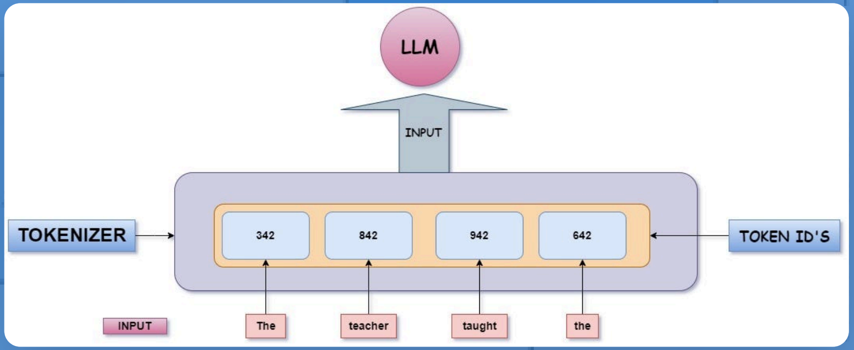
Data Preprocessing: The ~~Engine~~ of AI

Fuel

From Cleaning Tables to Training LLMs



Workshop Overview

Our journey through data preprocessing fundamentals to advanced LLM techniques

1 **The Foundation**
General data preprocessing for tabular and image data

2 **The Pivot**
Transition from traditional NLP to LLM preprocessing

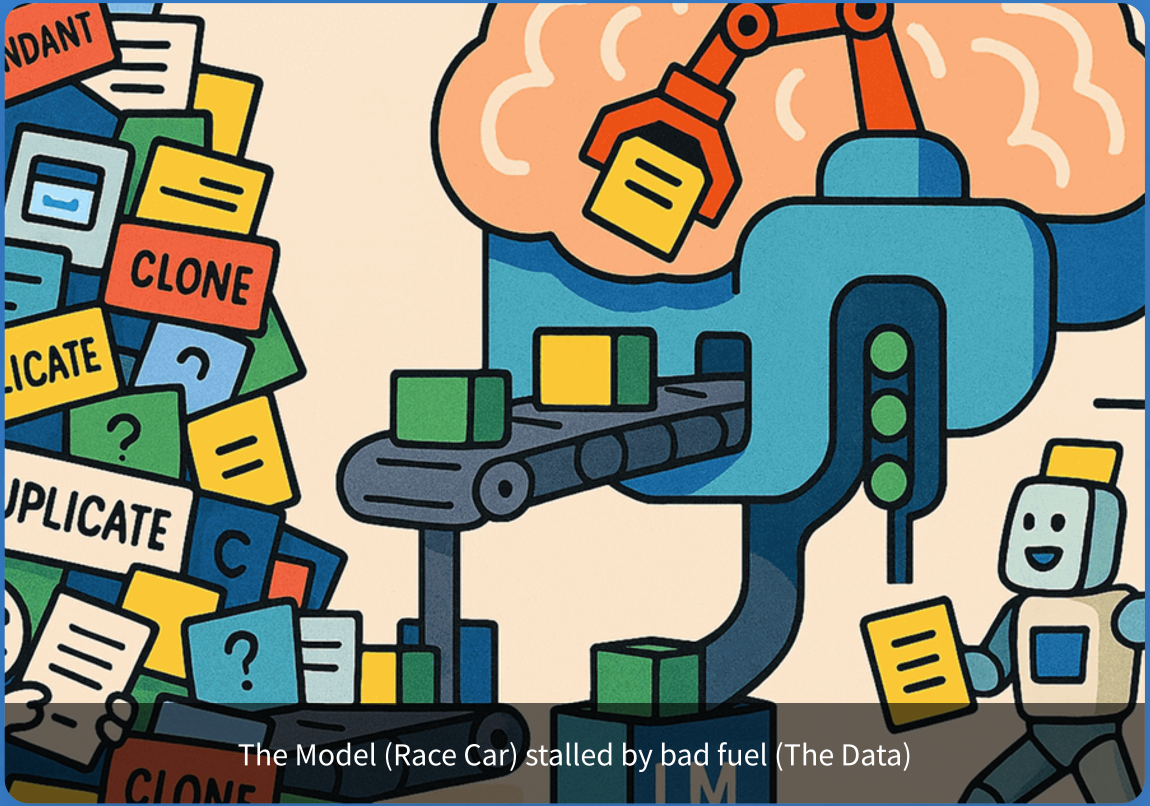
3 **Deep Dive - LLM Data Pipeline**
Ingestion, filtering, deduplication, and PII redaction

