Wikipedia Edit War Analyzer

End-to-End Python + Streamlit Data Analysis Project

Edit War Detection | Controversy Scoring | Interactive Dashboard

100%	4	6	5000	6
Python	Libraries	Modules	Max Revisions	Charts

1. Project Overview

The Wikipedia Edit War Analyzer is a complete data analysis and visualization system that fetches the full revision history of any Wikipedia article using the public MediaWiki REST API. It automatically detects edit wars, classifies editors as bots or humans, performs keyword-based NLP on edit comments, and computes a weighted Controversy Score (0-100). Results are displayed in a dark-themed Streamlit dashboard with six interactive Plotly charts and a CSV export feature.

Problem Statement

Wikipedia articles on controversial topics (politics, science, history) frequently experience 'edit wars' -- repeated reversions between editors with opposing views. Identifying these patterns manually is tedious. This tool automates detection.

Core Objectives

* Fetch and paginate thousands of Wikipedia revisions automatically.
* Detect reverts (key signal of edit wars) via keyword analysis.
* Classify bot vs. human editors by username convention.
* Identify abnormal edit-frequency spikes (rapid-fire editing periods).
* Compute a single, interpretable Controversy Score per article.
* Visualise all findings in an interactive web dashboard.

2. Project Architecture & File Structure

File	Responsibility
config.py	All constants: API URL, keywords, score weights
api_fetcher.py	MediaWiki API requests + rvcontinue pagination
data_processor.py	Revert, bot, spike & NLP detection on DataFrame
controversy_score.py	Weighted log-normalised Controversy Score (0-100)
visualizer.py	6 Plotly charts returned as Figure objects
app.py	Streamlit dashboard -- entry point
requirements.txt	4 pip dependencies
README.md	Full project documentation

Data Flow

```
User Input (Article Title)
        |
api_fetcher.py  ->  MediaWiki API  ->  raw DataFrame
        |
data_processor.py  ->  enriched DataFrame
   (is_revert, is_bot, is_spike_day, nlp_*, size_delta)
        |
controversy_score.py  ->  score dict  {score, label, breakdown}
        |
visualizer.py  ->  6 Plotly Figures
        |
app.py (Streamlit)  ->  Dashboard at http://localhost:8501
```

3. Module Deep-Dives

config.py

* WIKIPEDIA_API_URL: https://en.wikipedia.org/w/api.php
* REVISIONS_PER_REQUEST = 500 (API maximum per call)
* MAX_REVISIONS = 5000 (cap to keep large articles manageable)
* RATE_LIMIT_DELAY = 0.5 seconds between paginated API calls
* REVERT_KEYWORDS: revert, undo, rv, vandalism, rollback, restore ...
* CONFLICT_KEYWORDS: 4 groups -- hostility, dispute, revert_lang, protection
* Score weights: reverts 40%, unique editors 30%, edit spikes 30%

api_fetcher.py -- fetch_revision_history(article_title)

* Opens a requests.Session() for connection pooling efficiency.
* Sends rvprop=ids|timestamp|user|comment|size to get all needed fields.
* rvdir=newer ensures oldest-first ordering (ideal for time-series charts).
* Follows rvcontinue token automatically to get all pages of results.
* Raises ValueError for empty title or missing article.
* Raises RuntimeError for connection errors, timeouts, or API errors.
* Returns a clean pandas DataFrame with 5 columns, sorted by timestamp.

data_processor.py -- process_revisions(df)

Column Added	Method	Logic
is_revert	detect_reverts()	Comment contains REVERT_KEYWORDS
is_bot	detect_bots()	Username matches \bbot\b regex
is_spike_day	detect_spikes()	Day edits >= mean_daily * 3.0
nlp_hostility	nlp_conflict_flags()	Comment has hostility keywords
nlp_dispute	nlp_conflict_flags()	Comment has dispute keywords
nlp_revert_lang	nlp_conflict_flags()	Comment has revert language
nlp_protection	nlp_conflict_flags()	Comment has protection keywords
nlp_any_conflict	nlp_conflict_flags()	OR of all 4 NLP columns
size_delta	compute_edit_size_delta()	Byte change per revision

4. Controversy Score Algorithm

The Controversy Score is a composite metric in the range [0, 100] derived from three independently normalised components. A logarithmic scale ensures that both small and very large articles produce meaningful, comparable scores.

