

MODULE – 4

Information Retrieval & Lexical Resources

Information Retrieval: Design Features of Information Retrieval Systems, Information Retrieval Models - Classical, Non-classical, Alternative Models of Information Retrieval -Cluster model, Fuzzy model, LSTM model, Major Issues in Information Retrieval.

Lexical Resources: WordNet, FrameNet, Stemmers, Parts-of-Speech Tagger, Research Corpora.

Textbook 1: Ch. 9, Ch. 12.

Overview:

The huge amount of information stored in electronic form, has placed heavy demands on information retrieval systems. This has made information retrieval an important research area.

4.1 Introduction

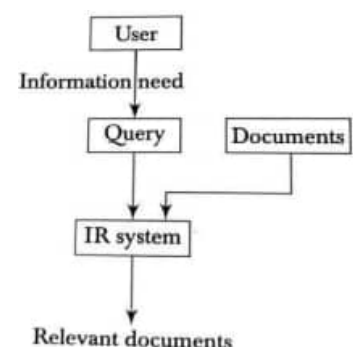
- Information retrieval (IR) deals with the organization, storage, retrieval, and evaluation of information relevant to a user's query.
- A user in need of information formulates a request in the form of a query written in a natural language.
- The retrieval system responds by retrieving the document that seems relevant to the query.

*“An information retrieval system does not inform (i.e., change the knowledge of) the user on the subject of their inquiry. It **merely informs on the existence** (or non-existence) and **whereabouts of documents** relating to their request”.*

- This chapter focuses on **text document retrieval**, excluding question answering and data retrieval systems, which handle precise queries for specific data or answers.
- In contrast, IR systems deal with vague, imprecise queries and aim to retrieve relevant documents rather than exact answers.

4.2 Design Features of Information Retrieval Systems

- It begins with the user's information need.
- Based on this need, he/she formulates a query.
- The IR system returns documents that seem relevant to the query.
- The retrieval is performed by matching the query representation with document representation.
- In information retrieval, documents are not represented by their full text but by a set of **index terms or keywords**, which can be single words or phrases, extracted automatically or manually.
- **Indexing**, provides a *logical view of the document* and helps reduce computational costs.



- A commonly used data structure is the **inverted index**, which *maps keywords to the documents* they appear in.
- To further reduce the number of keywords, **text operations** such as **stop word elimination** (removing common functional words) and **stemming** (reducing words to their root form) are used.
- **Zipf's law** can be applied to *reduce the index size* by filtering out extremely frequent or rare terms.
- Since not all terms are equally relevant, **term weighting** assigns numerical values to keywords to reflect their importance.
- Choosing appropriate index terms and weights is a complex task, and several **term-weighting schemes** have been developed to address this challenge.

4.2.1 Indexing

IR system can access a document to decide its relevance to a query. Large collection of documents, this technique poses practical problems. A collection of raw documents is usually transformed into an *easily accessible representation*. This process is known as **indexing**.

- **Indexing** involves *identifying descriptive terms* (keywords) that capture a document's content and distinguish it from others.
- **Effective descriptors** aid in both content representation and document discrimination.
- **Luhn (1957, 1958)** introduced automatic indexing based on word frequency, suggesting that terms with **middle-range frequency** are the most effective discriminators.
- **Indexing represents text**—both documents and queries—using selected terms that reflect the original content meaningfully.

The word term can be a single word or multi-word phrases.

For example, the sentence, *Design features of information retrieval systems*, can be represented as follows:

Design, features, information, retrieval, systems

It can also be represented by the set of terms:

Design, features, information retrieval, information retrieval systems

- **Multi-word terms** can be extracted using methods like n-grams, POS tagging, NLP, or manual crafting.
- **POS tagging** aids in resolving word sense ambiguity using contextual grammar.
- **Statistical methods** (e.g., frequent word pairs) are efficient but struggle with word order and structural variations, which **syntactic methods** handle better.
- **TREC approach**: Treats any adjacent non-stop word pair as a phrase, retaining only those that occur in a minimum number (e.g., 25) of documents.
- **NLP** is also used for identifying proper nouns and **normalizing noun phrases** to unify variations (e.g., "President Kalam" and "President of India").

- **Phrase normalization** reduces structural differences in similar expressions (e.g., "text categorization," "categorization of text," and "categorize text" → "text categorize").

4.2.2 Eliminating Stop Words

- **Stop words** are high-frequency, low-semantic-value words (e.g., articles, prepositions) that are commonly removed during lexical processing.
- They play **grammatical roles** but offer little help in distinguishing document content for retrieval.
- **Eliminating stop words** reduces the number of index terms and enhances efficiency.
- **Drawbacks** include potential loss of meaningful terms (e.g., "Vitamin A") and inability to search for meaningful phrases composed entirely of stop words (e.g., "to be or not to be").

about	above	accordingly	across	after
afterwards	again	against	all	almost
alone	along	already	also	although
am	among	amongst	always	an
and	another	any	anybody	anyhow
anyone	anything	anywhere	apart	are
around	as	aside	at	away
awfully	be	because	been	before
beforehand	behind	being	below	beside
besides	best	better	between	beyond
both	brief	but	by	can
could	did	during	each	even
everybody	everyone	everything	everywhere	else
even	ever	every	for	from
further	had	has	have	her
herself	him	himself	he	furthermore
many	near	shall	she	self
whose	why	will	where	ex
except	far	first	five	former
formerly	over	overall	usually	appropriate

Sample stop words in English

4.2.3 Stemming

- **Stemming** reduces words to their *root form by removing affixes* (e.g., "compute," "computing," "computes," and "computer" → "compute").
- This helps normalize **morphological variants** for consistent text representation.
- **Stems**, are used as index terms.
- The **Porter Stemmer (1980)** is one of the most widely used stemming algorithms.

The stemmed representation of the text, *Design features of information retrieval systems*, is

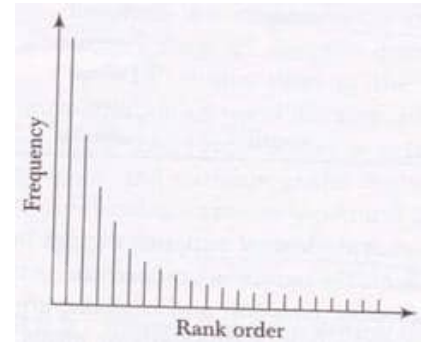
{design, feature, inform, retrieval, system}

- **Stemming can sometimes reduce effectiveness** by removing useful distinctions between words.
- It may **increase recall** by conflating similar terms, but can also **reduce precision** by retrieving irrelevant results (e.g., "computation" vs. "personal computer").

- **Recall and precision** are key metrics for evaluating information retrieval performance

4.2.4 Zipf's Law

- **Zipf's Law** describes the distribution of words in natural language.
- It states that **word frequency** $\times$ **rank** $\approx$ **constant**, meaning frequency is **inversely proportional to rank**.
- When words are sorted by decreasing frequency, higher-ranked words occur more often, and lower-ranked words occur less frequently.
- This pattern is consistent across large text corpora.
- This relationship is shown in **Figure 7**
- **Zipf's Law in practice** shows that human language has:
 - A few **high-frequency** words,
 - Many **low-frequency** words, and
 - A moderate number of **medium-frequency** words.
- In **information retrieval (IR)**:
 - **High-frequency words** lack discriminative power and are not useful for indexing.
 - **Low-frequency words** are rarely queried and can also be excluded.
- **Medium-frequency words** are typically **content-bearing** and ideal for indexing.
- *Words can be filtered* by setting **frequency thresholds** to drop too common or too rare terms.
- **Stop word elimination** is a practical application of Zipf's law, targeting high-frequency terms.



4.3 Information Retrieval Models

- An **Information Retrieval (IR) model** defines how documents and queries are represented, matched, and ranked.
- Core components of an IR system include:
 - A **document model**
 - A **query model**
 - A **matching function** to compare the two
- The primary goal is to **retrieve all relevant documents** for a user query.
- Different IR models exist, varying in:
 - **Representation**: e.g., as sets of terms or vectors of weighted terms
 - **Retrieval method**: based on term presence or **similarity scoring**

- Some models use **binary matching**, while others use **vector space models** with numerical scoring for ranking results.

These models can be classified as follows:

- Classical models of IR
- Non-classical models of IR
- Alternative models of IR

1. **Classical IR models** (Boolean, Vector, Probabilistic):

Based on well-known mathematical foundations.

Simple, efficient, and widely used in commercial systems.

Example:

- Boolean: Query $\rightarrow$ ("machine" AND "learning") OR "AI"
- Vector: Query and documents represented as vectors $\rightarrow$ cosine similarity used to rank results.
- Probabilistic: Estimates the probability that a document is relevant to a given query

2. **Non-classical IR models:**

Use principles beyond similarity, probability, or Boolean logic.