4 **Tokenization & Instruction Tuning**
Subword tokenization and chat formatting techniques

5 **Conclusion & Tools**
Key takeaways and modern preprocessing frameworks

Garbage In, Garbage Out

- **80%** of an ML project is data preparation
- Model architecture **cannot fix** broken data
- "Data is **code**"



The Model (Race Car) stalled by bad fuel (The Data)

Cleaning Structured Data

Practical Examples

Missing Values

Imputation (Mean/Median/KNN) vs. **Dropping**

Outliers

Detection via **Z-Score** or **IQR**

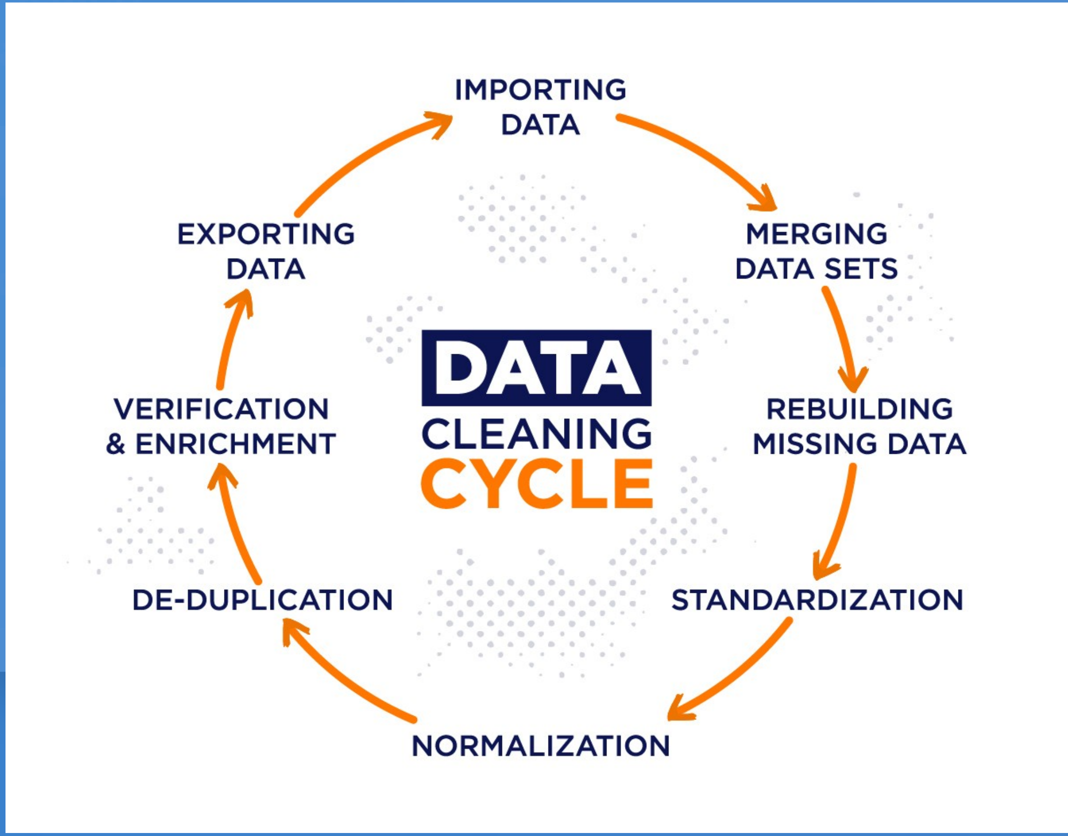
Scaling

Standardization vs. **Normalization** (Crucial for gradient descent)

Logistic Regression, SVM, PCA

distance-based algorithms (KNN)

