E) optical character Recognition (OCR)

→ Reading printed Text.

* Converts scanned text into editable text.

Input : A scanned book page.

output : Digital text format.

Used in : Digitizing old books, invoice scanning.

→ Tokenizer

A Tokenizer is a tool that breaks text into smaller parts called tokens. These tokens can be words, subwords (or) characters.

I Love NLP.

Ex : "I" "love" "NLP"

Types

1. Word Tokenization : splits text into words

Ex: AI is powerful

"AI" "is" "powerful".

2) Subword Tokenization: Breaks words into smaller meaningful parts.

Ex: "Unhappiness" → ["un", "happiness"]

Used in: GPT, BERT, and Transformer models.

3) Character Tokenization: splits text character by character.

"hello" → ["h", "e", "l" ...]

useful for handling unknown words.

4) Sentence Tokenization: splits text into into sentences

Ex: "Hellow world. How are you?

"Hello world"., "How are you".

Techniques Tokenization.

1. Rule-Based Tokenization.

* uses grammer rules (like spaces, punctuation) to split text.

Ex: splitting words by spaces in English.

* Doesn't work well in chinese (or) Japanese.

2) Regular Expression (Regex) Tokenization

* uses patterns to find words.

Ex: \wt -> Find all words in a sentence.

3) Byte pair Encoding (BPE) Tokenization.

* used in GPT and BERT models.

* merges common letters pairs to form subwords.

Ex: Lowest" ["low", "est"].

4) unigram Tokenization.

* Breaks words into multiple subword options and picks the best one.

Ex: "Internationalization" → ["inter", 'nation al', 'ization"]

challenges in Tokenization

1. Handling multiple meanings (Ambiguity)

Ex: I saw the man with a tele scope.

2. Dealing with compound words.

Ex: New York → should it be ["New", "York"] (or) ["New York"]

3. Languages without spaces

* Chinese.

4) Handling special characters & emojis,

Ex: "python is awesome! 😊"

Detecting and Correcting spelling Errors.

When, we type, we sometimes make spelling mistakes, computers use spell checkers to detect and correct these errors automatically. This process is used in word processors, search Engines, and chatbots.

1. Types of spelling Errors:

   1. Typographical Errors.

      Mistakes from fast (or) incorrect typing.

      Ex: 'hte' → 'the'.

   2. phonetic Errors → when a word is spelled as it sounds.

      frend → "friend".

3. Omission Errors : Leaving out letters.

"recieve" — "receive"

4. Insertion Errors : Adding extra letters.

Ex : "happenn" → "happen"

5. Transposition Errors — swapping two letters.

Ex : "ten" → "the"

6. Homophone Errors : Words that sound the same but are spelled differently.

"there" vs. "their"

2) Techniques for Detecting spelling Errors

* Dictionary - Based Approach.
  * compares words to a predefined word list.
  * If a word isn't found, it is marked as incorrect.

Ex :- If "recive" is not in the dictionary, it is flagged as a mistake.

* Edit Distance.
  * Measures how many changes (insert, delete, replace) are needed to fix a word.

Ex :- "hte" → "the" (1 change: swap 'h' to 't')

* N - gram Analysis
  * Breaks words into small parts (n-grams) and checks common patterns,

4) probabilistic Language models.

* uses models like n-grams (or) deep learning to check if a word fits well in a sentence.

5) Neural networks & AI Based Approaches.

* uses deep learning models (like GPT, BERT) to predict the correct word.

"I wnet to teh market"

→ I nent to the market.

Techniques for correcting spelling Errors.

1. Rule-Based correction.

* uses grammer rules to fix. mistakes.

Ex: "I before e except after c" corrects "recieve" → "receive".

2) Edit Distance correction.

* Suggests words that need the fewest changes.

Ex : "hte" → "the" (1 change).

3) Context - Based correction.

Looks at the entire sentence to fix mistakes.

Ex : "Their going to the park"

"They're going to the park".

4) Auto - complete & Auto _ correct.

* Suggests and fixes words while typing.

"definately" → "definitely".

Minimum Edit Distance

Minimum Edit distance is a way to measure how different two words (or text) are counting the smallest number of changes needed to turn one word into another. They include.

1. Insertion → Adding a letter.
   Ex :- "aple" → "apple" (insert "p").
2. Deletion → Removing a letter
   Ex :- "hello" → "helo" (delet "l").
3. Substitution → changing one letter to another.
   Ex :- "cat" → "bat" (change "c" to "b")

Advantages of Minimum Edit Distance :-

* Simplicity :- Easy to understand and implement.
* Versatility :- Applicable across multiple NLP domains.

optimality :- Guarantees the minimum number of edits.

Challenges with minimum Edit Distance

* computational complexity :

O(m×n) where m and n are the lengths of the strings.

* Context Ignorance : Does not consider word Semantics.

* Handling Large Datasets :- Inefficient for real-time spell correction in large corpora.

Applications of Minimum Edit Distance in NLP.

* Spell checking : Suggesting corrections based on edit distance.

2) Machine Translation :- Aligning phrases between languages.

3) Plagiarism Detection :- Comparing text similarity.

4) Speech Recognition :- Aligning predicted and transcribed words.

5) Bioinformatics :- Comparing DNA sequences.

Regular Expressions (Regex).

* A Regular Expression (Regex) is like a smart search tool. that helps find and match patterns in text.
* It is used for searching, replacing and validating text in programming and data processing.

Basic Regex patterns.

Pattern	Meaning	Example.
.	Any character.	"c.t" → Cat, cot, cot.
^	Start of a line	^Hello → Matches "Hello" only if it's at the start.
$	End of a line.	end$ → Matches "end" only at the end.
\d	Any digit (0-9).	"I have 3 apples" → Matches "3"
\w	Any letter, digit or underscore.	"hello_123" → Matches everything.
\s	Any space, tab, or newline.	"hello world" → Matches the space.
*	Repeat 0 or more times	"go*d" → "gd", "god", "good", "goooood"!

