29/July/2025 1:00:00

Generative AI: Kuch new generate karna based on sample data it was trained on. Example: ap training me poems waghera sikhao bht saari examples k saath aur phir wo khud se kuch similar create kare jo usne seekha hy training me . It uses LLM.

Supervised Learning: Isme apko X (questions/input) k saath uska Y (answers/output) b diya jata aur ap onme relationship sekhty ho . Phir jab new input ata to ap predict karty k output kia hoga. We use techniques called Classification and regression. It is used in machine and deep learning.

Example: spam detection , image recognition, speech recognition, medical diagnosis

Unsupervised Learning: Isme apko sirf X (input) diya jata hy aur ap os input me patterns learns karty ho. We use techniques called Clustering e.g K-Means. It is widely used in generative AI.

Example: recommendation systems etc.

LLM: Ik esa model jo k large amount of data par train kia hua ho. Ye models hamare pas human-like text ko understand aur generate karskty easily aur images waghera pr b kaam krskty .

Example: GPT, Gemini, Llama

Generative AI pipeline: New content generate krne k liye AI k step by step processes

1) Data Acquisition: Collect raw data (like text, images, or numbers) from sources to use for training the model.

- ➔ Available data (CSV, TXT, PDF, DOCS, XLSX)
- ➔ Other data (Db , internet , API , Web scrapping)
- ➔ No data (Create your own data , you can see LLM for that)
  Note: If you create your own data then you will not have enough data , so you can perform data augmentation techniques

Data Augmentation techniques:

- ➔ Replace with synonyms
  Example: “I am a data scientist” . After data augmentation: “I am an AI engineer”
  If you have images, you can also perform image augmentation like flipping it , brightness etc.

- ➔ Biagram flip
  Example: “I am Riyan” . (It can be) “Riyan is my name”

- ➔ Back Translate
  Example: ik language se dosri language me convert kia phir us language se dosri language me convert kia phir dosri language se jo text apke pas original language me tha usme convert kia. To ese kuch words change hojaye ge aur new data mil jayega

- ➔ Add additional data/noise
  Example: “I am a data scientist” (Now in this you can add one more line) “I am a data scientist and I love this job”

2) Data preparation: Clean and format the collected data (remove errors, handle missing values) so it's ready for analysis.

➔ Cleanup
Example: Remove HTML, Emojis, Spelling check

➔ Basic preprocessing

Tokenization: Ik esa process jo text ko chotay tukro me toor de (also called tokens)

Sentence level tokenization: Splits the text into sentences

Example: Text: "I love AI. It's amazing!"
After sentence-level tokenization:
["I love AI.", "It's amazing!"]

Word level tokenization: Splits the text into individual words

Example: Text: "I love AI"
After word-level tokenization:
["I", "love", "AI"]

➔ Optional preprocessing

1) Stop words removal

Example: common words (like "the", "is", "in", "on") that do not add much meaning to the text

2) Stemming: process of reducing words to their root form (base form)
Example: Words: "running", "runner", "ran"
After stemming: "run"

3) Lemmatization: word k context ko smjh kr k wo noun hy ya verb apko real word me convert kar k deta .

Example: Words: "running", "ran", "better"
After lemmatization:
"running" → "run"
"ran" → "run"
"better" → "good"

4) Punctuation removal: Removing dots(.) commas(,) etc.

5) Lower case: converting all the text to lowercase letters

6) Language detection: identifying the language

➔ Advance preprocessing

1) Parts of speech tagging: Nouns, verbs, adjectives identify karna
2) Parsing: In depth grammar aur sentence structure ko identify karna.
3) Coreference resolution: Ye figure karna k sentence me kitnay words ese hyn jo same hi cheeze ko reference karrhy hyn

3) Feature engineering: Converting text into vector representation

- ➔ Text Vectorization

  1) TFIDF (Term Frequency - Inverse Document Frequency): Is a way to measure how important a word is in a specific document compared to other documents in a collection(corpus). It gives high importance to words that are unique or rare in corpus.
     Example, if a word appears often in one document but rarely in others, it will have a higher TF-IDF score, indicating it's more significant for that document.