Before/After: Data Distribution Centering



Preprocessing for Computer Vision

Normalization

Rescaling pixel values **0-255 → 0-1**

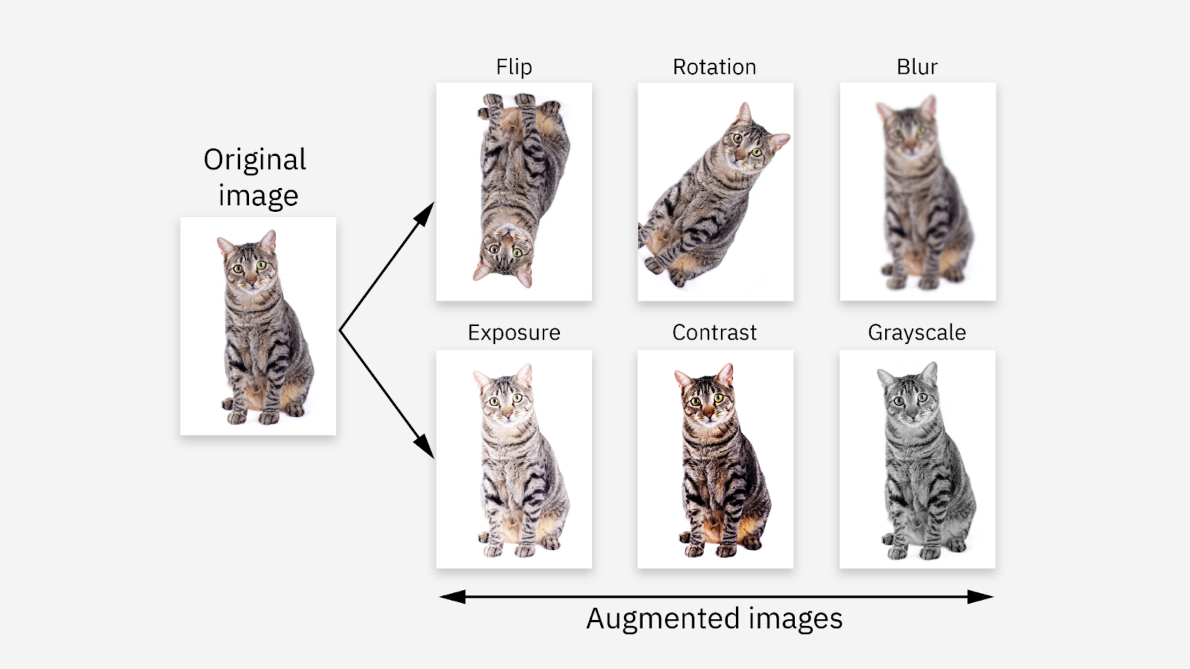
