MAINPIPE – Data Pipeline Report

Prepared by: Sania Khan

1. Executive Summary

Mainpipe is a modular, scalable, and production-inspired data preprocessing pipeline.

designed for LLM pre-training. It performs ingestion, cleaning, quality filtering, PII detection, optional toxicity scoring, deduplication, tokenisation, sharding, and metrics generation.

The system mirrors real-world pipelines used by OpenAI, MosaicML, Meta, and other industry teams.

Its streaming architecture ensures memory efficiency and deterministic processing.

2. Configuration Rationale (local.yaml)

- data: This section defines where the raw dataset is located and ensures the pipeline knows exactly which file to ingest. Using a configurable path makes the pipeline flexible and reusable for any incoming dataset, without modifying code.
- output: The output section determines where all processed files will be stored - including the final training shards, audit logs, and metric files. Using dedicated folders for each output improves organization, reproducibility, and compatibility with downstream training systems. The choice of 8 shards strikes a balance between parallel loading speed and manageability for smaller datasets.
- lang: The language configuration ensures the pipeline keeps only English text with high confidence. Texts shorter than 100 characters are excluded from language detection because they produce unreliable classifications. This helps eliminate not detected languages and ensures the final dataset is consistently English.
- filter: Quality filtering is performed here. The minimum character length of 160 ensures the pipeline keeps only meaningful, multi-sentence English text, avoiding short fragments that harm LLM performance. The maximum character limit prevents extremely long or noisy documents that slow down tokenisation. Additional filters like markup ratio and character diversity remove boilerplate, repetitive, or HTML-heavy content, which improves overall dataset cleanliness and linguistic richness.
- pii: PII settings control how the pipeline handles sensitive information such as emails, IP addresses, and phone numbers. Instead of dropping these records, the pipeline simply annotates them so they can be reviewed later. This preserves dataset volume while still supporting safety monitoring and compliance workflows.
- toxicity: Toxicity scoring is disabled by default to avoid the high computational cost of running Detoxify on a laptop. When enabled, the pipeline evaluates only a small sample of documents and caps the maximum size of text sent to the

model. This design provides useful safety insights without slowing the overall pipeline.

- tokenizer: The tokenizer configuration specifies GPT-2 BPE, chosen for its speed, stability, and wide adoption in LLM research. GPT-2 tokenisation produces consistent token distributions and works well for English-heavy datasets, making it ideal for evaluation and downstream model training.

3. Rationale for Key Metric Parameters (metrics.py)

Each metric in metrics.py is chosen to enable observability, alignment with LLM-data quality standards, and strong evaluability.

• **total_records / kept / dropped :** Tracks pipeline strictness and ensures auditability.

• **drop_reason_counts :** Identifies bottleneck filters (e.g., low diversity), vital for debugging dataset quality.

• **language_counts** : Ensures English-only constraint and distribution transparency.

• **pii_summary** : Required for safety evaluation and compliance (emails, IPs, phone numbers).

• **token_len_summary (mean, p50, p90, p99, max)** Critical for:

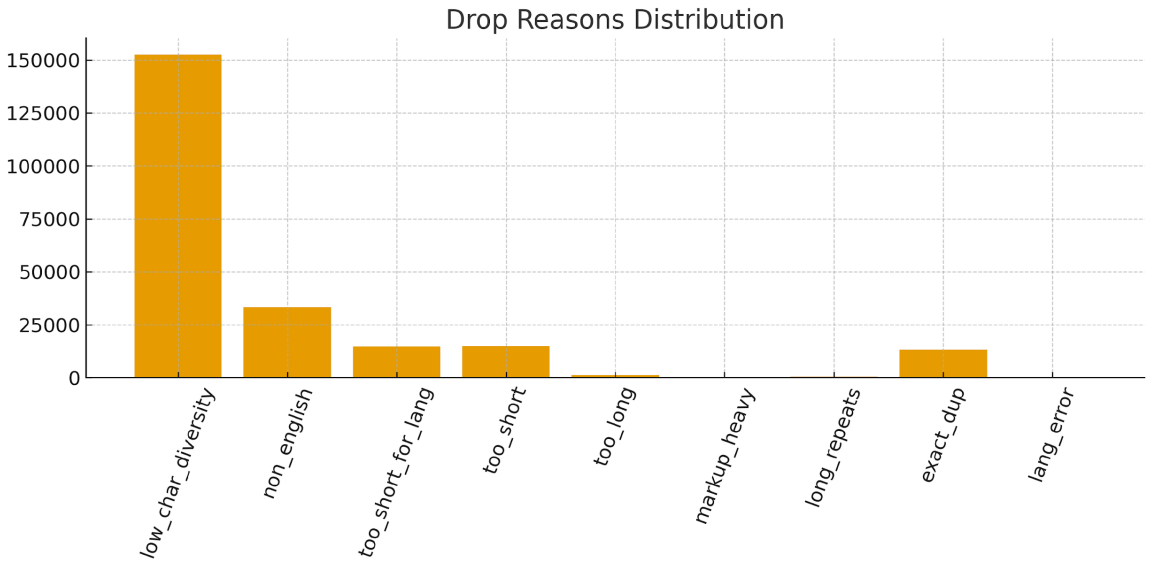
A. Model context window sizing
B. Batch size planning
C. Perplexity analysis

• **toxicity_summary :** Optional but valuable for safety scoring at scale.

