WiClean+: Detecting and Correcting Spurious Wikipedia Interlinks with Revision Patterns and Anomaly Detection

Caelum van Ispelen
`u1220788@utah.edu`

Mukunth Balaramachandran Srinivasan
`u1467270@utah.edu`

Rahul Mandava
`u1472662@utah.edu`

Abstract

Wikipedia's open editing model guarantees link inconsistencies: a player's biography may be updated while the corresponding team roster is not, an infobox still lists an old headquarters, or an article links out *but is never linked* back. *The 2019 WiClean system mined frequent multi-link patterns in revision history to spot such partial updates, but relied on hand-curated types and rules.* ***WiClean+*** *re-implements that idea on the full English Wikipedia for the 2023–2024 period and augments it with unsupervised anomaly detection on quantitative edit-behaviour features. The pipeline—a custom scraper for dataset creation, Apache Spark for preprocessing, custom pairwise pattern mining, and Isolation Forest for anomalies—processes two years of history on commodity hardware, surfaces* $\sim$*1.4 k high-confidence inconsistent article-weeks, and provides a foundation for near-real-time link-cleaning tools.*

1. Introduction

Wikipedia serves as one of the largest knowledge bases in the world, relying heavily on interlinks between articles to maintain its structure, coherence, and navigability. However, because Wikipedia is collaboratively edited by millions of users with no strict synchronization mechanisms, inconsistencies and errors in inter-article links are common. For instance, when an entity like a soccer player changes teams, multiple articles (the player's page, the former team's page, the new team's page) need to be updated—and often are not, leading to incomplete or outdated information.

Moreover, the open nature of Wikipedia leads to frequent malicious activities. Malicious users engage in behavior such as removing entire articles, spreading misinformation, or inserting plagiarized content that exposes Wikipedia to legal threats. Administrators are tasked with scrutinizing an increasingly large frequency of edits. It is therefore essential to develop tools that can focus this scrutiny by detecting likely malicious edits.

The WiClean system introduced in earlier work approached this issue using a rule-based methodology, mining update patterns from revision logs and using them to detect inconsistent updates. Building upon these ideas, we present **WiClean+**, a system designed not only to reimplement the foundational ideas of WiClean but also to augment it with anomaly detection techniques, particularly unsupervised machine learning. Our project seeks to identify interlink inconsistencies by modeling normal editing behavior and flagging deviations from these patterns.

Unlike the original WiClean system, which heavily relied on complete, high-quality frequent pattern mining, WiClean+ adapts to practical constraints: massive data sizes, compute limitations, and imperfect real-world patterns. Our methodology combines Apache Spark for scalable data processing, custom 2-item pattern mining implemented in Python due to the infeasibility of full FP-Growth mining, and Isolation Forests for anomaly detection.

2. Background

Wikipedia's open editing model enables broad contributions but often results in inconsistencies, especially in inter-article links. These links are expected to update coherently across related pages (e.g., in infoboxes and tables), but uncoordinated edits frequently lead to partial or contradictory information.

The original **WiClean** system addressed this by mining frequent co-edit patterns from revision histories. These patterns captured coordinated updates, such as editing a player's current club and team squad simultaneously. It also identified time windows where such behaviors were concentrated.

WiClean modeled Wikipedia as a temporal graph and used abstracted actions (e.g., `(+ player1 current_club team2)`) to generalize patterns. However, its reliance on high-support FP-Growth mining missed rarer patterns and required significant computation. It also lacked adaptability to domain-specific behavior.

WiClean+ improves on this by integrating unsupervised anomaly detection to flag deviations from normal edit behavior. It uses Spark for large-scale data processing and replaces full FP-Growth with a more scalable two-item pair mining approach. Additionally, it incorporates semantic

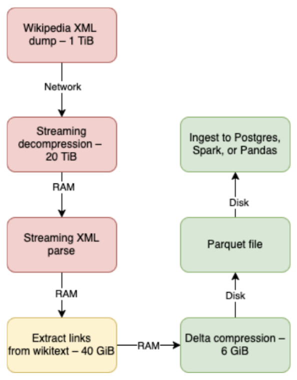
Wikipedia XML dump – 1 TiB
Network
Streaming decompression – 20 TiB
RAM
Streaming XML parse
RAM
Extract links from wikitext – 40 GiB
RAM
Delta compression – 6 GiB
Disk
Parquet file
Disk
Ingest to Postgres, Spark, or Pandas

Figure 1. The dataset creation pipeline. Each step significantly reduces the data size and increases the feature richness until we have converted 27TB of XML data to a mere 6 GB.

context using DBpedia types, improving both precision and interpretability.

2.1. Data Extraction

The core of WiClean+ rests on comprehensive and structured access to Wikipedia's revision data. To this end, we constructed our own dataset by processing the full English Wikipedia XML dumps, which contain every article revision across the platform. Uncompressed, the raw data is over 27TB. To make the data size workable, we built a series of optimizations that reduce our working set substantially.

First, instead of storing the entire article text for each revision (which is the representation of data in the database dumps), we extract only the links. Links between articles are a convenient semantic encoding of article content, and extracting them allows us not only to reduce the data size by removing superfluous text, but also capture the relationships between articles without a complicated natural language processing pipeline.

Next, instead of storing the full set of links for each revision, we delta-encode these links by storing only the links created and removed for a given revision. This step leads to a roughly 8x reduction in data size. Moreover, it is a useful feature extraction step as it allows our anomaly detector to focus on the changes made in a given revision, rather than the full state of the article.