Resizing

Fixed dimensions for batch processing

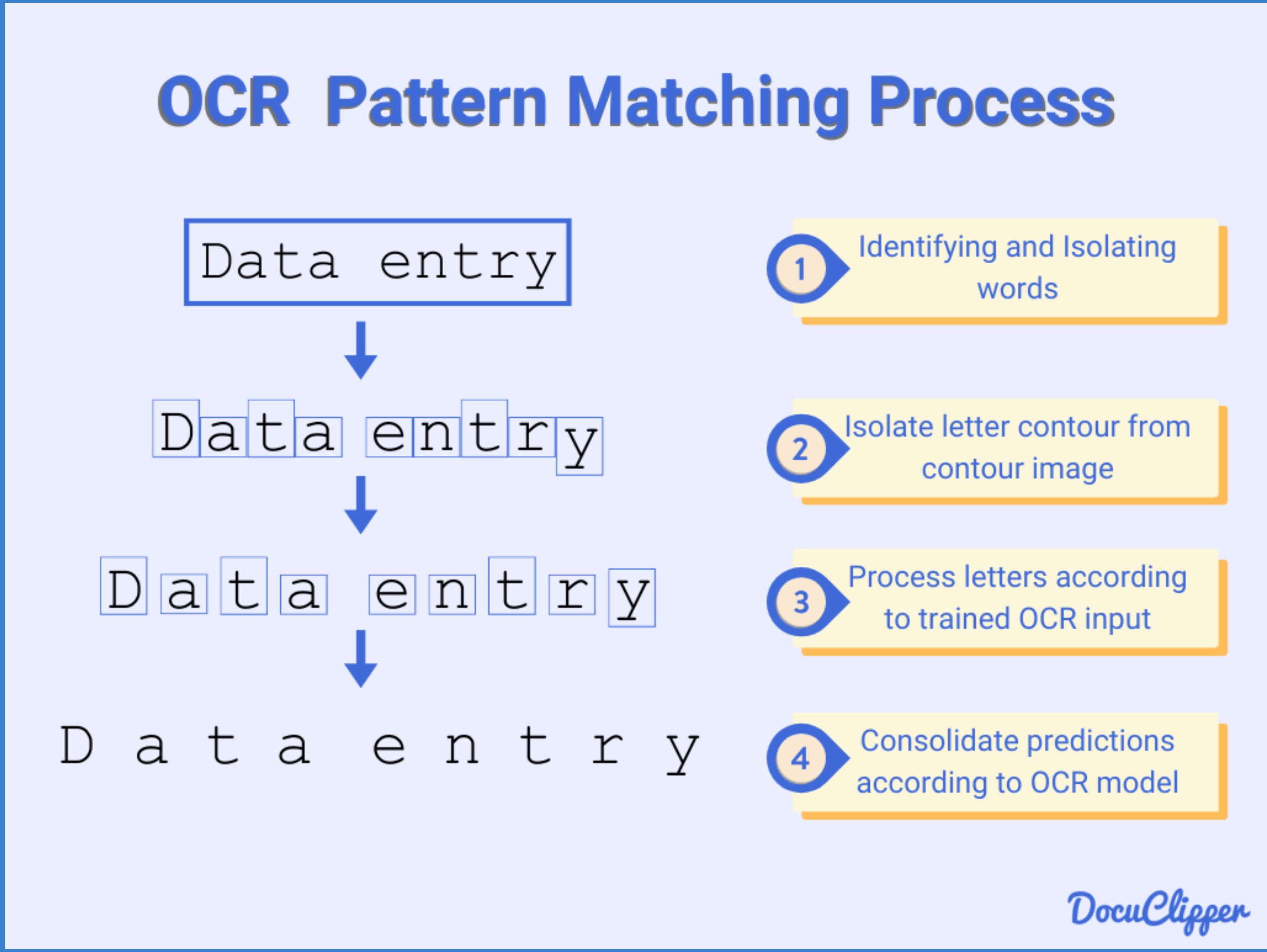
Augmentation

Rotation, Flip, Zoom (Fighting overfitting)