  2) Bag of words: It represents text by counting the frequency of each word In a document, ignoring grammar or word order.
     Example: "I love dogs" → [1, 1, 1] (for the words "I", "love", and "dogs")
     "Dogs are cute" → [0, 1, 1] (for the words "I", "love", and "dogs")

  3) Word2vec: it converts words into vectors based on their content and semantic meanings
     Example: "King" and "queen" will have similar vector representations because they often appear in similar contexts (e.g., "royalty" or "monarchy").

  4) One hot encoding: creates new binary columns for each unique category in a feature. Each column represents one unique category, and a value of 1 or 0 indicates the presence or absence of that category
     Example, if we have a dataset with a single categorical feature, Color, which can take on three values: Red, Green, and Blue, one-hot encoding would transform this feature into three new binary columns, each representing one of the colors.

  5) Transformers model (advance): It captures complex relationships between words, considering how each word relates to every other word in context
     Example: In the sentence "The cat sat on the mat", a transformer understands that "sat" relates to both "cat" and "mat", and "cat" is the subject of the sentence.

4) Modeling: Train a machine learning model using the prepared data to make predictions or generate output.

- ➔ Open-source models
- ➔ Paid models

5) Evaluation: Evaluate the model's performance using separate data to see how well it works and if it's accurate.

- ➔ Intrinsic evaluation: It measures the quality of the model's output based on internal criteria like grammatical correctness or coherence.

- ➔ Extrinsic evaluation: It measures the models performance on a specific task like translation or sentiment analysis to see if its effective in real world situations. It can be done after deployment in production.

6) Deployment: Deploy the model into production, making it available for use in real-world applications (e.g., website, app).

7) Monitoring and model updating: Keep track of how the model is performing over time and make updates when necessary (e.g., if performance drops or data changes).

30/July/2025	1:20:00	(20 minutes)	31/July/2025	1:25:00	(5 minutes)
01/August/2025	2:07:00	(42 minutes)			

How does one hot encoding works?

Let us say I have tabular data

D1	My name is Riyan
D2	People watch Riyan
D3	Hello guys welcome me
D4	Welcome Riyan!

Unique Words:

name , Riyan , People, watch, Hello, guys, welcome (total N = 7)

Now I will make a table of these unique words and for each word in a sentence I will mark 1 for the presence of that word and the rest of the words will be zero.

	Name	**Riyan**	**People**	**Watch**	**Hello**	**Guyz**	**welcome**
D1 (name)	1	0	0	0	0	0	0
D1(Riyan)	0	1	0	0	0	0	0
D2(people)	0	0	1	0	0	0	0
D2(watch)	0	0	0	1	0	0	0
D2(Riyan)	0	1	0	0	0	0	0
D3(Hello)	0	0	0	0	1	0	0
D3(guys)	0	0	0	0	0	1	0
D3(welcome)	0	0	0	0	0	0	1
D4(welcome)	0	0	0	0	0	0	1
D4(Riyan)	0	1	0	0	0	0	0

Now representations:

```
D1 = [ [1 0 0 0 0 0 0 ],
       [ 0 1 0 0 0 0 0 ] ]

D2 = [ [ 0 0 1 0 0 0 0 ] ,
       [ 0 0 0 1 0 0 0 ] ,
       [ 0 1 0 0 0 0 0 ] ]

D3 = [ [ 0 0 0 0 1 0 0 ] ,
       [ 0 0 0 0 0 1 0 ] ,
       [ 0 0 0 0 0 0 1 ] ]

D4 = [ [ 0 0 0 0 0 0 1 ] ,
       [ 0 1 0 0 0 0 0 ] ]
```

Takeaways: Matlb k jitney hamare pas unique words hogay utni hi bari mere pas input layer hogi . To isse hamare pas neurons increase hogay computations cost increase hogi.

Ik aur masla yeh hy k agar mene evaluation k doraan ko esa word dia hy jo testing k waqt mere corpus me nahi that to phir mera model wo word nahi smjh payega. (Out of vocabulary issue OOV).

