Merimënga: A Manifest-First Pipeline for Reproducible Albanian Web Corpus Construction

Besim Kabashi and Michael Ruppert
Computational Linguistics, University of Tuebingen
besim.kabashi@uni-tuebingen.de

Computational Corpus Linguistics, FAU Erlangen-Nuremberg
michael.ruppert@fau.de

Abstract

We present Merimënga, a pipeline for reproducible Albanian web-corpus construction from Common Crawl. Rather than distributing a static text dump, we publish versioned manifests and append-only JSONL ledgers that make every retrieval and filtering decision replayable at record level. Records are addressed by (WARC filename, byte offset, byte length) and retrieved via HTTP range requests with checksum validation, enabling selective download, resumability, and exact re-materialization. On top of deterministic cleaning and deduplication, Merimënga supports teacher–student filtering: a large LLM labels a stratified sample; the resulting policy is distilled into a faster student model applied at corpus scale. The paper contributes (i) a reproducibility specification for web-corpus construction based on coordinate-addressed retrieval and decision ledgers, (ii) a concrete instantiation for Albanian with language-specific filtering, and (iii) an evaluation protocol for rerun equivalence and filter-stack ablation. Large-scale download and full-corpus filtering are ongoing; this submission focuses on methodology and auditable artifacts rather than final corpus statistics.

Keywords: Common Crawl, Reproducibility, Corpus Construction, Learned Filtering, Albanian

1. Introduction

Large web corpora are easy to grow, but much harder to reproduce, audit, and share responsibly. For lower-resource languages such as Albanian, these issues are amplified by domain concentration, noisy page structure, and practical licensing constraints around redistribution. Yet the need for such corpora is pressing: Albanian NLP, lexicography, and corpus-linguistic research all depend on large, clean text collections that are currently available only through a small number of multilingual releases with limited control over construction decisions.

Our approach combines two ideas: (1) recording every retrieval and filtering decision in append-only ledgers so that the entire construction process can be replayed from machine-readable artifacts, and (2) using LLM-based teacher–student filtering to scale quality decisions. Compared with CCNet/RefinedWeb-style quality pipelines (Wenzek et al., 2020; Penedo et al., 2023), the methodological contribution is the reproducibility framework itself, and the primary output is a replayable construction process rather than a static text release.

This design addresses a gap that is especially relevant for lower-resource languages. Large multilingual projects like OSCAR and HPLT provide invaluable data, but their release cycles are tied to project timelines, and replicating their pipelines independently requires comparable infrastructure. Merimënga targets a different use case: a research group working on a single language can build and extend a corpus from Common Crawl on modest hardware, inspect every filtering decision, adjust parameters, and incorporate new crawl snapshots incrementally—without depending on the release schedule of a multilingual project.

2. Problem Setting and Goals

The project addresses three concrete requirements that are common in large-corpus management:

1. **Reconstruction requirement:** a third party should be able to replay corpus construction decisions from machine-readable artifacts.

2. **Operational requirement:** long runs must survive partial failures, rate limits, and interrupted execution without losing state.

3. **Governance requirement:** publication should support reuse while reducing redistribution risk where possible.

These requirements motivate the core design choices described in the following sections.

3. Related Work and Positioning

OSCAR and related Common-Crawl pipelines established large multilingual releases and evolved substantially across versions (Ortiz Suarez et al., 2019; Abadji et al., 2021, 2022; OSCAR Project,

Proceedings of the 12th Workshop on Challenges in the Management of Large Corpora (CMLC-12) @ LREC 2026, pages 25–31. Palma, Mallorca. May 11, 2026.

Dimension	OSCAR-style releases	Merimënga
Primary objective	release-oriented corpus product	reproducible construction process
Primary artifact	release corpus + metadata	manifests + code + ledgers
Audit granularity	release version level	per-record decision trail
Retrieval model	download full release	fetch individual records by coordinate
Redistribution strategy	text release policies	manifest-only by default

Table 1: Positioning of our approach relative to OSCAR-style corpus releases.