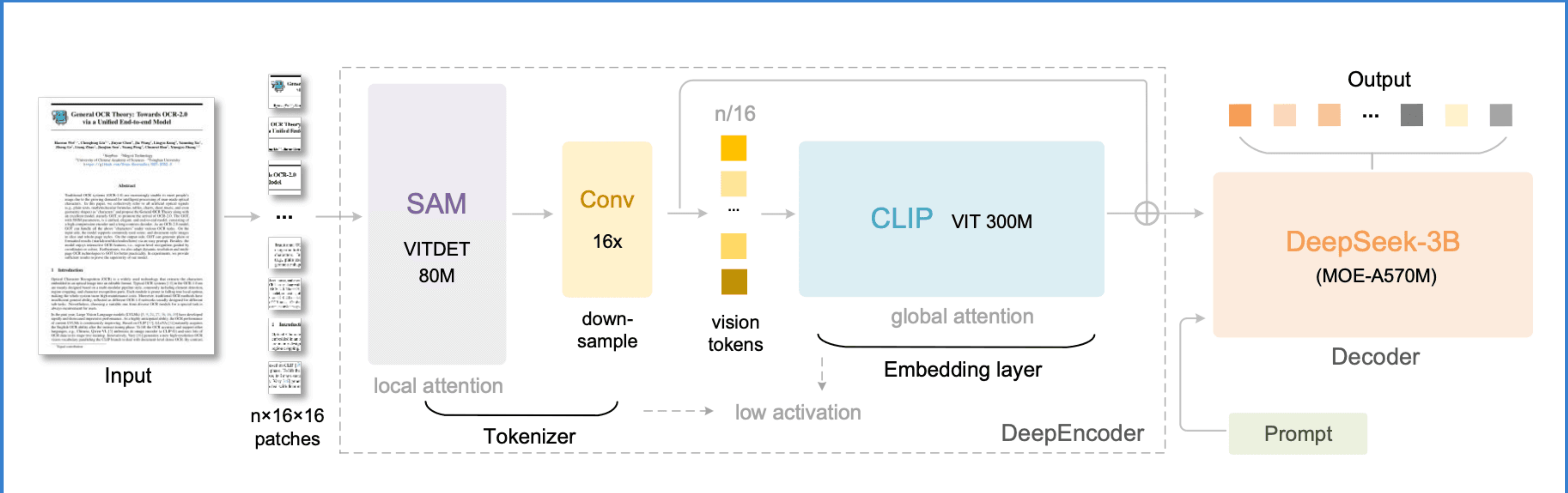
Image Augmentation Variations



Preprocessing for Computer Vision



Preprocessing for Computer Vision



Traditional NLP vs. LLMs

Traditional (TF-IDF/LSTM)

- **Remove** stopwords
- **Stem** words
- **Lowercase** everything
- Focus on **keywords**

LLMs (Transformers)

- **Keep** the context!
- **Case** matters
- **Stopwords** matter
- Focus on **language structure**

Goal: We want the model to learn language structure, not just keywords

From Raw Web to Pre-training

1

Data Ingestion

Common Crawl, GitHub, Wikipedia

2

Cleaning & Filtering

Quality control at scale

3

Deduplication

Critical Step

4

PII Redaction

Privacy protection

5

Tokenization

Breaking text into tokens

Key Data Sources

- Common Crawl
- GitHub
- Wikipedia
- Books
- Forums
- News Articles

Quality Control at Scale

Language ID

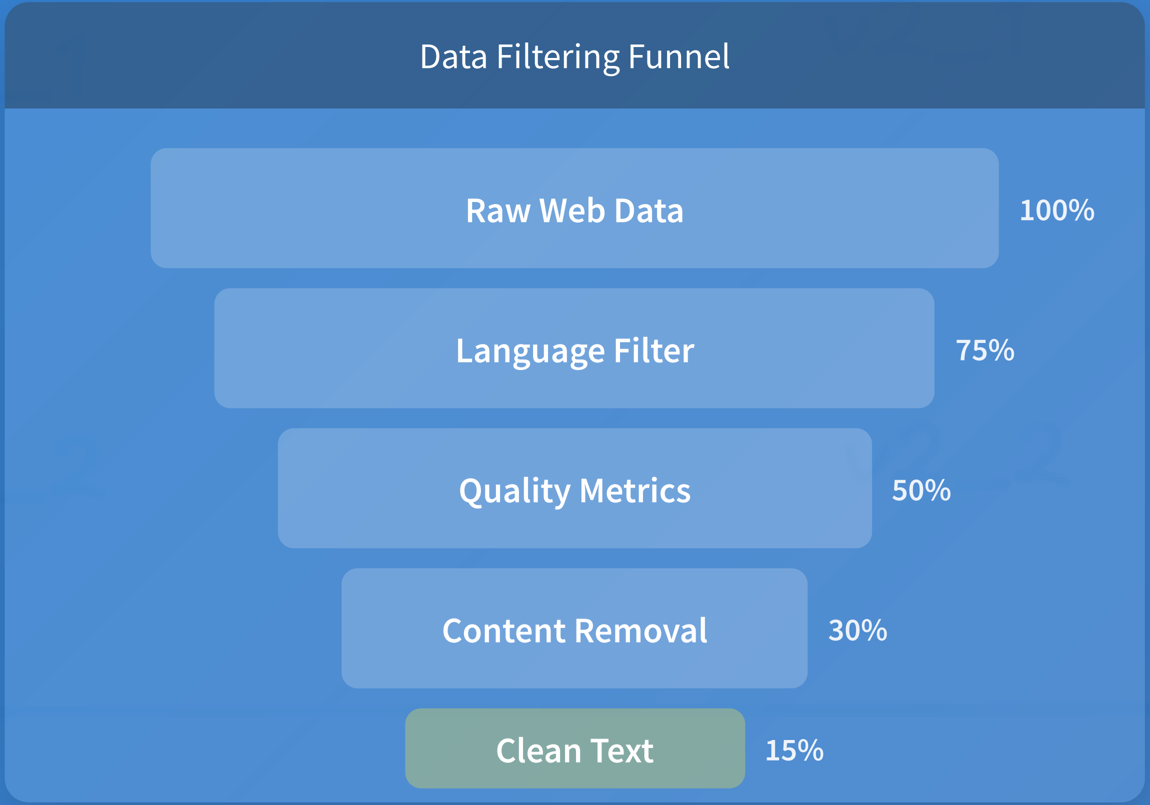
Filtering out **wrong languages** (e.g., keeping only English/Persian)

