

Housing Crisis Analysis System: Social Media Analytics on the Cloud

COMP90024 - Cluster and Cloud Computing - Assignment 2
Team 28

Ian Chi - 1579898
Yang Jin - 1498220
Yuhong Wu - 1507649
Ling Du - 1675590

May 20, 2025

Contents

1	Introduction	2
1.1	Project Background and Motivation	2
1.2	Project Objectives	2
1.3	Data Sources Overview	3
2	System Architecture	4
2.1	Architecture Overview	4
2.2	Module Descriptions	5
2.2.1	Data Collection	5
2.2.2	Scheduling & Ingestion	6
2.2.3	Elasticsearch Index Design	6
2.2.4	NLP Processing Pipeline	7
2.2.5	API Design	8
2.2.6	Visualization	9
3	Implementation Details	9
3.1	Technology Stack Overview	9
3.2	Key Module Implementations	9
3.2.1	Data Collection Method	9
3.2.2	Fission Deployment Process	10
3.2.3	NLP Module Architecture and Models	10
3.2.4	API Integration	11
3.2.5	Visualization Implementation	11
3.2.6	Fault Tolerance & Scalability	12
4	Results Presentation	12
4.1	Function Interpretation	13
4.2	Analysis & Discussion	16
5	Challenges and Discussion	16
5.1	System Performance & Challenges	16
5.2	NeCTAR Experience	17
5.3	Kubernetes Configuration & Lessons	17
5.4	Data Collection Issues	17
5.5	Fission Deployment Issues	17
5.6	Elasticsearch Limitations	17
5.7	NLP Limitations	18
5.8	Frontend & Visualization	18
6	Conclusion	18
7	Team Roles & Responsibilities	19
8	Appendix	20
8.1	Demo Video Link	20
8.2	GitLab Repository	20

1 Introduction

1.1 Project Background and Motivation

In recent years, Australia has experienced a growing housing and cost-of-living crisis, significantly impacting citizens across different income groups and regions. The sharp increase in rental prices, property market volatility, and public dissatisfaction with housing policies have sparked widespread discussions on digital platforms.

Social media has emerged as a powerful tool to capture real-time public sentiment and track emerging social trends. Platforms like Reddit and Mastodon host thousands of posts and discussions reflecting citizens' experiences, frustrations, and expectations around the housing system. Analyzing these discussions provides unique insights into public perception that traditional surveys often overlook.

By focusing on Australia, a country with diverse cities facing varying degrees of housing stress, we aim to understand how different regions and demographics are affected and represented in online discourse.

Analysis Theme Overview

The main theme of this project is the Australian housing and living cost crisis. Our analysis focuses on the following sub-themes:

- Housing affordability by city and region
- Public sentiment in response to housing policies and rental changes
- Topic distribution across income and age groups
- Sentiment trends around key events (e.g., interest rate hikes, policy announcements)
- Frequency and co-occurrence of key terms, including: housing crisis, affordable housing, rent increase, housing shortage, eviction, homelessness, rent control, mortgage rates, public housing, first-time homebuyer, real estate, landlord, and tenant.

By structuring our analysis around these themes and anchoring it in a curated set of housing-related keywords, we aim to explore both the broader policy-related discussions and the everyday experiences shared by Australians through social media platforms.

1.2 Project Objectives

This project is guided by four key analytical questions addressing the Australian housing and cost-of-living crisis:

1. **How has public sentiment surrounding the Australian housing and cost-of-living crisis evolved over the past 12 months?** The analysis examines monthly sentiment trends to identify significant shifts or emerging patterns in public attitudes over the last year.
2. **How does public sentiment regarding the housing and cost-of-living crisis vary across different Australian states?** The analysis compares sentiment across states to highlight geographical disparities and local nuances in how the crisis is perceived.

3. **What are the extreme instances of public sentiment (both highly positive and highly negative) observed in discussions of the housing and cost-of-living crisis?** The analysis identifies peaks and troughs in sentiment to pinpoint when public emotions have intensified or abated. These extremes are further examined for underlying events or factors that might have contributed to such surges or dips.
4. **How do weekly patterns in public discourse reflect changes in attention to the housing and cost-of-living crisis?** The analysis investigates week-by-week fluctuations in discussion volume. This approach reveals short-term trends and illustrates how public focus on the crisis ebbs and flows over time.