Based on advanced theories like *special logic*, *situation theory*, or *interaction models*.

Example: Modal or fuzzy logic, Contextual information, Dialogue or iterative process

3. **Alternative IR models:**

Enhance classical models with techniques from other fields.

Examples include the Cluster model, Fuzzy model, and Latent Semantic Indexing (LSI).

Example: Hierarchical or k-means clustering of documents, **partial matching** between query and documents using fuzzy logic, **Singular Value Decomposition (SVD)** to identify **hidden semantic structures**.

4.4 Classical Information Retrieval Models

4.4.1 Boolean model

- Introduced in the 1950s – Oldest of the three classical information retrieval models.
- Based on Boolean logic and set theory – Uses binary logic (true/false) operations.
- Document representation – Documents are represented as sets of keywords.

- Uses inverted files – A data structure listing keywords and the documents they appear in.
- Query formulation – Users must write queries using Boolean operators (AND, OR, NOT).
- Retrieval method – Documents are retrieved based on the presence or absence of query terms.

Example: Let the set of original documents be $D = \{D_1, D_2, D_3\}$

Where,

D_1 = Information retrieval is concerned with the organization, storage, retrieval, and evaluation of information relevant to user's query.

D_2 = A user having an information needs to formulate a request in the form of query written in natural language.

D_3 = The retrieval system responds by retrieving the document that seems relevant to the query.

Let the set of terms used to represent these documents be:

$$T = \{\text{information, retrieval, query}\}$$

Then, the set D of document will be represented as follows:

$$D = \{d_1, d_2, d_3\} \text{ Where}$$

$$d_1 = \{\text{information, retrieval, query}\}, d_2 = \{\text{information, query}\}, d_3 = \{\text{retrieval, query}\}$$

Let the query be Q : $Q = \text{information retrieval}$

First, the sets R_1 and R_2 of documents are retrieved in response to Q , Where,

$$R_1 = \{d_j \mid \text{information} \in d_j\} = \{d_1, d_2\}$$

$$R_2 = \{d_j \mid \text{retrieval} \in d_j\} = \{d_1, d_3\} \text{ Then,}$$

the following documents are retrieved in response to query Q

$$\{d_j \mid d_j \in R_1 \cap R_2\} = \{d_1\}$$

- Index terms set:
 $T = \{t_1, t_2, \dots, t_m\}$
- Document set:
 $D = \{d_1, d_2, \dots, d_n\}$
- Boolean query representation:
Query Q is written in normal form as:
$$Q = \bigwedge (\bigvee \theta_j), \quad \theta_i \in \{t_b, \neg t_i\}$$
- Retrieval process (2 steps):
 1. Document retrieval per term:
For each term t_i , construct set R_i as:
$$R_i = \{d_j \mid \theta_i \in d_j\}, \quad \theta_i \in \{t_b, \neg t_i\}$$

where $\neg t_i \in d_j$ means $t_i \notin d_j$
 2. Use of set operations:
Apply union/intersection operations on the R_i sets to retrieve documents that satisfy the query Q .

Advantages:

They are simple, efficient, and easy to implement and perform well in terms of recall and precision if the query is well formulated.

Drawbacks:

- The Boolean model retrieves only fully matching documents; it cannot handle documents that are partially relevant to a query (No partial relevance).
- It does not rank the retrieved documents by relevance—documents either match or don't (No ranking of results).
- Users must formulate queries using strict Boolean expressions, which is unnatural and difficult for most users (Strict query format).

4.2 Probabilistic Model

- Applies probability theory to information retrieval (Robertson and Jones, 1976).
- Documents are ranked by the probability of being relevant to a given query.
- A document is considered relevant if: $P(R/d) \geq P(I/d)$
(i.e., relevance probability is greater than or equal to irrelevance)
- A document is retrieved only if its probability of relevance is greater than or equal to a threshold value α .
- The retrieved set S consists of documents meeting both criteria:

$$S = \{d_j \mid P(R/d_j) \geq P(I/d_j), P(R/d_j) \geq \alpha\}$$

Assumptions & limitations:

- Assumes terms occur **independently** when calculating relevance probabilities.
- This simplifies computation and aids in parameter estimation.
- However, **real-world terms co-occur**, making this assumption often inaccurate.
- The probabilistic model allows **partial matching** of documents to queries.
- A threshold (α) must be set to filter relevant documents.
- Difficult to estimate accurately, especially when the **number of relevant documents is small**.

Process overview:**1. Given:**

- A set of documents D
- A query q
- A cut-off value α

2. Compute:

- $P(R/d)$: Probability that document d is relevant to q
- $P(I/d)$: Probability that document d is irrelevant

3. Rank:

- Documents satisfying $P(R/d) \geq P(I/d)$
- In decreasing order of $P(R/d)$

4. Retrieve:

- Documents for which $P(R/d) \geq \alpha$

Result:

The retrieved set S consists of documents meeting both criteria:

$$S = \{d_j \mid P(R/d_j) \geq P(I/d_j), P(R/d_j) \geq \alpha\}$$

4.3 Vector Space Model

- Representation:
- Documents and queries are represented as vectors of features (terms).
- Each vector exists in a multi-dimensional space, with each dimension corresponding to a unique term in the corpus.
- Numerical vectors: Terms are assigned weights, often based on their frequency in the document (e.g., TF-IDF).
- Similarity computation:
- Ranking algorithms (e.g., cosine similarity) are used to compute the similarity between a document vector and the query vector.
- The similarity score determines how relevant a document is to a given query.
- Retrieval output:
- Documents are ranked based on their similarity scores to the query.
- A ranked list of documents is presented as the retrieval result.

Given a finite set of n documents: $D = \{d_1, d_2, \dots, d_j, \dots, d_n\}$ and

a finite set of m terms: $T = \{t_1, t_2, \dots, t_j, \dots, t_m\}$

Each document is represented by a column vector of weights as follows:

$$(w_{1j}, w_{2j}, w_{3j}, \dots, w_{ij}, \dots, w_{mj})^t$$

Where w_{ij} is the weight of the term t_i in document d_j , the document collection as a whole is represented by an $m \times n$ term-document matrix as:

$$\begin{pmatrix} w_{11} & w_{12} & \dots & w_{1j} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2j} & \dots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{i1} & w_{i2} & \dots & w_{ij} & \dots & w_{in} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \dots & w_{mj} & \dots & w_{mn} \end{pmatrix}$$

Example:

Consider the documents and terms in previous section Let the weights be assigned based on the frequency of the term within the document. Then, the associated vectors will be

$$(2, 2, 1)$$

$$(1, 0, 1)$$

$$(0, 1, 1)$$

The vectors can be represented as a point in Euclidean space,

$$\begin{pmatrix} 2 & 1 & 0 \\ 2 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

To reduce the importance of the length of document vectors, we normalize document vectors.

Normalization changes all vectors to a standard length.

We convert document vectors to unit length by dividing each dimension by the overall length of the vector.

Elements of each column are divided by the length of the column vector given by $\sqrt{\sum_i w_{ij}^2}$

For $\vec{d}_1 = (2, 2, 1)^T$:

$$||\vec{d}_1|| = \sqrt{2^2 + 2^2 + 1^2} = \sqrt{4 + 4 + 1} = \sqrt{9} = 3$$

For $\vec{d}_2 = (1, 0, 1)^T$:

$$||\vec{d}_2|| = \sqrt{1^2 + 0^2 + 1^2} = \sqrt{2}$$

For $\vec{d}_3 = (0, 1, 1)^T$:

$$||\vec{d}_3|| = \sqrt{0^2 + 1^2 + 1^2} = \sqrt{2}$$

$$A_{norm} = \begin{pmatrix} \frac{2}{3} & \frac{1}{\sqrt{2}} & 0 \\ \frac{2}{3} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{3} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \quad \begin{pmatrix} 0.67 & 0.71 & 0.00 \\ 0.67 & 0.00 & 0.71 \\ 0.33 & 0.71 & 0.71 \end{pmatrix}$$

9

Let Query be $Q = (1,1,0)$

Now **normalize** this vector:

$$\text{Length of } Q = \sqrt{1^2 + 1^2 + 0^2} = \sqrt{2} \Rightarrow Q = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0 \right) \approx (0.71, 0.71, 0)$$

Compute Cosine Similarity:

Cosine similarity between vectors Q and D_j is the **dot product** since all vectors are unit length.

