F1-score, MAP, nDCG** used in advanced systems.

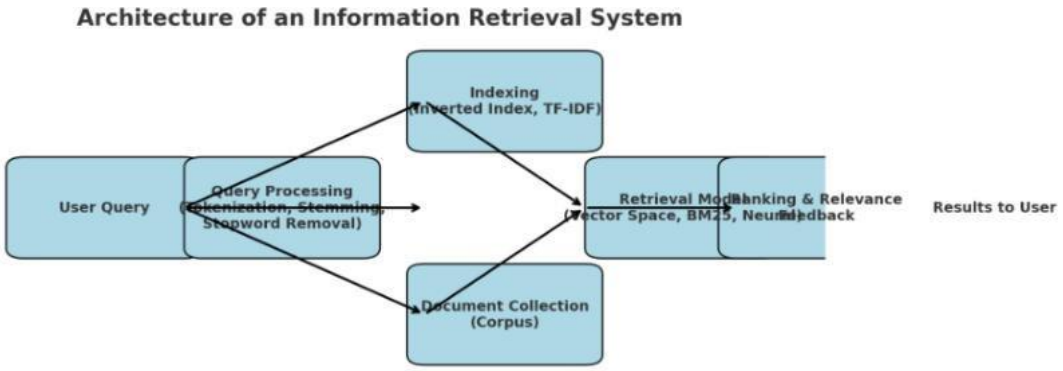


diagram of an Information Retrieval (IR) System Architecture showing the key design features:

- **User Query → Query Processing** (tokenization, stemming, stopword removal)
- **Indexing** (inverted index, TF-IDF, etc.)
- **Document Collection (Corpus)**
- **Retrieval Models** (vector space, BM25, neural models)
- **Ranking & Relevance Feedback**
- **Results back to User**

INFORMATION RETRIEVAL MODELS: CLASSICAL

Classical IR models are the earliest approaches to retrieving relevant documents from a collection based on a user's query. They rely on mathematical and statistical principles to model documents, queries, and their relationships.

1. Boolean Model

- **Concept**:
  Uses **set theory** and **Boolean logic** (AND, OR, NOT) to match queries with documents.
- **Representation**:
  - Documents and queries are represented as sets of terms.
  - Example: Query = *AI AND Healthcare* → retrieves only documents containing both terms.
- **Advantages**:
  - Simple and easy to implement.
  - Provides precise matching (all or nothing).
- **Disadvantages**:

- No ranking of results (all matching documents are equally relevant).
- Too rigid (slight mismatch in keywords leads to no results).

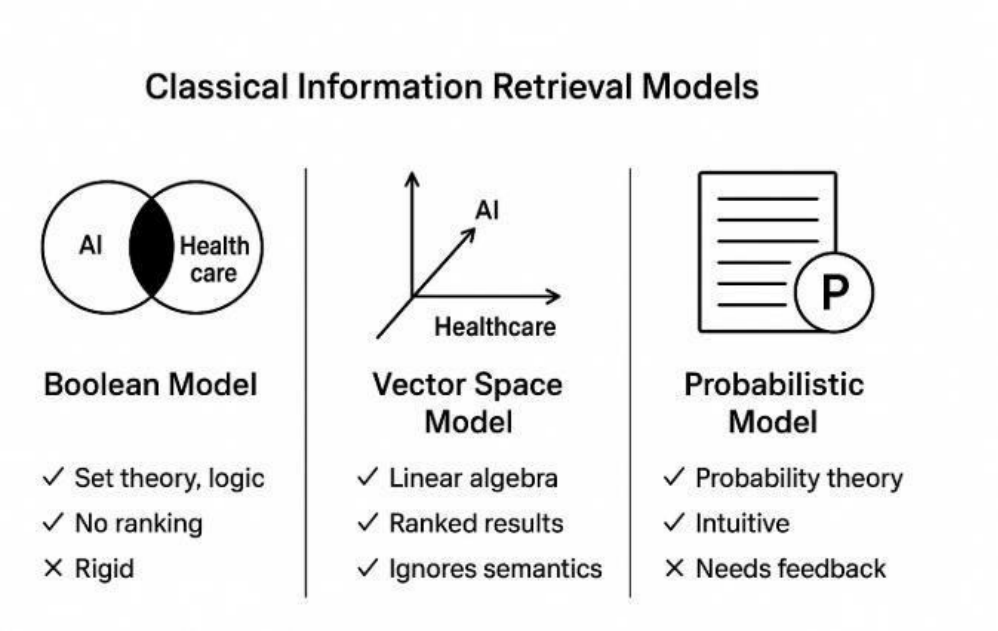
2. Vector Space Model (VSM)

- **Concept**:
  Represents documents and queries as vectors in a **multi-dimensional term space**.
- **Similarity**:
  - Uses **cosine similarity** between query and document vectors.
  - Weights terms using **TF-IDF** (Term Frequency–Inverse Document Frequency).
- **Advantages**:
  - Provides ranking of documents by relevance.
  - Can handle partial matches.
- **Disadvantages**:
  - Assumes term independence (ignores semantic relationships).
  - High-dimensional space is computationally expensive.

3. Probabilistic Model

- **Concept**:
  Estimates the probability that a document is relevant to a given query.
- **Principle**:
  Documents are ranked by their probability of relevance.
- **Example**:
  Binary Independence Model (BIM) assumes each term is either present or absent and independent.
- **Advantages**:
  - Intuitive probabilistic interpretation.
  - Forms the basis for modern ranking models (e.g., BM25).
- **Disadvantages**:
  - Assumes independence between terms.
  - Requires initial relevance feedback to refine probabilities.