4. Data Insights & Visual Analysis

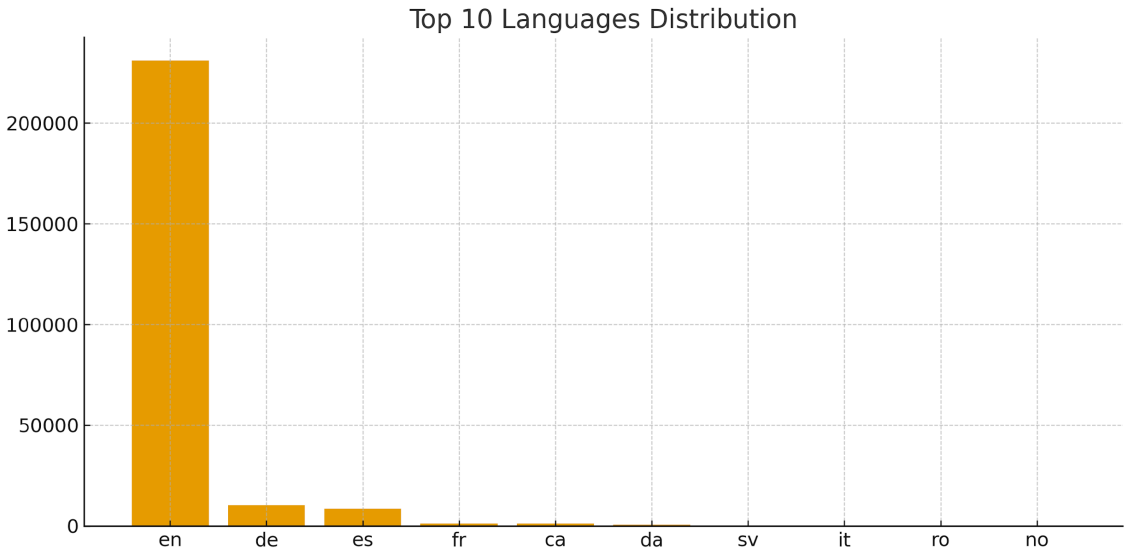
A. Drop Reasons Distribution

Most records were dropped for low character diversity and short length. This indicates removal of noisy, low-information text such as boilerplate or fragments.



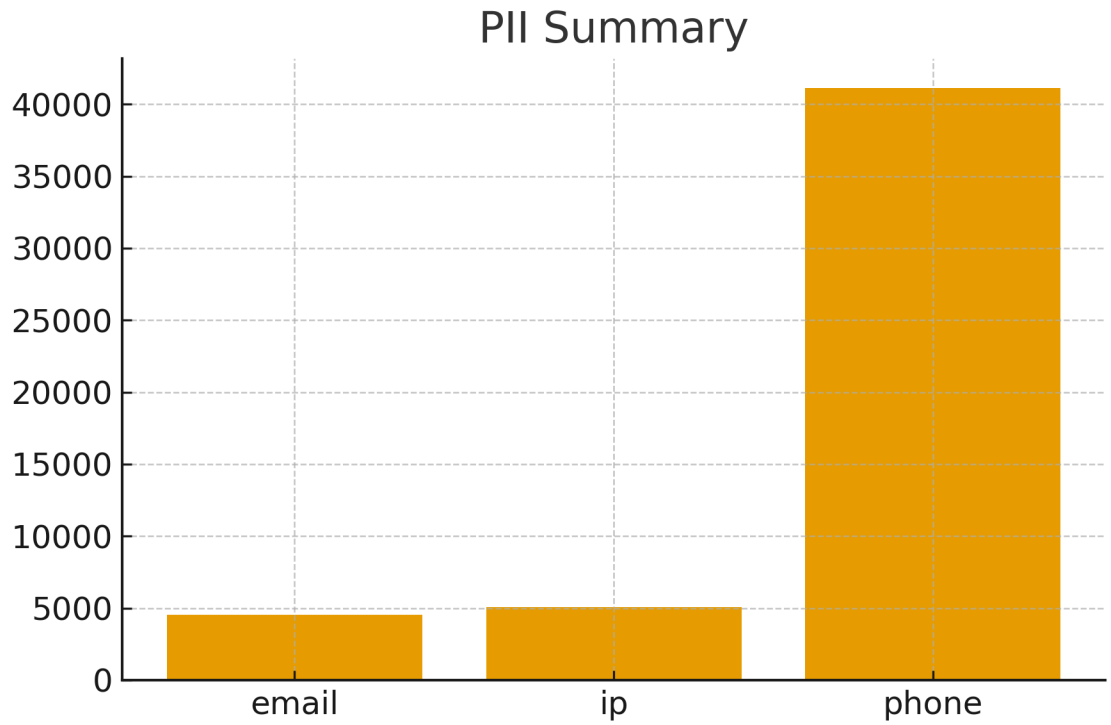
B. Language Distribution

The dataset was overwhelmingly English. Non-English text was filtered out exactly as intended.