1.3 Data Sources Overview

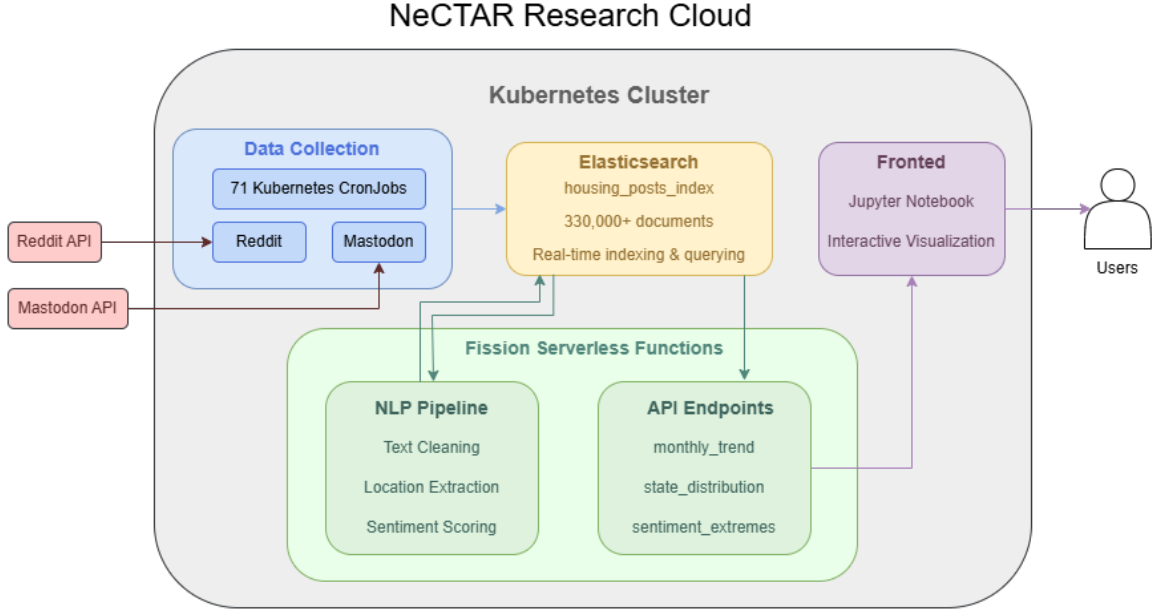
Reddit: Our analysis focused specifically on the r/Australia subreddit, a centralized platform where Australians discuss various topics, including housing and cost-of-living issues. Reddit offers a highly active environment with long-form, debate-style posts, detailed personal experiences, and transparent community feedback via upvotes, downvotes, and threaded replies.

Mastodon: A decentralized social network where users interact within independent instances, often built around specific regions or interests. Mastodon tends to host smaller, more focused communities, which makes it a valuable source for low-noise, context-rich discussions, particularly within Australian-themed or regional servers.

These two platforms differ not only in structure—federated vs. centralized—but also in tone, user demographics, and engagement dynamics. When analyzed together, they offer a more nuanced and layered understanding of how the housing crisis is perceived and discussed across Australia.

2 System Architecture

2.1 Architecture Overview



Our system follows a cloud-native microservices architecture deployed on the Melbourne Research Cloud (NeCTAR) Kubernetes platform, as illustrated in the diagram:

The architecture consists of five distinct layers, each responsible for a specific aspect of the data processing pipeline:

1. **Data Collection Layer:** Implemented as serverless Fission functions triggered by 71 Kubernetes CronJobs. Our collectors for Reddit (r/Australia) and Mastodon harvest data across different housing-related topics on a staggered schedule to distribute the workload efficiently.
2. **Storage Layer:** Elasticsearch serves as our central data repository, with a custom schema optimized for housing-related social media content. It stores both raw posts and processed results in a structured format, currently containing approximately 330,000+ documents for efficient querying and aggregation.
3. **Processing Layer:** A dedicated NLP processing function analyzes unprocessed documents from Elasticsearch, enriching them with sentiment scores and geographic information. This function writes the processed results back to Elasticsearch on an hourly schedule, ensuring recently collected data is promptly analyzed and available for querying.
4. **API Layer:** Four specialized API endpoints (/month, /state, /extreme, /weekly) transform the processed data from Elasticsearch into analytical insights. Each endpoint is implemented as a serverless function that returns JSON data tailored for specific visualization needs.

5. **Presentation Layer:** A Jupyter Notebook running in Kubernetes serves as our front-end, consuming the API endpoints to create interactive visualizations that reveal patterns in housing crisis discussions across temporal and geographic dimensions.

This microservices approach ensures strong component isolation and fault tolerance: if one component fails, the others continue operating independently. The serverless framework also provides efficient resource utilization, allocating computing power only when functions are actively processing data.

2.2 Module Descriptions

2.2.1 Data Collection

Our data collection system harvests content from both Reddit and Mastodon through specialized Fission functions deployed on the Kubernetes cluster. Each collector exposes an HTTP endpoint that, when invoked, retrieves posts for a specific housing-related topic:

Reddit Collection (PRAW)

- **Authentication:** Uses "script"-type OAuth with client credentials and a custom User-Agent for Reddit API access
- **Batch Mode:** Implements `reddit.subreddit("Australia").search(query=topic, sort="new", limit=None)` which pages automatically up to the 1000-item API cap
- **Comments:** For each submission, calls `submission.comments.replace_more(limit=0)` then flattens with `submission.comments.list()`
- **Normalization:** Maps posts and comments to a unified JSON schema (id, resource, text, timestamp, like_count, repost_count)

Mastodon Collection (mastodon.py)

- **Authentication:** Utilizes OAuth client credentials with a long-lived ACCESS_TOKEN
- **Paginated Fetch:** Loops through `mastodon.timeline_hashtag(tag, limit=40, max_id=last_id-1, local=True)` until exhaustion
- **Keyword Filter:** Applies in-memory filtering on `status.content.lower()` (e.g., "australia")
- **Normalization:** Strips HTML and maps to the same JSON schema used for Reddit content

All collectors batch up to 500 records per HTTP POST to the ingestion service, with detailed logging of pages fetched, items retained, and processing latency.