2019, 2022, 2023; Brack et al., 2024). Later OSCAR versions move toward document-oriented releases with richer metadata and quality signals (OSCAR Project, 2021, 2022, 2023). More recent web-corpus projects (CCNet, mC4, RefinedWeb, FineWeb, DCLM, HPLT) emphasize strong filtering, deduplication, and ablation-driven development (Wenzek et al., 2020; Xue et al., 2021; Penedo et al., 2023, 2024; Li et al., 2024; Samuel et al., 2024). Teacher-student filtering specifically is present in FineWeb-Edu, which uses LLM-generated supervision to train a scalable quality classifier (Penedo et al., 2024; Hugging Face, 2024).

These projects share a common pattern: they process the full Common Crawl archive centrally, across all languages, and publish the result as a versioned release. While several of them release their code, replicating the full pipeline independently requires comparable infrastructure and access to the same crawl data. In practice, this means that (a) release schedules are tied to project capacity, and (b) the barrier to running such a pipeline for a single language is disproportionately high. Merimënga addresses a different point in this design space: it enables targeted, single-language corpus construction that is fully replayable, incrementally extensible, and feasible on modest hardware. The filtering techniques we employ (language identification, deduplication, learned quality scoring) are not novel in themselves; our contribution is the framework that makes their application reproducible and auditable at record level.

4. Corpus Generation and Distribution

This section describes how records are selected from Common Crawl and how the resulting corpus is distributed.

4.1. Selective Retrieval from Common Crawl

Rather than processing an entire Common Crawl snapshot, the pipeline queries Common Crawl's index for Albanian records, retrieves only those, and applies staged cleaning and filtering. Common Crawl is published in standardized archive formats (WARC and derived representations) (Common Crawl Foundation, 2026a; International Organization for Standardization, 2017).

Selection is index-based via Common Crawl index services. Index rows provide file path, byte offset, and record length, enabling targeted retrieval through HTTP range requests instead of full-segment downloads (Common Crawl Foundation, 2026b). This is a form of selective sampling: rather than downloading and processing an entire crawl snapshot (tens of terabytes), we query the index for Albanian pages only (`language=sqi, status=200, mime=text/html`) and retrieve just the matching records by their byte coordinates. Albanian accounts for roughly 0.05% of the Common Crawl archive (for comparison, OSCAR 23.01 lists approximately 497 k Albanian documents out of 1.13 billion total; HPLT v1.2 provides 1.24 million cleaned Albanian documents totalling 1.34 billion words (Samuel et al., 2024)). This reduces the data volume by several orders of magnitude compared to full-crawl processing, making single-language corpus construction feasible on a single workstation without cloud infrastructure. The same approach scales to larger setups—on a more capable server, multiple languages can be processed in parallel—but the entry barrier for a single low-resource language is low. This stands in contrast to projects like OSCAR or HPLT, which process the full archive across all languages simultaneously and therefore require substantial compute resources (Ortiz Suarez et al., 2019; Samuel et al., 2024).

For archive parsing and response extraction, we follow standards-compatible handling of WARC/HTTP records (Webrecorder, 2026; International Organization for Standardization, 2017). Deterministic reconstruction is supported by pinning each crawl snapshot (for example `CC-MAIN-2026-08`) and persisting retrieval coordinates with provenance metadata.

A limitation of this index-based sampling is that it depends on the language labels assigned by Common Crawl's own classifier. Pages that are Albanian but mislabeled (or unlabeled) in the index will be missed. We accept this trade-off in exchange for the practical efficiency of targeted retrieval, and note that the same limitation applies to any index-based approach.

Once records are retrieved, they pass through a multi-stage filtering stack—heuristic cleaning, language identification, deduplication, perplexity scor-

ing, and learned quality filtering—described in detail in Sections 5–6. Because Common Crawl is heterogeneous and noisy, we prioritize quality over volume, following established curation strategies (Wenzek et al., 2020; Ortiz Suarez et al., 2019; Abadji et al., 2022).