For D_1 :

$$\text{sim}(Q, D_1) = (0.71 \cdot 0.67) + (0.71 \cdot 0.67) + (0 \cdot 0.33) \approx 0.951$$

For D_2 :

$$\text{sim}(Q, D_2) = (0.71 \cdot 0.71) + (0.71 \cdot 0) + (0 \cdot 0.71) \approx 0.504$$

For D_3 :

$$\text{sim}(Q, D_3) = (0.71 \cdot 0.00) + (0.71 \cdot 0.71) + (0 \cdot 0.71) \approx 0.504$$

Rank documents based on similarity:

1. D_1 — 0.951 **Retrieved**
2. D_2 — 0.504
3. D_3 — 0.504

4.4 Term Weighting

- Each selected indexing term distinguishes a document from others in the collection.
- Mid-frequency terms are the most discriminative and content-bearing.
- Two key observations refine this idea:
 1. A document is more about a term if the term appears frequently in it.
 2. A term is more discriminative if it appears in fewer documents across the collection.

Term Frequency (TF):

- A term that appears more frequently in a document likely represents its content well.
- TF can be used as a weight to reflect this.

Inverse Document Frequency (IDF):

- Measures how unique or discriminating a term is across the corpus.
- Terms common across many documents are less useful for distinguishing content.
- Calculated as:

$$\text{IDF} = \log(n / n_i)$$

- n = total number of documents
- n_i = number of documents containing term i
- **Note:** A term in all documents gets lowest IDF (1), while a term in one document gets highest IDF (n before taking log).

4.4.1 TF & IDF:

To assign higher weight to terms that occur frequently in a particular document but are rare across the corpus

- **tf-idf** (term frequency-inverse document frequency) weighting scheme combines both term frequency and inverse document frequency.

$$\text{tf-idf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D)$$

Where:

- t is the term,
- d is a specific document,
- D is the entire collection (corpus) of documents,
- $\text{tf}(t, d)$ is the frequency of term t in document d ,
- $\text{idf}(t, D) = \log\left(\frac{n}{n_t}\right) + 1$

Here:

- n is the total number of documents in the corpus,
- n_t is the number of documents that contain the term t ,
- The "+1" in the idf formula is added to prevent zero weights and to smooth the value.

The **tf-idf weighting scheme** combines two components to determine the importance of a term:

- **Term frequency (tf):** A *local* statistic indicating how **often a term appears in a document**.
- **Inverse document frequency (idf):** A *global* statistic that reflects **how rare or specific a term is across the entire document collection**.
- **tf-idf is Widely used** in information retrieval and natural language processing to assess the relevance of a term in a document relative to a corpus.

Example:

Consider a document represented by the three terms *{tornado, swirl, wind}* with the raw **tf** {4, 1, and 1} respectively. In a collection of 100 documents, 15 documents contain the term tornado, 20 contain swirl, and 40 contain wind.

The idf of the term tornado can be computed as $\log\left(\frac{n}{n_t}\right) = \log\left(\frac{100}{15}\right) = 0.82$

The idf of other terms are computed in the same way. Table shows the weights assigned to the three terms using this approach.

Term	Frequency (tf)	Document Frequency (n _i)	IDF [log(n/n _i)]	Weight (tf × idf)
Tornado	4	15	0.824	3.296
Swirl	1	20	0.699	0.699
Wind	1	40	0.398	0.398

Note:

Tornado: **highest TF-IDF weight (3.296)**, indicating both high frequency in the document and relatively low occurrence across all documents.

Swirl: rare but relevant

Wind: least significant

4.4.2 Weight normalization:

Normalization prevents longer documents from being unfairly weighted due to higher raw term counts.

Term frequency (tf) can be normalized by dividing by the frequency of the most frequent term in the document, known as **maximum normalization**, producing values between 0 and 1.

Inverse document frequency (idf) can also be normalized by dividing it by the logarithm of the total documents (log(n)).

be normalized by number of

$$w_{ij} = \frac{tf_{ij}}{\max(tf_{ij})} \times \frac{\log\left(\frac{n}{n_i}\right)}{\log n}$$

Most weighting schemes can thus be characterized by the following three factors:

- Within-document frequency or term frequency (tf)
- Collection frequency or inverse document frequency (idf)
- Document length

Table: Calculating weight with different options for the three weighting factors

♦ A – Term Frequency Handling (tf)

Code	Formula	Description
n	tf_{ij}	Raw term frequency
b	0 or 1	Binary weighting (presence/absence)
a	$0.5 + 0.5 \cdot \frac{tf_{ij}}{\max(tf_{ij} \text{ in } D_j)}$	Augmented tf (scaled by max frequency)
l	$\ln(tf_{ij}) + 1.0$	Logarithmic tf
L	$\frac{\ln(tf_{ij} + 1.0)}{1.0 + \ln(\text{mean}(tf \text{ in } D_j))}$	Average tf normalization

♦ B – Inverse Document Frequency (idf)

Code	Formula	Description
n	No conversion (use tf directly)	No idf used
t	$tf \cdot \log\left(\frac{n}{n_i}\right)$	Multiply tf with IDF component

♦ C – Document Length Normalization

Code	Description
n	No normalization
c	Normalize by cosine: divide weights by $\sqrt{\sum (wt)^2}$

Term weighting in IR has evolved significantly from basic tf-idf. **Different combinations of tf, idf, and normalization** strategies form various weighting schemes, each affecting retrieval performance. Advanced models like **BM25** further refine this by incorporating document length and probabilistic reasoning.

4.4.3 A simple automatic method for obtaining indexed representation of the documents is as follows.

Step 1: Tokenization This extracts individual terms from a document, converts all the letters to lower case, and removes punctuation marks.

Step 2: Stop word elimination This removes words that appear more frequently in the document collection.

Step 3: Stemming This reduces the remaining terms to their linguistic root, to obtain the index terms.

Step 4: Term weighting This assigns weights to terms according to their importance in the document, in the collection, or some combination of both.

Example:

Vector representation of sample documents after stemming

Sample documents

Document 1: Vector space model

Document 2: Probabilistic retrieval model

Document 3: Intelligent techniques in information retrieval



Stemmed Term	Document 1	Document 2	Document 3
inform	0	0	1
intellig	0	0	1
model	1	1	0
probabilist	0	1	0
retriev	0	1	1
space	1	0	0
technique	0	0	1
vector	1	0	0

4.5 Similarity Measures

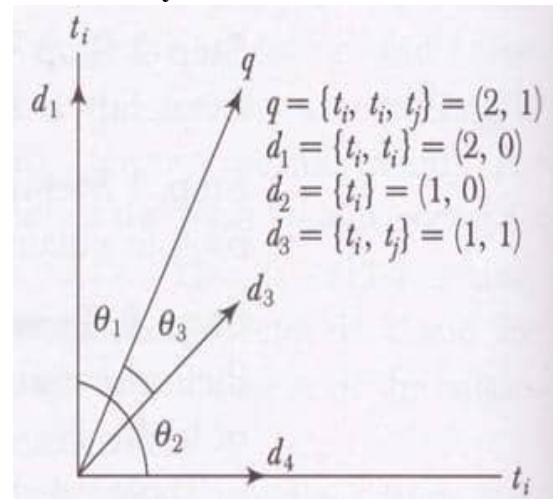
- **Vector Space Model (VSM)** represents documents and queries as vectors in a multi-dimensional space.
- **Retrieval** is based on measuring the *closeness* between query and document vectors.
- **Documents are ranked** according to their numeric similarity to the query.
- **Selected documents** are those geometrically closest to the query vector.
- The model assumes that **similar vectors represent semantically related documents**.
- Example in a 2D space using terms t_i and t_j :

○ Document d_1 : 2 occurrences of t_i ○

Document d_2 : 1 occurrence of t_i ○ Document

d_3 : 1 occurrence each of t_i and t_j

- **Term weights** (raw term frequencies) are used as vector coordinates.
- **Angles** θ_1 , θ_2 , θ_3 represent direction differences between document vectors and the query.
- **Basic similarity measure**: counting common terms.
- **Commonly used similarity metric**: *inner product* of query and document vectors.



$$\text{sim}(d_j, q_k) = (d_j, q_k) = \sum_{i=1}^m w_{ij} \times w_{ik}$$

m : total number of terms (dimensions).

w_{ij} : weight of term i in document d_j .

w_{ik} : weight of term i in query q_k .

The Dice's coefficient:

Measures similarity by **doubling the inner product** and **normalizing by the sum of squared weights**.

$$\text{sim}(d_j, q_k) = \frac{2 \times \sum_{i=1}^m w_{ij} \times w_{ik}}{\sum_{i=1}^m w_{ij}^2 + \sum_{i=1}^m w_{ik}^2}$$

Jaccard's Coefficient:

Computes similarity as the ratio of the inner product to the union (sum of squares minus intersection).

$$\text{sim}(d_j, q_k) = \frac{\sum_{i=1}^m w_{ij} \times w_{ik}}{\sum_{i=1}^m w_{ij}^2 + \sum_{i=1}^m w_{ik}^2 - \sum_{i=1}^m w_{ij} \times w_{ik}}$$

The cosine measure:

Computes the **cosine of the angle** between the document vector d_j and the query vector q_k . It gives a **similarity score between 0 and 1**:

- **0**: No similarity (vectors are orthogonal, angle is 90°).
- **1**: Maximum similarity (vectors point in the same direction).