Ik aur masla yeh hy one hot encoding me k apke pas 0 bht ziada hotay hyn to isse computation me unnecessary calculation hoti hy. (Sparse matrix bantay hyn)

Ye semantic meanings b catch nahi karta.

When to use it? Sirf us waqt jab hamare pas vocabulary ziada nahi banegi , limited words hone ka yaqeen hoga

N-gram: Isme ham ‘n’ words ko as a single token use karty hyn. Baaz oqaat hame 2 words ko as a single token use karne ki zarorat parti hy .

Example: This movie is very good

This movie is not very good

Ab agar isme ham 2 words k bajaye agar 1 word legy to capture nahi kr payege semantic meanings

How does bag of words (BOW) work?

Let us say I have tabular data

D1	My name is Riyan
D2	People watch Riyan
D3	Hello guys welcome me
D4	Welcome Riyan! welcome

Unique Words:

name , Riyan , People, watch, Hello, guys, welcome (total N = 7)

Now I will make a table of these unique words and for each word in a sentence I will count how many times It occurs

	Name	**Riyan**	**People**	**Watch**	**Hello**	**Guyz**	**welcome**
D1	1	1	0	0	0	0	0
D2	0	1	1	1	0	0	0
D3	0	0	0	0	1	1	1
D4	0	1	0	0	0	0	2

Now representations:

D1 = [1 1 0 0 0 0 0]

D2 = [0 1 1 1 0 0 0]

D3 = [0 0 0 0 1 1 1]

D3 = [0 1 0 0 0 0 2]

Takeaways: very few 0's as compared to one-hot encoding

It is better when we have sentiment analysis or classifications types of tasks like positive, negative etc.

It is not capturing semantic meanings, also some of the zero are still present.

What is NumPy? Used for scientific computing . It used to be Efficient large , multidimensional array operations , mathematical functions. It is very memory efficient.

What is SkLearn? Used for data mining and data processing. Also, for building, training, evaluating, and tuning machine learning models

02/August/2025 2:19:00 (12 minutes)

How does TF-IDF work?

Let us say I have tabular data

D1	My name is Riyan
D2	People watch Riyan
D3	Hello guys welcome me
D4	Welcome Riyan! Welcome

Unique Words:

name , Riyan , People, watch, Hello, guys, welcome (total N = 7)

Now For each document and each word

TF (word,doc) = (Number of times the word appears in doc) / (Total number of words in doc)

Data	Word	Total words in document (stop words removed)	TF
D1	Name	2	1/2 = 0.5
	Riyan	2	1/2 = 0.5
D2	People	3	1/3 = 0.33
	watch	3	1/3 = 0.33
	Riyan	3	1/3 = 0.33
D3	Hello	3	1/3 = 0.33
	Guyz	3	1/3 = 0.33
	welcome	3	1/3 = 0.33
D4	Welcome	3	2/3 = 0.67
	Riyan	3	1/3 = 0.33

Now we will calculate Document Frequency (DF) and Inverse Document Frequency (IDF)

Number of Documents N = 4

word	Appears in	DF	IDF [(log(4/DF)]
name	D1	1	log(4/1)=1.386
Riyan	D1 , D2 , D4	3	log(4/3) = 0.288
people	D2	1	log(4/1) = 1.386
Watch	D2	1	log(4/1) = 1.386
hello	D3	1	log(4/1) = 1.386
guys	D3	1	log(4/1) = 1.386
welcome	D3,D4	2	log(4/2) = 0.693

Now multiply TF x IDF for each word/document

Data	Word	TF	IDF	TF x IDF
D1	Name	0.5	1.386	0.693
	Riyan	0.5	0.288	0.144
	others	0	0	0
D2	People	0.33	1.386	0.462
	Watch	0.33	1.386	0.462
	Riyan	0.33	0.288	0.095
	Others	0	0	0
D3	Hello	0.33	1.386	0.462
	guys	0.33	1.386	0.462
	welcome	0.33	0.693	0.231
	Others	0	0	0
D3	Welcome	0.67	0.693	0.464
	Riyan	0.33	0.288	0.095
	Others	0	0	0