We release code, manifests, and decision ledgers rather than extracted full text. Downstream users retrieve content directly from Common Crawl and remain responsible for compliance with applicable copyright and terms (Common Crawl Foundation, 2024; Schäfer and Bildhauer, 2016). Reproducibility holds as long as the referenced snapshots remain publicly available (Common Crawl Foundation, 2026c,b). The pipeline design is language-agnostic: adapting it to another language requires replacing the language code in the index query and the language-specific filter resources. Because retrieval and filtering across multiple snapshots are still ongoing, we do not report a final corpus size in this version.

5. Pipeline Overview

The pipeline is implemented in Python 3.10. The retrieval and manifest-management components use only the standard library; filtering stages additionally rely on FastText for language identification (Joulin et al., 2017), KenLM for n-gram perplexity scoring (Heafield, 2011), and llama.cpp with LM Studio for local LLM inference in the teacher-labeling stage.

Crawl snapshots are processed in reverse chronological order, starting with the most recent and working backwards. This prioritizes fresh content and allows incremental expansion: each additional snapshot contributes only records not yet seen, and cross-snapshot deduplication keeps the most recent version of each page by default. Manifests have been generated for 19 crawl snapshots covering a total of 1.87 million candidate Albanian records. Download of the most recent snapshot (`CC-MAIN-2026-04`, 396,347 records, ∼12 GB) has completed successfully; the remaining snapshots are queued for incremental processing. The pipeline can be run continuously as new snapshots are published, growing the corpus over time without re-processing earlier data.

1. **Manifest generation (index level).** Either via Athena SQL or the Common Crawl Index API. Typical filters include `status=200`, `mime=text/html`, and `language=sqi`. The output is a CSV with WARC filename, byte offset, record length, and metadata.

2. **Reproducible download.** Records are retrieved by HTTP range requests against Common Crawl's S3-hosted WARC files. Neighboring byte ranges are merged to reduce the number of requests. Each attempt is logged in the fetch ledger (success/failure, HTTP status, content hash, timestamp), making runs resumable: on restart, already-fetched records are skipped. From the fetch ledger, a success manifest of retrieved records can be exported and shared so that others can retrieve the same record set.

3. **Cleaning and language verification.** Text is extracted from the retrieved WARC payloads. Heuristic quality checks (minimum text length, alphabetic character ratio, token repetition, boilerplate ratio) and a language-specific plausibility check are applied; each decision is logged to a separate cleaning ledger. The language check and the subsequent FastText filtering stage are described in Section 5.1.

4. **Deduplication and perplexity scoring.** Near-duplicate detection via shingling (Broder, 1997) removes redundant content across records. Documents that pass deduplication are optionally scored for perplexity using a KenLM n-gram language model (Heafield, 2011); implausibly high perplexity (indicating garbled text, encoding artifacts, or non-natural-language content) is a drop signal.

5. **Learned filtering.** A larger LLM labels a stratified sample of cleaned documents for quality; these labels are used to train or calibrate a faster student filter. The student's keep/drop decisions are again logged per record and exported as a keep manifest. This stage is described in detail in Section 6.

6. **Reporting.** A summary of the full pipeline funnel—records in, records kept at each stage, failure categories, and deduplication statistics—is generated from the ledgers.

The release artifact is therefore the pipeline itself rather than a text dump.

5.1. Language Filtering for Albanian

Language filtering proceeds in two stages, each targeting a different failure mode.

Heuristic pre-filter. The cleaning step includes a lightweight Albanian plausibility check that serves as a fast pre-filter before the more expensive FastText classification. It combines two signals: (a) the ratio of Albanian stopwords among all tokens, and (b) the ratio of Albanian-specific diacritics—specifically *ë* and *ç*—to total characters. These signals are combined into a composite score in

which the diacritic ratio is weighted substantially higher than the stopword ratio (currently by a factor of 12, chosen empirically). The rationale is linguistic: while Albanian shares many high-frequency function words with neighboring Balkan languages, the diacritics *ë* and *ç* are distinctive and occur frequently in Albanian text but rarely in Romanian, Serbian, or other regional languages. This makes the diacritic signal a strong discriminator even for short documents where stopword counts alone would be unreliable. The exact weighting factor is a candidate for systematic ablation; as part of our evaluation, we also test the pipeline without the heuristic pre-filter entirely, relying on FastText alone, to measure its contribution to precision and throughput.