2.2.2 Scheduling & Ingestion

We employ Kubernetes CronJobs to trigger each topic’s collector evenly distributed over the hour:

- **Distributed Scheduling:** Each of the 71 topics is assigned a specific minute within the hour, ensuring even distribution of workload and preventing API rate-limit issues
- **Idempotency:** Each collector tracks the last fetched `max_id` per topic and only retrieves newer items to prevent duplication
- **Concurrency Control:** By spacing jobs at different minutes, we avoid API rate-limit bursts
- **Retry Policy:** CronJobs use `restartPolicy: OnFailure` and will retry up to 3 times on non-200 responses

The ingestion endpoint validates incoming JSON arrays, performs any necessary transformations, and executes an Elasticsearch Bulk API call (`refresh=false, replication=async`) to index documents efficiently. A summary of successes and failures is returned to the caller.

2.2.3 Elasticsearch Index Design

To support efficient querying, filtering, and aggregation of social media data related to the Australian housing crisis, we implemented a custom Elasticsearch index schema named "housing-post". The index is structured to accommodate both raw post attributes and the results of downstream NLP processing.

Key Field Design:

- **Exact-match fields** (e.g., `id`, `resource`, `city`, `state`) use the `keyword` type for fast filtering and aggregations without tokenization
- **Full-text field** `text` uses the `text` type to support flexible match queries, enabling sentiment analysis and keyword extraction
- **Numeric fields** (`like_count`, `favourite_count`, `repost_count`, `sentiment_score`) are stored as integer or float to enable statistical aggregation and time-series trend analysis
- **Temporal field** `timestamp` (type: `date`) enables efficient time-based filtering and aggregation, such as grouping by day or month
- **Nested field** `named_entities` allows structured access and aggregation on named entity pairs (entity, type) extracted by NLP tools
- **Boolean flag** `processed` (values 0 or 1) indicates whether the post has been successfully processed by the NLP pipeline:
 - `processed = 0`: Post has not yet been analyzed

- **processed = 1:** Post has completed NLP enrichment and is ready for downstream querying and visualization

This design ensures that only fully enriched posts are included in analytical queries, improving consistency and accuracy of the results.

Implementation Details:

- Query logic is encapsulated in specialized API endpoints supporting multiple analytics tasks:
 - City-level post count aggregation
 - Monthly sentiment trends
 - State-level sentiment distribution
 - Weekly sentiment patterns

This modular design provides a clean interface for downstream components to consume preprocessed and structured data efficiently.

2.2.4 NLP Processing Pipeline

For both our NLP Processing Pipeline and API functions, we built a customized Docker image that was used to create a Fission environment. This approach ensured consistency between the processing and API layers, with all necessary dependencies pre-installed and optimized for our specific workload.

The NLP Processing Pipeline includes several key components designed to extract structured insights from unstructured social media posts and write them back to Elasticsearch:

Text Cleaning: Raw post content is preprocessed to remove emojis, control characters, and punctuation, and to normalize casing and spacing. This prepares the input for downstream tasks.

City and State Recognition: A custom location extraction module identifies and assigns geographic information (city and state name) from post text using a rule-based approach. It leverages regular expressions in combination with a structured dictionary derived from the Australian Statistical Geography Standard (ASGS), specifically the 2021 Statistical Areas Level 3 (SA3) allocation file published by the Australian Bureau of Statistics (ABS).

The module first attempts to match city names mentioned in the text to those listed in the SA3 dataset. Once a match is found, the corresponding state is inferred. If a state name or abbreviation appears explicitly in the post, it is used directly. In the absence of recognizable location-related content, both city and state are assigned as "Unknown".

Sentiment Analysis: Each post is analyzed using the TextBlob sentiment engine. The model returns a polarity score in the range $[-1, 1]$, which is linearly rescaled to a $[0, 10]$ sentiment score for better interpretability in downstream analytics.

Data I/O: The pipeline connects to the Elasticsearch instance to retrieve unprocessed records from the housing-post index and write the processed results back with a `processed: 1` flag to prevent reprocessing. The system processes data in batches to ensure efficient handling of large datasets.

This unified pipeline supports both batch and real-time processing and is deployed as a serverless function via Fission with an hourly schedule.

2.2.5 API Design

Our system exposes four analytical endpoints—`/month`, `/state`, `/extreme`, and `/weekly`—as independent, stateless APIs deployed using the same custom Docker image created for the NLP processing pipeline.

