

Text & Web Intelligence

- The subject Text and Web Intelligence deals with techniques to extract, process and retrieve useful knowledge from unstructured text and large scale web data.
- It combines concepts from Information Retrieval, NLP, ML and Data Mining.
- The goal is to build intelligent systems (like search engine, chatbots, etc) that can understand, organize and respond to human queries effectively.

Example:- When a user searches on Google, text & web intelligence methods process the query, analyze web content and return the most relevant ranked results.

Text Intelligence:- Extracting knowledge from Unstructured text using NLP & ML.

Web Intelligence:- Applying intelligent methods to analyze and mine web data for smart applications.

Text Mining

Data Mining

Web Mining

Defⁿ

Process of extracting useful info. and patterns from unstructured text data.

Process of discovering hidden patterns, trends and knowledge from structured data.

Process of extracting and analyzing info. from the web.

Data Type

Unstructured or Semi-structured (doc., emails, social media posts)

Structured (databases, data warehouses)

Semi-structured & Unstructured (web pages, hyperlinks)

Techniques Used

NLP (tokenization, stemming, sentiment analysis, topic modeling)

Machine Learning, Classification, Clustering, Association Rules

Web content mining, Web str. mining, Web Usage mining.

Goal

Understand meaning, context and hidden knowledge from text

Find patterns, correlation and predictions from numerical/categorical data.

Discover patterns in web data for improving user experience, personalization & search

Application

Spam filtering

Marketing, Health care

Recommended System, Web logs

Fact

A fact is a true and objective statement about something, which cannot be disputed.

It is raw truth about an event or object.

Ex:- Water boils at 100°C
India Won the 2011 World Cup.

Data

Data is a collection of raw facts and figures without context or meaning.

Data alone does not provide knowledge until it is processed.

Ex:- Numbers: 100, 200, 300
Student marks: 85, 92, 78, 60

Information

Information is processed, organized and meaningful data that helps in decision-making.

It gives context to raw data.

Ex:- If student marks = 85, 92, 78, 60

information: Avg. marks = 78.75

Information Retrieval

Information Retrieval is the process of finding relevant information from a large collection of unstructured or semi-structured data.

Types of Information Retrieval System

Based on Functionality

① Boolean Retrieval - Exact match retrieval

- Uses logical operators (AND, OR, NOT)

- Ex:-

Query = "data AND mining"

② Ranked Retrieval - Best match retrieval

- Return documents in order of relevance score

- Ex:-

Google Search

Based on Data Types

① Text Retrieval

- IR from doc, books, web pages
- Ex :- Google search, digital libraries

② Multimedia Retrieval

- IR from images, videos, audio
- Ex :- Youtube search, Google images

③ Structured/Hybrid Retrieval

- Combination of structured queries (SQL) with unstructured search
- Ex :- Searching job listings by title + salary + description

Based on User Interaction

① Interactive IR

- User refines search step by step
- Ex :- Research databases like PubMed

② Automatic IR

- System retrieves & ranks doc without

uses refinement

Ex:- Google search.

Methods of Information Retrieval

IR methods defines how documents are matched and ranked against queries.

① Boolean Retrieval Method

Uses Boolean logic (AND, OR, NOT)

② Vector Space Model (VSM)

Represents documents & queries as vectors of terms.

Uses similarity measures (like Cosine similarity) to rank results.

Example:- If Query vector = (AI, healthcare)
and Doc. vector = (AI, healthcare, data), similarity score is calculated.

//_

③ TF-IDF

A statistical method used to evaluate how important a word is in document relative to the entire collection (corpus).

④ Page Rank

An Algorithm developed by Google.

It measures the importance of web pages based on the number & quality of links pointing to them.

⑤ Probabilistic Model

Estimates the probability that a document is relevant to a query.

Uses Bayesian probability.

⑥ Latent Semantic Indexing (LSI)

Uses matrix factorization (SVD) to capture hidden relationship between words and documents.

Information Retrieval Problem

The main problem of IR is :

How to find and retrieve the most relevant information from a large collection of unstructured or semi-structured data in response to a user's query.

Issues in Information Retrieval

① Ambiguity

Words have multiple meanings.

Ex:- Apple can mean fruit or company.

② Synonymy

Different words may mean the same thing.

③ Vocabulary Mismatch

User and author may use different terms.

Ex:- User query:- "doctor", but document uses "physician".

④ Ranking and Relevance

Many documents may match the query
→ IR must rank them

Ex:- Searching "AI applications" gives thousands of results, but only top 10 are truly useful.

⑤ Noisy or Incomplete Data

Documents may contain irrelevant, duplicate or incomplete information.

Ex:- Spam websites in search results.

⑥ Scalability

Search engines must handle billions of documents & millions of queries per sec.

⑦ User Query Problem

⑧ Evaluation of IR System

⑨ Multimedia Retrieval Challenges

⑩ Information Overload

Boolean Retrieval

Boolean Retrieval is the simplest information retrieval model.

It uses Boolean logic (AND, OR, NOT) to decide whether a document is relevant (1) or not relevant (0) for a query.

Term Incidence Matrix

It is a binary matrix showing whether a term occurs in a document.

1 $\rightarrow$ present , 0 $\rightarrow$ absent.

Suppose we have 3 documents:

D1 : Data mining technique

D2 : Machine learning & data Science

D3 : Mining and deep learning.

Term Incidence Matrix :

Term	D1	D2	D3
data	1	1	0
mining	1	0	1
learning	0	1	1
machine	0	1	0

Query Example :-

Query = data AND mining $\rightarrow (D1=1, D2=0, D3=0)$
 $\rightarrow D1$ only

Inverted Index

Instead of storing a big matrix, IR systems store an inverted index.

It maps each term $\rightarrow$ list of docs. where it appears.

Ex :-

D1 : Data mining technique

D2 : Machine learning & data science

D3 : Mining and deep learning

//_

data $\rightarrow \{D1, D2\}$
mining $\rightarrow \{D1, D3\}$
learning $\rightarrow \{D2, D3\}$
machine $\rightarrow \{D2\}$

Query Example :-

data and mining $\rightarrow \{D1, D2\} \cap \{D1, D3\}$
 $\rightarrow \{D1\}$

Dictionary - Posting List

It is a part of information retrieval & data structure.

It store indexing information about collection of document.

It map each term to the list of document where their term appear including term position.

It is calculating using term frequency and position frequency.

D1: "The cat sat on the mat"

D2: "The Dog chased the cat"

//_

Doc terms :- "The", "Cat", "Sat", "On", "mat"
"dog", "chased"

Posting List Cat : $[(\text{Document 1}, 1), (\text{Document 2}, 1)]$

↓ ↓
Doc. No Term freq.

Posting List Dog :- $[(D1, 0), (D2, 1)]$

Processing Boolean Queries

Processing Boolean Queries means defining precisely what you are looking for in a search by using logical operators.

- The most common operators are:
AND, OR, NOT

The goal is to identify the matches that satisfy the given logical condition.

Example :- * → Wild Card operator

Cat *

↳ Cat, Cats, Category, Catbills, etc.

Strategies for Processing Boolean Queries

① Processing Order

The query is executed step by step in order.

- Used for short queries.

② In-place Operation

Operations are applied directly while processing.

- Used for long queries.

Skip Pointers

- When we process Boolean queries (like AND, OR), we often need to intersect postings lists (the lists of documents where each term occurs).
- If postings lists are very long, comparing them one by one is slow.
- Skip pointers are extra links we add inside postings lists

//_

to allow the search system to "jump ahead" instead of checking every document.

Example Without Skip pointers

Suppose we have 2 terms:

data $\rightarrow [2, 4, 6, 8, 12, 14]$

mining $\rightarrow [1, 2, 3, 6, 8, 11, 14]$

We want to process query: data AND mining

- Compare 2 vs 1 $\rightarrow$ move mining forward
- Compare 2 vs 2 $\rightarrow$ ✓ Match found
- Compare 4 vs 3 $\rightarrow$ move mining forward
- Compare 4 vs 6 $\rightarrow$ move mining forward
- Compare 6 vs 6 $\rightarrow$ ✓ Match found

Example With Skip pointers

data $\rightarrow [2 \rightarrow 6 \rightarrow 10 \rightarrow 14]$

mining $\rightarrow [1 \rightarrow 6 \rightarrow 11 \rightarrow 14]$

2 vs 1 $\rightarrow$ move mining forward

2 vs 6 $\rightarrow 6 > 2$, skip from 2 $\rightarrow$ 6

6 vs 6 $\rightarrow$ ✓ Match found

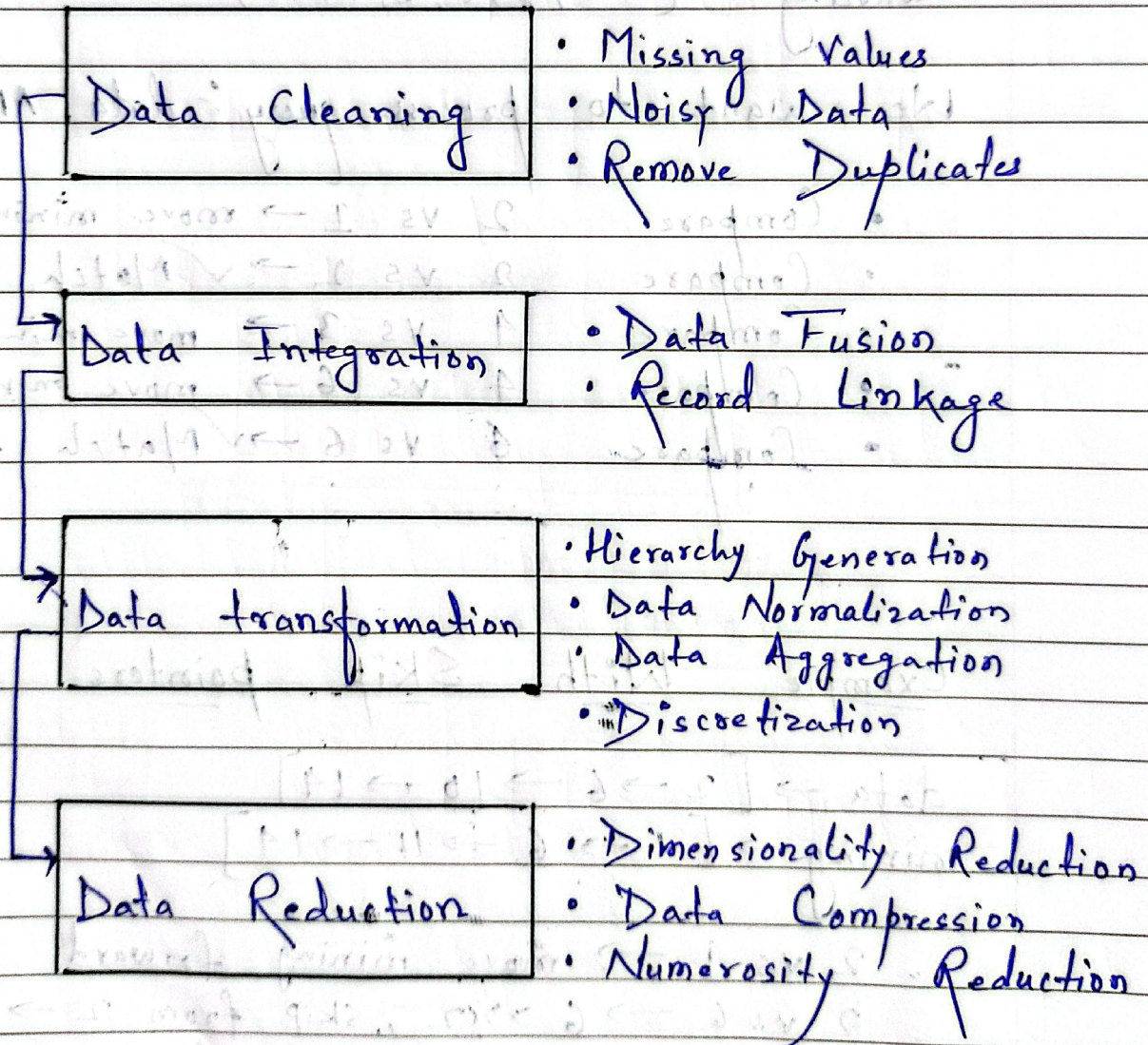
10 vs 11 $\rightarrow 11 > 10$, skip ahead