Metrics

Perplexity scores, text length, symbol-to-word ratio

Removal

HTML tags, "lorem ipsum", toxic content



Why Deduplication Matters?

The Problem

Duplicate data causes **memorization**, not **generalization**

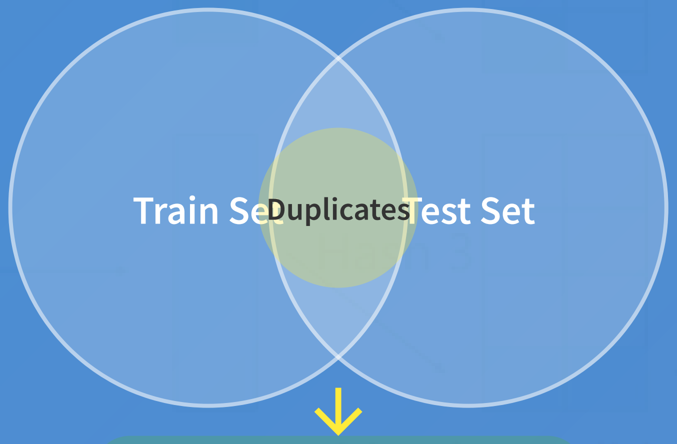
Exact Match

Removing identical strings using **SHA-256**

Fuzzy Deduplication

MinHash & LSH for similar documents

Data Leakage Prevention



No Overlap After Deduplication

PII Redaction

PII Must Be Scrubbed

Personally Identifiable Information removal is critical for privacy compliance

Detecting

Emails, **IP addresses**, **Phone numbers**, **Keys/Passwords**

Techniques

Regex patterns + **Named Entity Recognition** (NER)

PII Redaction Example

Hello, my name is John Smith and I work at TechCorp. You can reach me at **<EMAIL>** or call my office at **<PHONE>**. Our company is located at 123 Main Street, New York, NY 10001. My employee ID is **<ID>** and my IP address is **<IP>**. Please don't share my password: **<PASSWORD>**.

PII Replaced with Tags

How LLMs Read

Not Just Splitting

It is **NOT** just splitting by space

Sub-word Tokenization

BPE (Byte Pair Encoding) & **WordPiece**

Efficiency

Common words = 1 token
Rare words = multiple tokens

Sub-word Tokenization Example

Unbelievable ↓

Un	##believ	##able
Token ID: 1234	Token ID: 5678	Token ID: 9012

Instruction & Chat Formats

Pre-training

Raw Text prediction

Fine-tuning

Chat Template structure

Format

Structured conversation with roles

Masking

Loss calculated only on Assistant's response

JSON to ChatML Conversion

Original JSON

```
{ "instruction": "What is data preprocessing?",
"input": "", "output": "Data preprocessing is the
process of transforming raw data into an
understandable format." }
```

ChatML Format

```
<|im_start|>user What is data preprocessing?
<|im_end|> <|im_start|>assistant Data preprocessing
```

Beyond Basic Tokenization

SentencePiece

Unsupervised text tokenizer for language-agnostic tokenization

Special Tokens

BOS, **EOS**, **PAD**, **UNK** - Control model behavior

Vocabulary Size Impact

Larger vocab = fewer tokens per word but more parameters

Tokenization Algorithm Comparison

Algorithm	Input Text	Output Tokens
WordPiece	Tokenization	Token ##ization
BPE	Tokenization	Token ization
SentencePiece	Tokenization	To ken ization
Unigram	Tokenization	Tok en iz ation