Architecture & Workflow: Each endpoint is accessible via GET request and returns structured JSON data retrieved from Elasticsearch. All functions are deployed independently and remain stateless, enabling horizontal scalability and fault tolerance. This architecture ensures reliable access from external tools such as dashboards, notebooks, or automated scripts, even under concurrent usage.

The deployment process is unified across all endpoints and includes:

- Preparing Python functions with consistent input/output formats
- Installing dependencies and packaging using `build.sh`
- Uploading and exposing functions via defined routes (`deploy.sh`, `creat_route.sh`)
- Binding accessible URLs through Fission routing

This repeatable workflow simplifies updates and supports rapid integration or scaling as needed.

Roles of Each Endpoint:

- `/month` — Returns aggregated sentiment data per month for the past year from Elasticsearch, enabling long-term trend monitoring
- `/state` — Aggregates sentiment scores and post counts by region from the processed data in Elasticsearch
- `/extreme` — Highlights cities with the highest and lowest sentiment averages based on Elasticsearch queries
- `/weekly` — Provides sentiment distribution and post counts for the past seven days from Elasticsearch

Response Format: All responses are returned in consistent JSON structures, organized by time or geographic unit, and include associated metrics such as average sentiment score, total post volume, or sentiment label breakdown. This design ensures compatibility with downstream analytics and visualization pipelines.

2.2.6 Visualization

For deploying our frontend Jupyter Notebook, we used Kubernetes YAML files applied directly with `kubectl apply -f` commands, rather than shell scripts. The Jupyter Notebook frontend drives the four REST endpoints to fetch pre-aggregated data from Elasticsearch and render interactive visualizations:

- **Monthly Trend:** A dual-axis chart showing post volume and average sentiment over the past 13 months
- **State Distribution:** A comparative view of post counts and sentiment scores across Australian states
- **Sentiment Extremes:** Bar charts highlighting cities with the most positive and negative sentiment
- **Weekly Statistics:** Recent patterns in post volume and sentiment distribution over the past 7 days

All visualizations support interactive features like hover information, zooming, and legend toggling. The Jupyter Notebook can be exported as a standalone HTML dashboard for sharing results.

3 Implementation Details

3.1 Technology Stack Overview

Our system leverages a modern cloud-native technology stack:

- **Container Orchestration:** Kubernetes running on Melbourne Research Cloud
- **Serverless Platform:** Fission for function-as-a-service capabilities
- **Database:** Elasticsearch for storing and querying social media data
- **Programming Languages:** Python for all components
- **NLP Libraries:** NLTK, TextBlob
- **API Clients:** PRAW (Reddit), Mastodon.py
- **Visualization:** Jupyter Notebook, Pandas, Matplotlib, Plotly

3.2 Key Module Implementations

3.2.1 Data Collection Method

We adopt a hybrid streaming-batch approach:

- **Streaming:** Real-time cron-driven pulls for recent posts, ensuring low latency
- **Batch:** Bulk re-sync of historical partitions for backfill operations

Collectors maintain a per-topic cursor (max.id for Mastodon, timestamp for Reddit) stored in a lightweight key-value store. This enables exactly-once semantics and incremental updates, preventing duplicate data collection.

The system distributes workload across 71 different topics related to housing, with each topic assigned to a specific minute within the hour. This approach prevents API rate limiting issues while ensuring comprehensive coverage of the subject matter.

3.2.2 Fission Deployment Process

- **Packaging:** Each Python collector is packaged with its dependencies using a `build.sh` script that installs requirements in the deployment package
- **Function Creation:** The `deploy.sh` script uses Fission CLI commands to register the handler, specifying environment variables for credentials and configuration
- **Routing:** The `creat_route.sh` script establishes HTTP routes that bind to the functions
- **Autoscaling:** Fission’s pool manager dynamically scales pods based on concurrent requests
- **Security:** Credentials are injected via Kubernetes Secrets mounted as environment variables

This standardized deployment approach ensures consistency across all components and simplifies the process of updating or extending the system.

3.2.3 NLP Module Architecture and Models

The NLP module adopts a modular design with well-structured functions for each sub-task:

Text Cleaning (`clean_text`): This function standardizes input text by removing emojis, special characters, control symbols, and punctuation. The text is converted to lower-case and normalized for consistent downstream processing.

Location Recognition (`extract_locations`): A rule-based approach using regular expressions identifies geographic references from the post. Cities are matched against a predefined dictionary based on the 2021 ABS SA3 allocation file and mapped to their respective states. If no city is found, the module attempts to match known state names or abbreviations.

Sentiment Scoring (`analyze_sentiment`): The sentiment of each post is assessed using TextBlob, a lightweight lexicon-based tool. The polarity score (ranging from -1 to $+1$) is scaled to a continuous $[0, 10]$ sentiment score to support flexible thresholding and visual analytics.