Model	Basis	Ranking	Pros	Cons
Boolean	Set theory, logic	No	Simple, precise	No ranking, rigid
Vector Space Model	Linear algebra	Yes	Ranked results, partial match	Ignores semantics, high-dimensional
Probabilistic Model	Probability theory	Yes	Intuitive, modern basis	Needs assumptions, feedback



INFORMATION RETRIEVAL MODELS NON-CLASSICAL

Non-Classical Information Retrieval (IR) models were developed to overcome the limitations of classical models (Boolean, Vector Space, Probabilistic). They aim to handle semantics, uncertainty, context, and linguistic structure more effectively.

- **Fuzzy & Extended Boolean:** Handle vagueness and flexibility.
- **LSI & PLSA:** Capture latent semantics and topics.
- **Neural & Language Models:** Power modern search engines with context, probability, and deep learning.

Here are the main **Non-Classical IR Models**:

1. Fuzzy Set Model

- **Basis:** Fuzzy logic and set theory.

- **Key Idea:** Documents and queries are not treated as exact matches but as having degrees of relevance.
- **Features:**
  - Handles vagueness in user queries.
  - Provides graded (soft) matching instead of rigid matching.
  - Useful when query terms are imprecise.

2. Extended Boolean Model

- **Basis:** Combines Boolean logic with Vector Space concepts.
- **Key Idea:** Replaces strict AND/OR logic with *soft decision functions*.
- **Features:**
  - Allows partial matching between documents and queries.
  - Improves ranking over classical Boolean model.
  - More flexible in handling user queries.

3. Latent Semantic Indexing (LSI) Model

- **Basis:** Linear algebra (Singular Value Decomposition, SVD).
- **Key Idea:** Captures hidden semantic relationships between words and documents.
- **Features:**
  - Handles synonymy (different words with similar meaning).
  - Reduces dimensionality of term-document matrix.
  - Improves retrieval accuracy for concept-based queries.

4. Neural Network Model

- **Basis:** Machine learning using neural networks.
- **Key Idea:** Learns patterns of relevance between queries and documents.
- **Features:**
  - Models complex, non-linear relationships.
  - Can incorporate semantics and context.
  - Used in modern search engines with deep learning.

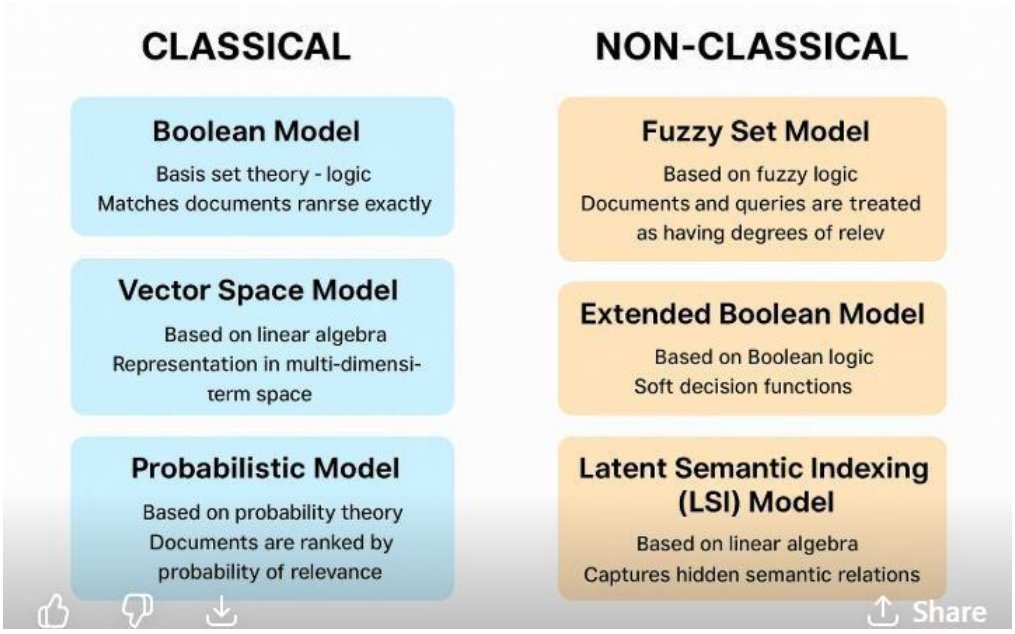
5. Probabilistic Latent Semantic Analysis (PLSA)

- **Basis:** Statistical modeling, probability distributions.

- **Key Idea:** Each document is represented as a mixture of latent topics.
- **Features:**
  - Extracts topics automatically.
  - Useful in text mining and clustering.
  - Predecessor of LDA (Latent Dirichlet Allocation).

6. Language Models for IR

- **Basis:** Statistical language modeling.
- **Key Idea:** Each document is treated as a probabilistic language model. The query likelihood is estimated for ranking.
- **Features:**
  - Very effective in modern IR.
  - Handles term dependencies better.
  - Forms the basis of many current search engines.



INFORMATION RETRIEVAL MODELS ALTERNATIVE MODELS

Here are some **Alternative Information Retrieval (IR) Models** beyond classical and non-classical approaches:

1. Inference Network Model

- Uses a probabilistic graphical model.

- Represents dependencies between queries, documents, and terms.
- Computes the probability of a document being relevant using Bayesian inference.

