Discourse Analysis Research Pipeline

Technical Overview and Methodological Notes

Uses app.py and index.html (requires an OS appropriate tool to launch and integrate)

Purpose	Track discursive normalisation of terms across social media platforms over time
Platforms	YouTube, Reddit, Bluesky
Technology	Python · Flask · SQLite · VADER NLP · Matplotlib
Outputs	SQLite DB, CSVs, 7 chart types, HTML toxicity report
Architecture	REST API server + background worker threads

1. Purpose and Research Goal

The pipeline is a self-contained web application designed for discourse research. Its central purpose is to collect posts and comments from social media platforms — YouTube, Reddit, and Bluesky — and then analyse how particular terms and phrases are used, framed, and how toxic the discourse surrounding them is over time.

The guiding concept is discursive normalisation: the hypothesis that certain terms, initially treated with scepticism or used in explicitly critical contexts, gradually appear in more affirmative or matter-of-fact contexts over time. The pipeline operationalises this by measuring framing ratios, toxicity scores, and pattern frequencies year by year.

2. System Architecture

The application is a Flask web server. When the script is launched it automatically finds a free TCP port (starting from 5000, scanning up to 50 ports) and exposes a REST API. A frontend — served separately — talks to this API via HTTP.

Long-running tasks (data collection and NLP analysis) are executed in background daemon threads so that the API remains responsive. Job state is held in two global dictionaries:

- job_status — tracks whether each job is idle, running, completed, or failed
- job_logs — stores timestamped log lines, returning the last 30 entries per status poll

SQLite is used as the storage layer. The WAL (Write-Ahead Logging) journal mode is enabled for safer concurrent writes.

2.1 REST Endpoints

Endpoint	Purpose
POST /api/create-session	Creates a timestamped output folder and writes session metadata JSON
POST /api/validate-config	Light validation of config — returns platform, term count, year span
POST /api/start-collection	Launches collection background thread
GET /api/collection-status	Returns current status + last 30 log lines
POST /api/start-analysis	Launches analysis background thread
GET /api/analysis-status	Returns current status + last 30 log lines
GET /api/results	Lists CSV and image files in a given output folder
GET /api/preview-image/<file>	Serves a PNG or HTML file for in-browser preview
GET /api/download/<file>	Downloads a file as an attachment
GET /api/health	Liveness check — returns port number

Note: The validate-config endpoint currently always returns valid: true — it is a placeholder for future stricter input checking.

3. Data Collection

Each platform has a dedicated collector class. Collection is driven by a year loop: for each year in the configured range, each search term is queried and the resulting posts are stored in a per-platform-per-year SQLite table (e.g. youtube_posts_2022). Comments are then fetched for those posts.

3.1 YouTube

Uses the official YouTube Data API v3 via the google-api-python-client library. The search endpoint is queried with the term wrapped in quotes (exact phrase match). Results are then enriched by fetching full video statistics (view count) and snippet data. A secondary content filter checks that the search term actually appears in the combined title and description text — this guards against the API occasionally returning loosely related results.

- Score field: view count (integer)
- Comment collection: top-level comments only, fetched via the commentThreads endpoint
- Pagination: handled via nextPageToken until the requested maximum is reached

3.2 Reddit

Uses the PRAW library (Python Reddit API Wrapper) in read-only mode (no user credentials required, only a client ID and secret). Subreddits are specified as a plus-separated string (e.g. ukpolitics+politics) and searched individually. A timestamp filter is applied after retrieval because PRAW's search does not reliably support date ranges natively — posts outside the year window are silently dropped.

- Score field: Reddit upvote score
- Comment collection: flattened comment trees, MoreComments objects are skipped (replace_more(limit=0))

Note: PRAW search returns results ordered by relevance, not chronology. Date filtering is done client-side, which may result in fewer results than the configured maximum for sparse time periods.

3.3 Bluesky

Uses the atproto Python SDK to authenticate with the AT Protocol API. Bluesky's search endpoint does not currently expose reliable date-range parameters, so all results are returned and stored without year-level filtering at collection time. The date is parsed and filtered during analysis instead.