Formula

```
revert_rate    = total_reverts / total_edits x 100
editor_density = unique_editors / total_edits x 100
spike_fraction = spike_days / total_days x 100

revert_norm  = log10(revert_rate   + 1) / log10(30 + 1)   [max = 30]
editor_norm  = log10(editor_density + 1) / log10(60 + 1)  [max = 60]
spike_norm   = log10(spike_fraction + 1) / log10(20 + 1)  [max = 20]

score = (revert_norm x 0.40
       + editor_norm x 0.30
       + spike_norm  x 0.30) x 100
```

Score Range	Label	Interpretation
0 - 24	Low	Mostly peaceful editing, rare reversions
25 - 49	Medium	Some dispute activity, moderate revert rate
50 - 74	High	Active edit warring, many reversions
75 - 100	Extreme	Heavy and persistent edit war detected

5. Visualizations (Plotly, dark theme)

Chart	Type	What It Shows
Weekly Edit Activity	Line chart	Edit volume per week over full history
Reverts vs Normal Edits	Stacked bar	Weekly revert count vs normal edits
Bot vs Human Edits	Horizontal bar	Total edits attributed to humans/bots
Top 10 Most Active Editors	Horizontal bar	Leaderboard; bots highlighted orange
Conflict Language	Donut pie chart	NLP keyword categories in comments
Controversy Gauge	Gauge / indicator	0-100 speedometer with colour zones

6. Streamlit Dashboard (app.py)

* Sidebar: free-text article title input + 7 example article quick-launch buttons.
* Analysis triggered by 'Analyze Article' button or example click.
* Spinner shown during API fetch and data processing stages.
* Controversy gauge + per-component metric cards at the top.
* 8 stat cards: total edits, reverts, revert rate, unique editors, human edits, bot edits, spike days, NLP conflicts.
* All 5 charts in a 2-column responsive grid.
* Raw data table (last 500 revisions) with one-click CSV download button.
* Custom CSS: dark mode, gradient cards, coloured stat borders.
* Full error handling: invalid titles, API failures, empty revision lists.

7. Installation & Running

```
# 1. Navigate to project folder
cd d:\Project\wiki_edit_war_analyzer

# 2. Create virtual environment  (already done)
python -m venv venv

# 3. Activate it
venv\Scripts\activate          # Windows

# 4. Install dependencies  (already done)
pip install -r requirements.txt

# 5. Run the dashboard
streamlit run app.py
#  --> Opens at http://localhost:8501
```

requirements.txt (4 packages only)

```
requests>=2.31.0
pandas>=2.0.0
plotly>=5.18.0
streamlit>=1.32.0
```

8. Example Articles & Expected Scores

Wikipedia Article	Expected Label	Why
Flat Earth	Extreme	Persistent fringe-theory reversions
Vaccine hesitancy	High	Medical misinformation disputes
Climate change	High	Scientific consensus vs denialism
Israel-Hamas war	Extreme	Active geopolitical conflict
Donald Trump	High	Heavy partisan editing
Homeopathy	Medium/High	Alternative medicine controversy
Genetically modified organism	Medium	Scientific vs. public opinion clash
Python (programming language)	Low	Technical article, few disputes

Live Smoke-Test Result (Verified)

Article: 'Flat Earth' | Revisions fetched: 5,000 | Score: 71 / 100 | Label: Extreme | Exit code: 0

9. Limitations & Known Issues

* Keyword-based revert detection may produce false positives or miss creatively-worded reverts.
* Bot detection relies on the 'bot' username convention -- custom-named bots may be misclassified.
* Large articles are capped at 5,000 revisions (configurable via MAX_REVISIONS in config.py).
* English Wikipedia only -- API URL targets en.wikipedia.org.
* No ML/AI -- the controversy score is a deterministic heuristic formula.
* Rate-limiting is courtesy-based (0.5 s delay); no auto-retry on HTTP 429.

* Suppressed / deleted revisions appear as 'Unknown' user with empty comments.

10. Future Improvement Ideas

* Implement WP:3RR detection -- flag any editor reverting the same content 3+ times in 24 h (the official Wikipedia edit-war policy).
* Add local caching (SQLite or Parquet) to skip re-fetching unchanged articles.
* Multi-article comparison mode -- plot controversy scores side-by-side.
* Train a text classifier on labelled edit comments for precision revert detection.
* Multi-language support -- allow picking any Wikipedia language edition.
* Wikipedia protection-log integration -- semi/fully protected articles are a strong edit-war signal.
* Time-range filter in the dashboard -- focus on a specific year or event window.
* In-app PDF report export directly from the Streamlit dashboard.

11. Tech Stack Summary

Layer	Tool / Library	Purpose
HTTP Client	requests	MediaWiki API calls with timeout & session
Data Wrangling	pandas	DataFrame manipulation, resampling, groupby
Visualisation	plotly	Interactive Plotly charts (dark theme)
Dashboard	streamlit	Web UI, sidebar, stat cards, CSV download
Language	Python 3.10+	All logic, scripting, and NLP
API	MediaWiki REST API	Wikipedia revision history (free, public)