- w_{ij} : weight of term i in document d_j
- w_{ik} : weight of term i in query q_k
- m : total number of terms

4.5 Non-Classical Models of IR

1. Information Logic Model

- Based on *logical imaging* and *inference* from document to query.
- Introduces **uncertain inference**, where a measure of uncertainty (from van Rijsbergen's principle) quantifies how much additional information is needed to establish the truth of an implication.
- Aims to address classical models' limitations in effectiveness.

2. Situation Theory Model

- Also grounded in van Rijsbergen's principle.
- Uses **infons** to represent information and its truth in specific situations.

- Retrieval is seen as an **information flow** from document to query.
- Incorporates **semantic transformations** (e.g., synonyms, hypernyms) to establish relevance even if a document does not directly support a query.

3. Interaction Model

- Inspired by quantum mechanics' concept of interaction (Copenhagen interpretation).
- Documents are interconnected; retrieval emerges from the **interaction between query and documents**.
- Implemented using **artificial neural networks**, where documents and the query are neurons in a dynamic network.
- Query integration reshapes connections, and the **degree of interaction** guides retrieval.

4.6 Alternative Models of IR

4.6.1 Cluster Model

Reduces the number of document comparisons during retrieval by grouping similar documents.

Cluster Hypothesis (Salton)

- *“Closely associated documents tend to be relevant to the same clusters.”*
- Suggests that documents with high similarity are likely to be relevant to the same queries.

Cluster Improves Efficiency

- Instead of comparing a query with every document:
 - The query is first compared with **cluster representatives** (centroids).
 - Only documents in **relevant clusters** are checked individually.
- This significantly **reduces search time and computational cost**.

Clustering can be applied to:

- **Documents** (group similar documents).
- **Terms** (group co-occurring terms; useful for dimensionality reduction or building thesauri).

Similarity Matrix-Based Clustering

- Let $D = \{d_1, d_2, \dots, d_m\}$ be the document set.
- $E = (e_{ij})$ is the similarity matrix, where e_{ij} denotes similarity between documents d_i and d_j .
- A threshold T is chosen.
- If $e_{ij} \geq T$, documents d_i and d_j are grouped into the same cluster.
- Remaining documents (not meeting the threshold with any other) form their own clusters.

Cluster Representation

- Each cluster C_k has a **representative vector (centroid)**:
 - $\Gamma_k = \{a_{1k}, a_{2k}, \dots, a_{mk}\}$, where each element represents the average of corresponding term weights in the documents of that cluster.
 - An element a_i in this vector is computed as $a_{ik} = \frac{\sum_{d_j \in C_k} a_{ij}}{|C_k|}$
 - where a_{ij} is weight of the term t_i , of the document d_j , in cluster C_k . During retrieval, the query is compared with the cluster vectors $(r_1, r_2, \dots, r_k, \dots, r_p)$
- This comparison is carried out by computing the similarity between the query vector q and the representative vector Γ_k as

$$s_{ik} = \sum_{i=1}^m a_{ik} q_i, \quad k = 1, 2, \dots, p$$

- A cluster C_k whose similarity S_k exceeds a threshold is returned and the search proceeds in that cluster.

Example:

Consider 3 documents (**d1, d2, d3**) and 5 terms (**t1 to t5**). The term-by-document matrix is:

t/d	d1	d2	d3
t1	1	1	0
t2	1	0	0
t3	1	1	1
t4	0	0	1
t5	1	1	0

So, document vectors are: $d1 = (1, 1, 1, 0, 1)$, $d2 = (1, 0, 1, 0, 1)$, $d3 = (0, 0, 1, 1, 0)$ Calculate

cosine similarity between the documents:

- **sim(d1, d2)** $\text{dot}(d1, d2) = 1 \times 1 + 1 \times 0 + 1 \times 1 + 0 \times 0 + 1 \times 1 = 3$

$$|d1| = \sqrt{(1^2+1^2+1^2+0^2+1^2)} = \sqrt{4} = 2$$

$$|d2| = \sqrt{(1^2+0^2+1^2+0^2+1^2)} = \sqrt{3} \approx 1.73$$

$$\text{sim} = 3 / (2 \times 1.73) \approx \mathbf{0.87}$$

- **sim(d1, d3)** dot = $1 \times 0 + 1 \times 0 + 1 \times 1 + 0 \times 1 + 1 \times 0 = 1$ $|d3| = \sqrt{(0^2+0^2+1^2+1^2+0^2)} = \sqrt{2} \approx 1.41$
sim = $1 / (2 \times 1.41) \approx \mathbf{0.35}$
- **sim(d2, d3)** dot = $1 \times 0 + 0 \times 0 + 1 \times 1 + 0 \times 1 + 1 \times 0 = 1$ sim = $1 / (1.73 \times 1.41) \approx \mathbf{0.41}$

Similarity matrix:

	d1	d2	d3
d1	1.0		
d2	0.87	1.0	
d3	0.35	0.41	1.0

Clustering with threshold 0.7

- d1 and d2 $\rightarrow$ sim = 0.87 $\rightarrow$ **Cluster C1** • d3 has low similarity with both $\rightarrow$ **Cluster C2**

Clusters:

- C1 = {d1, d2}
- C2 = {d3}

Cluster representatives

Average the vectors in each cluster:

- **r1 = avg(d1, d2)**
= $((1+1)/2, (1+0)/2, (1+1)/2, (0+0)/2, (1+1)/2)$
= **(1, 0.5, 1, 0, 1)**
- **r2 = d3 = (0, 0, 1, 1, 0)**

Retrieval is performed by matching the query vector with r1 and r2.

Retrieval using a query

Assume the **query vector q = (1, 0, 1, 0, 1)**

This means the query contains terms t1, t3, and t5.

Similarity with cluster vectors:

- **sim(q, r1)** dot = $1 \times 1 + 0 \times 0.5 + 1 \times 1 + 0 \times 0 + 1 \times 1 = 3$
 $|q| = \sqrt{(1^2 + 0^2 + 1^2 + 0^2 + 1^2)} = \sqrt{3} \approx 1.73$ $|r1| = \sqrt{(1^2 + 0.5^2 + 1^2 + 0^2 + 1^2)} = \sqrt{3.25} \approx 1.80$ sim = $3 / (1.73 \times 1.80) \approx \mathbf{0.96}$
- **sim(q, r2)** dot = $1 \times 0 + 0 \times 0 + 1 \times 1 + 0 \times 1 + 1 \times 0 = 1$ $|r2| = \sqrt{(0^2+0^2+1^2+1^2+0^2)} = \sqrt{2} \approx 1.41$
sim = $1 / (1.73 \times 1.41) \approx \mathbf{0.41}$

Query is **closer to r1**, so we retrieve documents from **Cluster C1 = {d1, d2}**

4.6.2 Fuzzy Model

In the **fuzzy model** of information retrieval, each document is represented as a **fuzzy set of terms**, where each term is associated with a **membership degree** indicating its importance to the document's content. These weights are typically derived from term frequency within the document and across the entire collection.

Each document d_j is modelled as a **vector of term weights**: $(w_{1j}, w_{2j}, \dots, w_{mj})^T$

where w_{ij} is the degree to which term t_i belongs to document d_j .

Each term t_i defines a **fuzzy set** f_i over the documents: $f_i = \{(d_j, w_{ij})\}$

For queries:

- A **single-term query** returns documents where the term's weight exceeds a threshold.
- An **AND query** uses the **minimum** of term weights (fuzzy intersection).
- An **OR query** uses the **maximum** of term weights (fuzzy union).

This model allows ranking documents by their **degree of relevance** to the query.

Example:

Documents:

- $d1 = \{\text{information, retrieval, query}\}$
- $d2 = \{\text{retrieval, query, model}\}$
- $d3 = \{\text{information, retrieval}\}$

Term Set:

- $T = \{t1: \text{information, } t2: \text{model, } t3: \text{query, } t4: \text{retrieval}\}$

Fuzzy sets (term-document weights):

- $f1(t1): \{(d1, 1/3), (d2, 0), (d3, 1/2)\}$
- $f2(t2): \{(d1, 0), (d2, 1/3), (d3, 0)\}$
- $f3(t3): \{(d1, 1/3), (d2, 1/3), (d3, 0)\}$
- $f4(t4): \{(d1, 1/3), (d2, 1/3), (d3, 1/2)\}$

Query:

- $q = t2 \wedge t4$ (i.e., **model AND retrieval**)

In fuzzy logic, the AND operation ($\wedge$) is typically interpreted using the **minimum** of the memberships.