2. Neural IR Models

- Based on deep learning techniques.
- Uses embeddings (e.g., Word2Vec, BERT) to represent queries and documents.
- Can capture semantic meaning beyond keyword matching.
- Includes models like DSSM (Deep Structured Semantic Model), ColBERT, and Transformer-based retrieval.

3. Language Models for IR

- Treats each document as a probabilistic language model.
- Query likelihood model: ranks documents by the probability of generating the query from the document model.
- Popular example: LMIR with smoothing methods (Jelinek-Mercer, Dirichlet).

4. Cluster-Based Retrieval Models

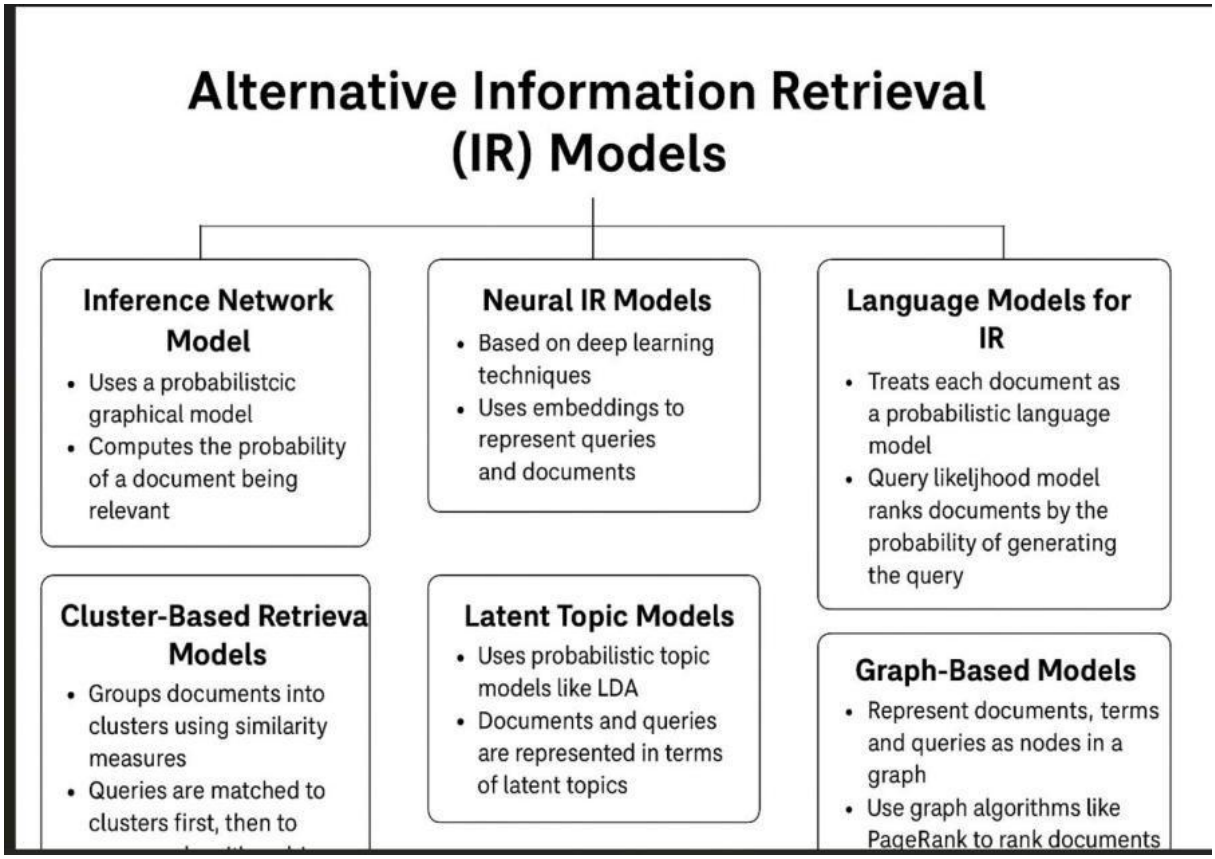
- Groups documents into clusters using similarity measures.
- Queries are matched to clusters first, then to documents within clusters.
- Improves efficiency and sometimes effectiveness.

5. Latent Topic Models

- Uses probabilistic topic models like Latent Dirichlet Allocation (LDA).
- Documents and queries are represented in terms of latent topics.
- Helps in semantic matching by bridging vocabulary gaps.

6. Graph-Based Models

- Represent documents, terms, and queries as nodes in a graph.
- Use graph algorithms like PageRank to rank documents based on structural relationships.
- Example: Hyperlink-Induced Topic Search (HITS), PageRank-based IR.



CLUSTER MODEL

The **Cluster Model** in Information Retrieval (IR) is a **non-classical retrieval model** that improves search efficiency and effectiveness by grouping similar documents together. Instead of matching a query against the entire collection, it matches the query against clusters of documents first.