Final Manual TF-IDF Table

	name	**Riyan**	**people**	**watch**	**hello**	**guys**	**welcome**
D1	0.693	0.144	0	0	0	0	0
D2	0	0.095	0.462	0.462	0	0	0
D3	0	0	0	0	0.462	0.462	0.231
D4	0	0.095	0	0	0	0	0.464

So, your new TF-IDF representations:

- **D1**: [0.693, 0.144, 0, 0, 0, 0, 0]
- **D2**: [0, 0.095, 0.462, 0.462, 0, 0, 0]
- **D3**: [0, 0, 0, 0, 0.462, 0.462, 0.231]
- **D4**: [0, 0.095, 0, 0, 0, 0, 0.464]

Takeaways: Not catching semantic meanings , there are better options than that.

USING DEEP LEARNING APPROACH FOR TEXT VECTORIZATION

1) WORD2VEC

Some important concepts

Weights: hamare neural network me input layer se hidden layer aur hidden layer se output layers k drmiyaan weights hotay hyn. It determines k har input kitna strongly mere output ko determine Karega. Esa smjho jese Ik dial ya multiplier hy.

Example:

Suppose you're mixing paint:

- You have 3 basic colors: red, blue, yellow (inputs).
- You want to make purple.
- The "weights" would decide how much of each color to use. For purple, maybe:

  Red: 0.7 (weight), Blue: 0.8 (weight), Yellow: 0.1 (weight).

In a neural network, the weights are *learned*—the network finds the right mix to get as close as possible to your target color (or output).

Word embeddings: Learned weights between the input and the hidden layers.

Loss: How much wrong the network prediction is from our desired result

Back propagation: The process of adjusting the weights by calculating the errors(loss) . It helps the network understand which connections need to be stronger or weaker.

WORD2VEC TECHNIQUES

A. CBOW (Continuous Bag of Words)

- **Goal:** Predict the current (center) word using the surrounding context words.
- **Example:**
  Sentence: "The cat sat on the mat"
  For "cat" (center), context is ["the", "sat"]
  So, CBOW tries to predict "cat" from ["the", "sat"]

- **CBOW** is faster and works better with large datasets and frequent words.

B. Skip-Gram

- **Goal:** Predict the surrounding context words using the current (center) word.
- **Example:**
  Sentence: "The cat sat on the mat"
  For "cat" (center), try to predict ["the", "sat"]

- **Skip-Gram** works better with small datasets and captures rare words well.

Intuition: So basically, for catching semantic meanings , I can use word2vec . For this I have 2 options (CBOW, skip grams) and I will use neural network to get my word embeddings and that word embeddings will have my semantic vector.

Manual Example (Skip-Gram)

Let’s use a tiny corpus:

- "cat sat mat"

Vocabulary:

["cat", "sat", "mat"]

Step 1: One-hot encoding

- "cat": [1, 0, 0]
- "sat": [0, 1, 0]
- "mat": [0, 0, 1]

Step 2: Training Pair Creation

- For "sat" as center, context window = 1:
  - Pairs: ("sat", "cat"), ("sat", "mat")

Step 3: Initial Weights (Random)

- Suppose our embedding size = 2 (just for simplicity)
- Input-to-hidden weights matrix: 3 words × 2 dims (to be learned)
- Start with random numbers, e.g.:
  - "cat": [0.2, 0.5]
  - "sat": [0.4, 0.1]
  - "mat": [0.9, 0.7]

Step 4: Training Process (Single Step)

For ("sat", "cat"):

1. Input: "sat" = [0, 1, 0]
2. Multiply by weights → [0.4, 0.1] (current "sat" embedding)
3. Use another set of weights to predict all possible context words ([softmax] output layer).
4. Compute loss (is “cat” predicted strongly? If not, error is high).
5. Adjust weights so next time "sat" is more likely to predict "cat".

Repeat for ("sat", "mat").

After training:

- "sat"’s embedding will be close to "cat" and "mat" because their contexts overlap.

