

Scalable RDF Statistics: Streaming Analysis of Billion-Triple Graphs

Dominik Tomaszuk¹

¹University of Białystok, Poland

Abstract

We present `rdfstats_stream.py`, a lightweight streaming tool for computing descriptive statistics and predicate co-occurrence patterns over very large RDF graphs directly from N-Triples dumps. The tool runs in bounded memory using a single pass combined with external sorting and multi-way merges. It outputs: (i) global predicate distributions with entropy and Gini indices; (ii) per-predicate distinct counts and degree summaries for subjects and objects; (iii) a subject-level predicate co-occurrence matrix with association measures (support, confidence, lift, PMI, Jaccard); (iv) optional association rules filtered by user-defined thresholds; and (v) interest-focused summaries with alias-aware aggregation across HTTP/HTTPS and CURIE expansions. The demo emphasizes scalable processing, robust parsing, and practical CSV outputs (with optional plots) for fast inspection without loading the graph into a triplestore.

Keywords

RDF, large-scale graph analytics, streaming, external sorting, association rules, co-occurrence

1. Introduction

The Web of Data contains increasingly large and heterogeneous RDF graphs, spanning domains¹ such as open government data, knowledge bases, scientific metadata, and cultural heritage. While small RDF examples are valuable pedagogically, operational knowledge engineering often requires inspection and profiling of *very large* dumps where loading into a triplestore is expensive or infeasible under commodity constraints. In practice, dataset users and integrators frequently need *reproducible summaries* that characterize predicate prevalence and skew, capture predicate co-occurrence patterns on shared subjects, and reveal recurring structural signatures.

Several profiling approaches exist, including large-scale platforms for RDF analytics and cluster-based profiling, as well as systems built around compact RDF representations. For example, LODStats and DistLODStats provide broad statistics computed using dedicated pipelines and distributed execution [1, 2], while HDT focuses on compressed RDF storage and exchange [3]. In contrast, our goal is a minimal-dependency tool for fast, local, streaming profiling of plain N-Triples dumps, emphasizing bounded-memory execution on commodity machines.

We address this need with `rdfstats_stream.py`, which follows an external-memory approach and leverages standard external sorting and merge-based aggregation [4, 5]. One command produces multiple CSV summaries and optionally generates plots for rapid inspection.

Contributions. This demo contribution is fourfold: (i) a streaming and external-sorting architecture for scalable RDF profiling from N-Triples; (ii) global and per-predicate metrics including entropy/Gini, distinct subject/object counts, and degree summaries; (iii) predicate co-occurrence and association rules computed from subject-level predicate sets; and (iv) optional alias-aware aggregation to consolidate HTTP/HTTPS and CURIE-based predicate variants².

K-CAP 2025 Companion Volume

✉ dtomaszuk@uwb.edu.pl (D. Tomaszuk)

🌐 <https://orcid.org/0000-0003-1806-067X> (D. Tomaszuk)

🆔 0000-0003-1806-067X (D. Tomaszuk)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

¹<https://www.rdfdata.org/data.html>

²<https://www.w3.org/TR/curie/>

2. Design Goals and Data Model Assumptions

Input scope. The tool targets RDF graphs serialized in N-Triples³, optionally compressed using .gz, .bz2, or .xz. It treats each triple as a line-level record and performs no inference; all statistics are derived from explicit triples in the dump.

Scalability objective. The core design constraint is bounded main-memory execution, even when the number of triples exceeds RAM capacity by orders of magnitude. The tool therefore separates computation into a single streaming pass (materializing compact intermediate files) followed by external sorting and merge phases that compute exact aggregations.

Pragmatic interpretability. Outputs are CSV-based and stable under deterministic sorting/merging, enabling reproducible analytics and downstream use in notebooks or dashboards.

3. System Overview

The implementation is provided as a single Python script (`rdfstats_stream.py`, v1.3). The pipeline is organized as a sequence of external-memory transformations operating on intermediate tab-separated files.