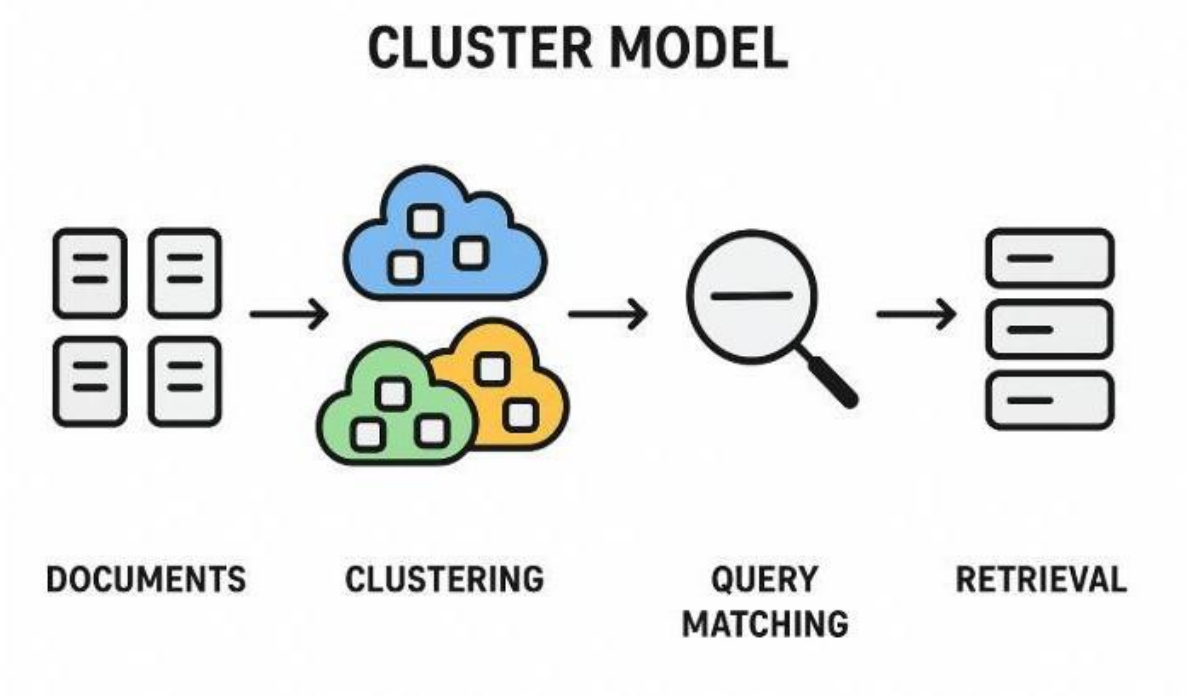
Key Concepts of Cluster Model:

1. **Clustering Documents**
   - Documents are grouped into clusters based on similarity (e.g., cosine similarity, Jaccard similarity).
   - Clustering methods include **k-means, hierarchical clustering, or density-based clustering**.
2. **Query Matching**
   - When a query is submitted, it is first compared to **cluster centroids** (representative summaries of clusters).
   - The most relevant cluster(s) are selected.
   - Finally, documents within those clusters are ranked and retrieved.

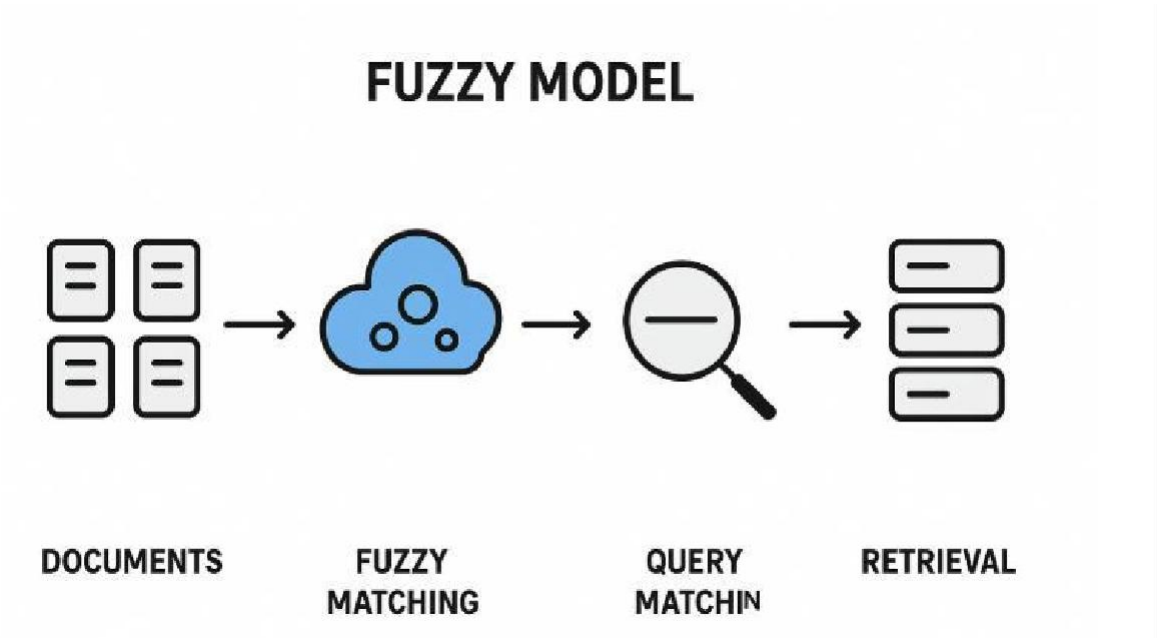
3. **Advantages**
    - Reduces search space → faster retrieval.
    - Groups similar documents → improves relevance.
    - Useful for exploratory search (e.g., browsing topics).
4. **Limitations**
    - Clustering is computationally expensive for large datasets.
    - Cluster quality strongly affects retrieval performance.
    - Not all queries benefit equally from clustering.

Example Use Case:

- A digital library groups research papers into clusters (e.g., AI, ML, NLP).
- A user searching "neural networks" will first be directed to the ML/NLP clusters, avoiding irrelevant clusters (like databases).



