Analyzing Public Discussions for Product Insights

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

From noisy threads ⟶ actionable product signals.

Problem & Value

- Which products break, why, and how often—using public discussions.
- Turn unstructured Reddit and Tiki text into actionable product signals.
- Output insights teams can ship on: issues, trends, severity, examples.

Objectives

- Which issues? How frequent? How trending? What severity?
- Where do issues cluster (by product/subreddit/time)?
- How reliable are methods vs baselines?
- Deliver a repeatable pipeline and clear, prioritized findings.

Executive Summary

- Battery swelling dominates r/macbookpro in Q2–Q3 (placeholder data).
- Smart-home connectivity drops spike post-firmware X (placeholder data).
- Headphones: left-channel crackle clusters on model Y (placeholder data).
- [Placeholder image: multi-subreddit bar chart of top 5 issues by frequency]

Why These Platforms?

- Reddit: rich, threaded discussions; public API; strong NLP support.
- Tiki (e-commerce): structured, purchase-verified reviews; VN market signal.
- Complementary: text-rich threads vs short reviews ⟶ broader coverage.
- [Placeholder image: two-column pros/cons cards + coverage map]

Data Collection

- **Initial:** PRAW (Python Reddit API Wrapper) for prototyping.
  - **Limitation:** 1,000 post cap, rate limits.
- **Current (Hybrid):**
  - Reddit historical archives (Academic Torrents) to bypass API caps.
  - Tiki review dumps for cross-source validation.
  - **Result:** Broader time windows, more volume.
- [Placeholder image: timeline ribbon PRAW → Archives → Tiki]
- [Placeholder image: flowchart with rate-limit icon on PRAW; time-filter lock]

How PRAW Traverses Comments

- Reddit returns comment trees with `MoreComments` placeholders.
- We expand lazily, traverse, and serialize—no one-click “download”.
- Captures full depth where needed; avoids rate-limit explosions.
- [Placeholder image: comment tree diagram with expansion arrows]

Alternatives Considered

- Pushshift, Academic Torrents, Personal Archive compared on access, filtering, scale.
- Chosen: history + scale via archives; complement with live API when needed.
- [Placeholder image: scorecard table with decision badge]

Subreddits Chosen

- A diverse mix of tech and lifestyle communities:
- r/macbookpro, r/GamingLaptops
- r/iphone, r/AppleWatch, r/Monitors
- r/headphones, r/homelab, r/photography
- …and several others covering home, audio, and PC building.
- Example coverage: N posts, M comments, YYYY–YYYY. (placeholder)
- [Placeholder image: grid of subreddit badges sized by sample count]

EDA Highlights

- [Placeholder image: time series of comments/week by subreddit]
- [Placeholder image: length distribution (boxplot or histogram)]
- [Placeholder image: top product mentions (bar) via keyword/NER]
- Takeaways (placeholder): spikes follow launch X; r/homelab comments 2× longer.

Preprocessing Pipeline

- **Ingestion:** Ingested JSONL into typed Parquet schemas using Polars & Nushell.
- **Cleaning:**
  - Removed URLs, stripped markup, normalized whitespace.
  - Filtered for English-only content for initial analysis.
- **Modeling Choice:** Modeled each comment individually after experiments showed parent context polluted sentiment signals.
- Randomized 100 posts/sub (seeded) to reduce virality bias. (placeholder)
- [Placeholder image: before/after text snippet (URLs/emojis removed)]
- [Placeholder image: language donut (kept vs dropped)]
- [Placeholder image: schema alignment diagram JSONL ⟶ typed Parquet]

Context Experiment (Per-Comment SA)

- Parent-context vs per-comment classification compared on small labeled set.
- Result: per-comment avoids 512-token truncation and context pollution.
- [Placeholder image: ablation bar chart (F1 with vs without parent context)]

Sentiment Analysis (SA)

- **Approach:** Pretrained transformers over simpler baselines (VADER, TextBlob).
- **Model:** `lxyuan/distilbert-base-multilingual-cased-sentiments-student`
  - **Why:** Good multilingual support, lightweight, strong zero-shot performance.
- **Execution:** Rented GPU on `vast.ai` for large-scale inference.
- **Next Step:** Fine-tune on a domain-specific labeled dataset.
- [Placeholder image: model compare bar (VADER/TextBlob/DistilBERT) on labeled set]
- [Placeholder image: throughput/cost bar CPU vs GPU on vast.ai]
- [Placeholder image: confusion matrix or calibration curve]
- [Placeholder panel: 2–3 qualitative examples with predicted labels]

Topic Modeling (Planned)

- **Goal:** Discover key themes and issues per subreddit.
- **Method:** Evaluate BERTopic vs. traditional methods (LDA/NMF).
- **Process:** Preprocess with domain stopwords, lemmatization.
- **Output:** Top topics, representative comments, and trend lines.
- Target: coherence c_v threshold and stable topics across subsamples. (placeholder)
- [Placeholder image: intertopic distance map (BERTopic mock)]
- [Placeholder image: top-words table for 2 sample topics]

CLI Tool (Planned)

- A pipeline for repeatable analysis:
- `ingest`: Raw data to Parquet.
- `clean`: Filter and normalize data.
- `sentiment`: Run batch sentiment analysis.
- `topics`: Train and apply topic models.
- `report`: Aggregate results and export.

Risks & Mitigations

- **Bias:** Sampled randomly across multiple, diverse subreddits.
- **Time Drift:** Included time slices to compare cohorts.
- **Reproducibility:** Pinned environments, config-driven runs, and stored seeds.
- **Ethics:** Used public data, followed platform ToS, aggregated results.

Next Steps

- Finalize EDA visualizations.
- Run sentiment analysis at scale.
- Pilot and select a topic modeling approach.
- Build and demo the core CLI workflow.
- Begin labeling for fine-tuning sentiment model.

Q & A