Step 1: Retrieve memberships for t2 and t4

From f2 (t2 - model):

- d1: 0
- d2: 1/3
- d3: 0

From f4 (t4 - retrieval):

- d1: 1/3 • d2: 1/3
- d3: 1/2

Step 2: Compute query membership using min (t2, t4)

Apply **min** operator for each document:

- d1: $\min(0, 1/3) = 0$
- d2: $\min(1/3, 1/3) = 1/3$
- d3: $\min(0, 1/2) = 0$

Step 3: Determine which documents are returned

Assuming a non-zero membership indicates relevance (typical in fuzzy IR), only documents with **non-zero** membership values for the query will be returned. So, only: **d2** has a non-zero value (1/3)

4.6.3 Latent Semantic Indexing Model

Latent Semantic Indexing (LSI) applies **Singular Value Decomposition (SVD)** to information retrieval, aiming to uncover **hidden semantic structures** in word usage across documents. Unlike traditional keyword-based methods, LSI captures **conceptual similarities** between terms and documents, even when there's no exact term match.

- **Term-document matrix (W):** Represents the frequency or weighted usage of terms (rows) in documents (columns).
- **SVD Decomposition:** The matrix W is decomposed into three matrices:

$$W = TSD^T$$

Where, T: Term vectors, **S:** Diagonal matrix of singular values, **D:** Document vectors

- **Truncated SVD:** Retain only the top **k singular values** and corresponding vectors to form a lower-dimensional approximation $W_k = T_k S_k D_k^T$, capturing the main semantic structure and removing noise.
- **Query Transformation:** Queries are projected into the same reduced **k-dimensional**

latent $q_k = q^T T_k S_k^{-1}$

- **Similarity Computation:** Documents are ranked using similarity measures (e.g., **cosine similarity**) between the query vector and document vectors in the latent space.

Advantages:

- Captures **semantic relationships** between terms and documents.
- Can retrieve **relevant documents** even if they don't share any terms with the query.
- Reduces the impact of **synonymy and polysemy**.

Example:

- An example is given with a 5-term, 6-document matrix reduced to 2 dimensions using truncated SVD. This shows how documents originally in a 5D space (based on terms like *tornado*, *storm*, etc.) are projected into a 2D **concept space**, revealing deeper connections among them.
- In essence, **LSI enhances retrieval effectiveness** by operating on meaning (latent semantics) rather than surface-level word matching.

The SVD of X is computed to get the three matrices T , S , and D . $X_{5 \times 6} = T_{5 \times 5} S_{5 \times 5} (D_{6 \times 5})^T$

Term Vector

$$T = \begin{bmatrix} 0.3318 & 0.3338 & 0.8064 & -0.2426 & -0.2634 \\ 0.6693 & 0.1616 & -0.2737 & 0.5853 & -0.3293 \\ 0.5514 & 0.1038 & -0.0961 & -0.2667 & 0.7777 \\ 0.3583 & -0.5745 & -0.2148 & -0.5778 & -0.4021 \\ 0.0974 & -0.7223 & 0.4684 & 0.4400 & 0.2362 \end{bmatrix}$$

Singular values

$$S = \begin{bmatrix} 2.3830 & 0 & 0 & 0 & 0 \\ 0 & 1.6719 & 0 & 0 & 0 \\ 0 & 0 & 1.2415 & 0 & 0 \\ 0 & 0 & 0 & 0.8288 & 0 \\ 0 & 0 & 0 & 0 & 0.5454 \end{bmatrix}$$

Document Vector
Consider the first two
largest singular

$$D^T = \begin{bmatrix} 0.6515 & 0.1392 & 0.6626 & 0.1912 & 0.2809 & 0.0409 \\ 0.3584 & 0.1996 & -0.1848 & -0.7756 & 0.0967 & -0.4320 \\ 0.3516 & 0.6495 & -0.4710 & 0.2042 & -0.2205 & 0.3773 \\ 0.0916 & -0.2927 & -0.3127 & -0.1662 & 0.7062 & 0.5309 \\ 0.3392 & -0.4829 & 0.0849 & -0.3042 & -0.6037 & 0.4330 \end{bmatrix}$$

values of S , and rescale $D^T_{2 \times 6}$ with singular values to get matrix $R_{2 \times 6} = S_{2 \times 2} D^T_{2 \times 6}$, as shown in below Figure. R is a reduced dimensionality representation of the original term-by-document matrix X .

$$R = \begin{bmatrix} 1.5526 & 0.3318 & 1.5790 & 0.4557 & 0.6603 & 0.0974 \\ 0.5992 & 0.3338 & -0.3090 & -1.2967 & 0.1616 & -0.7223 \end{bmatrix}$$

To find out the changes introduced by the reduction, we compute document similarities in the new space and compare them with the similarities between documents in the original space.

The document-document correlation matrix for the original n -dimensional space is given by the matrix $Y = X^T X$. Here, Y is a square, symmetric $n \times n$ matrix. An element Y_{ij} , in this matrix gives the similarity between documents i and j . The correlation matrix for the original document vectors is shown in Figure (Z). This matrix is computed using X , after normalizing the lengths of its columns.

$$Z = \begin{bmatrix} 1.0000 & 0.5774 & 0.6667 & 0.0000 & 0.5774 & 0.0000 \\ 0.5774 & 1.0000 & 0.0000 & 0.4082 & 0.5774 & 0.7071 \\ 0.6667 & 0.0000 & 1.0000 & 0.4082 & 0.0000 & 0.7071 \\ 0.0000 & 0.4082 & 0.4082 & 1.0000 & 0.0000 & 0.0000 \\ 0.5774 & 0.5774 & 0.0000 & 0.0000 & 1.0000 & 0.0000 \\ 0.0000 & 0.7071 & 0.7071 & 0.0000 & 0.0000 & 1.0000 \end{bmatrix}$$

The document-document correlation matrix for the new space is computed analogously using the reduced representation R . Let N be the matrix R with length-normalized columns. Then, $M = N^T N$ gives the matrix of document correlations in the reduced space. The correlation matrix M is given in Figure.

$$M = \begin{bmatrix} 1.0000 & 0.9131 & 0.8464 & -0.0304 & 0.9914 & -0.2322 \\ 0.9131 & 1.0000 & 0.5557 & -0.4353 & 0.8518 & -0.6086 \\ 0.8464 & 0.5557 & 1.0000 & 0.5066 & 0.9089 & 0.3215 \\ -0.0304 & -0.4353 & 0.5066 & 1.0000 & 0.1008 & 0.9793 \\ 0.9914 & 0.8518 & 0.9089 & 0.1008 & 1.0000 & -0.1027 \\ -0.2322 & -0.6086 & 0.3215 & 0.9793 & -0.1027 & 1.0000 \end{bmatrix}$$

The similarity between document d_1 , d_4 (-0.0304), and d_6 (-0.2322) is quite low in the new space because document d_1 is not topically similar to documents d_4 and d_6 .

In the original space, the similarity between documents d_2 and d_3 and between documents d_2 and d_5 is 0. In the new space, they have high similarity values (0.5557 and 0.8518 respectively) although documents d_3 and d_5 share no term with the document d_2 . This topical similarity is recognized due to the co-occurrence of patterns in the documents.

4.7 Major Issues in Information Retrieval

1. **Vocabulary Mismatch:** Users often express queries using terms that differ from those in relevant documents, leading to retrieval failures.
2. **Ambiguity and Polysemy:** Words with multiple meanings can cause confusion in interpreting user intent, affecting retrieval accuracy.

3. **Scalability and Performance:** As data volumes grow, IR systems must efficiently index and retrieve information without compromising speed or accuracy.
4. **Evaluation Metrics:** Determining the relevance and effectiveness of IR systems is challenging due to the subjective nature of "relevance" and the lack of standardized evaluation methods.
5. **User Behavior Modeling:** Understanding and predicting user behavior is essential for refining search results and improving user satisfaction.
6. **Integration with Natural Language Processing (NLP):** Incorporating NLP techniques can enhance IR systems by enabling better understanding of context and semantics, but it also introduces complexity.

These issues highlight the multifaceted nature of IR and the need for interdisciplinary approaches to address them effectively.

LEXICAL RESOURCES

1. Introduction

The chapter provides an overview of freely available tools and lexical resources for natural language processing (NLP), aimed at assisting researchers—especially newcomers to the field. It emphasizes the importance of knowing where to find resources, which can significantly reduce time and effort. The chapter compiles and briefly discusses key tools such as stemmers, taggers, parsers, and lexical databases like WordNet and FrameNet, along with accessible test corpora, all of which are available online or through scholarly articles.