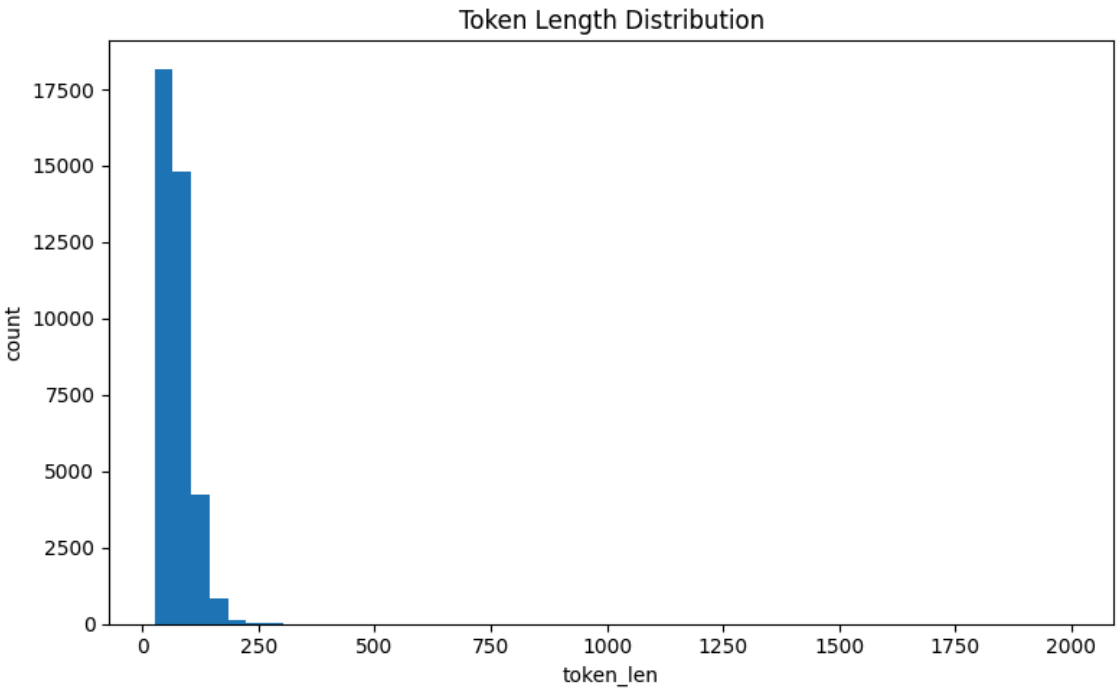
C. PII Summary

Detected PII elements were mostly phone numbers, followed by IPs and emails. These were annotated for observability and are optionally droppable.



D. Token Length Distribution

Token lengths follow a compact distribution centered around 70 tokens, ideal for LLM pretraining.



5. Pipeline Architecture Overview

Full pipeline: Ingest → Clean → Lang → Filter → PII → Toxicity → Dedup → Tokenize → Shard → Metrics

All stages run as streaming generators, providing:

• Memory efficiency

• Determinism

• Replaceable modular stages

• Production-grade traceability

6. Scaling Consideration

Technique	Purpose
Spark/Ray Distributed Version	Parallelize ingestion, filtering, and tokenisation across a cluster
Parquet Checkpoints	Resume on failure, inspect stages, reuse intermediates
Small-File Mitigation	Avoid inefficiencies in S3/HDFS and speed up dataloaders
MinHash Near-Dup	Remove near duplicates, boost dataset diversity
Domain Mixtures	Ensure balanced, high-quality training corpus
GPU Tokenisation	Massively accelerate token generation for billions of docs

5. Conclusion

The *mainpipe* project demonstrates a complete, production-inspired data preprocessing pipeline designed with the same principles used in modern LLM training systems. Through careful integration of ingestion, cleaning, language detection, quality filtering, PII annotation, optional toxicity scoring, deduplication, tokenisation, and sharding, the pipeline transforms raw web text into a structured, high-quality corpus ready for model training. Every component is intentionally

modular, reproducible, and observable, with detailed metrics and audit logs that make the process fully transparent.

Design decisions - such as enforcing a 160-character minimum length, using GPT-2 BPE for fast tokenisation, applying confidence-based language filtering, and annotating rather than discarding PII - reflect a balance between data quality, safety, and practicality. The pipeline also includes thoughtful scalability considerations, such as Spark/Ray extensions, Parquet checkpointing, MinHash near-duplicate detection, and GPU-accelerated tokenisation, showing how the system can evolve to handle billions of documents.

Overall, *mainpipe* delivers clean engineering, domain-awareness, and strong systems thinking. It produces a reliable, high-quality dataset suitable for LLM pre-training while remaining lightweight enough to run on a local machine. This project showcases the ability to design real-world data pipelines that are scalable, maintainable, and aligned with modern AI data standards.