14 vs 14 $\rightarrow$ ✓ Match found

Data Preprocessing

Data preprocessing is the process of preparing raw data for analysis by using data cleaning, data integration, data transformation, data reduction and it is usable format.

∴ Steps :-



Improves data quality & ensures accuracy of dataset.

//_

It Also help to handle missing value, normalization data & remove duplicate data.

Data Fusion: Combining multiple data inputs into one meaningful output.

Record Linkage: It is the process of identifying and linking records from one or more datasets that refer to the same real world entity, even if the records are not identical.

Data Aggregation: It is the process of collecting and summarizing data from multiple sources into a single, more useful form for analysis.

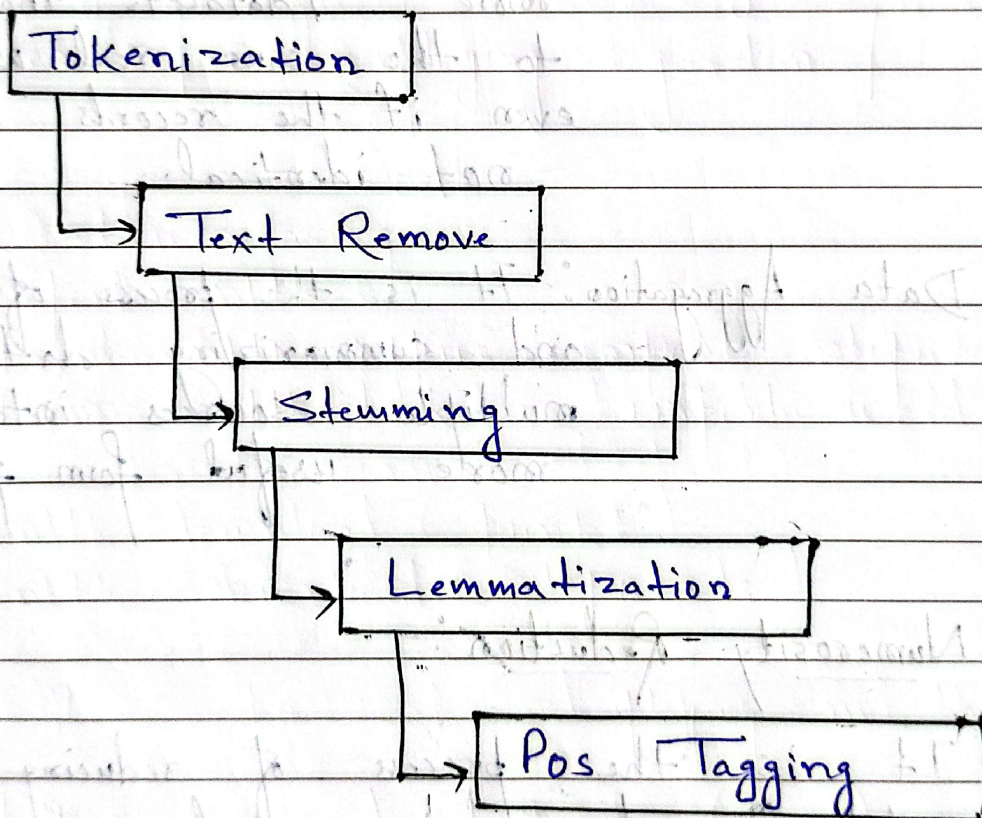
Numerosity Reduction:

It is the process of reducing the volume of data by representing it in a smaller, more compact form without losing significant information.

Bining Method :

Bining is the data preprocessing and discretization technique used to handle noisy, smooth values, and group continuous attributes into bins (intervals).

Text Preprocessing



① Tokenization

Tokenization is the process of splitting text into smaller units called tokens.

These tokens can be words, subwords or sentences depending on the method.

Ex:- "I Love NLP" → ["I", "Love", "NLP"]

—: Code :-

```
import nltk
from nltk.tokenize import word_tokenize
nltk.download("punkt")
```

```
text = "I love Natural Language Processing,  
it's amazing!"
```

```
tokens = word_tokenize(text)
print(tokens)
```

Output

```
[ "I", "love", "Natural", "Language", "Processing",  
  ", ", "it", "s", "amazing", "!" ]
```


② Stop Word Removal

- Stop words are very common words like is, am, are, the, in, on, at, a, an, and etc.
- These words often don't add much meaning and can be removed to reduce noise.

Ex:- "The cat is playing in the garden"

After:- ["cat", "playing", "garden"]

Code:-

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

```
nltk.download("punkt")
nltk.download("stopwords")
```

```
text = "The cat is playing in the garden"
```

```
tokens = word_tokenize(text)
```

```
stop_words = set(stopwords.words("english"))
```

```
filtered_tokens = [word for word in tokens
                    if word.lower() not
                    in stop_words]
```


//_

```
print (filtered_tokens)
```

Output

`["cat", "playing", "garden", "."]`

③ Normalization

Normalization means converting text into a standard and consistent form so that similar words are treated the same.

It reduces variations in text caused by case, punctuation, spelling or formatting.

Common Types of Normalization

① Lowercasing :-

- Convert all characters to lowercase
- "NLP IS FUN" → "nlp is fun"

② Removing Punctuation :-

"Hello !!! How are you?" → "Hello How are you"

iii) Expanding Contractions :-

"don't" → "do not"
"it's" → "it is"

iv) Removing Numbers (if not useful)

"I have 2 cats" → "I have cats"

v) Removing Extra Spaces :-

"Hello world" → "Hello world"

∴ Code :-

i) Lower casing

```
text = "NLP is FUN And Powerful"  
normalized_text = text.lower()  
print(normalized_text)
```

Output :-

nlp is fun and powerful

ii) Removing Punctuation

```
import string  
text = "Hello !!! How are you?"  
normalized_text = text.translate(str.  
make_translate("", "", string.punctuation))  
print(normalized_text)
```


//_

Output :- Hello How are you

(iii) Expanding Contractions

```
import contractions
```

```
text = "I don't like NLP, it's  
confusing sometimes."
```

```
expanded_text = contractions.fix(text)  
print(expanded_text)
```

Output :-

I do not like NLP, it is confusing
sometimes.

(iv) Removing Numbers

```
import re
```

```
text = "I have 2 cats and 3 dogs."  
normalized_text = re.sub(r'\d+', "", text)  
print(normalized_text)
```

Output :-

I have cats and dogs.

⑤ Removing Extra Spaces

```
text = "Hello World !"  
normalized_text = " ".join(text.split())  
print(normalized_text)
```

Output

Hello World !

⑥ Handling Emojis

Emojis are special characters that carry meaning.

In preprocessing you may:

- Remove them
- Convert them to text meaning

Code :-

```
import emoji  
text = "I love NLP 😊🔥"  
normalized_text = emoji.demojize(text)  
print(normalized_text)
```

Output

I love NLP : smiling-face : fire :

(VII) Spell Checker

It is used to correct spelling errors in the text. It imports Spell Checker.

Spell correction ensures consistency.

∴ Code :-

```
from spellchecker import SpellChecker
```

```
spell = SpellChecker()
```

```
words = ["I", "love", "mlp", "processing"]
```

```
misspelled = spell.unknown(words)
```

```
corrected = [spell.correction(word) if word  
in misspelled else word for  
word in words]
```

```
print("Corrected words:", corrected)
```

Output

```
Corrected words: ['I', 'love', 'mlp', 'processing']
```


④ Stemming

Stemming is the process of reducing a word to its root/base form by chopping off prefixes or suffixes.

The result may not always be a real word in English.

Example :-

playing, played, plays $\rightarrow$ play

Stemming is fast but less accurate than Lemmatization.

Types of Stemmers in NLTK

① Porter Stemmer

② Snowball Stemmer

③ Lancaster Stemmer

$\therefore$ Code :-

```
from nltk.stem import PorterStemmer  
stemmer = PorterStemmer()
```


//_

```
words = ["play", "playing", "played", "plays"]
```

```
stems = [stemmer.stem(word) for  
          word in words]
```

```
print(stems)
```

Output

['play', 'play', 'play', 'play']

⑤ Lemmatization

- Lemmatization reduces a word to its lemma (dictionary form).
- Unlike stemming, it considers grammar and meaning (pos tags).
- Always returns a real word, unlike stemming.

Example :-

running → run

better → good

studies → study

//_

Why Lemmatization is Better than Stemming?

Stemming :- studies → studi

Lemmatization :- studies → study

Code :-

```
import nltk
import nltk.stem
from nltk.corpus import wordnet
```

```
nltk.download('punkt')
```

```
nltk.download('wordnet')
```

```
nltk.download('omw-1.4')
```

```
lemmatizer = WordNetLemmatizer()
```

```
words = ["running", "runs", "studies", "better",  
         "cars"]
```

```
lemmas = [lemmatizer.lemmatize(word) for  
           word in words]
```

```
lemmas_pos = [lemmatizer.lemmatize(word,  
                                     pos="v")] for word in words  
print(lemmas_pos)
```

Output

```
['run', 'run', 'study', 'better', 'car']
```


⑥ POS tagging

POS tagging is the process of assigning a grammatical category (part of speech) to each word in a text, such as noun, verb, adjective, adverb etc.