FUZZY MODEL



The **Fuzzy Model** in Information Retrieval (IR) is a **non-classical model** that uses **fuzzy set theory** to represent documents and queries. Unlike Boolean models, which use rigid "yes/no" matching, the Fuzzy Model allows **partial matching** with degrees of relevance.

Key Features of the Fuzzy Model:

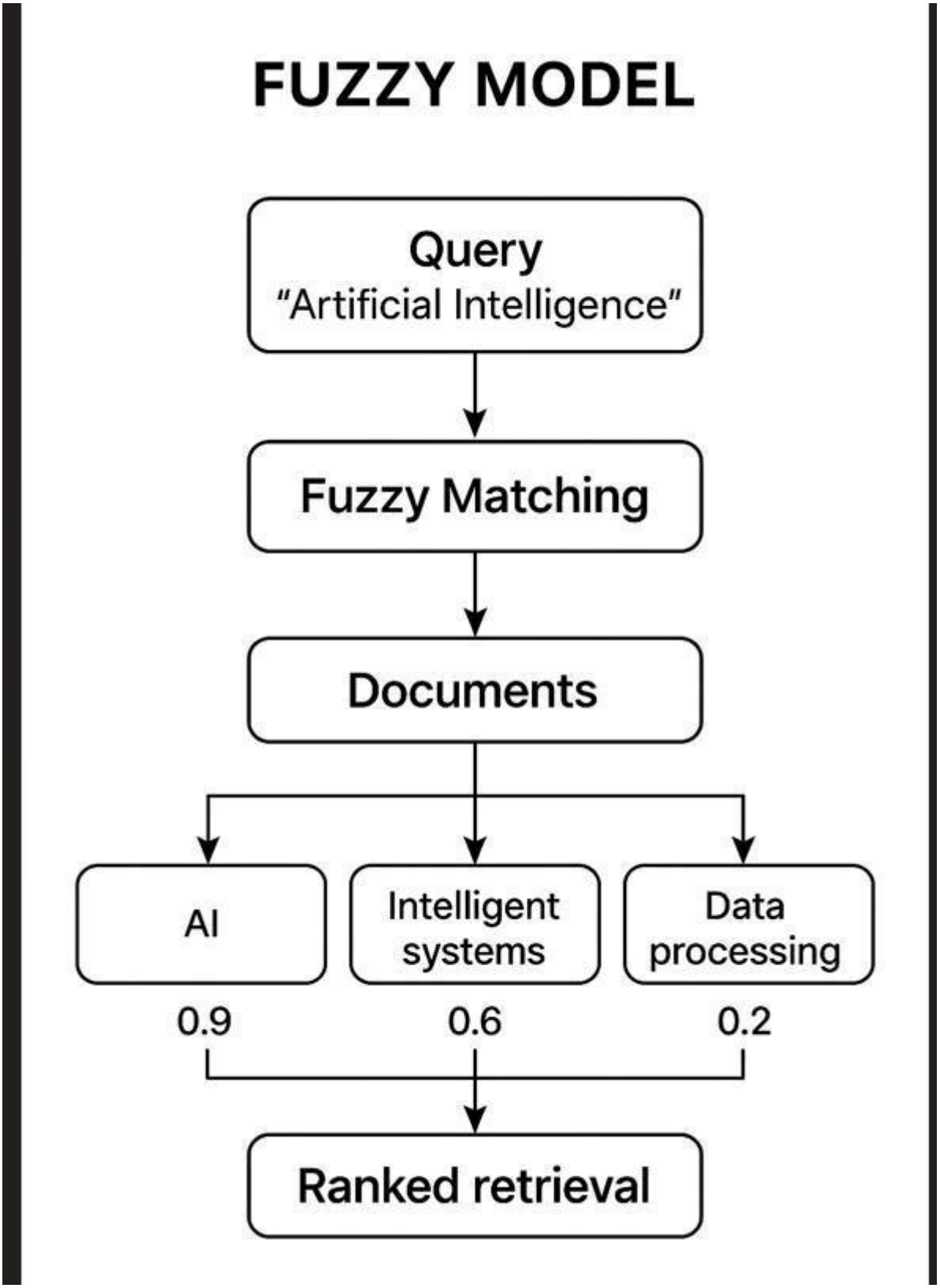
1. **Fuzzy Sets**
   - Documents and queries are represented as fuzzy sets of terms.
   - Each term has a **membership value** (between 0 and 1), indicating its importance in the document or query.
2. **Similarity Measurement**
   - Relevance between a document and a query is measured using **fuzzy similarity functions**.
   - Example:
     - In Boolean → "machine learning" must appear exactly.
     - In Fuzzy → If "learning algorithms" appears, it is still considered **partially relevant**.
3. **Advantages**
   - Captures uncertainty and vagueness in human language.
   - Provides ranking of results rather than strict true/false output.
   - Handles synonymy, polysemy, and partial matches better.

4. **Limitations**
    - More computationally complex than Boolean models.
    - Membership values and similarity functions must be well-defined, which can be challenging.

Example

- Query: *"Artificial Intelligence"*
- Document A: contains "AI" → high fuzzy match (0.9).
- Document B: contains "intelligent systems" → partial fuzzy match (0.6).
- Document C: contains "data processing" → low fuzzy match (0.2).

Thus, the fuzzy model retrieves A > B > C.