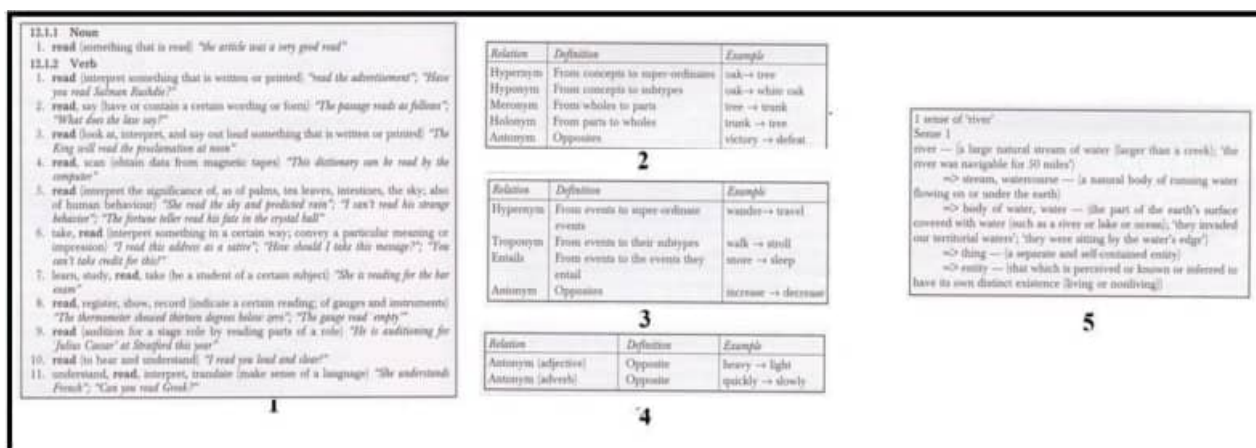
2. WORDNET

A comprehensive lexical database for the English language developed at Princeton University under George A. Miller based on psycholinguistic principles, WordNet is divided into three databases: nouns, verbs, and a combined one for adjectives and adverbs.

Key features include:

- **Synsets:** Groups of synonymous words representing a single concept.
- **Lexical and semantic relations:** These include synonymy, antonymy, hypernymy/hyponymy (generalization/specialization), meronymy/holonymy (part/whole), and troponymy (manner-based verb distinctions).
- **Multiple senses:** Words can belong to multiple synsets and parts of speech, with each sense given a **gloss**—a dictionary-style definition with usage examples.
- **Hierarchical structure:** Nouns and verbs are arranged in taxonomic hierarchies (e.g., 'river' has a hypernym chain), while adjectives are grouped by antonym sets.

The figure 1 shows the entries for the word 'read'. 'Read' has one sense as a noun and 11 senses as a verb. Glosses help differentiate meanings. Figures 2, 3, and 4 show some of the relationships that hold between nouns, verbs, and adjectives and adverbs.



Nouns and verbs are organized into hierarchies based on the hypernymy/hyponymy relation, whereas adjectives are organized into clusters based on antonym pairs (or triplets). Figure 5 shows a hypernym chain for 'river' extracted from WordNet. Figure 6 shows the troponym relations for the verb 'laugh'.

Sense 1	
laugh, express joy, express mirth — (produce laughter)	
⇒ bray — (laugh loudly and harshly)	
⇒ bellylaugh — (laugh a deep, hearty laugh)	
⇒ roar, howl — (laugh unrestrainedly and heartily)	
⇒ snicker, snigger — (laugh quietly)	
⇒ giggle, titter — (laugh nervously; 'The girls giggled when the rock star came into the classroom')	
⇒ break up, crack up — (laugh unrestrainedly)	
⇒ cackle — (emit a loud, unpleasant kind of laughing)	
⇒ guffaw, laugh loudly — (laugh boisterously)	
⇒ chuckle, chortle, laugh softly — (laugh quietly or with restraint)	
⇒ convulse — (be overcome with laughter)	
⇒ cachinnate — (laugh loudly and in an unrestrained way)	

The availability and multilingual extensions of **WordNet**:

- **English WordNet** is **freely available** for download at <http://wordnet.princeton.edu/obtain>.
- **EuroWordNet** extends WordNet to multiple **European languages**, including English, Dutch, Spanish, Italian, German, French, Czech, and Estonian. It includes both **language-internal** relations and **cross-lingual links** to English meanings.
- **Hindi WordNet**, developed by **CFILT at IIT Bombay**, follows the same design principles as the English version but includes **language-specific features**, such as **causative relations**. It currently includes:
 - Over **26,208 synsets** and **56,928 Hindi words** ○ **16 types of semantic relations** ○ Each entry contains a synset, gloss (definition), and its position in the ontology.
- Figure 7 shows the Hindi WordNet entry for the word **आकांक्षा** (aakanksha).
- Hindi WordNet can be obtained from the URL <http://www.cfilt.iitb.ac.in/wordnet/webhwn/>. CFLIT has also developed a Marathi WordNet.
- Figure 8 shows the Marathi WordNet (<http://www.cfilt.iitb.ac.in/wordnet/webmwn/wn.php>) entry for the word 'qa' (pau).

1. (R) अपेक्षा, आकांक्षा, अन्वेषणा — किसी पर मनोसा रखने की क्रिया कि अमुक कार्य उसके द्वारा हो जायेगा “ हर पिता की अपने पुत्र से यह अपेक्षा रहती है कि वह अपने जीवन में सफल हो ”
2. (R) इच्छा, अभिलाषा, आकांक्षा, उवाहिना, आरप्ता, तमन्ना, कामना, तल्लव, चेष्टा, हसस्त, मुराव, पिपासा, प्यास, तुष्णा, मनोकामना, मनोकामना, मनोवांछा, मनोमेष, मनोभावना, मरपी, रणा, मर्पी, मन, रणा, मंशा, लिप्सा, लालसा, तुषा, चाह, अरमान, क्षुधा, भूख, भूक, सुखा, हवस, स्तुहा, अनीप्सा, अनु, अपेक्षिता, अभिकांक्षा, अभिकाम, वांछा, वावछा, वावछा, अभिष्टा, अभिलाष, अभिप्रीति, अभिमत, अभिमतता, अभिमति, अभिभूता, अभिलासा, अभिलाखा, अभिलाख, अभिलाखना — यह मनोप्रति जो किसी बात या वस्तु की प्राप्ति की ओर ध्यान ले जाती है “ इंसान की हर इच्छा पूरी नहीं होती / उसकी ज्ञान पिपासा बढ़ती जा रही है / मेरा आज खाने का मन नहीं है ”

1. (R) पाव एक चतुर्थांश-चौथा भाग “मी बाजारातून एक पाव चुरमुरे आणले”
2. (R) पाव, पावरीटी-मळाचे पीठ आंबवून कोलेल एक खाद्यविशेष “मुंबईत बरेच लोक पाव खाऊन नुजराण करतात”
3. (R) पाव-पावाचे वजन “बुकानवार चहाच्या पूडीचे वजन करपासाठी पाव शोधत आहे”
4. (R) पाव-एक चतुर्थांश बाटली वारू “एक पाव व्यावसायिक तो बटवट करू लागला”

Figure 8 WordNet entry for the Marathi word **पाव** (pau)

Figure 7. WordNet entry for the Hindi word **आकांक्षा** (aakanksha).

2.1 Applications of WordNet

The key applications of WordNet in Information Retrieval (IR) and Natural Language Processing (NLP):

1. Concept Identification:

WordNet helps identify the underlying **concepts** associated with a term, enabling more accurate understanding and interpretation of user queries or texts by capturing their full **semantic richness**.

2. Word Sense Disambiguation (WSD):

WordNet is widely used for disambiguating word meanings in context. Its value lies in:

- Providing **sense definitions** and **examples**
- Organizing words into **synsets**
- Defining **semantic relations** (e.g., synonymy, hypernymy)

These features make WordNet the most prominent and frequently used resource for WSD.

- **Early research:** One of the first uses of WordNet in WSD for IR was by **Voorhees (1993)**, who applied its **noun hierarchy** (hypernym/hyponym structure).
- **Further work:** Researchers like **Resnik (1995, 1997)** and **Sussna (1993)** also utilized WordNet in developing WSD techniques.

Additional applications of WordNet Automatic

□ Query Expansion:

WordNet's **semantic relations** (e.g., synonyms, hypernyms, hyponyms) can enhance query terms, allowing a broader and more meaningful search.

- **Voorhees (1994)** used these relations to expand queries, improving retrieval performance by going beyond simple keyword matching.

□ Document Structuring and Categorization:

WordNet's **conceptual framework** and semantic relationships have been employed for **text categorization**, helping systems classify documents more effectively.

- **Scott and Matwin (1998)** leveraged this approach for document classification tasks.

□ Document Summarization:

WordNet aids in generating **lexical chains**—sequences of semantically related words—that help identify key topics and coherence in texts.

- **Barzilay and Elhadad (1997)** used this technique to improve **text summarization**.

3. FRAMENET

FrameNet, a rich lexical database focused on **semantically annotated English sentences**, grounded in **frame semantics**.

1. Frame Semantics:

Each word (especially verbs, nouns, adjectives) evokes a specific **situation or event** known as a **frame**.