Elasticsearch Integration (`load_from_elasticsearch`, `write_results_to_es`): The module connects to the Elasticsearch instance, fetches unprocessed records, and updates each record with new annotations, including location, sentiment, and a processed status. This ensures idempotent, batch-based processing and enables incremental updates.

Orchestration Logic (`process_nlp_logic`, `main`): The main function orchestrates the entire NLP workflow, processing documents in batches and continuing until all unprocessed documents have been handled.

3.2.4 API Integration

This project deploys four core analytical endpoints using the Fission framework on a Kubernetes cluster. The integration process consists of three parts:

Function Packaging and Deployment: Each function is packaged using a `build.sh` script that installs dependencies and prepares the deployment directory. The `deploy.sh` script then uses the Fission CLI to upload and register the functions on the cluster.

Route Binding and Service Registration: After deployment, each function is exposed via a specific HTTP route using a corresponding `creat_route.sh` script. For example, the `weekly-counts` function is bound to `/weekly` using the command: `fission route create --url /weekly --function weeklycounts --name weekly-counts --createingress`

Integration with External Analysis Tools: Once active, the APIs can be integrated into external environments such as Jupyter Notebook. Analysts can use libraries like `requests` to call endpoints and receive structured JSON responses. The data can then be loaded into tools like `Pandas` for visualization and further analysis.

3.2.5 Visualization Implementation

In our Jupyter Notebook (`housing-crisis-analysis.ipynb`), we consume four REST API endpoints exposed by our Fission HTTP layer to drive all visualizations:

Monthly Trend (`/month`): We call `GET /month`, ingest the response—an array of records with `month`, `count`, and `avg_sentiment`—into a `DataFrame`, then plot a dual-axis Plotly chart showing monthly post volume (bar) and average sentiment (line).

State Trend (`/state`): We call `GET /state`, receiving per-state sentiment and count data. This is reshaped into a `DataFrame` and plotted as a comparative chart with clear visual distinction between states.

Sentiment Extremes (`/extreme`): We fetch the top-3 most positive and top-3 most negative city data via `GET /extreme`. We render two side-by-side charts to highlight the geographic contrast in sentiment.

Recent 7-Day Statistics (`/weekly`): We query `GET /weekly` for the last seven days' sentiment distribution data. This is visualized to show recent patterns and short-term changes in public discourse.

3.2.6 Fault Tolerance & Scalability

Our system employs multiple fault tolerance mechanisms working together as a safety net:

Component-level isolation ensures failures remain contained within specific services. For instance, NLP processing failures don't impact data collection workflows. This architecture prevents cascading failures across the system.

Error handling with fallbacks is implemented throughout our codebase, particularly in NLP functions. These catch exceptions and assign appropriate default values - for example, if sentiment analysis fails on a complex post, we assign a sentiment value of -1 and continue processing. Similarly, if location extraction encounters an unrecognizable place name, we default to 'Unknown' rather than crashing.

Our **distributed processing model** with 71 independent topic harvesters and batch-based data processing ensures that failures in one area don't cascade through the system. Behind the scenes, Kubernetes provides automatic pod restart capabilities, while Elasticsearch maintains replicated data across nodes to prevent data loss.

For scalability, our system incorporates multiple approaches:

Horizontal scaling allows the Kubernetes cluster to add more worker nodes as demand increases. Our distributed approach divides work across 71 topic-specific harvesters scheduled throughout the hour, making it easy to add more topics without restructuring.

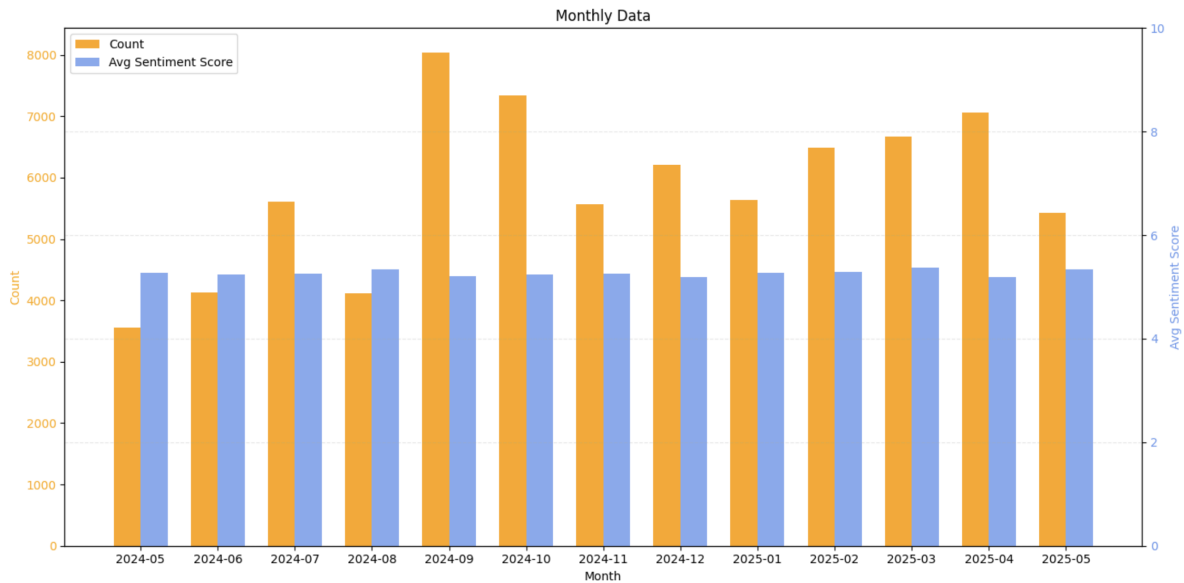
For **efficient resource usage**, we leverage Fission's serverless model, where computing resources are allocated only when functions execute and released afterward - automatically scaling with actual demand rather than peak capacity. Meanwhile, Elasticsearch's sharding distributes our data across multiple nodes, while our NLP component uses configurable batch sizes to optimize processing throughput as the dataset grows.

