Analyzing Public Discussions for Product Insights

Mining Reddit and Tiki for product issues, sentiment, and trends

Team: Add names and roles here

Date: datetime(year: 2025, month: 8, day: 26)

Contents

1. Title & Team

- Project: Mining Reddit (and Tiki) to extract product issues, pros/cons, and topic trends
- Deliverable: CLI + datasets + analyses + slides
- Team: Add names and roles here

2. Objectives

- Identify common issues across consumer assets (laptops, phones, home gear, etc.)
- Produce sentiment scores per comment and topics per subreddit
- Deliver a CLI for repeatable analysis and export

3. Why Reddit? (+ Tiki for diversity)

- Reddit: rich, threaded discussions; public API; text-first; strong English NLP support
- Excluded:
  - Facebook ⟶ heavy bots, limited Vietnamese NLP support
  - TikTok ⟶ media-first, no practical API, scraping slow/inefficient
- Tiki (e-commerce reviews): complements Reddit with structured, purchase-verified user feedback and Vietnam market signal

4. Data Collection — Initial vs. Now

- Initial (PRAW)
  - Pros: simple API, good for prototyping
  - Cons: 1,000 post cap per subreddit (hot/new/top/rising), limited time filtering, rate limits
- Now (hybrid)
  - Reddit historical via downloaded archives (e.g., Academic Torrents) to bypass caps
  - Tiki review dumps (where available) for cross-source validation
  - Result: broader time windows, more volume, better representativeness

5. How PRAW Traverses Comments (and why it's not "just save to JSON")

- CommentForest structure: each submission has `submission.comments` (a tree)
- MoreComments placeholders: large threads include `MoreComments` nodes; they require extra API calls
- Exhausting the tree: call `submission.comments.replace_more(limit: none)` to recursively fetch remaining branches
- Traversal: iterate via `for c in submission.comments.list():` or DFS/BFS over `comment.replies`
- Not automatic export: PRAW returns Python objects lazily; you must walk the tree, handle rate limits, deleted/removed entries, pagination, and serialize yourself (e.g., to JSONL/Parquet)

6. Alternatives Considered (to bypass API limits)

- Pushshift (mod-gated now): powerful time filters; access constraints ⟶ not feasible
- Academic Torrents: downloadable Reddit snapshots for specific periods ⟶ used for scale and history
- Personal archive: long-running collector; impractical (hardware/time), misses older content

7. Subreddits Chosen (assets and ecosystems)

- r/macbookpro, r/GamingLaptops, r/HomeDecorating, r/photography, r/iphone, r/mac, r/AppleWatch, r/Monitors, r/SmartThings, r/PcBuild, r/laptops, r/hometheater, r/headphones, r/ErgoMechKeyboards, r/homelab, r/BudgetAudiophile

8. EDA and Visualization

TODO Planned figures: volume over time (posts/comments) per subreddit; comment length and user activity distributions; top products/models; sentiment by subreddit/product (variance and outliers); co-occurrence networks (issue terms ↔ products).

9. Preprocessing Pipeline

- Format normalization
  - Built a Polars + Nushell wrapper to ingest JSONL ⟶ Parquet (`posts.parquet`, `comments.parquet`)
  - JSONL has variable schemas ⟶ enforced typed schema in Parquet
  - Identified one malformed file ⟶ removed problematic column; recreated empty field to align schema
- Cleaning
  - Remove URLs; strip markup; collapse whitespace
  - Language filter: drop non-English for core run (keeps model assumptions cleaner)
- Context experiment (hierarchy)
  - Hypothesis: including parent context improves SA
  - Trial: concatenated parent chains (up to 512 tokens)
  - Result: worse performance (context pollution + truncation) ⟶ chose per-comment modeling
- Sampling
  - Initially planned top-100 per subreddit ⟶ biased to virality
  - Switched to random 100 posts per subreddit for representativeness (set random seed)

10. Sentiment Analysis (SA)

- Baselines (considered)
  - VADER (rule-based): valence lexicon + heuristics
    - Handles intensifiers, negations, punctuation/caps; outputs compound $\in [-1, 1]$ plus pos/neu/neg
    - Fast, interpretable; English-centric; brittle on domain jargon/emojis outside lexicon
  - TextBlob: simple polarity/subjectivity; similar limitations
- Pretrained transformers (chosen)
  - Tried twitter-roberta ⟶ too heavy for local CPU
  - Selected `lxyuan/distilbert-base-multilingual-cased-sentiments-student`
    - Pros: multilingual coverage, lighter footprint, good zero-shot behavior
    - Cons: still needs GPU for throughput ⟶ rented GPU on vast.ai
- Planned fine-tuning
  - Goal: domain-adapt to product/support discourse
  - Approach: label a stratified sample (by subreddit/product/sentiment)
  - Tooling: use gemini-cli to assist labeling/QA; track IAA (inter-annotator agreement)
  - Metrics: accuracy/F1 on held-out; calibration of thresholds for {neg, neu, pos}
- Output
  - Per-comment: `sent_score` $\in [-1, 1]$ + label; attach `subreddit`, `product`, `timestamp`

11. Topic Modeling

TODO Evaluate BERTopic (embeddings + clustering + topic words) vs. LDA/NMF (TF-IDF). Preprocess with domain stopwords (brand names?), lemmatization, min/max df. Assess coherence (c_v) and topic stability across subsamples. Deliver top topics per subreddit, representative comments, and trend lines.

12. CLI — Demo Plan

TODO Commands to implement and demo: `ingest` (JSONL → Parquet with schema checks), `clean` (URL/language filtering; dedup), `sentiment` (batch inference; GPU flag; progress/resume), `topics` (train/infer; export artifacts), `report` (aggregate by product/subreddit; export CSV/Parquet + charts).

13. Data Quality, Risks, and Mitigations

- Bias (subreddit culture): use multiple communities; random sampling
- Time drift: include time slices; compare older vs. newer cohorts
- Language: English-first results; future pass on Vietnamese with dedicated models
- Rate limits / duplicates: cached IDs; respectful backoff
- Schema drift: enforce Parquet schema; validate on ingest

14. Reproducibility and Ops

- Pinned env: Python/Polars/Nushell versions; `requirements.txt` / `poetry.lock`
- Config-driven runs: seeds, subreddit lists, date windows
- Deterministic sampling: stored RNG seeds and sampled ID lists
- Artifacts: Parquet datasets, SA predictions, topic artifacts, run logs

15. Ethics and Compliance

- Public data; follow platform terms; remove PII where feasible
- Respect community norms; avoid deanonymization; aggregate reporting

16. Next Steps

TODO Finalize EDA visuals and narrative; run SA at scale and sanity-check class balance; pilot topic modeling, tune parameters, and choose final method; wire up CLI demo path and sample outputs; optionally begin small-scale fine-tuning with labeled set.

17. Appendix — Tools and Environment

- Data: Reddit archives (Academic Torrents), Tiki reviews (exports)
- Stack: Python, Polars, PyTorch/Transformers, BERTopic/Scikit-learn, Nushell
- Compute: local CPU for ETL; rented GPU (vast.ai) for transformer inference

17.0.1. Slide Notes (optional)

- Keep bullets short; speak to details (e.g., how PRAW's `replace_more` works, why context hurt SA).
- Emphasize decisions and evidence: random sampling vs. top posts; per-comment SA vs. hierarchical.
- Show at least one chart per claim (volume/time, sentiment by product, example topics).