Beyond Basic Tokenization

OOV = Out Of Vocabulary

SentencePiece

Unsupervised text tokenizer for language-agnostic tokenization

Special Tokens

BOS, **EOS**, **PAD**, **UNK** - Control model behavior

Vocabulary Size Impact

Larger vocab = fewer tokens per word but more parameters

Tokenization Algorithm Comparison

Feature	WordPiece	Byte-Pair Encoding (BPE)	Unigram
Developer	Google	Philip Gage (originally for compression)	Google (used by SentencePiece)
Core Principle	Segmentation based on **maximum likelihood** of words being split into components that already exist in the vocabulary (like morphemes).	Iteratively merging the **most frequent pair** of bytes/characters/tokens until the desired vocabulary size is reached.	Chooses the segmentation that **maximizes the probability** of the sequence based on a language model (often trained using the Expectation-Maximization algorithm).
Typical Separator	`##` prefix on subwords (e.g., `run` + `##ning`).	Underscore `_` or a special hidden space byte (e.g., `_token` + `iz` + `ation`).	Leading space or special token to denote the start of a word.
Handling of OOV Words	Excellent. Splits the unknown word into known root forms, prefixes, and suffixes.	Excellent. Guaranteed to break down any word into known subwords or individual characters/bytes.	Excellent. Offers multiple segmentation options and selects the most probable one.
Common LLMs	BERT, DistilBERT, ELECTRA	GPT-2, GPT-3, Llama 1/2, RoBERTa	Llama 3, XLNet, ALBERT, T5 (with SentencePiece)
Farsi Example (پاییز)	`##یز` + `پای` (Segmentation based on known parts)	`_ یز` + `پای` (Merging frequent pairs)	`ز` + `ایی` + `پ` (Choosing the most probable sequence)

Beyond Basic Fine-Tuning

Supervised Fine-Tuning

SFT - Direct training on labeled examples

RLHF

Reinforcement Learning from Human Feedback

DPO

Direct Preference Optimization - Simpler than RLHF

Parameter-Efficient

LoRA, QLoRA - Fine-tune with fewer resources

Fine-Tuning Approaches

Method	One-Line Explanation	Primary Benefit	Need for Additional Data/Model	Complexity
SFT	Training on labeled examples to specialize in a task.	Simple, reliable, and well-understood.	Requires high-quality labeled (input/output) data.	Low
RLHF	Uses human feedback to train a Reward Model, then uses RL for alignment.	Aligns the model with complex human preferences.	Requires comparative data and a separate Reward Model.	High
DPO	Directly tunes the model based on human preferences without a Reward Model.	Simpler and more stable than RLHF.	Requires comparative data (preferred/rejected pairs).	Medium
PEFT (LoRA/QLoRA)	Fine-tuning by training a small number of parameters to drastically reduce resource needs.	Significant reduction in computational and memory resources.	Needs SFT data (but more efficient training).	Medium