4 Results Presentation

The findings are organized into four key areas: monthly trends, regional distribution, sentiment score extremes, and short-term (weekly) sentiment patterns. Each area is summarized below with accompanying insights from the corresponding charts.

4.1 Function Interpretation

Monthly Trends (May 2024 – May 2025)

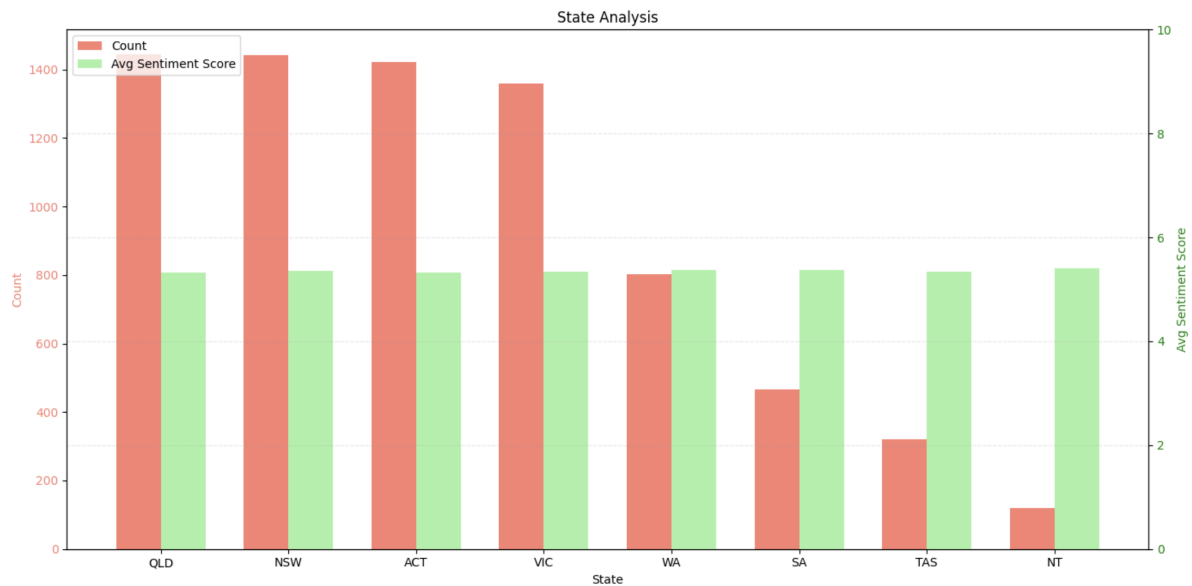


Over the 13-month period, there is a clear fluctuation in posting activity with an overall upward trend in certain periods. Monthly post counts started around 3,500 in May 2024 and rose to a peak of around 8,000 in September 2024, indicating a surge in user activity toward the end of 2024. After this peak, activity dipped slightly (e.g., ~5,500 posts in January 2025) before rising again in early 2025 (another high of roughly 7,500 posts in April 2025). By May 2025, the count settled to about 5,800 posts, suggesting a moderate decline after the second peak.

Alongside these volume changes, the average sentiment score per month (on a scale of 0–10, with 5 being neutral) shows very modest variation, hovering around the neutral mid-4 to 5 range.

Even though the number of posts varied a lot from month to month, the average sentiment score stayed relatively stable. This is likely because most posts were neutral in tone, without enough extremely positive or negative content to significantly shift the average. The sentiment model also tends to assign moderate scores unless the emotion is particularly strong. Overall, the community sentiment remained in a calm, slightly positive zone, indicating a fairly steady and balanced platform environment.