It is the process of marking up, word into text.

POS tag Representation

noun, singular	→	NN
noun, plural	→	NNS
verb	→	VB
Adjective	→	JJ
Adverb	→	RV
determiner	→	DT
preposition	→	IN
pronoun	→	PRP
Conjunction	→	CC
Interjection	→	UH
Symbol	→	SY
to	→	TO

Example:

I	want	an	early	promotion.
↓	↓	↓	↓	↓
PRP	VB	DT	JJ	NN

I Like to read book.
 ↓ ↓ ↓ ↓ ↓
 PRP VB TO VB NN

They refuse to go.
 ↓ ↓ ↓ ↓
 PRP VB TO VB

POS Tagging Techniques

① Rule Based POS tagging

It has pre-defined technique to assign tags.

Example:-

If word end with "ing" → Verb

If word follows "the" → noun

② Statistical POS tagging

It provide probability & statistical relation between words & tags.

③ Machine Learning Based POS tagging

Machine learning based POS tagging is an approach where we train a supervised learning model on a large, annotated corpus.

//_

The model learns patterns and features from the training data and then predicts the POS tags for new, unseen text.

POS is classified into 2 categories

① Open Class

These are open to new additions, new words can be created and added anytime.

- Noun
- Verb
- Adjective
- Adverb

② Closed Class

These are fixed sets of words, new words are rarely added.

- Pronoun
- Preposition
- Conjunction
- Interjection
- Determiners
- Symbol

Wildcard Queries

A wildcard query allows you to use special symbols to represent one or more unknown characters in a search term.

It is useful when you don't know the exact spelling of a word or want to search for word variations.

Example:- Query : colo*r
Matches : color, colour

Types of Wildcard

① * (Asterisk)

Matches zero or more characters

Ex:- Query : comput*
Matches : computer, computing, computation

② ? (Question Mark)

Matches exactly one character.

Ex:- Query : te?t
Matches : text, test, tent

③ Character Range / Set []

Ex: Query: gr[ae]y

Matches: gray, grey

It is used in situation where:

(i) User is uncertain of spelling of query term.

(ii) Where user is aware of multiple variant of spelling term.

(iii) When user is uncertain of correct spell of foreign word and sentences.

(iv) When user read root document containing various terms capture by stemming.

Application:

(i) Data base search

(ii) Search for pattern

(iii) file system search

(iv) Information retrieval system

(v) Search Engine / Elastic search

(vi) Text editor & programming language.

Data

Data refers to raw facts, figures or symbols that can be processed or analyzed to extract meaningful information.

Types of data

① Structured

Organized into row & columns
(like a table).

Eg:- Excel, SQL, Database.

②. Unstructured

No fixed format

Eg:- Text, images, videos.

③ Semi-Structured

Partial Structure

Eg:- XHTML, XML, JSON, JS.

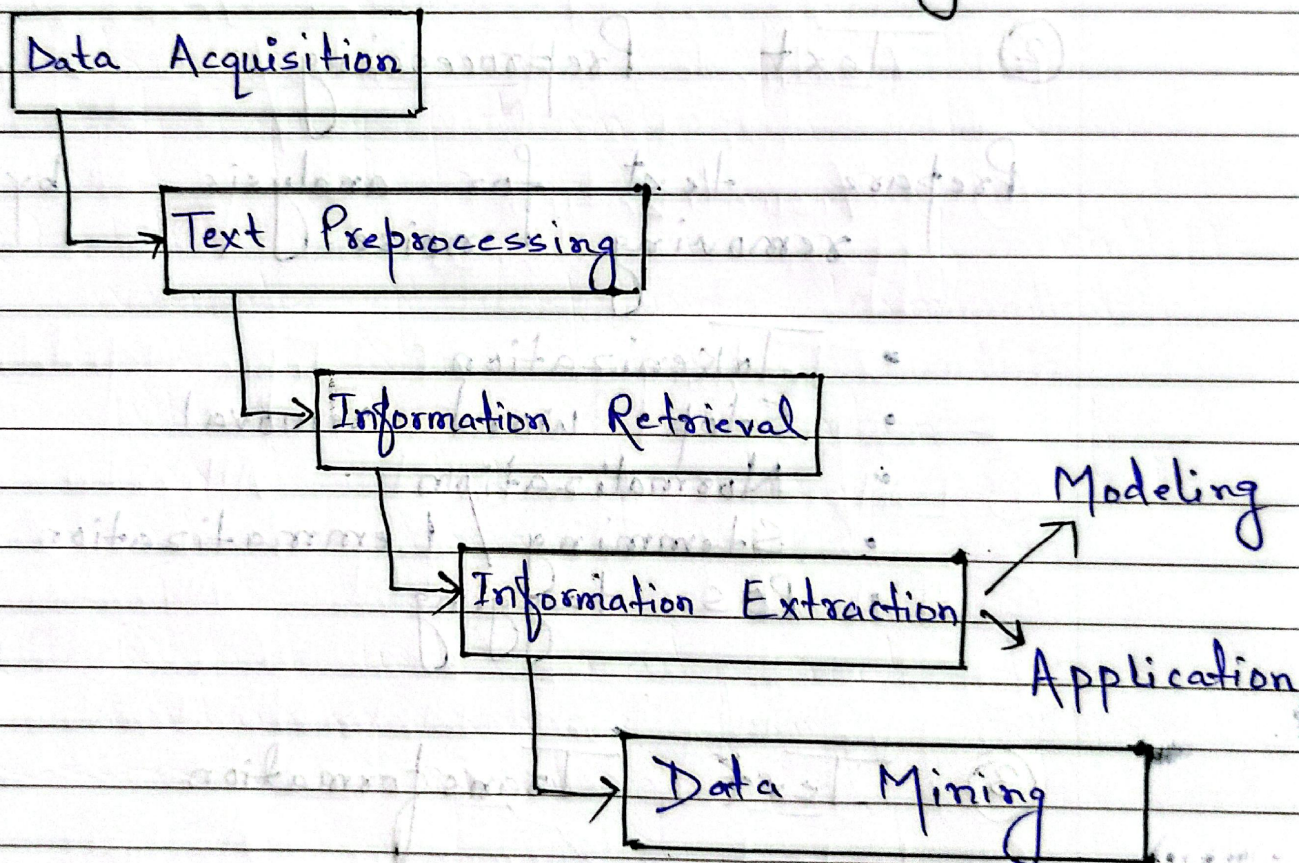
UNIT- 2

__/__/__

Text Mining

Text Mining is the process of extracting meaningful information, hidden patterns and knowledge from unstructured text data using NLP, IR and ML techniques.

Process of Text Mining



Techniques used in Text Mining

- ① Classification
- ② Clustering
- ③ Summarization

Steps of Text Mining

① Text Collection

Collect raw text from multiple sources:

- Websites, blogs, social media, emails, customer reviews.

② Text Preprocessing

Prepare text for analysis by removing noise!

- Tokenization
- Stop word removal
- Normalization
- Stemming / Lemmatization
- POS tagging

③ Text Transformation

Cleaned text into a structured format.

- Bag of Words
- TF-IDF
- Word Embeddings

④ Text Mining / Analytics

Apply algorithms to find hidden patterns.

- Classification/clustering
- Topic Modeling

⑤ Evaluation

⑥ Knowledge Representation & Visualization

Topic Modeling

Topic Modeling is a text mining technique used to automatically discover hidden topics within a collection of documents.

- It does not need labeled data.
- Each document can contain multiple topics and each topic is represented by a set of words.

Example :-

If you have 1000 news articles, topic modeling can group into topics like politics, sports, technology, health etc.

Steps in Topic Modeling

- ① Collect Data
- ② Preprocess Text
- ③ Convert to Numerical Representation
- ④ Apply Topic Modeling Representation
- ⑤ Evaluate & Interpret Algorithm
- ⑥ Visualization

Techniques of Topic Modeling

- ① Latent Dirichlet Allocation
- ② Probabilistic Dirichlet Allocation
- ③ Non-Negative Matrix Factorization
- ④ Latent Semantic Analysis
- ⑤ Deep Learning

Application

- ① Organizing Document
- ② Recommendation System

- _/_/_
- (iii) Customer Feedback Analysis
 - (iv) Social Media Analysis
 - (v) Healthcare.

Paradigmatic Relation Discovery

Two words have a paradigmatic relation if one can be replaced by another in the same context without breaking grammar.

These relation provide the relationship between words that can be substituted by words with same class.

There are different types of class:

- Synonymy, antonymy, hyponymy

Example

Sentence :- The dog is barking

Paradigmatic Alternatives : dog → cat, lion, puppy

Because these words can replace "dog" still make sense grammatically.

"The puppy is barking"

//_

Used in Text Mining

Helps in synonym discovery, word embeddings, semantic similarity.

Syntagmatic Relation > Discovery

Syntagmatic relations are about co-occurrence or combination.

Two words have a syntagmatic relation if they frequently appear together in a sequence.

Example is:

Sentence: The dog barks loudly.

Words dog → barks have a syntagmatic relation because they co-occur.

New → York, Machine → Learning.

Eg: The dog is white.

The bark is white.

TF - Term Frequency

Term Frequency measures how often a term (word) appears in a document compared to the total number of terms in that document.

$$TF(t, d) = \frac{\text{No. of times term } t \text{ appears in document } d}{\text{Total no. of terms in document } d}$$

Example :-

Doc 1: The cat sat on the mat.

Doc 2: The dog played in the park.

Doc 3: Cats & Dogs are great pets.

$$TF(\text{cat}, \text{Doc 1}) = \frac{1}{6}$$

$$TF(\text{cat}, \text{Doc 2}) = \frac{0}{6}$$

$$TF(\text{cat}, \text{Doc 3}) = \frac{1}{6}$$

IDF - Inverse Document Frequency

IDF measures how important a term is across the entire set of documents.

If a word appears in many docs, its IDF will be low (less informative).

$$IDF(t, D) = \log \frac{\text{Total no. of doc. in corpus } D}{\text{No. of doc. containing term } t}$$

Example :-

Doc 1: The cat sat on the mat

Doc 2: The dog played in the park.

Doc 3: cats and dogs are great pets.

$$IDF(\text{cat}, D) = \log \frac{3}{2} \approx 0.176$$

$$IDF(\text{dog}, D) = \log \frac{3}{2} \approx 0.176$$

TF-IDF

- TF tells us how frequent a word is in a document.
- IDF tells us how unique (important) that word is across all documents.

So, TF-IDF highlights words that are frequent in a document but rare across the corpus.