03/august/2025 2:29:00 (10 minutes)

Gensim: NLP library especially for working with vector space models like word2vec, doc2vec etc.

04/August/2025 3:00:00 (31 minutes)

Confusion Matrix: The confusion matrix helps you see where the model is making mistakes

Transformers: A type of deep learning architecture model.

When we work on text to hame words k drmiyaan relationship capture karna parta hy ta k ham sahi meaning ko identify kar saken. Hamare pas jo traditional models hyn like RNN etc yeh sab at a time 1 word ko capture karty hyn jiski wajah se long texts me distant words k drmiyaan relation capture karty waqt problem hoti hy.

Transformers is problem ko solve karty hyn by looking at all words at a once and they use an attention mechanism to figure out which words are important for each other , no matter where they are in text.

Popular Transformer Models

1. **BERT (Bidirectional Encoder Representations from Transformers)**
   - What it does: BERT looks at the entire sentence at once (both left and right context) to understand meaning. It’s great for tasks like text classification, question answering, and language understanding.
2. **GPT (Generative Pretrained Transformer)**
   - What it does: GPT generates text one word at a time, using context to predict the next word in the sequence. It’s great for text generation, translation, summarization, and other creative tasks.

05/August/2025 3:12:00 (12 minutes)

Zero-shot learning: The model is asked to perform a task without any prior examples, relying solely on the prompt.
Example: "Translate the sentence 'I love programming' to French."

F**ew-shot learning**: The model is given a few examples in the prompt to help it understand the task.
Example: "Translate the following sentences to French: 'I love programming' → 'J'aime la programmation', 'Hello' → 'Bonjour'. Now, translate 'Goodbye'."

CNN: Type of neural network designed specifically for image processing. It uses filters or kernels which are like small window that moves over the image to recognize edges, shapes and textures.

RNN: Type of neural network designed to work with sequential data like text, speech etc.

06/August/2025 4:01:00 (49 minutes)

Reinforcement learning: is when a model learns by trial and error, receiving rewards or penalties based on its actions.
Example: Teaching a robot to walk by rewarding it when it moves forward and penalizing it when it falls.

07/August/2025 4:12:00 (11 minutes)

08/August/2025 4:25:00 (13 minutes)

09/August/2025 4::27:00 (2 minutes)

Epoch: means one full pass through your entire training dataset.

Think of it like studying a book:

- You have 100 pages (dataset).
- Epoch 1 → you read all 100 pages once.
- Epoch 2 → you go back and read all 100 pages again.
- And so on…

Batch size: how many training examples your model sees at the same time before updating its weights.

Example:
You have 1000 sentences in your dataset.

- Batch size = 10 → The model looks at 10 sentences, calculates the loss, updates weights → then takes the next 10, and so on.
- Batch size = 100 → The model looks at 100 sentences at once before updating weights.

10/August/2025	4:34:00	(7 minutes)
11/August/2025	4:43:00	(9 minutes)
12/August/2025	4:43:00	(0 minutes)
13/August/2025	4:43:00	(0 minutes)
14/August/2025	4:43:00	(0 minutes)
15/August/2025	4:43:00	(0 minutes)

Data Collator: prepares your batch of data before sending it to the model

16/August/2025	4:55:00	(12 minutes)
17/August/2025	5:06:00	(9 minutes)

Automodelforseq2seqlm: encoder+decoder models e.g BART, Pegasus

AutoModelForCausalLM: decoder only models e.g GPT-2 , Llama , most chat models

Text to image generation: model learns patterns from huge dataset of images + captions and it maps text -> images. Two main model families:

1) Diffusion Models

- Start with pure noise (random pixels).
- Gradually “denoise” it while conditioning on your text prompt until it forms an image.
- Examples: Stable Diffusion, DALL·E 2

2) **Transformer-based Models**

- Some models like DALL·E mini / DALL·E 2 use transformers to predict pixels or image tokens directly from text.
- Less popular now because diffusion gives higher-quality images.