2. Target Word / Predicate:

The word that evokes the frame (e.g., *nab* in the ARREST frame).

3. Frame Elements (FEs):

These are **semantic roles** or participants in the frame-specific event (e.g., AUTHORITIES, SUSPECT, TIME in the ARREST frame).

- These roles define the **predicate-argument structure** of the sentence.

4. Annotated Sentences:

Sentences, often drawn from the **British National Corpus**, are tagged with frame elements to illustrate how words function in context.

5. Ontology Representation:

FrameNet provides a **semantic-level ontology** of language, representing not just grammatical but also **contextual and conceptual relationships**.

Example:

In the sentence, “The police *nabbed* the suspect,” the word *nab* triggers the **ARREST** frame:

- *The police* → **AUTHORITIES**
- *The suspect* → **SUSPECT**

[Authorities The police] nabbed [Suspect the snatcher]

FrameNet thus provides a structured and nuanced way to model meaning and roles in language, making it valuable for tasks such as semantic role labeling, information extraction, and natural language understanding.

The **COMMUNICATION** frame includes roles like **ADDRESSEE**, **COMMUNICATOR**, **TOPIC**, and **MEDIUM**. The **JUDGEMENT** frame includes **JUDGE**, **EVALUEE**, and **REASON**. Frames can inherit roles from others; for instance, the **STATEMENT** frame inherits from **COMMUNICATION** and includes roles such as **SPEAKER**, **ADDRESSEE**, and **MESSAGE**.

The following sentences show some of these roles:

[Judge She] [Evaluee blames the police] [Reason for failing to provide enough protection].

[Speaker She] told [Addressee me] [Message 'I'll return by 7:00 pm today'].

Figure 9 shows the core and non-core frame elements of the COMMUNICATION frame, along with other details.

Communication	
Frame Elements	
Core:	
Addressee [Add]	Receiver of Message from the Communicator.
Communicator [Com]	The person conveying (written or spoken) a message to another person.
Message [Msg]	A proposition or set of propositions that the Communicator wants the Addressee to convey
Topic [Top]	The entity that the proposition(s) are about.
Non-core:	
Amount_of_information [Amo]	The amount of information exchanged when communication occurs.
Depictive [Dep-Act]	The Depictive describes the state of the Communicator.
Duration []	The length of time during which the communication takes place.
Manner [Mann]	The Manner in which the Communicator communicates.
Means [Mns]	The Means by which the Communicator communicates.
Medium [Medium]	The physical or abstract setting in which the Message is conveyed.
Time []	The time at which the communication takes place.
Inherits From:	
Is Inherited By:	Communication_noise, Statement
Subframe of:	
Has Subframes:	
Uses:	Topic
Is Used By:	Claim_ownership, Communication_response, Contacting, Deny_permission, Discussion, Hear, Questioning, Reasoning, Reporting, Request, etc
Is Inchoative of:	
Is Causative of:	
See Also:	
Sample Predicates	
communicate, indicate, signal, speech	

Figure 9 Frame elements of communication frame

3.1 FrameNet Applications

FrameNet supports **semantic parsing** and **information extraction** by providing **shallow semantic roles** that reveal meaning beyond syntax. For example, the noun "*match*" plays the same **theme role** in both sentences below, despite differing syntactic positions:

The umpire stopped the match.

The match stopped due to bad weather.

FrameNet also enhances **question-answering systems** by enabling role-based reasoning. For instance, in the **TRANSFER** frame, verbs like "*send*" and "*receive*" share roles such as **SENDER**, **RECIPIENT**, and **GOODS**, allowing a system to infer that:

Q: *Who sent a packet to Khushbu?*

A: *Khushbu received a packet from the examination cell.*

Additional applications of FrameNet include:

- **Information retrieval (IR)**
- **Machine translation (interlingua design)**
- **Text summarization**
- **Word sense disambiguation**

These uses highlight FrameNet's importance in understanding and processing natural language at a deeper semantic level.

4. STEMMERS:

Stemming (or **conflation**) is the process of reducing **inflected or derived words** to their **base or root form**. The resulting **stem** doesn't need to be a valid word, as long as related terms map to the same stem.

Purpose:

- Helps in **query expansion**, **indexing** (e.g., in search engines), and various **NLP tasks**.

Common Stemming Algorithms:

- **Porter's Stemmer** – Most widely used (Porter, 1980).
- **Lovins Stemmer** – An earlier approach (Lovins, 1968).
- **Paice/Husk Stemmer** – A more recent and flexible method (Paice, 1990).

These tools, called **stemmers**, differ in how aggressively they reduce words but all aim to improve text processing by grouping word variants.

Figure 10 shows a sample text and output produced using these stemmers.

Input Text: Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation.
Output: Lovins stemmer: such an analys can rev feature that ar not eas vis from the vari in th individu gen and can lead to a picture of expres that is mor biolog transpar and acces to interprese Porter's stemmer: such an analysi can reveal feature that ar not easily visible from the variat in the individu gene and can lead to a picture of express that is more biolog transpar and access to interpret Paice stemmer: Such an analys can rev feat that are not easy vis from the vary in the individ gen and can lead to a pict of express that is mor biolog transp and access to interpret

4.1 Stemmers for European Languages:

- **Snowball** provides stemmers for many European languages:
 - Examples: **English, French, Spanish, Russian, Portuguese, German, Dutch, Hungarian, Italian, Swedish, Norwegian, Danish, Finnish** ◦

Available at:

<http://snowball.tartarus.org/texts/stemmersoverview.html>

4.2 Stemmers for Indian Languages:

- **Standard stemmers** for Indian languages like **Hindi** are limited.
- Notable research:

- **Ramanathan and Rao (2003)**: Used handcrafted suffix lists for Hindi.
- **Majumder et al. (2007)**: Used a **cluster-based approach**, evaluated using **Bengali data**, and found that stemming improves **recall**.
- **CFILT, IIT Bombay** has developed stemmers for Indian languages:
 - <http://www.cfilt.iitb.ac.in>

4.3 Stemming Applications:

- Widely used in **search engines** and **IR systems**:
 - Reduces word variants to a common form, improving **recall** and reducing **index size**.
 - Example: "*astronaut*" and "*astronauts*" are treated as the same term.

However, for **English**, stemming may **not always improve precision**.
- Also applied in:
 - **Text summarization** ○ **Text categorization** ○ Helps in **term frequency analysis** by consolidating word forms into stems.

5. PART-OF-SPEECH TAGGER

Part-of-speech tagging is a crucial **early-stage NLP technique** used in applications like **speech synthesis**, **machine translation**, **information retrieval (IR)**, and **information extraction**. In IR, it helps with **indexing**, **phrase extraction**, and **word sense disambiguation**.

5.1 Stanford Log-linear POS Tagger

- **Model Type**: Maximum Entropy Markov Model
- **Key Features**:
 - Uses **preceding and following tag contexts** via a **dependency network**.
 - Employs a **wide range of lexical features**.
 - Incorporates **priors** in conditional log-linear models.
- **Accuracy**: 97.24% on Penn Treebank WSJ
- **Improvement**: 4.4% error reduction over previous best (Toutanova et al., 2003)
- **More Info**: <http://nlp.stanford.edu/software/tagger.shtml>

5.2 A Part-of-Speech Tagger for English

- **Model Type**: Maximum Entropy Markov Model (MEMM)
- **Inference**: **Bi-directional inference** algorithm ○ Enumerates **all possible decompositions** to find the **best sequence**.
- **Performance**:

- Outperforms **unidirectional** methods.
- Comparable to top algorithms like **kernel SVMs**.
- **Reference:** Tsuruoka and Tsujii (2005)

5.3 TnT Tagger (Trigrams'n'Tags)

- **Model Type:** Hidden Markov Model (HMM)
- **Features:**
 - Uses **trigrams**, **smoothing**, and **handling of unknown words**.
- **Efficiency:** Performs as well as other modern methods, including **maximum entropy models**.
- **Reference:** Brants (2000)

Table 12.1 shows tagged text of document #93 of the CACM collection.

A	DT	simple	JJ
technique	NN	algebraic	JJ
is	VBZ	formulas	NNS
shown	VBN	into	IN
for	IN	a	DT
enabling	VBG	three	CD
a	DT	address	NN
computer	NN	computer	NN
to	TO	code	NN
translate	VB		

5.4 Brill Tagger

- **Type:** Rule-based, transformation-based learning.
- **Key Features:**
 - Learns tagging rules automatically.
 - Handles unknown words.
 - Supports **k-best tagging** (multiple tags in uncertain cases).
- **Performance:** Comparable to statistical methods.
- Brill tagger is available for download at the link http://www.cs.jhu.edu/~brill/RBT1_14.tar.Z.