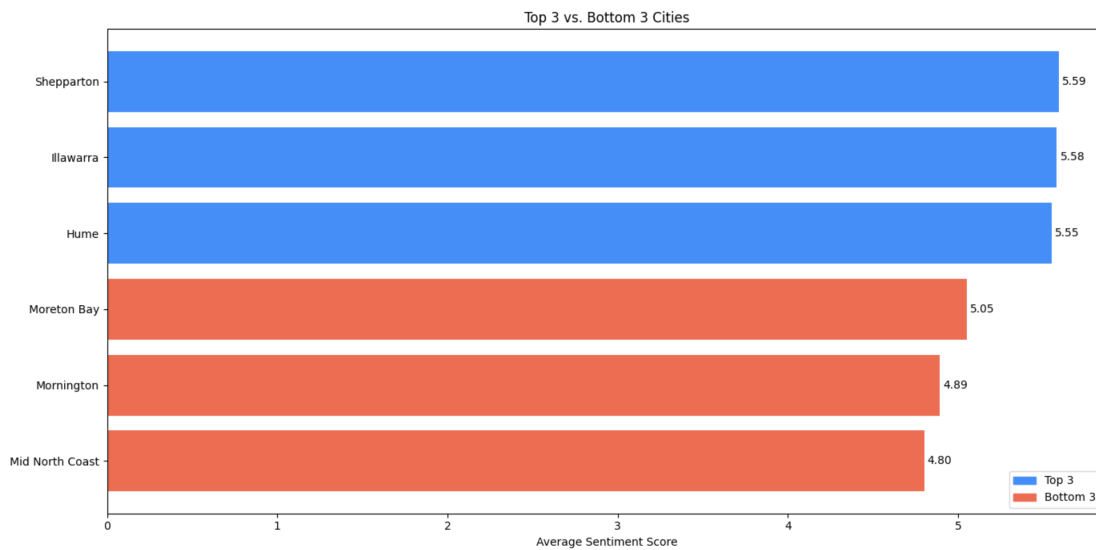
Regional Distribution by State



The distribution of posts across states reveals that content generation is unevenly spread by region. The largest contributors are Queensland (QLD), New South Wales (NSW), and the Australian Capital Territory (ACT), each with approximately 1,300–1,500 posts in the dataset. Victoria (VIC) also shows high activity ($\sim 1,200+$ posts), though slightly less than the top three. Western Australia (WA) and Tasmania (TAS) form a mid-tier with around 700–800 posts each. In contrast, South Australia (SA) and especially the Northern Territory (NT) contributed far fewer posts – SA in the few hundreds, and NT only a few dozen, reflecting their smaller populations and possibly lower engagement with the platforms studied.

Despite these volume differences, average sentiment scores across all states are relatively close, generally in the mid-range (approximately 5.0 to 6.0 on the sentiment scale). This suggests that user sentiment across regions is generally neutral to slightly positive, with no state showing particularly extreme opinions about housing issues.

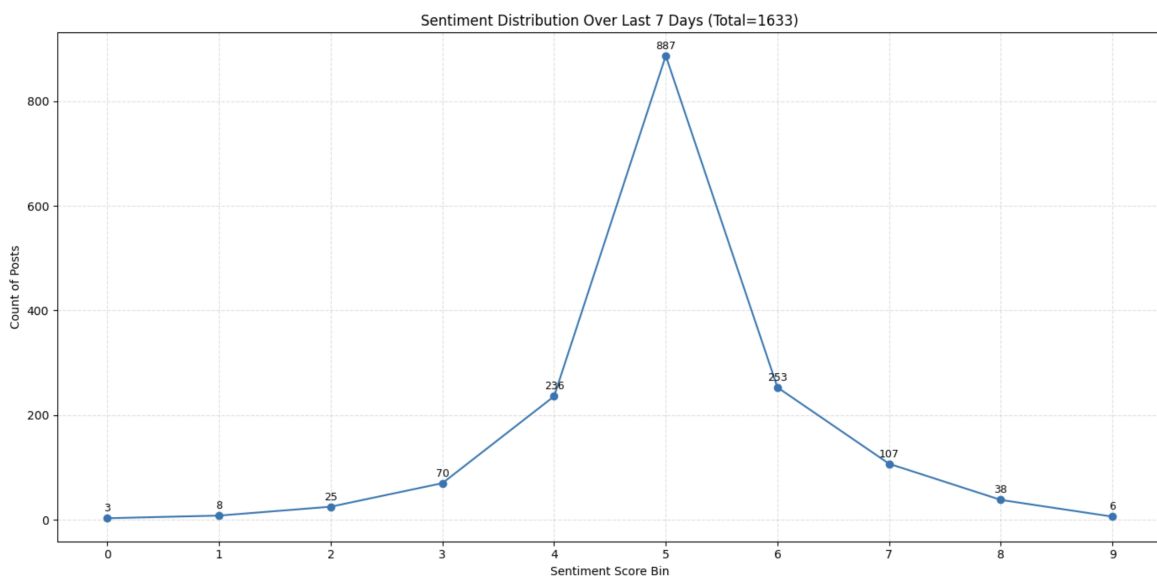
Sentiment Score Extremes by Location



The analysis of sentiment extremes shows that the top three most positive regions—Shepparton (5.59), Illawarra (5.58), and Hume (5.55)—all had average scores just above 5.5, indicating a mildly positive tone. In contrast, the bottom three—Moreton Bay (5.05), Mornington (4.89), and Mid North Coast (4.80)—were only slightly below neutral. The overall gap between the highest and lowest averages is small (about 0.8 on a 0–10 scale), suggesting no region displays strong sentiment polarity.