3.1. Streaming pass and intermediate materialization

During the streaming scan of the input, each parsed triple (s, p, o) contributes to:

- **Predicate counts:** a global in-memory counter `pred_counts[p]++`, used to compute predicate shares, entropy and Gini.
- **Intermediate raw files:** tab-separated records written to a temporary directory:
 - `ps.raw`: pairs (p, s) for distinct-subject counting and per-predicate subject degree statistics,
 - `po.raw`: pairs (p, o) for distinct-object counting and per-predicate object degree statistics,
 - `sp.raw`: pairs (s, p) to construct subject-level predicate sets and compute predicate co-occurrence,
 - `pso.raw` (optional): triples (p, s, o) used to count unique (s, o) pairs and duplicate excess per predicate.

Input parsing relies on a robust N-Triples pattern that extracts the first two tokens (subject and predicate) and captures the remaining object token up to the terminating dot. Lines that do not match the expected syntax are skipped, and the tool reports the number of rejected lines as a warning.

3.2. External sorting and multi-way merge

To remain memory-bounded, each raw file is processed via external sorting: the file is read in chunks of `--chunk-lines` records, sorted in memory, and materialized as a temporary chunk file. Subsequently, chunk files are merged using a multi-way merge iterator [4, 5]. The merge streams are then aggregated to compute exact statistics by maintaining only the current key group (e.g., the current (p, s) block) plus small per-predicate summary maps.

Interest-focused profiling and alias consolidation (optional). The tool supports a user-provided set of “interesting” predicates (`--predicates-file`) and consolidates HTTP/HTTPS variants under a single display key. It also expands CURIEs using configurable prefix maps, enabling stable reporting even when datasets mix different predicate spellings and schemes.

³<https://www.w3.org/TR/n-triples/>

Output	Content
1_predicate_distribution.csv	predicate counts and shares
1_meta_entropy_gini.csv	entropy (bits) and Gini-style concentration
2_per_predicate_basic.csv	distinct subjects/objects, degree averages, coverage
3_predicate_cooccurrence.csv	co-occurrence counts + measures
4_association_rules.csv	rules $A \rightarrow B$ filtered by support/confidence
5_per_predicate_graph_metrics.csv	centralities on Top- K co-occurrence graph
8--13 (optional)	interest-focused summaries (with aliasing)

Table 1

Main CSV outputs produced by `rdfstats_stream.py`.

4. Computed Statistics and Measures

The script produces a set of numbered CSV files. Table 1 summarizes their purpose.

4.1. Predicate distribution: entropy and Gini

Let N be the number of triples and $c(p)$ the count of predicate p . The script computes predicate probabilities $\Pr(p) = c(p)/N$. Shannon entropy is reported in bits:

$$H = - \sum_p \Pr(p) \log_2 \Pr(p),$$

and a simple concentration proxy is computed as:

$$G = 1 - \sum_p \Pr(p)^2.$$

Values closer to 0 indicate dominance of few predicates, while higher values indicate a more even spread.

4.2. Per-predicate distinct counts and degree summaries

From the sorted `ps.raw`, the merge phase groups consecutive records with identical (p, s) . For each predicate p , the tool computes: (i) the number of unique subjects $\#\{s \mid \exists o : (s, p, o)\}$, (ii) the average multiplicity of p per subject (out-degree under p), and (iii) min/max multiplicities as distribution boundaries.

Analogously, from `po.raw`, it computes unique objects and object-side multiplicities under p . Additionally, the script estimates predicate coverage as the percentage of all distinct subjects that exhibit p :

$$\text{pct}(p) = \frac{\#\{s \mid \exists o : (s, p, o)\}}{\#\{s \mid \exists p', o : (s, p', o)\}} \cdot 100.$$

4.3. Subject-level predicate co-occurrence and association rules

To extract structural patterns, the tool computes co-occurrence of predicates on the same subject. The sorted `sp.raw` stream is grouped by subject s to form a set P_s of predicates used with s . To limit combinatorial growth, the algorithm restricts attention to the Top- K most frequent predicates and optionally includes user-selected predicates. A safety bound `--max-preds-per-subject` limits the number of predicates per subject.