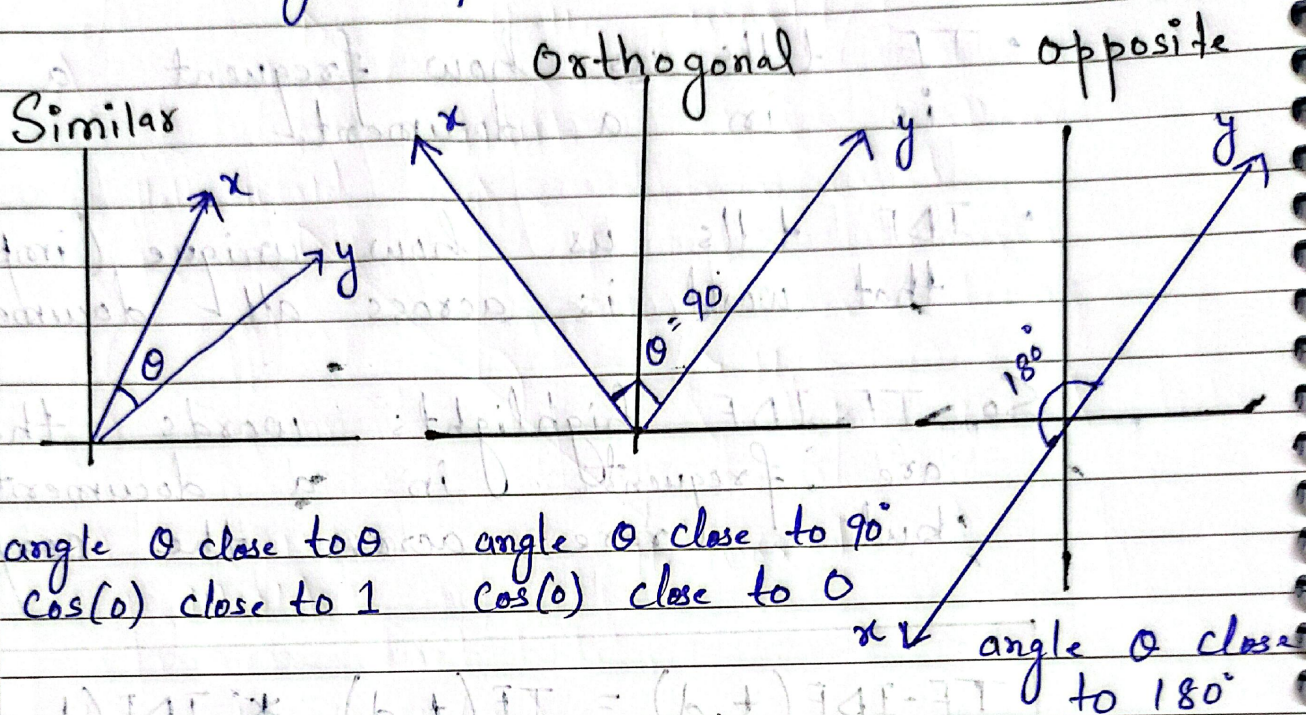
$$\text{TF-IDF}(t, d) = \text{TF}(t, d) * \text{IDF}(t, d)$$

Cosine Similarity

Cosine similarity is a measure of similarity between two documents or vectors based on the cosine of the angle between them.

- If two vectors point in the same direction, cosine similarity = 1
- If they are orthogonal (90° apart), cosine similarity = 0
- If they are opposite, similarity = -1

Its range from -1 to $+1$.



Steps

① Vectorization

Convert text data into vector data matrix.

Example:

D1: "I love machine learning"

D2: "I love deep learning"

D1 $\rightarrow [1, 1, 1, 0, 1]$

D2 $\rightarrow [1, 1, 0, 1, 1]$

Techniques of Vectorization

① Bag of Words

② TF-IDF

③ Word Embedding

② Angle Calculation

The cosine of the angle between these vectors is calculated using cosine similarity angle calculation.

Angle calculation having 3 types of vector.

(i) Similar vector

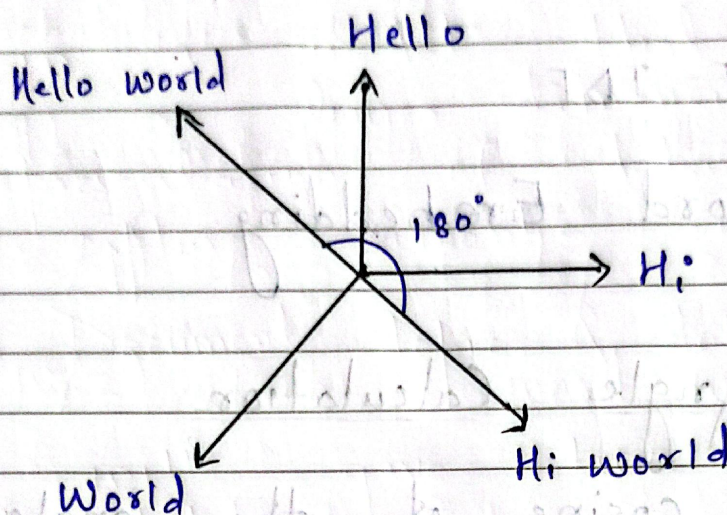
(ii) Orthogonal vector

(iii) Opposite vector.

$$\text{Cosine Similarity}(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|}$$

$$= \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2 \sum_{i=1}^n B_i^2}}$$

Eg:- $D_1 = \text{"Hi, world!"}$
 $D_2 = \text{"Hello, world!"}$



Example

$$A = [1, 2, 3]$$

$$B = [2, 3, 4]$$

$$A \cdot B = (1 \times 2) + (2 \times 3) + (3 \times 4) \\ = 20$$

$$\|A\| = \sqrt{1^2 + 2^2 + 3^2} = \sqrt{14} \approx 3.74$$

$$\|B\| = \sqrt{2^2 + 3^2 + 4^2} = \sqrt{29} \approx 5.39$$

Cosine Similarity

$$\cos(\theta) = \frac{20}{3.74 \times 5.39} = \frac{20}{20.16} \approx 0.99$$

$$\theta = \cos^{-1}(0.99) \approx 8.1$$

So, the vectors are very close (almost in the same direction)

PLSA - Probabilistic Latent Semantic Analysis

It is a probabilistic method to uncover latent topics hidden in a collection of documents.

Each document is represented as a mixture of topics and each topic is a distribution of words.

It helps in topic modeling: finding hidden structures (topics) in large text collections.

It is used to find particular word within a document, it is also called PLST.

PLST: Probabilistic Latent Semantic Indexing

Working and Steps

① Generative Model

It defines the process of each document by generator by mixture of topics and each topic is categories by a distribution of words.

② Latent Variable

A latent variable is a variable in a model that allows for a lower-dimensional representation of data and captures semantic connections.

③ Document topic & topic-word distrⁿ

It will give two set of probability

① $P(T/D)$

② $P(W/T)$

$P(T/D)$: Probability of topic T given document D

$P(W/T)$: Probability of word w given topic T

④ EM - Expectation Maximization

The model parameters are fitted using EM Algorithm.

Application

- Document Analysis
- Topic Discovery
- Statistical & Probability
- Linear Algebra based Research.

PLSA model can be understand
in 2 ways:

- (i) Latent Variable Model
- (ii) Matrix Factorization

(i) Latent Variable Model

A latent variable model is a probabilistic model that introduces hidden variables to explain relationship between observed data.

Variables:

- (i) D : Document = $\{d_1, d_2, d_3 \dots d_n\}$
- (ii) W : Words = $\{w_1, w_2, w_3 \dots w_m\}$
- (iii) Z : Topics = $\{z_1, z_2, z_3 \dots z_k\}$

N = No. of document
 d_i = Bags of words.

//_

Where n is vocabulary size, each word represents Unique terms.

Probabilities used :

$P(d)$ = Probability of choosing a document

$P(z/d)$ = Prob. of topic z given doc. d

$P(w/z)$ = Prob. of word w given topic z .

Common Probability :

$$P(w, d, z) = P(d) * P(z/d) * P(w/z)$$

Marginal Probability :-

$$P(w) = \sum P(w, d, z)$$

for all d, z

Conditional Probability :

$$P(z/d, w) = (P(w/z) * P(z/d)) / P(w)$$

② Matrix Factorization

A matrix factorization model represents data by approximating a large matrix with the product of smaller matrices.

The main component of a term document matrix represent the occurrence of word in the document.

It is usually denoted by $A = N \times M$

$N = \text{Vocab. Size}$

$M = \text{No. of Doc.}$

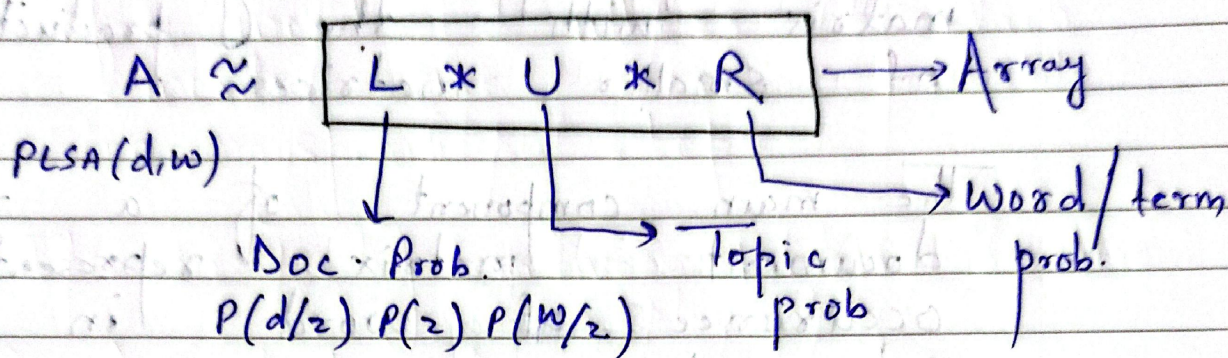
$A(i, j)$ represent the no. of occurrence of word j in doc. i .

We can visualize the matrix.