FastText verification. Documents that pass the heuristic check are classified by a FastText language-identification model (Joulin et al., 2017). Rather than a single confidence threshold, the filter applies a two-tier acceptance strategy: a document is kept if the model's top prediction is Albanian with confidence ≥ 0.80, *or* if Albanian appears among the top three predictions with confidence ≥ 0.60. These thresholds are conservative defaults; the ledger-based design makes it straightforward to re-evaluate with different values without re-downloading or re-extracting text. The second, more permissive gate addresses a pattern common in Albanian web content: news articles and institutional pages frequently contain English headlines, embedded quotations in other languages, or code-switched passages that depress the top-1 confidence for Albanian while still leaving it among the most likely languages. Without this fallback, such documents—which are often substantively Albanian—would be systematically lost.

Both stages log their decisions per record (including the scores, thresholds applied, and reason codes), so the effect of each gate can be inspected and adjusted in subsequent runs.

To illustrate the interaction of these filters: a news article from a `.al` domain with consistent Albanian prose, frequent use of *ë*, and a FastText top-1 confidence of 0.92 passes both stages easily. An Albanian government page with an English-language header and bilingual body might score only 0.65 on top-1 but 0.72 on top-3, and would be rescued by the fallback gate. Conversely, a page in Macedonian that was misclassified as `sqi` in the Common Crawl index would fail both the diacritic check (Macedonian uses Cyrillic or Latin without *ë*) and the FastText gate. A cookie-consent dialog or navigation-only page, even if in Albanian, would be caught earlier by the minimum-text-length and boilerplate-ratio checks in the cleaning stage.

5.2. Operational Properties

Index acquisition and download are resumable: the builder persists state per page and per TLD, so interrupted runs can continue without re-fetching. Rate limiting from Common Crawl's servers is handled by circuit-breaker logic with cooldown windows; when a TLD's index queries are repeatedly throttled, that TLD is temporarily set aside and retried later. When records from multiple crawl snapshots are merged, explicit priority rules and deterministic tie-breaking ensure reproducible deduplication outcomes.

6. Learned Filtering Stage

The learned-filter stage applies a score-based keep/drop policy in two steps:

- `evaluate-threshold` selects a score threshold from labeled data under precision/recall constraints (e.g., "keep at least 90% of good documents while dropping at least 70% of noise"),
- `apply` applies this threshold to all documents: each document's score is compared against the threshold, and the keep/drop decision is logged with a machine-readable reason.

The stage is independent of how scores are produced. In the teacher–student setup, scores come from a student model trained on LLM-generated labels; the same policy mechanism works with any external scorer. The end-to-end quality gain of this stage over baseline filtering is not yet demonstrated on the final corpus.

6.1. Segment-First Student Filtering Architecture

We define the student stack as a segment-first cascade with four deterministic stages:

1. **Stage S (BlockTagger Student).** Segment-aware classification predicts functional block labels (for example MAIN/TITLE vs NAV/COOKIE/FOOTER/ADS/TEMPLATE). Outputs are (a) canonical main text, (b) boilerplate ratios, and (c) template signatures.
2. **Stage Q (DocScorer Student).** A document-level scorer produces multiple output signals from main text and structural features: a quality score, a language-purity estimate, a content-type label, and a preliminary drop reason. These signals are computed jointly by a single model with multiple output heads.
3. **Stage R (Compact Reranker, optional).** Documents where Stage Q's quality score falls in an uncertain range (neither clearly good nor

clearly bad) are re-scored by a small transformer model. The majority of documents bypass this stage entirely, keeping throughput high.

4. **Stage D (Deterministic set selection).** Final keep decisions are computed under cluster constraints (near-duplicate clusters on main text and template clusters on boilerplate signatures), without per-host caps.

This decomposition allows each stage to be evaluated and ablated independently.