LSTM-BASED INFORMATION RETRIEVAL

The **LSTM-Based Information Retrieval model** applies **deep learning**—specifically **Long Short-Term Memory (LSTM) networks**—to improve how queries and documents are matched. Unlike classical IR models, LSTM-based models capture **sequential and contextual dependencies in text**, making them powerful for natural language understanding.

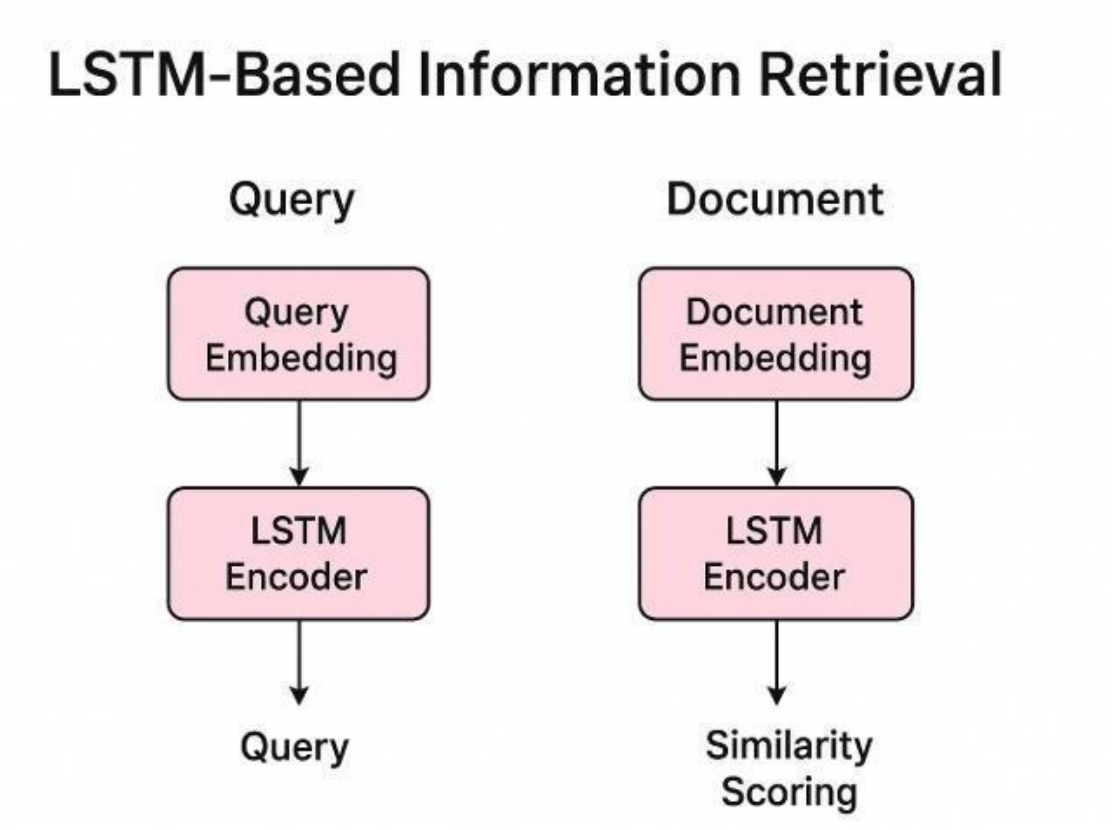
ey Concepts

1. **Why LSTM for IR?**
    - Traditional models (Boolean, Vector Space, BM25) treat queries and documents as bags of words, ignoring word order.
    - LSTMs handle **sequential data** and can capture **long-term dependencies**, which is critical in understanding natural language queries.
2. **Architecture**
    - **Input Layer**: Queries and documents are represented as word embeddings (e.g., Word2Vec, GloVe, BERT embeddings).
    - **LSTM Encoder**: Processes the sequence of embeddings and captures contextual meaning.
    - **Representation Layer**: Produces dense vector representations of queries and documents.
    - **Similarity Scoring**: Cosine similarity, dot product, or a learned function compares query and document representations.
    - **Ranking Layer**: Ranks documents based on similarity scores.
3. **Training**
    - LSTM models are trained on large-scale query-document pairs.
    - Objective: maximize the score of relevant documents and minimize that of irrelevant ones.
    - Loss functions: **contrastive loss, hinge loss, or cross-entropy loss**.
4. **Advantages**
    - Captures word order and semantic dependencies.
    - Learns query-document relationships automatically.
    - Handles synonyms, paraphrases, and context much better than classical models.
5. **Limitations**
    - Requires **large labeled datasets**.

- Training is **computationally expensive**.
- LSTMs are being replaced in practice by **Transformers (BERT, GPT, etc.)** which perform even better.

Example Flow

- Query: *"best deep learning books"*
- Document A: "Top resources for deep learning study" → LSTM captures context → High relevance score.
- Document B: "Library contains many programming books" → Lower relevance score.



WORD SENSE DISAMBIGUATION (WSD)

Word Sense Disambiguation (WSD) is the process of **determining the correct meaning (sense) of a word** in a given context when the word has multiple possible interpretations.
Example:

- *"He went to the **bank** to deposit money."* → **bank = financial institution**
- *"He sat on the **bank** of the river."* → **bank = river shore**

Approaches to WSD

1. **Knowledge-based Methods**

- Use lexical resources like **WordNet**, dictionaries, and thesauri.
- Algorithms:
  - *Lesk Algorithm*: Chooses the sense with the most word overlap between dictionary definitions and context.
  - *Semantic Similarity*: Selects the sense closest in meaning to surrounding words.