		Term/Word			
		T_1	T_2	T_3	T_4
Document	D_1	0.2	0.2	0.2	0.2
	D_2	0.2	0.2	0.2	0.2
	D_3	0.2	0.2	0.2	0.2
	D_4	0.2	0.2	0.2	0.2

//_

In PLSA, the document-word matrix A is factorized into:



L = document topic distribution ($N \times K$)

R = topic-word distribution ($K \times M$)

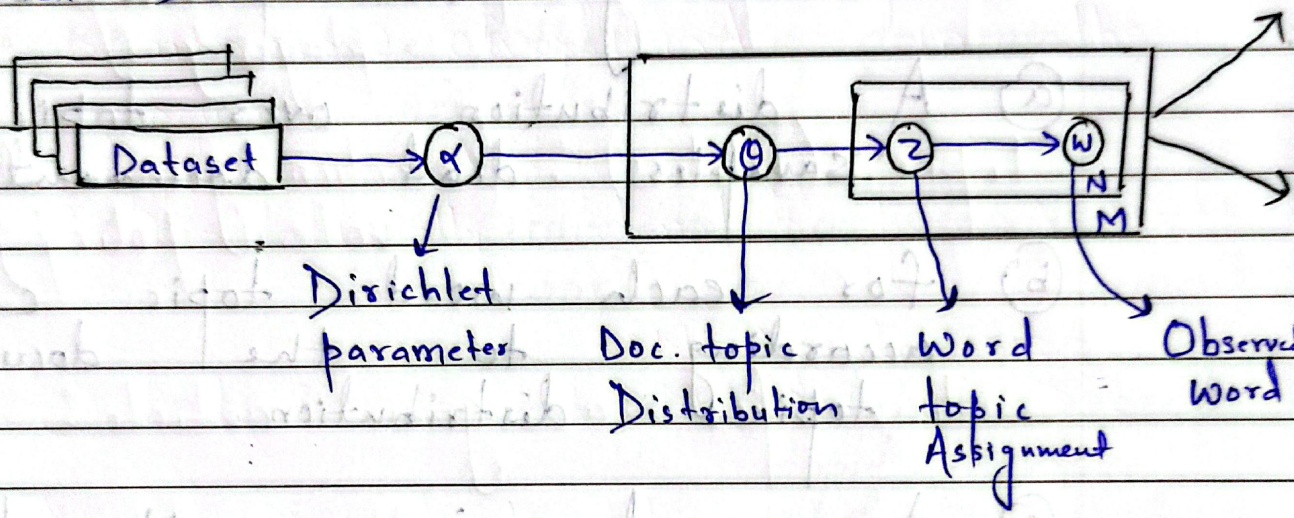
U = diagonal scaling

Meaning: instead of representing documents by thousands of words, we represent them by few latent topics, reducing dimensionality.

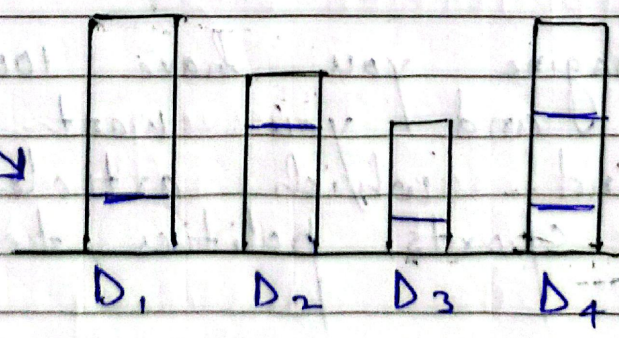
LDA : Latent Dirichlet Allocation

LDA is a topic modeling algorithm used in NLP. It helps in discovering hidden (latent) topics in a collection of documents.

Collⁿ of Text Doc..



① ② ③ ④ ⑤ Creating topic



Frequency of topic per document

//_

It Assume that it is a mixture of various latent (hidden topics). It Also Assume each topic is mixture of word where certain words are more probable in specific topic.

It is generative process because:

- (a) A distribution over topic is samples each document.
 - (b) for each word, topic chosen according to the document topic distribution.
 - (c) A word is sample from chosen topic word distribution.
- LDA does this without supervision (no labels).
 - Imagine you have 1000 news articles, and you want to automatically find which articles talk about sports, politics, technology etc.

How LDA Works

① Choose number of topics (k).

Example: $k=3$ (Sports, politics, technology).

② For each document:

- LDA assumes that the document is generated by:

- Picking a mixture of topics
- For each word in the document

- Randomly choose a topic based on the topic mixture.

- Pick a word from that topic's word distribution.

③ Reverse Engineering (Inference)

- We don't know topics beforehand.

- LDA tries to infer:

- The topic mixture per document.

- The word distribution per topic.

Example :-

Suppose we run LDA on a set of news articles and ask for 2 topics ($K=2$)

It may output :

Topic 1 (Sports) : {team, match, goal, win, player}

Topic 2 (Politics) : {election, government, party, vote, policy}

Now for a given article :

• "The team won the match after election day"

• LDA might say : 60% Sports + 40% Politics

Application :

- i Topic Discovery
- ii Information Retrieval
- iii Text Summarization
- iv Doc. Clustering and Classification
- v Recommendation Systems

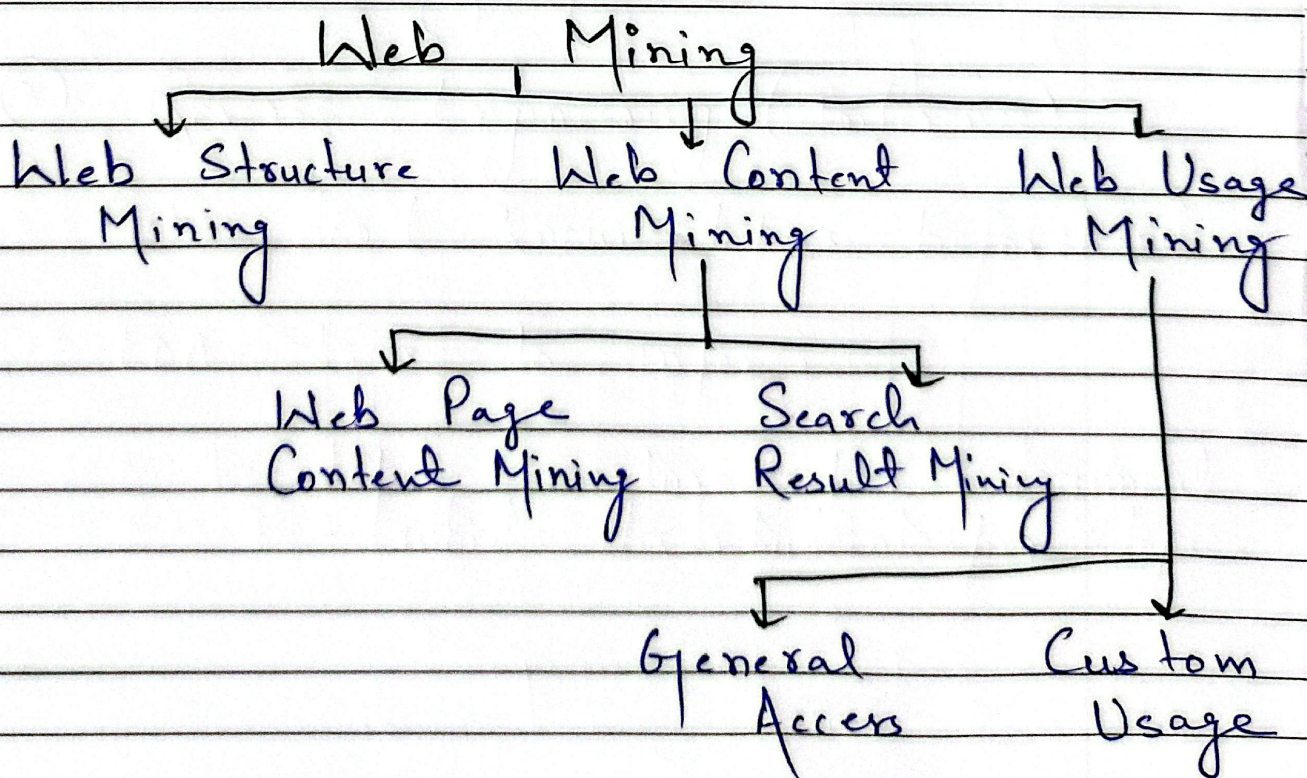
Web Mining

Web Mining is the process of applying data mining techniques to discover useful information and knowledge from the World Wide web (WWW).

It helps extract patterns, trends and insights from web data, which can be content (text, images, videos), structure (links, hierarchy) or usage (logs, clicks, visits).

Web Mining Techniques

- ① Sequential Pattern
- ② Association Mining & Clustering principle
- ③ Data Retrieval



UNIT-3

___/___/___

Sentiment Analysis

Sentiment Analysis is a branch of NLP that focuses on determining the emotional tone, attitude or opinion expressed in a piece of text.

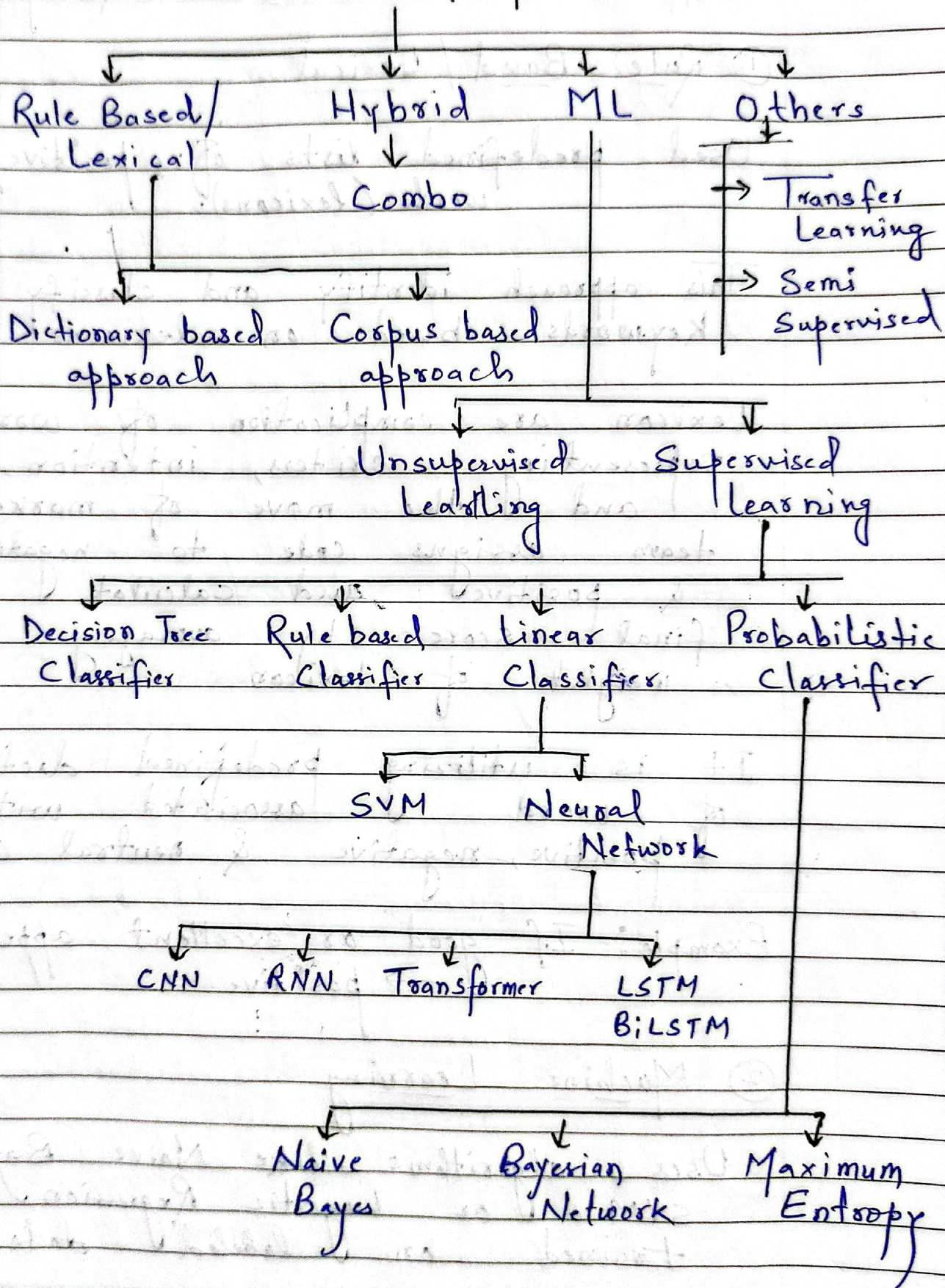
It helps machine understand whether the text expresses a positive, negative or neutral sentiment.