6.2. Training and Policy Learning

Training follows a teacher-to-student distillation approach. A large open-weight LLM serves as the teacher: we currently use GPT-OSS-120B, a 120-billion-parameter model that can be run locally on desktop hardware with large unified memory, such as NVIDIA DGX Spark or AMD Ryzen AI MAX 395 systems with 128 GB LPDDR5x. As a comparison teacher we are evaluating Qwen3.5-27B on workstations with NVIDIA RTX 5090 GPUs. Inference is served locally via llama.cpp and LM Studio. While this hardware is not inexpensive, it is far below the scale of a data-center deployment: a small number of local machines can label stratified samples at sufficient throughput for a single-language corpus.

The distillation process works as follows:

1. **Teacher labeling.** The teacher LLM labels stratified samples of cleaned documents at document level, assessing content quality, language purity, and topical relevance on a structured rubric. Stratification ensures coverage across quality tiers, domains, and document lengths so that the student sees representative positives and negatives. Where segment-level annotation is available, the teacher additionally labels functional blocks (main content vs. boilerplate).

2. **Student training.** Segment-level labels train the BlockTagger (Stage S); document-level quality and preference-style targets train the DocScorer (Stage Q). The optional reranker (Stage R) is trained only on cases where Stage Q is uncertain.

3. **Threshold calibration.** Deployment thresholds are selected on held-out labeled data under precision/recall constraints (e.g., keeping $\geq$90% of teacher-approved documents) and then frozen for the production run.

4. **Deterministic execution.** Stage D applies fixed tie-break rules and emits machine-readable drop reasons (e.g., near-duplicate, template-dominated, mixed-language, boilerplate-heavy).

For Albanian, training and evaluation emphasize hard negatives that are characteristic of the language's web presence: code-switching with closely related Balkan languages, template-heavy pages from a small number of dominant domains, and thin SEO or listing content.

7. Conclusion and Future Work

Merimënga demonstrates that reproducible web-corpus construction for a single language does not require the infrastructure of a large multilingual project. By querying Common Crawl's index for Albanian records and fetching only those via byte-level range requests, the pipeline reduces the data volume by orders of magnitude, making it feasible to run on desktop hardware. The construction process—from index query to final keep/drop decision—is fully recorded and can be replayed, inspected, and extended by third parties.

By separating the reusable construction recipe from the derived text, the manifest-first release model reduces redistribution concerns while preserving transparency. New crawl snapshots can be incorporated incrementally, and filter parameters can be revised without re-downloading data. For lower-resource languages, where research groups cannot always wait for the next release of a multilingual project, this independence is a practical advantage.

Planned evaluation work includes systematic ablation of the filter stack (heuristic pre-filter, FastText thresholds, learned filter), measurement of rerun equivalence across independent pipeline executions, and reporting of per-stage yield and drop-reason distributions once the full multi-snapshot run completes.

8. Ethics Statement

Web-crawled text can contain copyrighted material, personal data, and sensitive content. To reduce redistribution risk, we release manifests and code rather than full text; users retrieve content directly from Common Crawl. Each filtering decision is logged with a machine-readable reason, enabling post-hoc inspection. The approach does not guarantee that reconstructed material is free of personal or sensitive information; downstream users remain responsible for additional safeguards appropriate to their jurisdiction and use case.

9. Limitations

Full-corpus filtering is ongoing; we do not yet report final corpus size, yield, or end-to-end quality comparisons. Reproducibility depends on continued public availability of the referenced Common Crawl

snapshots; changes in access patterns or snapshot retention can affect replay feasibility. Deterministic record identifiers provide stable addressing within a snapshot but do not eliminate all sources of variance (e.g., transient retrieval failures or extraction corner cases), which are mitigated operationally via resumable ledgers and integrity checks.

Language identification and quality filtering are imperfect for Albanian, particularly in the presence of code-switching with closely related Balkan languages, template-heavy pages, and domain concentration. Learned filtering inherits the assumptions of the teacher model and may amplify annotation biases; we do not yet claim validated quality improvements from this stage on the final corpus.