2. **Supervised Learning Methods**
   - Treat WSD as a classification task.
   - Steps:

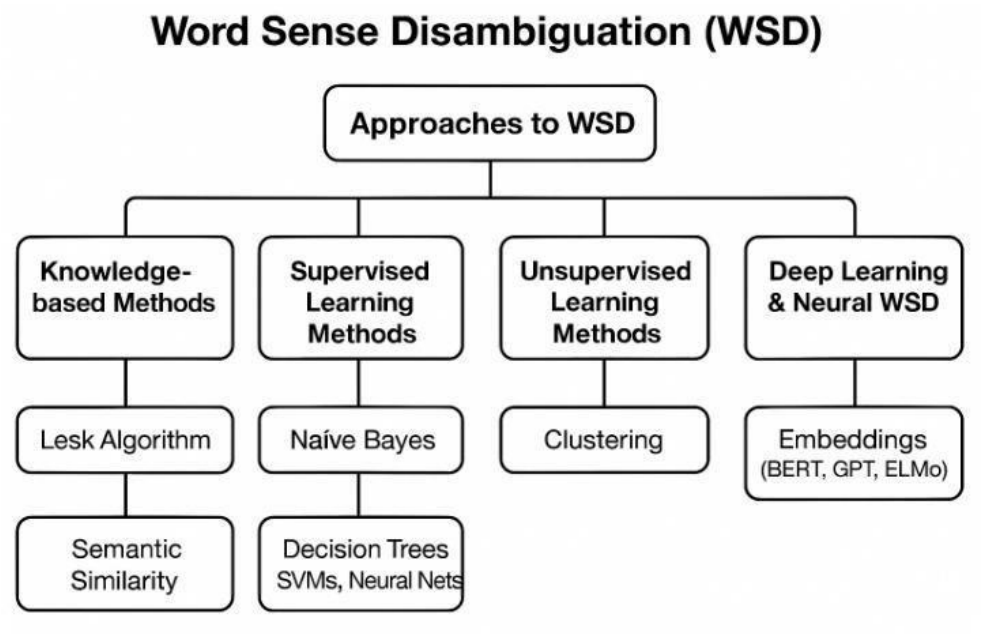
1. Extract features from the word's context (neighboring words, POS tags, etc.).
2. Train classifiers like Naïve Bayes, Decision Trees, SVMs, Neural Networks.
   - Limitation: Requires **large annotated datasets**.

3. **Unsupervised Learning Methods**
   - Cluster word occurrences into groups based on context similarity.
   - No labeled data needed.
   - Example: Using embeddings + clustering to group word senses.

4. **Deep Learning & Neural WSD**
   - Use embeddings (Word2Vec, GloVe, BERT).
   - Contextual embeddings (BERT, GPT, ELMo) **disambiguate senses dynamically** depending on context.
   - Example: *BERT automatically distinguishes "bank" in finance vs. river contexts.*

Applications of WSD

- **Machine Translation** (choosing correct word in target language).
- **Information Retrieval** (retrieving documents with correct sense).
- **Text Mining & NLP** (better semantic understanding).
- **Chatbots/Assistants** (improving intent understanding).



WORD SENSE DISAMBIGUATION (WSD) METHODS: SUPERVISED APPROACHES

Supervised WSD treats word sense disambiguation as a **classification problem**, where each occurrence of an ambiguous word is assigned to one of its possible senses.

Steps in Supervised WSD

1. **Data Preparation**
    - Requires a **sense-annotated corpus** (e.g., SemCor, OntoNotes).
    - Each ambiguous word in training data is labeled with the correct sense.
2. **Feature Extraction**
    - **Local Features**: Neighboring words, collocations.
    - **Global Features**: Sentence/paragraph-level words.
    - **Syntactic Features**: POS tags, dependency relations.
    - **Semantic Features**: Word embeddings (Word2Vec, GloVe, BERT).
3. **Model Training**
    - Train a classifier to predict the sense using extracted features.
    - Popular classifiers:
        - **Naïve Bayes**
        - **Decision Trees**

- **Support Vector Machines (SVMs)**
- **k-Nearest Neighbors (kNN)**
- **Neural Networks** (MLPs, CNNs, RNNs)

4. **Prediction**
    - Apply the trained model to unseen sentences to predict the correct sense of ambiguous words.

Example

Sentence: *"The fisherman sat on the **bank**."*

- Context features: {fisherman, sat, river}
- Classifier predicts: **bank = river shore**

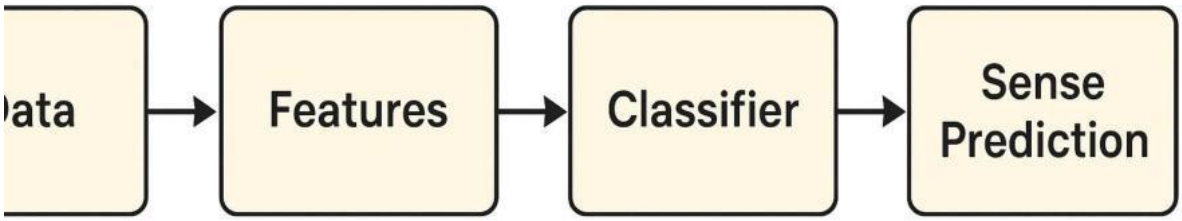
Advantages

- High accuracy when **large annotated datasets** are available.
- Can adapt to **domain-specific senses**.

Limitations

- **Expensive** to build sense-annotated corpora.
- Performance drops with **low-resource languages**.
- Classifiers may **overfit** on small datasets.