Sentiment Analysis is the computational study of people's emotion, opinions, sentiments, attitudes and emotions expressed in text.

For example :-

- This phone is amazing → Positive sentiment
- The movie was terrible → Negative sentiment
- It's an average product → Neutral sentiment

Sentiment Analysis



Approaches to Sentiment Analysis

① Rule Based / Lexical

Used predefined lists of positive/negative words (lexicons).

This approach identify and classify keywords based on lexicons.

Lexicon are compilation of words, representing writers, intention, emotions and the move of marketing team assigns code to negative & positive and calculate the final score by averaging the weight of lexicon.

It is utilizing predefined dictionary of word associated with positive, negative & neutral category.

Example:- If "good" or "excellent" appears
→ positive

② Machine Learning

Uses algorithms like Naive Bayes, SVM or Logistic Regression trained on labeled data.

③ Deep Learning

Uses neural network (RNN, LSTM, Transformers like BERT) to capture content and improve accuracy.

④ Hybrid - Combine both the approaches (Rule based + ML)

⑤ Other

Libraries and Tools

① NLTK

② VADER - Valence Aware Dictionary for Sentiment Reasoning

③ TextBlob

④ spacy

⑤ Hugging Face Transformers

Levels of Sentiment Analysis

① Document - Level :

Determines the overall sentiment of a full document or review.

Example :- I love this restaurant.
The food is great.

↳ positive

② Sentence - level :-

Analyzes sentiment in each sentences.

Ex:- The screen is good, but the battery drain fast.

↳ positive

↳ Negative

③ Aspect level (Feature based) :-

Focuses on specific aspects or features.

Ex:- The camera quality is excellent, but the battery life is poor.

• Camera → positive

• Battery → Negative

Steps Involved in Sentiment Analysis

① Data Collection :- Gather text data
(tweets, reviews, comments)

② Data Preprocessing :- Clean text (remove stopwords, punctuation, emojis, etc.)

③ Tokenization :- Split sentences into words.

④ Feature Extraction :- Convert text into numerical form.
(Bag of Word, TF-IDF)

⑤ Model Training :- Train ML/DL models.
(e.g- Naive Bayes, Logistic Reg.)

⑥ Prediction :- Classify new text as Positive, Negative or Neutral.

⑦ Evaluation :- Measure accuracy, precision, recall, F1-score.

Why Sentiment Analysis is Important

(i) Business Intelligence :- Companies use it to analyze customer feedback and product reviews

(ii) Social Media Monitoring :- Helps in understanding public opinion about a brand or topic.

(iii) Customer Support :- Automatically classifies user feedback as positive or negative.

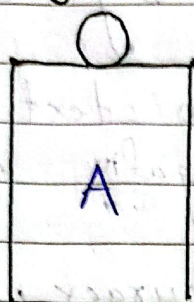
(iv) Political Analysis :- Used to study public sentiment about policies or candidates.

(v) Market Research :- Understand trends and preferences through online discussions.

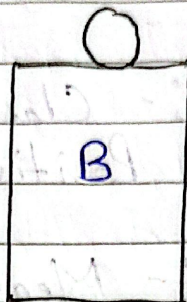
Negative

Positive

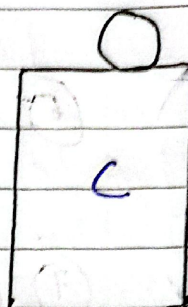
Neutral



Lemon



Levander



Rose

→ Waste of money	→ Totally Worth it	→ Okay Okay
→ Not good smell	→ Smell Awesome	→ Smell Nice,
→ Waste product	→ Fabulous product	→ Not Strong

Example Using NLTK - VADER

```
from nltk.sentiment import SentimentIntensityAnalyzer
```

```
sia = SentimentIntensityAnalyzer()
```

```
text = "The movie was fantastic! The acting  
was great but the ending was  
sad."
```

```
score = sia.polarity_scores(text)
```

```
print(score)
```

Output :-

```
{ 'neg': 0.17, 'neut': 0.43, 'pos': 0.40,  
  'compound': 0.51 }
```

→ Compound = 0.51 > 0 → sentiment is positive

Sentiment Classification

Sentiment Classification is a key part of Sentiment Analysis, where the goal is to automatically categorize text (such as a sentence, tweet or review) into sentiment classes — for example, Positive, Negative or Neutral.

Sentiment Classification is the process of assigning a sentiment label (like positive, negative or neutral) to a given piece of text based on its expressed opinion or emotion.

Example :-

I Love this phone → Positive

The service was awful → Negative

The product is okay → Neutral

Goal of Sentiment Classification

The main goal is to train a model that can predict sentiment for new, unseen text based on patterns it learned from previously labeled data.

Types of Sentiment Classification

① Binary Classification

- Two classes: Positive and Negative
- Ex - Movie reviews (Good or Bad)

② Multi-Class Classification

- More than two classes (e.g., Positive, Negative, Neutral)
- Ex - Product feedback

③ Fine-Grained Classification

- Multiple levels of polarity
- Ex - 5 star rating scale

④ Aspect Based Classification

- Analyzes sentiments towards specific features or aspects of a product.

Example:- The camera is great but the battery life is short

Camera → Positive

Battery → Negative

Approaches to Sentiment Classification

① Lexicon Based Approach

- Uses a dictionary (lexicon) of words labeled with their sentiment polarity.

Ex:- Excellent $\rightarrow +3$
Bad $\rightarrow -2$

Sentiment = Sum of all word score.

② Machine Learning Approach

- Uses supervised learning algorithms trained on labeled text data

Ex:- Algorithm - Naive Bayes, SVM, Logistic Reg., Decision Tree

③ Deep Learning Approach

- Uses neural networks that capture context and sequence in language

Example models:- RNN, LSTM, GRU, Transformers

Opinion Mining

Opinion Mining is a subfield of Sentiment Analysis that focuses on extracting and analyzing opinions — What people think or feel about specific entities, products, events or topics.

It doesn't just find whether a text is positive or negative, but also what the opinion is about and who expressed it.

Sentiment Analysis = What's the emotion?

Opinion Mining = Who said what, about what and how do they feel?

Example :- The camera of this phone is amazing, but the battery drains too fast.

Opinion Mining Output :-

<u>Entity</u>	<u>Aspect</u>	<u>Opinion</u>	<u>Sentiment</u>
Phone	Camera	amazing	Positive
Phone	Battery	drains too fast	Negative

//_

Opinion Mining process text to identify & classify sentiments which can be broken down into 4 parts:

- (i) Identifying the target
- (ii) Assessment Extracting
- (iii) Classifying Sentiment
- (iv) Determine Opinion above the target
(Aspect based sentiment Analysis)

Types of Opinion Mining

① Document Level Opinion Mining

Determines the overall sentiment of a whole document or review.

② Sentence Level Opinion Mining

Analyzes each sentence to find if it contains an opinion and its polarity.

③ Aspect Level Opinion Mining

Focuses on specific features or attributes of an entity.

Ex: The laptop's design is beautiful, but it heats up quickly.

Design → Positive
Heating → Negative

Techniques

- ① Lexicon / Rule Based
- ② ML - Machine Learning
- ③ POS
- ④ DL - Deep Learning

Steps in Opinion Mining

- ① Data Collection
- ② Preprocessing
- ③ Opinion Detection
- ④ Feature Extraction
- ⑤ Sentiment Classification
- ⑥ Summarization

Applications

- i Product Review
- ii Social Media Analysis
- iii Business & Marketing Research
- iv Reputation Management
- v Recommendation Systems
- vi National Security

Python NLTK Sentiment Analysis

NLTK Sentiment Analysis is the use of NLTK sentiment modules (like VADER or Naive Bayes Classifier) to classify text as positive, negative or neutral.

Approaches in NLTK Sentiment Analysis

NLTK supports two major approaches.

① Lexicon-Based Approach (VADER)

- Uses predefined sentiment dictionary

② Machine Learning Approach

- Trains models (like Naive Bayes) using labeled text data.
- Requires feature extraction and supervised learning.

① Lexicon-Based using VADER

VADER stands for Valence Aware Dictionary and sentiment Reasoner

It is specifically tuned for analyzing sentiments in social media text (tweets, comments, etc.).

Code

```
import nltk
nltk.download('vader_lexicon')

from nltk.sentiment import SentimentIntensityAnalyzer

sia = SentimentIntensityAnalyzer()

sentences = [
    "I love this phone! It's so fast and beautiful.",
    "The camera quality is bad and the battery drains quickly.",
    "The product is okay, not too good, not too bad."
]

for text in sentences:
    score = sia.polarity_score(text)
    print(f"Text: {text}")
    print(f"Scores: {score}")
```

Output

```
Text: I love this phone
Scores: {'neg': 0.0, 'neu': 0.33, 'pos': 0.67, 'compound': 0.89}

Text: The camera quality
Scores: {'neg': 0.53, 'neu': 0.47, 'pos': 0.0, 'compound': -0.67}

Text: The product is
Scores: {'neg': 0.10, 'neu': 0.75, 'pos': 0.15, 'compound': 0.05}
```


- _/_/_
- Positive Sentiment : $\text{Compound} \geq 0.05$
 - Negative Sentiment : $\text{Compound} \leq -0.05$
 - Neutral Sentiment : otherwise

② Machine Learning Approach

Code : Using Naive Bayes Classifier

```
import nltk
from nltk.corpus import movie-reviews
import random
```

```
nltk.download('movie-reviews')
```

```
documents = [(list(movie-reviews(fileid)), category)
               for category in movie-reviews.categories()
               for fileid in movie-reviews.fileids(category)]
```

```
random.shuffle(documents)
```

```
# Extract features
```

```
all-words = nltk.FreqDist(w.lower() for
                           w in movie-reviews.words())
word-features = list(all-words)[:2000]
```

```
def document-features(document):
    words = set(document)
    features = { }
```


//_

```
for word in word_features:
    features[f'contains({word})'] =
        (word in words)
return features
```

```
featuresets = [(document_features(d), c) for
                (d, c) in documents]
```

```
# Split data
```

```
train_set, test_set = featuresets[100:],
                             featuresets[:100]
```

```
# Train
```

```
Classifier = nltk.NaiveBayesClassifier.train
              (train_set)
```

```
print("Accuracy:", nltk.classify_accuracy
      (Classifier, test_set))
```

```
Classifier.show_most_informative_features(5)
```

∴ Output :-

Accuracy : 0.83

Most Informative Features

contains (outstanding) = True → Positive
contains (seagal) = True → Negative
contains (wonderfully) = True → Positive
contains (mess) = True → Negative
contains (ludicrous) = True → Negative

Recommendation System

A Recommendation System is an AI tool that analyzes user behaviour and preferences to suggest relevant items such as products, movies, music, books or news.