10. Bibliographical References

Julien Abadji, Pedro Ortiz Suarez, Laurent Romary, and Benoit Sagot. 2022. Towards a cleaner document-oriented multilingual crawled corpus. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 4344–4355, Marseille, France. European Language Resources Association.

Julien Abadji, Pedro Javier Ortiz Suarez, Laurent Romary, and Benoit Sagot. 2021. Ungoliant: An optimized pipeline for the generation of a very large-scale multilingual web corpus. In *Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC-9)*, pages 1–9, Mannheim. Leibniz-Institut fur Deutsche Sprache.

Manuel Brack, Malte Ostendorff, Pedro Ortiz Suarez, Jose Javier Saiz, Inaki Lacunza Castilla, Jorge Palomar-Giner, Alexander Shvets, Patrick Schramowski, Georg Rehm, Marta Villegas, and Kristian Kersting. 2024. Community oscar: A community effort for multilingual web data. In *Proceedings of the Fourth Workshop on Multilingual Representation Learning (MRL 2024)*, pages 232–235, Miami, Florida, USA. Association for Computational Linguistics.

Andrei Z. Broder. 1997. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences*, pages 21–29.

Common Crawl Foundation. 2024. Common crawl terms of use. https://commoncrawl.org/terms-of-use. Accessed 2026-02-25.

Common Crawl Foundation. 2026a. Common crawl: Get started. https://commoncrawl.org/get-started. Accessed 2026-02-25.

Common Crawl Foundation. 2026b. Common crawl index. https://index.commoncrawl.org/. Accessed 2026-02-25.

Common Crawl Foundation. 2026c. Common crawl on aws open data registry. https://registry.opendata.aws/commoncrawl/. Accessed 2026-02-25.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197.

Hugging Face. 2024. Huggingfacefw/fineweb-edu-classifier - model card. https://huggingface.co/HuggingFaceFW/fineweb-edu-classifier. Accessed 2026-02-25.

International Organization for Standardization. 2017. Iso 28500:2017 information and documentation – warc file format. International Standard.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 427–431.

Jeffrey Li, Alex Fang, Georgios Smyrnis, et al. 2024. Datacomp-lm: In search of the next generation of training sets for language models.

Pedro Javier Ortiz Suarez, Benoit Sagot, and Laurent Romary. 2019. Asynchronous pipelines for processing huge corpora on medium to low resource infrastructures. In *Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC-7)*, pages 9–16, Cardiff, UK. Leibniz-Institut fur Deutsche Sprache.

OSCAR Project. 2019. Oscar 2019 - oscar documentation. https://oscar-project.github.io/documentation/versions/oscar-2019/. Accessed 2026-02-25.

OSCAR Project. 2021. Oscar 21.09 - oscar documentation. https://oscar-project.github.io/documentation/versions/oscar-2109/. Accessed 2026-02-25.

OSCAR Project. 2022. Oscar 22.01 - oscar documentation. https://oscar-project.github.io/documentation/versions/oscar-2201/. Accessed 2026-02-25.

OSCAR Project. 2023. Oscar 23.01 - oscar documentation. https://oscar-project.github.io/documentation/versions/oscar-2301/. Accessed 2026-02-25.

Guilherme Penedo, Hynek Kydlicek, and all. 2023. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only.

Guilherme Penedo, Hynek Kydlicek, Loubna Ben Allal, et al. 2024. Fineweb: Decanting the web for the finest text data at scale.

David Samuel, Andrey Kutuzov, Satya Almasian, et al. 2024. Hplt: High-performance language technologies datasets.

Roland Schäfer and Felix Bildhauer. 2016. Common crawl and web corpus construction: Practical and legal considerations. https://commoncrawl.org/blog. Accessed 2026-02-25.

Webrecorder. 2026. warcio: Streaming warc reader/writer for python. https://github.com/webrecorder/warcio. Accessed 2026-02-25.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzman, Armand Joulin, and Edouard Grave. 2020. Ccnet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4003–4012.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mt5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498.