+	Repeat 1 (or) more times	"go+d" → "god", "good", "gooood" (not "gd").
?	optional (0 or 1 time)	"colou?r" → Matches "colour" and "color".
{n}	Repeat exactly n times	"a{3}" → Matches "aaa".
{n,m}	Repeat between n and m times	"a{2,4}" → Matching "aa", "aaa", "aaaa".

Python Regex

Python has a built-in module called re that lets you use Regular Expressions (Regex) to search, match, and manipulate text easily.

2) Basic Regex Functions in python.

re.search()

* Finds the first match in a string.

re.search("cat", "The cat sleeps").

re.findall()

* Finds all matches in a string.

re.findall("\d+", "price: 100, Tax: 20")

re.match()

* checks if a string starts with a pattern.

re.match("Hello", "Hello world")

re.sub()

* Replaces a pattern with new text.

re.sub("blue", "red", "blue sky")

re.split()

* Splits a string using a pattern

re.split("\s+", "Hello world")

UNit-2.

① UNderstanding N-grams.

N-grams are essential for understanding language patterns and structure. They help in predicting the next item in a sequence.

② N-grams comparison.

1. unsmoothed N-Grams (No smoothing).

* uses raw counts from the training data.
* If an N-gram wasn't seen before, its gets a probability of zero. meaning the model fails.

Ex:- (Bigram model).

Training data :- "I love NLP"

If we ask for "I love AI" - Fails.

(Since "love AI" was never seen).

② Smoothed N-Grams (with smoothing).

* Adjusts probabilities so unseen word combinations don't get zero probability.

* Helps make better predictions.

Ex:- Laplace Smoothing:- Adds a small value (like +1) to all word counts.

* Backoff & Interpolation:- uses smaller N-grams (bigram - unigram) when needed.

N- Grams models.

An N-Gram model is a type of language model that predicts the next word based on the previous words using N-grams.

1. Unigram model (N = 1)

* predicts the next word based only on individual word frequency.

Ex: "I love" → predicts next word only based on the most frequent word in the dataset.

② Bigram model (N = 3).

* predict the next word based on the previous word.

Ex: If "I love" is common, the next word might be "cybersecurity"

"I love cybersecurity" appears frequently

③ Trigram model (N = 3).

* predict the next word using two previous words.

Ex: "I love cybersecurity" → The model predicts the next word based on.

"love cyber security".

* Train and Test Corpora.

When building a machine learning (or) NLP model (like an N-gram model), we need data to train and evaluate it. This data is split into two main parts.

1. Training corpus (Train Data)
2. Testing corpus (Test Data).

1. Training Corpus (Train Data).

* This is the main dataset used to teach the model.
* The model learns patterns from this data.

Ex :- If you're training an autocomplete system, the training data would be books, articles, (or) chats.

Think of it as :- studying for an exam using textbooks.

2) Testing corpus (Test Data)

* This is a separate dataset used to check how well the model learned.
* The model does not see this data during training.

Ex :- If you trained an N-gram model (or) news articals, the test set might be new articales from a different website

Think of it as :- Taking an exam to see how well you studied.

Unknown words.

When we test an N-gram model (or) any NLP model, sometimes the test data contains words that were never seen in the training data. These called unknown words (oov - out of vocabulary words).

Handle unknown words?

1. special TOKen ("<UNK>")

   * During training, replace rare words with a placeholder like <UNK>.
   * When an unknown word appares in the test set, the model treats it as <UNK>.

2. smoothing Techniques.

   * Techniques like Laplace smoothing (or) Backoff help assign small probabilities to unknown words.

Perplexity.

Perplexity is a way to measure how good (or) bad a language model (like an N-gram model) is at predicting text.

* Low perplexity :- The model is confident and makes good predictions.

* High perplexity :- The model is confused and makes bad predictions.

smoothing

smoothing is a technique used in N-gram models to handle Zero probability problems.

Techniques

1. Laplace smoothing (. Add-one smoothing)

   * Adds 1 to every word count to prevent Zero probability.

2. Add-K smoothing

   * similar to Laplace, but instead of adding 1, we add a small value k (like 0.01 (or) 0.001).

3.) Backoff smoothing.

* If a higher-order N-gram (like a trigram) is missing, it "backe off" to a lower N-gram (like a bigram or unigram).

4) Interpolation Smoothing.

* combines different N-grams (trigram, bigram, unigram) instead of just back off.

* more advanced and balances short term and long-term dependencies.

5) Kneser-Ney smoothing

* one of the most powerful smoothing techniques.

* Instead of just adjusting probabilities. It also considers how diverse the word usage is.

POS Tagging

POS (part-of-speech) Tagging is the process of labeling each word in a sentence with its grammatical category. (noun, verb, adjective etc).

Sentence :- The cat sleeps on the mat.

The → Determiner (DET)

cat → Noun

sleeps → verb.

on → preposition.

the → Determiner

mat → noun.

Types of POS Tagging.

* Rule based POS Tagging.
* statistical POS Tagging
* Hybrid POS Tagging.

① Rule - Based pos Tagging.

* uses predefined grammer rules and lexical dictionaries to assign pos tags.

Ex: If a word end in -ing, it is likely a verb.

Types of Rule - Based pos Tagging

* Lexicon - Based Tagging.
  * uses a dictionary of words with their pos tags.

  Ex: run is a verb, "beautiful is an adjective.

* context - Based Tagging.
  * uses rules to analyze context and decide the correct tag.