A Recommendation System is a type of information filtering system that predicts and suggests items a user might like, based on their past behaviour, preferences or similar user interests.

Types of Recommendation System

① Content Based Filtering

- Recommends items similar to those the user liked in the past.
- Works by analyzing item features and user preferences.

Example :-

If you liked a movie with "action" and "Tom Cruise,"

→ it will recommend another action movie starring Tom Cruise.

② Collaborative Filtering

- Recommends items based on similar users preferences.
- It assumes that users with similar past behaviour will like similar items.

Example:- If User A and B both liked movies 1, 2 and 3 and User A liked movie 4 → Recommend movie 4 to User B.

Two types:-

- ① User-User Collaborative Filtering
- ② Item-Item Collaborative Filtering

③ Hybrid Recommendation System

- Combines Content-Based and Collaborative Filtering.
- More accurate and avoids individual limitations.

Example:-

Netflix uses both:-

- Content based → compares movie genres and actors you like
- Collaborative → compares your preferences with other similar users.

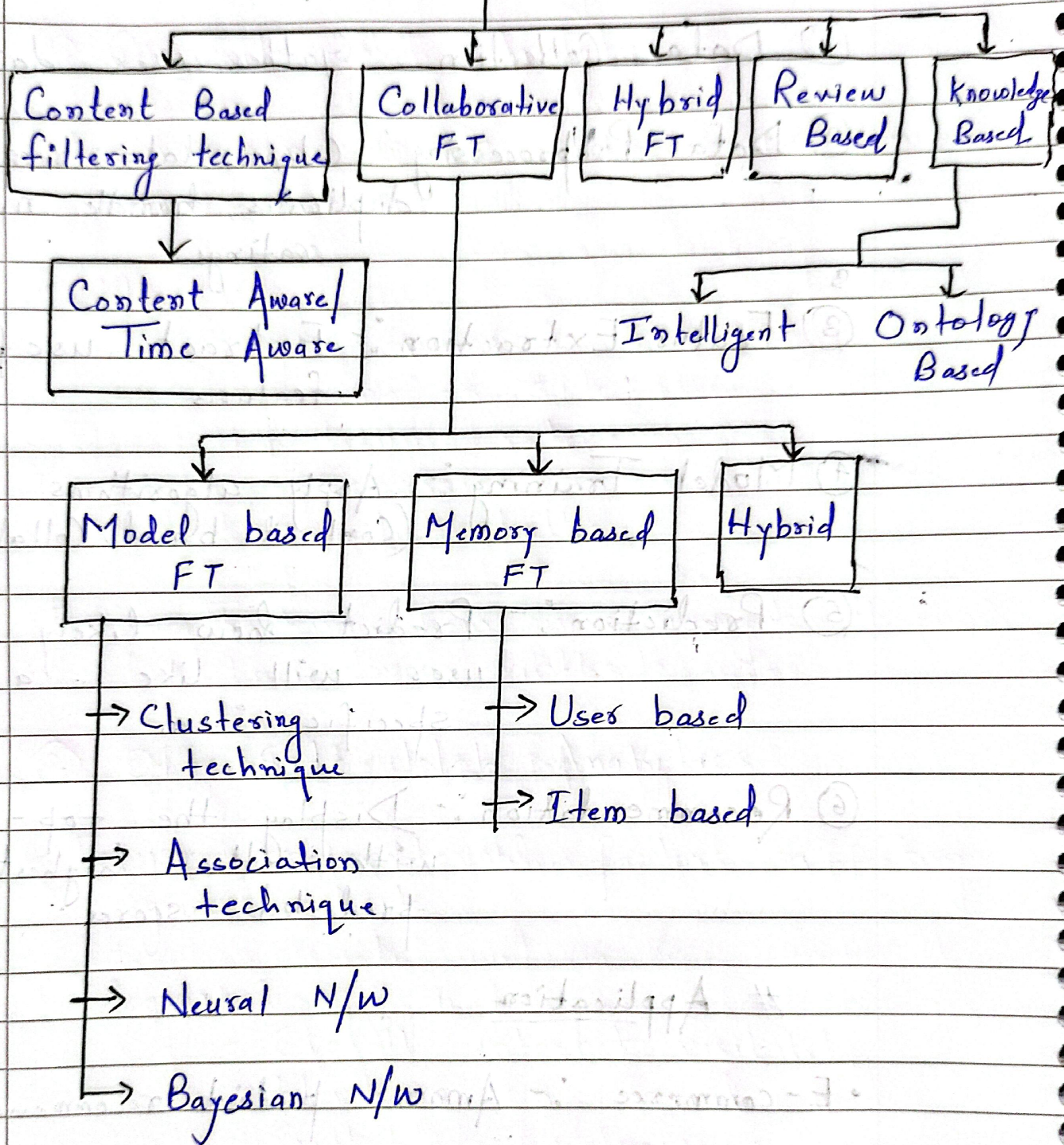
How it Works

- ① Data Collection :- Gather user data
- ② Data Preprocessing :- Clean data, remove duplicates, handle missing ratings.
- ③ Feature Extraction :- Extract useful features
- ④ Model Training :- Apply algorithms (Content based, Collaborative)
- ⑤ Prediction :- Predict how likely a user will like a specific item.
- ⑥ Recommendation :- Display the top-N items with the highest predicted scores.

Application

- E-commerce :- Amazon product recommendation
- Entertainment :- Netflix, Youtube, Spotify suggestion
- Social Media :- Instagram "Suggested for You"
- Education :- Course recommendation

Recommendation System



UNIT-4

__/__/__

Web Analytics

Web Analytics is the process of collecting, measuring, analyzing and reporting web data to understand and optimize web data usage.

It helps website owners, businesses and organizations to track user behaviour, improve website performance and make data driven decisions.

Purpose of Web Analytics

- To understand user behaviour on a website (who visits, what they do)
- To measure website performance (traffic, engagement, conversions)
- To identify strengths and weaknesses of a website or campaign.
- To optimize marketing efforts (like SEO, ads or content strategies)
- To increase conversion rates (turning visitors into customers)

Types of Web Analytics

① On-Site Web Analytics

Focuses on what happens on your own websites.

• Measures:

- Page views
- Time on site
- Bounce rate
- Conversion rate
- Click paths

Ex:- Tracking how users navigate through your website.

② Off-Site Web Analytics

Focuses on external factors affecting your website.

• Measures:

- Overall internet visibility
- Mentions on social media
- Competitor analysis
- Search engine ranking

Ex:- Monitoring how often your site is mentioned on other websites.

How it Works

① Data Collection

Gather raw data about website visitors

② Data Processing

Cleaning and organizing the raw data.

③ Data Analysis

Interpreting metrics and patterns
(like visitor trends, bounce rate)

④ Reporting

Presenting the insights through dashboards, charts or reports.

⑤ Decision Making and Optimization

Using insights to improve website design, content and marketing strategy.

Advantages

- Better understanding of audience behaviour
- Improved website performance
- Effective digital marketing strategy
- Higher Conversion rates

Web Mining

Web Mining is the process of discovering useful patterns, information and knowledge from the web.

It applies data mining techniques to web data such as web content, web structure and web usage.

It helps in understanding:

- User behaviour
- Trends and patterns on the internet
- And improving decision-making for businesses and organizations.

Types of Web Mining

① Web Content Mining

Extracting useful information from the content of web pages (text, images, videos, etc)

Ex:- Extracting product reviews or news articles

② Web Structure Mining

Analyzing the link structure of websites to discover relationship

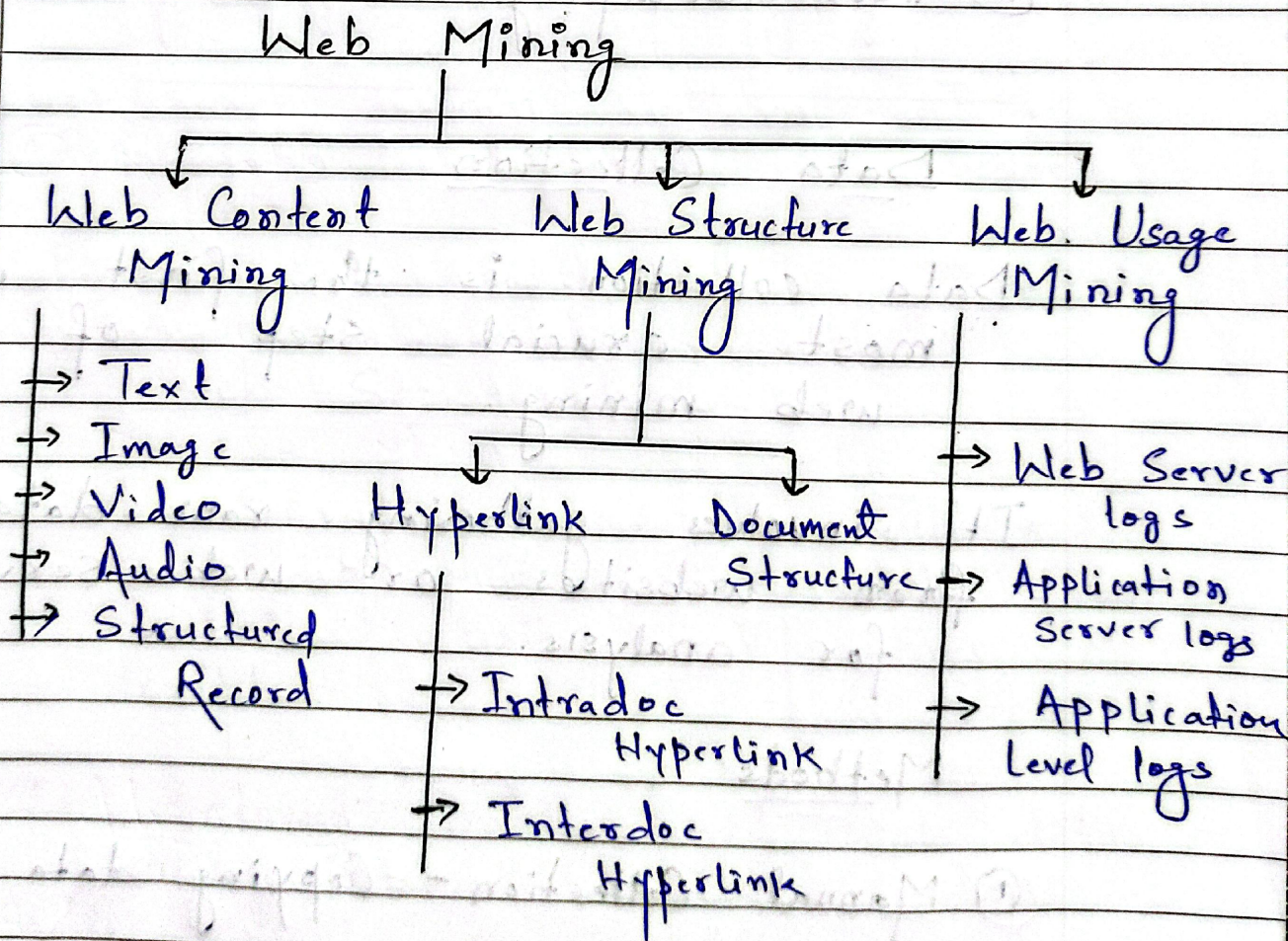
between web pages.