- Score field: like count
- Comment collection: direct replies (depth=1 thread fetch)

Note: Bluesky's search API is newer and less mature than YouTube or Reddit. Result sets may be smaller and ordering is less predictable.

4. NLP Analysis Methods

Analysis is performed in the run_analysis function after data collection completes. All posts and comments are loaded from SQLite and enriched with four types of annotation.

4.1 Sentiment Analysis — VADER

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon-based sentiment analyser from NLTK. It is specifically designed for short social media text. For each text, it returns four scores:

Score	Range	Meaning
neg	0 – 1	Proportion of negative valence
neu	0 – 1	Proportion of neutral valence
pos	0 – 1	Proportion of positive valence
compound	−1 to +1	Normalised overall score

The compound score is used to classify sentiment into three categories using a ±0.05 threshold: positive (≥ +0.05), negative (≤ −0.05), and neutral (between the two thresholds).

The toxicity score is simply the neg (negative polarity) score. This is an important assumption: toxicity is equated with negative sentiment, which works well for hostile or aggressive language but will not capture, for example, sarcastically framed positive language.

Note: VADER was trained primarily on English-language tweets and product reviews. Performance on other dialects or highly informal language may vary. It does not account for context window or sarcasm beyond simple punctuation heuristics.

4.2 Toxicity Banding

The raw toxicity score (0–1) is mapped to five ordinal bands:

Band	Score Range	Interpretation
Minimal	< 0.10	Largely neutral or positive content
Low	0.10 – 0.29	Some negative sentiment, not hostile
Moderate	0.30 – 0.49	Noticeably negative or critical tone
High	0.50 – 0.69	Strongly negative; likely hostile
Severe	≥ 0.70	Extreme negative sentiment

4.3 Discursive Pattern Detection

Eight named pattern categories are detected using regular expressions applied to lowercased text. The categories and representative patterns are:

Category	Example Matches
replacement_narrative	great replacement, replacement theory, demographic replacement
authenticity_claim	real british people, genuine british, legacy british
two_tier	two-tier policing, two standards of justice
dehumanising	cockroach, vermin, swarm, infestation

conspiratorial	deep state, globalist, NWO, cabal
nativist	send them back, take back control, reclaim
dogwhistle	cultural marxism, woke agenda, 1488, 14 words
procedural_delegit	two-tier, one rule for, unequal treatment

Only comments that match at least one pattern are retained for the main analysis charts and toxicity report. Comments with no pattern matches are excluded.

Note: Pattern detection is keyword-based and case-insensitive. It does not disambiguate usage (e.g. critical citation vs. sincere use of a phrase). The framing analysis (below) is designed to partially address this limitation.

4.4 Framing Detection

A two-class framing classifier is applied to each item of text using a second set of regular expressions:

- Critical framing: markers such as so-called, alleged, what they call, quote unquote. These suggest the author is distancing themselves from a term — using it whilst signalling scepticism.
- Affirmative framing: markers such as absolutely, exactly right, this is real, wake up, 100%, spot on. These suggest the author endorses the content.
- Neutral framing: the default if neither pattern is detected.

The key metric derived from this is the framing ratio over time (Chart 1): a rising proportion of affirmative framing, relative to critical or neutral, is treated as a signal of discursive normalisation — the hypothesis that a term or idea is becoming more casually accepted.

5. Outputs

5.1 Structured Data

- {platform}_posts_analysis.csv — all posts with NLP annotations appended
- {platform}_comments_analysis.csv — comments matching at least one discursive pattern, with all annotations
- {platform}_term_frequency_by_year.csv — annual term occurrence counts in a wide-format table

5.2 Visualisations (PNG)

#	Chart Name	What It Shows
1	Framing Over Time	Monthly proportions of critical / neutral / affirmative framing as a line chart. Rising affirmative line is the normalisation signal.
2	Term Frequency by Year	Grouped bar chart of raw term occurrence counts per year, one group per search term.
3	Toxicity Banding Over Time	Monthly stacked proportional bar chart showing the share of each toxicity band.
4	Mean Toxicity per Term	Annual mean toxicity line chart with separate series per search term. Reference lines at 0.3 (moderate) and 0.5 (high).
5	Pattern Heatmap	Grid of raw pattern hit counts per year (years on x-axis, patterns on y-axis), colour-scaled from white to dark red.