For each unordered pair $\{a, b\}$ with $a \neq b$ in P_s , the tool increments the co-occurrence count n_{ab} , and records per-predicate subject support $n_a = \#\{s \mid a \in P_s\}$.

From n_{ab} , n_a , n_b , and the number of distinct subjects S , the script computes standard association measures: support, confidence, lift, Jaccard, and PMI. Association rules $a \rightarrow b$ are filtered by `--min-support` and `--min-conf`.

4.4. Optional graph metrics on the co-occurrence network

When `networkx` is available, the tool builds an undirected weighted co-occurrence graph on Top- K predicates and computes degree centrality, weighted degree, and PageRank. These metrics highlight structurally prominent predicates in the reduced co-occurrence network.

5. Demonstration Scenario

The demo illustrates end-to-end profiling on multi-gigabyte RDF dumps. A representative invocation is:

```
python rdfstats_stream.py \  
  --input wikidata.nt.gz --outdir ./results \  
  --chunk-lines 500000 --topk 200 \  
  --min-support 0.01 --min-conf 0.5 \  
  --predicates-file ./interesting.txt \  
  --prefixes ./prefixes.json \  
  --plots --plots-topN 30 --plots-dpi 150 --plots-curie-labels
```

The resulting directory contains CSV summaries (Sections 4) and optional PNG plots. The plotting mode provides fast visual inspection of skew and structure, including: (i) a histogram of predicate counts, (ii) a Top- N predicate frequency bar chart, and (iii) a co-occurrence heatmap for the most supported predicates. If an interest list is provided, an additional heatmap emphasizes patterns around the selected predicate subset.

6. Conclusions and Future Work

We presented `rdfstats_stream.py`, a practical bounded-memory tool for profiling very large RDF graphs directly from N-Triples dumps. By combining a single streaming pass with external sorting and merge-based aggregation, it produces reproducible summaries of predicate distributions, per-predicate subject/object statistics, and subject-level predicate co-occurrence with association measures and rules. An optional interest mode consolidates aliases to support targeted reporting on heterogeneous datasets.

The current implementation focuses on explicit N-Triples analysis and does not compute inference-based statistics. Future work includes typed summaries (class-aware patterns), richer structural motifs beyond predicate pairs, and approximate early-feedback methods (e.g., sketches) for faster interactive exploration on extremely large inputs.

Availability

The code, example configurations, and a screencast are available at: https://github.com/domel/rdfstats_stream. The demo video is accessible at: https://domel.github.io/rdfstats_stream/.

Acknowledgments

This work has been supported by COST Action CA23147, *Global Network on Large-Scale, Cross-domain and Multilingual Open Knowledge Graphs (GOBLIN)*, funded by COST (European Cooperation in Science and Technology) under the Horizon Europe Framework Programme.

Declaration on Generative AI

During the preparation of this work, the author used GPT-5.4 to assist with manuscript formatting and to check grammar and spelling. The author reviewed and edited the suggested changes and takes full responsibility for the final content of the publication.

References

- [1] I. Ermilov, J. Lehmann, S. Auer, LODStats – large scale dataset analytics for linked open data, in: *The Semantic Web – ISWC 2013*, 2013. doi:10.1007/978-3-642-33876-2_31.
- [2] G. Sejdiu, J. Lehmann, M. Hartung, G. Rojas, S. Auer, DistLODStats: Distributed computation of rdf dataset statistics, in: *The Semantic Web – ISWC 2018*, 2018. doi:10.1007/978-3-030-00668-6_13.
- [3] J. D. Fernández, C. Gutierrez, M. A. Martínez-Prieto, A. Polleres, Binary RDF representation for publication and exchange (HDT), *Journal of Web Semantics* (2013). doi:10.1016/j.websem.2013.01.002.
- [4] J. S. Vitter, External memory algorithms and data structures: Dealing with massive data, *ACM Computing Surveys* 33 (2001) 209–271. doi:10.1145/384192.384193.
- [5] J. S. Vitter, External memory algorithms and data structures, in: *Handbook of Massive Data Sets*, Springer, 2002. doi:10.5555/779232.779243.