The Modern Tech Stack

General Data Processing

Pandas
Tabular data manipulation

NumPy
Numerical computing

OpenCV
Computer vision

LLM Specific

Hugging Face Datasets
Dataset management

Datatrove
Data processing pipeline

Text-Dedup
Text deduplication

Scaling Data Processing

Distributed Processing

Spark, Dask for handling massive datasets

Data Sharding Strategies

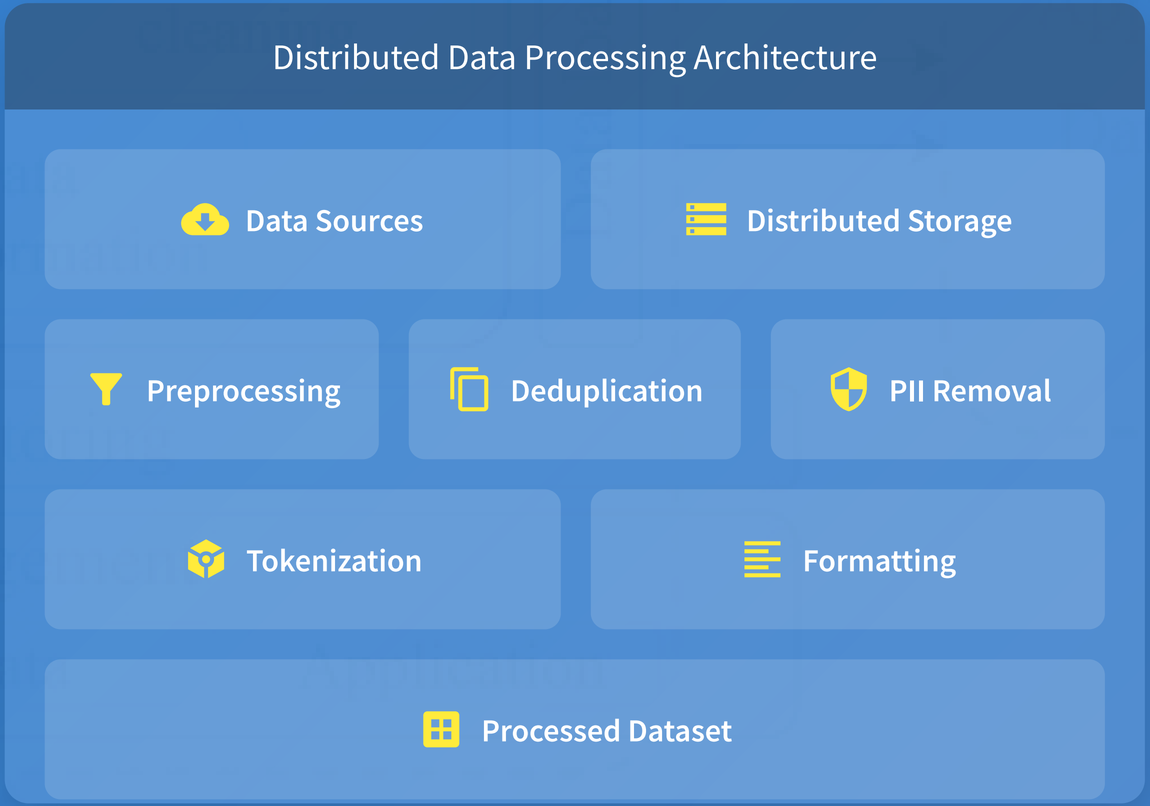
Horizontal partitioning for parallel processing

Quality Assessment

Automated metrics for data validation

Automated Pipelines

End-to-end data cleaning workflows



Summary

REAL WORLD
An example of Data Pre Processing
(But not for models, In life)

Data quality defines the **ceiling of performance**

LLMs require **"Fuzzy" cleaning**, not destructive cleaning

Deduplication is the most high-impact step for LLMs

Data Preprocessing Checklist

- Clean tabular data (missing values, outliers)
- Normalize and augment image data
- Filter and clean web data for LLMs
- Remove duplicates (exact and fuzzy)

Review - Importance of Data Cleaning

Data Cleaning is no longer a secondary step in data-driven projects; it is an indispensable part of their success.

Data that is not clean is unreliable, and making decisions based on it is useless or even harmful.

The reality is that the quality of the output of any analytical process is directly dependent on the quality of its input data.

Many data science professionals believe that without cleaning, data analysis will have no meaning or value.

Even the best algorithms cannot extract a correct result from incorrect data.

In the world of data, no analysis is reliable without clean data.

Clean Data = Correct Decision.

Data cleaning is a long-term investment for reducing direct and indirect costs.

What Works and What Doesn't

Best Practices

Start Small

Begin with a **subset** of data before scaling

Validate at Each Step

Check **data quality** after each transformation

Document Your Pipeline

Create **reproducible** preprocessing workflows

Iterate and Improve

Continuously **refine** based on model performance

Common Pitfalls

Over-cleaning Data

Removing **too much** context for LLMs

Ignoring Data Drift

Not monitoring **changes** in data distribution

Not Validating Steps

Skipping **quality checks** between stages

Neglecting Deduplication

Underestimating its **impact** on model performance

Questions & Discussion

Key Resources

- Hugging Face Documentation
- Data Preprocessing Courses
- GitHub Repository

Ryan Heida

ryan@ryanheida.com / mr.ryanheida@gmail.com

www.ryanheida.com / links.ryanheida.com

t.me/RyanHeida

https://t.me/ML_Topics

Thank You!

Scan for workshop feedback