6	Sentiment per Term	Side-by-side bar charts (one per term) showing the count of positive, neutral, and negative classified comments.
7	Word Cloud	Top 150 words across all comments, sized by frequency. Uses a red–yellow–green colour map for visual contrast.

5.3 HTML Toxicity Report

An interactive self-contained HTML report is generated for each platform run. It contains:

- Four KPI cards: total comments analysed, mean toxicity score, percentage severe, percentage high
- A visual toxicity bar scaled 0–1 with band labels
- A trend indicator (increasing / stable / decreasing) based on the difference between first and last year mean scores, using a ±0.05 threshold
- Peak and lowest month by mean toxicity
- Year-on-year table with absolute change between years
- Per-term toxicity summary table with inline bar charts
- Top 5 most toxic comments (selected by highest toxicity score)

6. Key Assumptions and Limitations

These assumptions are important for interpreting results correctly, particularly for a statistical or research audience.

6.1 Toxicity as Negative Sentiment

The toxicity score is the VADER negative polarity value, not a purpose-built toxicity classifier. This means:

- Content that is angry, sad, or pessimistic will score as toxic even if it is not hateful
- Sarcastic positivity (e.g. enthusiastically endorsing a harmful idea) will not be captured
- A dedicated model (e.g. Perspective API) would give more precise toxicity estimates but requires additional API credentials and quota

6.2 Framing Classifier Precision

The framing classifier is a keyword list, not a trained model. It will produce false positives (e.g. the word absolutely in an unrelated sentence will count as affirmative) and false negatives (novel or indirect framing language will be missed). The ratio should be read as a relative indicator across time rather than an absolute measure.

6.3 Pattern Filter and Selection Bias

Only comments matching at least one of the eight discursive patterns are included in charts and the toxicity report. This means the dataset is already filtered to discourse using these specific terms. Raw counts and sentiment distributions are therefore not representative of the full comment volume for a given search term — they reflect only the subset of comments engaged with these discursive categories.

6.4 Platform-Specific Collection Limits

Each platform imposes its own constraints:

- YouTube: the Data API has daily quota limits; large collection runs may exhaust quota
- Reddit: PRAW search returns at most ~1,000 results per subreddit; date filtering is post-hoc and client-side
- Bluesky: no server-side date filtering; all available results are retrieved and filtered during analysis

6.5 Thread-Safety and Concurrency

The job_status and job_logs dictionaries are global Python objects accessed from both the Flask request threads and the background worker threads without explicit locking. In practice this is unlikely to cause data corruption given the simple read/write patterns, but it is a known architectural imprecision.

6.6 Compound Score Threshold

Sentiment classification uses a ±0.05 compound threshold. This is the default threshold recommended by the VADER authors. Lowering it (e.g. to ±0.02) would classify more content as positive or negative rather than neutral; raising it (e.g. to ±0.1) would increase the neutral bucket. Results are sensitive to this choice for content near the boundary.

7. Running the Pipeline

The application is launched directly:

```
python app.py
```

On startup it prints the port it has bound to. The frontend is then pointed at that address. The typical workflow is:

1. POST /api/create-session — create a named output folder
2. POST /api/start-collection — begin data collection
3. Poll GET /api/collection-status until completed
4. POST /api/start-analysis — run NLP pipeline
5. Poll GET /api/analysis-status until completed
6. GET /api/results — retrieve file list for download or preview

Required dependencies: googleapiclient, praw, requests, pandas, numpy, matplotlib, nltk, wordcloud, regex, atproto, flask, flask-cors.