18/August/2025	5:20:00	(14 minutes)
19/August/2025	5:20:00	(0 minutes)
20/August/2025	6:09:00	(49 minutes)

Open AI completion API: It just *completes* the text, like autocomplete

Open AI Chat completion API: It’s *conversation-style*, supports roles (system, user, assistant), and is used for ChatGPT

Function Calling in Open AI: Function calling in OpenAI means: the model doesn’t just give you text, it can also “call” your own function with structured data when it recognizes it’s needed.

👉 Example:
You ask: *“What’s the weather in Paris?”*
Instead of just replying, the model outputs:

```
{ "name": "get_weather", "arguments": { "location": "Paris" } }
```

Then your app runs get_weather("Paris") and sends the result back.

21/August/2025	7:00:00	(51 minutes)

22/August/2025	7:43:00	(43 minutes)

Prompt Engineering Key Principles:

1. **Be Specific & Clear**: The more precise you are about the context and expectations, the better the output.
   - **Bad**: "Tell me about AI."
   - **Good**: "Explain the main differences between supervised and unsupervised learning in AI, with examples."
2. **Ask for Structure**: If you need a specific format, ask for it upfront.
   - **Example**: "Summarize the key points of this article in a bullet-point list."
3. **Use Examples**: Provide examples of what you're looking for. This helps the model understand your desired format, tone, or depth.
   - **Example**: "Can you summarize the article like this: [brief example of summary]?"
4. **Iterative Refinement**: If the initial response isn't quite right, ask for adjustments.
   - **Example**: "Expand the explanation with more details on point 2."
5. **Limit the Scope**: Narrow down what you're asking for to prevent overly broad answers.
   - **Bad**: "Tell me about AI."
   - **Good**: "In 100 words, explain how reinforcement learning works in AI."
6. **Temperature & Creativity**: For creative outputs (like story writing, poems, or brainstorming), ask for more creativity using a higher temperature (e.g., "Can you write a creative story about..."). For factual queries, keep it straightforward.

7. **Avoid Ambiguity**: Remove any vagueness from your prompt that could lead to confusion.
    - **Bad**: "Give me some data on the topic."
    - **Good**: "Provide recent statistics on AI adoption in healthcare in 2023."
8. **Role-Playing**: If you need a specific perspective, specify the role.
    - **Example**: "Act as a personal trainer and suggest a workout for someone who wants to build muscle."

AI hallucination: happens when the model generates incorrect or made-up information that sounds plausible but isn't grounded.

- Example: *"Albert Einstein invented the lightbulb,"* which is incorrect, but the model might "hallucinate" that as a fact.

RAG (Retrieval-Augmented Generation): combines retrieval-based methods with generation: the model first retrieves relevant information (like from a database) and then generates a response using that information.

Example: If asked, *"What's the GDP of Japan in 2023?"*, a RAG model first pulls the latest GDP data from an external source, then uses it to generate a factual response.

23/August/2025	8:17:00	(34 minutes)

Vector Database is a specialized database designed to store, search, and retrieve high-dimensional data (called vectors) efficiently. It captures semantic meanings and checks for similarity . In traditional databases like SQL, we can't like just search based on semantic or similarities.

Example: Pinecone, Weaviate (cloud based) , Chroma (local db), Faiss (local db) ,neo4j (graph based)

LangChain: Is a framework that integrates pre-trained models (and others) into more advanced applications. It provides tools for managing the flow of logic and context across multiple steps, making it easier to build more complex workflows with language models.

Cosine similarity: is a measure of similarity between two vectors by calculating the cosine of the angle between them. It ranges from -1 (completely opposite) to +1 (completely similar), with 0 indicating no similarity.

24/August/2025	8:17:00	(0 minutes)
25/August/2025	8:17:00	(0 minutes)
26/August/2025	8:55:00	(38 minutes)

Chunk: A piece of text split from a larger document**.** We split docs because LLMs/embeddings have limits (e.g., 512–1000 tokens).

Example doc (raw):

Microsoft raised $1B in funding.

The money will be used for AI research and infrastructure.

OpenAI also announced new partnerships.

Chunk Size: How big each piece should be (in characters or tokens).