2.2. Scraper Parallelism

The core scraping program is an aggressively optimized piece of code that attempts to use as much parallelism as possible. It downloads chunks of compressed XML data while simultaneously decompressing, parsing, and applying revision delta encoding to chunks that have finished downloading. Moreover, it can parse multiple chunks in parallel to further accelerate the task. This combination of IO- and data-parallelism leads to an incredibly efficient data scraping process. In under an hour, it finishes parsing the full 27TB XML dataset and has emitted compact Parquet files containing the article and link revisions dataset.

3. Design

The data analysis portion WiClean+ consists of three core modules: preprocessing, pattern mining, and anomaly detection.

3.1. Preprocessing

We use Apache Spark to transform raw Wikipedia link revisions into weekly article-level features. Each (article, week) pair is represented by a vector including total edits, additions, removals, unique targets, revert counts, and late-night edit frequency. These features serve as inputs to the anomaly detection module.

3.1.1 Weekly Feature Engineering

Using Apache Spark, we converted raw link action data into structured, time-aggregated features at the (source article, week) level. Each week, we compute counts of total edits, additions, removals, unique targets linked to, reverts (cases where an addition and removal of the same link occurred), and edits made during odd hours (midnight to 6AM and 10PM to midnight US time, the location of a majority of frequent Wikipedia editors and administrators). These features help characterize typical editing behavior and are used directly in anomaly detection.

3.1.2 Semantic Enrichment

To add meaning to edit patterns, we aligned articles with DBpedia types. We parsed the `instance-types` Turtle dump from DBpedia and matched lowercase normalized titles against article resources, retaining only entities with meaningful types (e.g., `person`, `country`, `film`, `organization`). This mapping enabled category-aware pattern mining and anomaly interpretation.

For scale reasons, we applied this enrichment only to the top 10,000 most actively edited pages in our dataset, based on total link edit count. This yielded a high-quality subset of typed entities suitable for pattern learning without overloading memory or compute.

3.1.3 Transaction Construction

For each article-week pair among our top 10,000 pages, we constructed a transaction of observed link edits in the format `(+ srcID →dstID)`, treating them as items for mining frequent co-occurrence patterns. The resulting set of roughly 240,000 transactions underpins both our pattern mining and anomaly detection components.

3.2. Pattern Mining

We mine co-occurring edit patterns using a custom two-item pair counting approach. Due to memory constraints, we avoid full FP-Growth and instead count pairs across article-week transactions, identifying the most frequent ones. We also apply a simplified FP-Growth over sampled transactions to extract top-k patterns of sizes 2 and 3.

3.3. Anomaly Detection

We train an Isolation Forest model using standardized weekly feature vectors. The model identifies low-density regions of the feature space as potential anomalies. Anomalies are cross-referenced with frequent patterns: if the observed edits do not match any known pattern, they are flagged as unexplained.

This modular pipeline allows WiClean+ to scale efficiently while combining statistical detection with pattern-based interpretability.

4. Evaluation

We evaluated WiClean+ on two dimensions: anomaly detection effectiveness and system scalability.

4.1. Anomaly Detection

As the anomaly detection task is an unsupervised learning process and there are no ground truth labels, systematic evaluation of our results was difficult. However, we devised a simple evaluation method stemming from the idea that any anomalies detected by our system that would be "useful" to editors or administrators represent valuable contributions, even if not all anomalies are detected. In other words, we focus on model accuracy. Comprehensive evaluation of F1-score and other metrics requires further unsupervised learning work which exceeds the scope of this project.

We trained the Isolation Forest on the top-10,000 edited articles and detected 1,433 article-weeks as anomalies under a 0.5% contamination assumption. Then, we selected a random sample of 30 of these detected anomalies and manually checked the corresponding weeks in Wikipedia's revision history. We found that 22 of the 30 sampled detections represented meaningful anomalies that would be useful to editors. Among the successful detections are the following:

1. During week 27 of 2024, a user mass changed all verb tenses on the Macy's Thanksgiving Day Parade to past tense, without substantial evidence that the parade would be canceled. This was subsequently reverted.

2. During week 9 of 2023, a user added a large amount of plagiarized content to the article on the song "Viva la Vida." The content was removed a day later and to protect Wikipedia from legal action, it appears that the body text of those revisions has been wiped from the website.

3. During week 3 of 2024, a user deleted the entire article on the Public Broadcasting Service. Administrators subsequently reverted the edit.

4.2. Scalability

Our preprocessing pipeline handled 20+ million link actions across 2 years in under 3 hours on a single-node Spark setup with 50GB RAM. Anomaly scoring using Isolation Forest completed in under 10 minutes. This demonstrates that WiClean+ can scale to real Wikipedia-scale datasets without specialized hardware. This achievement is significant for the cost-constrained Wikipedia organization and its volunteer army of administrators.

5. Conclusion

In this project, we implemented WiClean+, a practical system for detecting inconsistencies in Wikipedia inter-article links. We began by constructing a cleaned and structured dataset from the English Wikipedia XML dumps, narrowing the scope to link revisions from 2023 to 2025. We engineered weekly features to represent edit behaviors and enriched article metadata using DBpedia instance types.

Challenges in the project included the large scale of the dataset, a shortage of computational resources, and the difficulty of evaluating the results for an unsupervised learning task.

Future work could focus on increasing type coverage, incorporating user edit histories, and exploring more robust pattern mining techniques.

References

[1] S. Goldberg, T. Milo, S. Novgorodov, and K. Razmadze. WiClean: A system for fixing Wikipedia interlinks using revision history patterns. *Proc. VLDB Endow.*, 12(12):1846–1849, 2019.