5.5 CLAWS Tagger

- **Type:** Hybrid (probabilistic + rule-based).
- **Developed By:** University of Lancaster.
- **Accuracy:** 96–97%, depending on text type.
- **Adaptability:** Works with diverse input formats.
- **More Info:**

5.6 Tree-Tagger

- **Type:** Probabilistic (uses decision trees).
- **Strengths:**
 - Effective with sparse data.
 - Automatically selects optimal context size.
- **Accuracy:** Above 96% on Penn Treebank.
- The tagger is available at the link <http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html>

5.7 ACOPOST Collection • Language: C

- **Includes:**
 1. **Maximum Entropy Tagger (MET)** – Iterative feature-based method.
 2. **Trigram Tagger (T3)** – HMM-based using tag pairs.
 3. **Transformation-based Tagger (TBT)** – Based on Brill's rule learning.
 4. **Example-based Tagger (ET)** – Uses memory-based reasoning from past data.

5.8 POS Taggers for Indian Languages

- **Challenge:** Lack of annotated corpora and basic NLP tools.
- **Development Centers:**
 - **IIT Bombay:** Developing POS taggers for **Hindi** and **Marathi** using a bootstrapping + statistical approach.
 - **Other Institutes:** CDAC, IIIT Hyderabad, CIIL Mysore, University of Hyderabad.
 - **Urdu Tagging:** Reported by Hardie (2003) and Baker et al. (2004). ◦ More information can be found at <http://ltrc.iiit.net> and www.cse.iitb.ac.in

6. RESEARCH CORPORA

Research corpora have been developed for a number of NLP-related tasks. In the following section, we point out few of the available standard document collections for a variety of NLP-related tasks, along with their Internet links.

6.1 IR Test Collection

Glasgow University, UK, maintains a list of freely available IR test collections. Table lists the sources of those and few more IR test collections. LETOR (learning to rank) is a package of benchmark data sets released by Microsoft Research Asia. It consists of two datasets OHSUMED and TREC (TD2003 and TD2004).

LETOR	http://research.microsoft.com/users/tyliu/LETOR/
LISA	
CACM	
CISI	
MEDLINE	http://www.dcs.gla.ac.uk/ldom/ir_resources/test_collections/
Cranfield	
TIME	
ADI	

LETOR is packaged with extracted features for each query-document pair in the collection, baseline results of several state-of-the-art learning-to-rank algorithms on the data and evaluation tools. The data set is aimed at supporting future research in the area of learning ranking function for information retrieval.

6.2 Summarization Data

Evaluating a text summarizing system requires existence of 'gold summaries'. DUC provides document collections with known extracts and abstracts, which are used for evaluating performance of summarization systems submitted at TREC conferences. Figure 11 shows a sample document and its extract from DUC 2002 summarization data.

```
<DOC>
<DOCNO> AP880911-0016 </DOCNO>
<FILEID> APNR-09-11-08 0421EDT</FILEID>
<FIRST> 1 BC-Hurricane Gilbert 09-11-0339</FIRST>
<SECOND> BC-Hurricane Gilbert 09-11-0339</SECOND>
<HEAD> Hurricane Gilbert Heads Toward Dominican Coast</HEAD>
<BYLINE> By RUDDY GONZALEZ</BYLINE>
<BYLINE> Associated Press Writer</BYLINE>
<DATELINE> SANTO DOMINGO, Dominican Republic (AP) </DATELINE>
<TEXT>
Hurricane Gilbert swept toward the Dominican Republic Sunday, and the Civil Defense alerted its heavily populated south coast to prepare for high winds, heavy rains, and high seas.

The storm was approaching from the southeast with sustained winds of 75 mph gusting to 92 mph.

"There is no need for alarm," Civil Defense Director Eugenio Cabral said in a television alert shortly before midnight Saturday.

Cabral said residents of the province of Barahona should closely follow Gilbert's movement. An estimated 100,000 people live in the province, including 70,000 in the city of Barahona, about 125 miles west of Santo Domingo.

Tropical Storm Gilbert formed in the eastern Caribbean and strengthened into a hurricane Saturday night. The National Hurricane Center in Miami reported its position at 2 a.m. Sunday at latitude 16.1 north, longitude 67.5 west, about 140 miles south of Ponce, Puerto Rico, and 200 miles southeast of Santo Domingo.

The National Weather Service in San Juan, Puerto Rico, said Gilbert was moving westward at 15 mph with a "broad area of cloudiness and heavy weather" rotating around the center of the storm.

The weather service issued a flash flood watch for Puerto Rico and the Virgin Islands until at least 6 p.m. Sunday.

Strong winds associated with the Gilbert brought coastal flooding, strong southeast winds and up to 12 feet to Puerto Rico's south coast. There were no reports of casualties.

San Juan, on the north coast, had heavy rains and gusts Saturday, but they subsided during the night.

On Saturday, Hurricane Florence was downgraded to a tropical storm and its remnants pushed inland from the U.S. Gulf Coast. Residents returned home, happy to find little damage from 80 mph winds and sheets of rain.

Florence, the sixth named storm of the 1988 Atlantic storm season, was the second hurricane. The first, Debby, reached minimal hurricane strength briefly before hitting the Mexican coast last month.
</TEXT>
</DOC>
Extract
Tropical Storm Gilbert in the eastern Caribbean strengthened into a hurricane Saturday night. The National Hurricane Center in Miami reported its position at 2 a.m. Sunday to be about 140 miles south of Puerto Rico and 200 miles southeast of Santo Domingo. It is moving westward at 15mph with a broad area of cloudiness and heavy weather with sustained winds of 75mph gusting to 92mph. The Dominican Republic's Civil Defense alerted that country's heavily populated south coast and the National Weather Service in San Juan, Puerto Rico issued a flood watch for Puerto Rico and the Virgin Islands until at least 6 p.m. Sunday.
```

6.3 Word Sense Disambiguation

SEMCOR is a sense-tagged corpus used in disambiguation. It is a subset of the Brown corpus, sense-tagged with WordNet synsets.

Open Mind Word Expert13 attempts to create a very large sense-tagged corpus. It collects word sense tagging from the general public over the Web.

6.4 Asian Language Corpora

The EMILLE (Enabling Minority Language Engineering) corpus is a multilingual resource developed at Lancaster University, UK, aimed at supporting natural language processing (NLP) for South Asian languages. The project, in collaboration with the Central Institute for Indian Languages (CIIL) in India, provides extensive data and tools for various Indian languages. The corpus includes monolingual written and spoken corpora, parallel corpora, and annotated data. The monolingual written corpus covers 14 South Asian languages, while the spoken data, sourced from BBC Asia radio broadcasts, includes five languages: Hindi, Bengali, Gujarati, Punjabi, and Urdu. The parallel corpus consists of English texts and their translations into five languages, featuring materials like UK government advice leaflets, aligned at the sentence level. The annotated section includes part-of-speech tagging for Urdu and annotations of demonstrative usage in Hindi.

The EMILLE/CIIL corpus is available free of charge for research purposes at elda.org, with further details provided in the manual at emille.lancs.ac.uk. This resource is particularly valuable for research in statistical machine translation and other NLP applications involving Indian languages, despite challenges posed by the limited availability of electronic text repositories in these languages.

7. JOURNALS AND CONFERENCES IN THE AREA

- Major NLP Research Bodies and Conferences:
 - ACM (Association for Computing Machinery)
 - ACL (Association for Computational Linguistics) and EACL (European Chapter)
 - RIAO (Recherche d'Information Assistie par Ordinateur)
 - COLING (International Conferences on Computational Linguistics) □ Key Conferences:
 - ACM SIGIR: A leading international conference on Information Retrieval (IR); the 30th conference held in Amsterdam (July 23–27, 2007).
 - TREC (Text Retrieval Conferences): Organized by the US government (NIST), providing standardized IR evaluation results; formerly called the Document/Message Understanding Conferences.
 - NTCIR: Focuses on IR for Japanese and other Asian languages.
 - ECIR: European counterpart of SIGIR.
 - KES (Knowledge-Based and Intelligent Engineering & Information Systems): Focuses on intelligent systems, including NLP, neural networks, fuzzy logic, and web mining.

- HLT-NAACL: Sponsored by the North American chapter of ACL; covers human language technologies.
- Notable Journals:
 - Journal of Computational Linguistics: Focuses on theoretical and linguistic aspects.
 - Natural Language Engineering Journal: Focuses on practical NLP applications.
 - Information Retrieval (Kluwer), Information Processing and Management (Elsevier), ACM TOIS (Transactions on Information Systems), Journal of the American Society for Information Science.
- Other Relevant Journals:
 - International Journal of Information Technology and Decision Making (World Scientific)
 - Journal of Digital Information Management
 - Journal of Information Systems □ AI Journals Reporting NLP Work:
 - Artificial Intelligence
 - Computational Intelligence
 - IEEE Transactions on Intelligent Systems
 - Journal of AI Research