SUPERVISED WSD



WORD SENSE DISAMBIGUATION (WSD) – DICTIONARY-BASED APPROACHES

Dictionary-based (or **knowledge-based**) approaches exploit **lexical resources** such as **WordNet, Oxford Dictionary, or machine-readable dictionaries** to determine the correct sense of a word in context.
They do **not require annotated corpora** → useful for low-resource languages.

Major Dictionary-Based WSD Methods

1. Lesk Algorithm (1986)

- **Idea:** The correct sense of a word has the **highest overlap** between its dictionary definition (gloss) and the context words.
- **Steps:**
    1. Take all possible senses of the target word from the dictionary.
    2. For each sense, compare its gloss with the glosses of context words.
    3. The sense with maximum word overlap is chosen.
- **Example:**
  *"I went to the **bank** to deposit money."*
    - Gloss overlap with *money, deposit* → sense = *financial institution*.

2. Extended Lesk Algorithm

- Extends Lesk by considering **glosses of related words** (synonyms, hypernyms, hyponyms).
- Provides richer context and improves accuracy.

3. Semantic Similarity / Relatedness Methods

- Uses WordNet to compute **semantic distance** between senses.
- Example measures:
  - Path-based (shortest path in WordNet hierarchy).
  - Information content–based (Resnik, Lin, Jiang-Conrath measures).
- Chooses the sense most semantically similar to context words.

4. Selectional Preferences

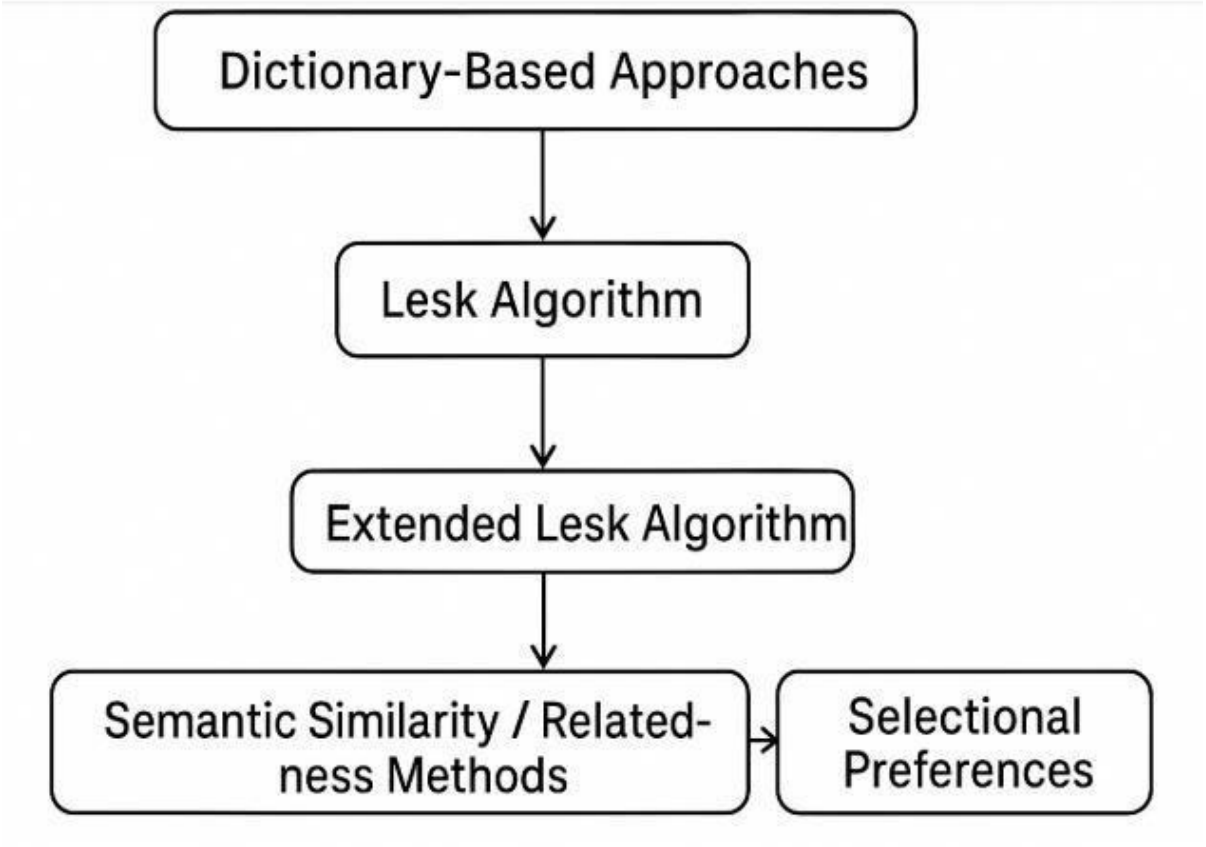
- Uses dictionary-based semantic restrictions.
- Example: *"He drank a **glass** of water."*
  - Verb *drink* expects a **liquid object** → sense of *glass* = container, not material.

Advantages

- No need for large labeled corpora.
- Effective for languages with **rich dictionaries** (e.g., English with WordNet).
- Easy to implement.

Limitations

- **Dictionary coverage problem** (not all senses well-defined).
- Struggles with **rare words or domain-specific terminology**.
- Performance often lower than supervised ML methods.


Dictionary-Based Approaches
Lesk Algorithm
Extended Lesk Algorithm
Semantic Similarity / Related-ness Methods
Selectional Preferences