Ex:- Page Rank algorithm used by Google.

③ Web Usage Mining

Analyzing user behaviours by examining web server logs or user sessions.

Ex:- Studying user click paths and visit durations.



Web Mining Process

- ① Data Collection
- ② Data Preprocessing
- ③ Data Analysis / Pattern Discovery
- ④ Pattern Evaluation
- ⑤ Knowledge Presentation

Techniques of Web Mining

- i Data Collection
- ii Web Scraping

Data Collection

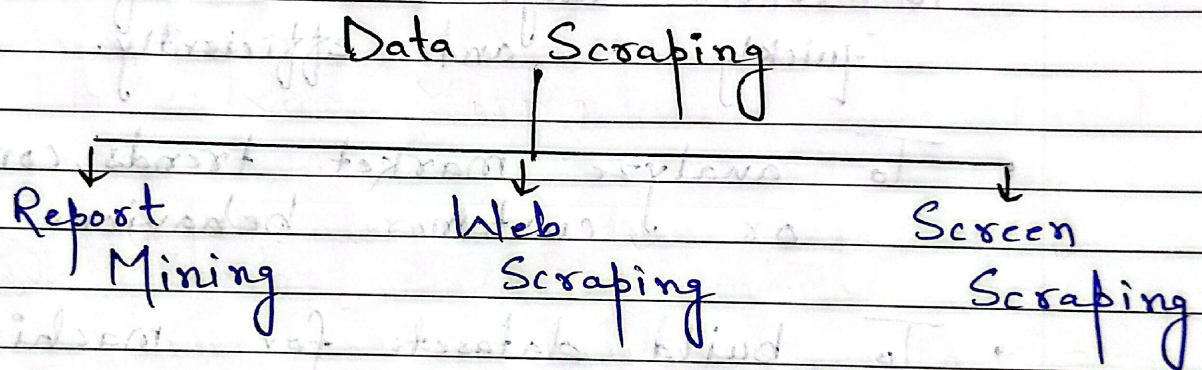
Data collection is the first and most crucial step of web mining.

It involves gathering raw data from websites or web services for analysis.

Methods

- ① Manual Collection - Copying data by hand
- ② Automated Collection - Using automated tools or scripts like:

- _/_/_
- Web Crawlers : Program that browse the web automatically to collect pages.
 - APIs : Many websites provide APIs to fetch structured data.
 - Web Scrapers : Custom scripts that extract specific data from websites.



Web Scraping

Web Scraping is the automated process of extracting data from websites using programs or scripts.

It convert unstructured web data (HTML) into structured data (like CSV, Excel or database) for analysis.

//_

It is like using a robot to read a website and collect useful information - such as product prices, reviews, news or statistics - automatically.

Purpose of Web Scraping

The main goals of web scraping are:

- To collect large amount of data quickly and efficiently.
- To analyze market trends, competitors or customer behaviours.
- To build datasets for machine learning, research or sentiment analysis.
- To monitor changes on websites (like price drops or news updates).

How it Works

① Sending an HTTP Request

- The scraper sends a request to a website's URL.
- Example: `https://www.taquexalam.com`

② Downloading the Web Page

The website sends back the HTML content of that page.

③ Parsing the HTML

The scraper reads and analyzes the HTML structure to find the desired data (like text, images or links)

④ Extracting Data

It extracts the relevant data based on tags (e.g., `<div>`, `<a>`, `<span>`)

⑤ Storing Data

The extracted data is stored in structured formats - such as

- CSV
- Databases (MySQL, MongoDB)
- JSON files

Methods of Web Scraping

① API,

② Pre built tools and Extension

③ Custom Code (e.g., python)

Tools and Libraries

- ① Beautiful Soup :- Parses and extracts data from HTML or XML
- ② Scrapy :- Full-Fledged web scraping framework.
- ③ Selenium :- Automates browsers (e.g. Chrome, fire fox).
- ④ Requests :- Sends HTTP requests and receives web pages.

Legal and Ethical Aspects

- Always follow a site's robot.txt file (it tells what's allowed or disallowed for bots)
- Avoid scraping personal or copyrighted data.
- Respect rate limits to prevent overloading a website's server.

Applications of Web Scraping

- i) SEO - Search Engine Optimization
- ii) Market Research
- iii) Social Media

- _/_/_
- ④ Job Portals
 - ⑤ Price Competition
 - ⑥ Financial Marketing

Code

```
import requests
from bs4 import BeautifulSoup
```

```
# Send a request to website
url = "https://taqueeralam.com"
response = requests.get(url)
```

```
# Parse the HTML
soup = BeautifulSoup(response.text,
                      'html.parser')
```

```
# Extract all links
```

```
for link in soup.find_all('a'):
    print(link.get('href'))
```

Output

A list of all URLs present on that page.

Ranking Techniques

In web mining and search engines, ranking techniques help determine the importance or relevance of web pages.

When you search something on Google, the top results are not random - they are ranked using algorithms like PageRank and HITS that analyze the link structure of the web.

Webpage Ranking Technique

Every web page can be connected to others through hyperlinks.

The basic idea is:

- A web page is considered important if many other important pages link to it.

This concept is called link analysis - and it is the foundation of:

- PageRank
- HITS - Hyperlink Induced Topic Search

Page Rank

PageRank is an algorithm developed by Larry Page and Sergey Brin (Google founder) to measure the importance of web pages based on incoming links.

Basic Idea:

- A page has a high rank if many pages link to it.
- Links from important pages contribute more value than links from less important ones.

So, the rank of a page depends on both:

- ① The number of links it receives
- ② The quality (importance) of those linking pages.

Mathematical Formula

$$PR(A) = (1-d) + d \cdot \sum_{i=1}^n \frac{PR(T_i)}{C(T_i)}$$

Where:- $PR(A)$ = PageRank of page A

d = damping factor

T_i = pages that link to A

$C(T_i)$ = No. of outbound links from page T_i

$(1-d)$ = probability that a user randomly jumps to another page.

Steps in PageRank Algorithm

- ① Initialize all pages with equal rank.
- ② Iteratively update ranks using the formula above.
- ③ Continue until the ranks converge. (do not change much)

Example:-

Suppose we have 3 pages:-

- A links to B and C
- B links to C
- C links to A

After a few iterations, PageRank values might be:

$$PR(A) = 0.37$$

$$PR(B) = 0.22$$

$$PR(C) = 0.41$$

Thus, C is the most important page.

Advantages of PageRank

- Considers both quantity and quality of links.
- Works globally
- Resistant to spam to some extent.

Limitations

- Doesn't consider content relevance (only links)
- Vulnerable to link farming (artificial link N/w)
- Needs iterative computation (time consuming)

HITS - Hyperlink Induced Topic Search

HITS is a link analysis algorithm developed by Jon Kleinberg in 1999 that ranks web pages based on their link structure.

It identifies two types of web pages:

- Authorities :- Pages that contain valuable information.
- Hubs :- Pages that link to many good authorities.

Basic Idea :

- A good hub points to many good authorities.

- A good authority is pointed to by many good hubs.

Hence, hub and authority scores reinforce each other.

Mathematical Representation

For each page p :

$$A(p) = \sum_{q \in I(p)} H(q)$$

$$H(p) = \sum_{q \in O(p)} A(q)$$

Where: $A(p)$ = Authority score of page p

$H(p)$ = Hub score of page p

$I(p)$ = Set of pages linking into p

$O(p)$ = Set of pages p links out to

Steps in HITS Algorithm

① Collect a set of relevant pages for a given query.

② Initialize all authority and hub scores to 1.

③ Iteratively update:

- Each page's authority score = sum of hub scores of page linking to it

- _/_/_
- Each page's hub score = sum of authority scores of pages it links to

④ Normalize scores after each iteration

⑤ Repeat until scores converge

Example :- Suppose :-

- Page A links to B and C
- Page B links to C
- Page C links to none

Then :-

A $\rightarrow$ Hub (links to other)

C $\rightarrow$ Authority (receives links)

B $\rightarrow$ Acts as both hub and authority

Advantages of HITS :-

- Distinguishes between hubs and authorities
- Works well for focused topics or small set of pages.
- Useful for topic specific ranking.

Limitations :-

- Computationally expensive
- Depends on initial query results.
- Can be affected by link spam.
- Sensitive to topic drift.
(irrelevant pages can get ranked)

How to Improve Webpage Ranking

There are methods like SEO, Google Ads which can improve the position on search engine or Google.

It can be improved by :-

- ① Keywords should be very creative
- ② Publish unique and genuine content
- ③ Focus on creating relevant links
- ④ Update your website content after specific time.
- ⑤ Describe your images and videos using alt tags.

Strategies

- i) Keyword Search
- ii) High Quality Content
- iii) Mobile friendly
- iv) Build High Quality Links
- v) On page Optimization - metadata, title, description
- vi) Website Speed
- vii) User Experience
- viii) Backlinks
- ix)

SEO

└─> Local
└─> Technical
- x) EAT - Expertise, Authoritativeness and Trustworthiness