If chunk_size = 30 →

["Microsoft raised $1B in fund",
 "ing. The money will be used f",
 "or AI research and infrastruct",
 "ure. OpenAI also announced new",
 " partnerships."]

Chunk Overlap: How much chunks should overlap with each other (to keep context between splits).

If chunk_size = 30, chunk_overlap = 10 →

["Microsoft raised $1B in fund...",
 "und. The money will be used f...",
 "used for AI research and infra...",
 "structure. OpenAI also announc..."]

Notice each chunk shares some text (10 chars) with the next one.
This avoids cutting important context (like breaking a sentence in half).

27/August/2025	8:55:00	(0 minutes)
28/August/2025	8:55:00	(0 minutes)
29/August/2025	9:20:00	(25 minutes)
30/August/2025	9:37:00	(17 minutes)
30/August/2025	9:37:00	(0 minutes)
31/August/2025	9:37:00	(0 minutes)
1/September/2025	9:37:00	(0 minutes)
2/September/2025	9:51:00	(14 minutes)
3/September/2025	10:05:00	(14 minutes)
4/September/2025	10:21:00	(16 minutes)
5,6,7,8/September/2025	10:21:00	(0 minutes) (project)
9/September/2025	10:41:00	(20 minutes)
10,11,12/September/2025	10:41:00	(0 minutes)
13/September/2025	14:10:00	(3 hours 29 minutes)

1. Character Text Splitter

Splits text purely based on **character count**.
Example: if chunk_size=500, it just cuts every 500 characters.

Pros: Very simple. Fast.

Cons: Might cut off mid-sentence or mid-word. Chunks can lose context.

Best for: quick experiments, logs, or raw text without structure.

2. Recursive Character Text Splitter

More intelligent.
Splits text recursively by **preferred separators**: paragraphs → sentences → words → characters.
Ensures chunks are within chunk_size but tries to keep **semantic meaning**.

Example:

Tries to split by \n\n first.
If still too long, splits by . (sentence).
If still too long, splits by space.
Finally by characters if needed.

Pros: Keeps text natural (sentences/paragraphs together). Minimizes mid-sentence breaks.

Cons: Slightly slower.

Best for: most RAG use cases (docs, PDFs, Q&A, general LLM pipelines).

3. Token Text Splitter

Splits text based on **LLM tokens** (using the tokenizer of your model).
Guarantees chunks align with token limits (chunk_size=500 means 500 tokens, not characters).

Pros:

Exact control over LLM input size.
Prevents chunk from exceeding model's context window.
Works well with multilingual text or languages where word length varies.

Cons: Needs the model's tokenizer (slower). More setup (depends on OpenAI/HF tokenizer).

Best for: when you care about LLM context window (production RAG, chatbots, embeddings).

Quantization: Shrinking a model's weights from high precision (like 16-bit floats) to lower precision (like 8-bit or 4-bit numbers) so it uses less memory and runs faster.

Fine-Tuning: Adapting a pretrained model to a narrower domain by updating its weights with additional task-specific data.

Parameter-Efficient Techniques (PEFT):

LoRA (Low-Rank Adaptation)

Instead of updating all weights, insert small trainable matrices (low-rank adapters) into certain layers.

During training → only these adapters are updated, not the whole model.
At inference → base model + adapters = fine-tuned model.

Benefits:

- Much smaller training footprint (hundreds of MBs instead of tens of GBs).
- You can stack multiple LoRAs for different tasks (plug-and-play).

QLoRA (Quantized LoRA)

Takes LoRA further:

First quantize the base model (reduce precision from 16-bit → 4-bit).
Then apply LoRA adapters on top.

Benefits:

- Allows fine-tuning a 65B model on a single 48GB GPU.
- Maintains quality close to full fine-tuning.
- Super popular in open-source projects (like Hugging Face PEFT).

14/September/2025	14:30:00	(20 minutes)
15,16,17/September/2025	14:30:00	(0 minutes)
18/September/2025	15:38:00	(1 hour 8 minutes)
19/September/2025	15:38:00	(minutes)