Overall, sentiment across all locations remains moderate and centered around the neutral midpoint, with minimal variation between regions.

Weekly Sentiment Distribution (Short-Term)



In the short term, sentiment scores are clearly concentrated around the neutral range. The weekly distribution shows a strong peak at score 5, with 887 posts—over half of the

total—falling into this neutral category. Scores 4 and 6 follow with 236 and 253 posts respectively, indicating that mildly negative and positive sentiments are also common. As scores move further from the center, post counts drop significantly: very negative scores (0–2) and very positive scores (8–9) each had fewer than 40 posts, with nearly none at the extremes (0 or 10). This forms a bell-shaped distribution centered at 5, suggesting that most posts express neutral or moderate sentiment. Overall, the data reflects a stable community mood with very few emotionally extreme posts.

4.2 Analysis & Discussion

Monthly & Regional Trends

Over the past year, user engagement showed notable fluctuations, with peaks in September 2024 and again in early 2025. These likely reflect seasonal patterns or event-driven activity. Despite changes in post volume, sentiment scores remained relatively stable, hovering around the neutral midpoint. This suggests that while user activity varied, emotional tone stayed moderate.

At the regional level, most posts came from populous states like QLD, NSW, VIC, and ACT. However, sentiment scores across all states were consistent, generally ranging between 5.0 and 5.6. Even less active regions like NT showed averages close to the national trend. This uniformity suggests that sentiment is shaped more by national discourse than by local issues, and no state displayed strongly polarizing sentiment.

Sentiment Polarity & Short-Term Patterns

An analysis of the most positive and negative cities shows only mild variation: top regions like Shepparton scored just above 5.5, while bottom regions like Mid North Coast stayed close to 4.8. The gap between extremes was less than one point on a 10-point scale, highlighting an overall moderate tone nationwide.

Looking at the most recent week, sentiment distribution formed a bell-shaped curve centered on score 5, with over half of posts falling into the neutral category. Very few posts were highly positive or negative, reinforcing the platform’s emotionally stable environment. These findings point to a community that is active, steady, and rarely extreme—providing a solid base for monitoring trends or guiding platform improvements.

5 Challenges and Discussion

5.1 System Performance & Challenges

Data Volume & Throughput: Ingesting approximately 330,000 documents (≈ 300 MB raw) revealed performance bottlenecks in our pipeline. Initial bulk-indexing with default Elasticsearch settings took over 4 hours; we optimized by disabling refresh during import and parallelizing per-day partitions, cutting import time to approximately 1 hour.

Processing Latency: The NLP enrichment Fission function processed batches of 100 documents in approximately 30 seconds; end-to-end from fetch to index averaged around 2 minutes per topic. Under peak load (all 71 topics running), queueing delays increased significantly until we implemented proper autoscaling configurations.

5.2 NeCTAR Experience

Resource Quotas: NeCTAR’s CPU and RAM quotas constrained our Kubernetes node pool; heavy parallel imports caused pods to be evicted when resource limits were reached.

Storage I/O: Shared NFS storage exhibited high latency under bulk Elasticsearch writes, slowing both import operations and Kibana visualizations.

Networking: Intra-cluster network congestion occasionally resulted in timeouts when calling the Fission router or Elasticsearch, necessitating retry mechanisms.

5.3 Kubernetes Configuration & Lessons

CronJob Management: Managing 71 CronJobs manually was error-prone. We automated YAML generation via scripts and grouped jobs by minute slot to simplify oversight.

The project utilized Kubernetes features that were configured during the cluster setup, though we did not explicitly implement custom settings for node autoscaling or pod affinity rules. The Kubernetes cluster managed resource allocation and scaling based on its default configuration, which was sufficient for our workload distribution patterns.

5.4 Data Collection Issues

API Rate Limits: Reddit’s 1,000-item search cap and Mastodon’s 40-item page limit required implementing efficient pagination and back-off strategies. Improper spacing of CronJobs initially led to request bursts and 429 (Too Many Requests) errors.

Authentication & Account Restrictions: We needed to adjust our Reddit harvesting to use the OAuth Authorization Code + Refresh Token flow, bind script apps to verified phone numbers, and use long-lived Mastodon tokens injected via Kubernetes Secrets to avoid authentication issues.

5.5 Fission Deployment Issues

Package Build Failures: ZIP archives stripped executable bits from `build.sh`, causing build-time errors. We fixed this by explicitly using `chmod +x build.sh` or inlining pip commands in `--buildcmd`.

Routing & Port-Forwarding: Developers struggled to discover the correct Fission router address; we standardized `kubectl port-forward svc/router -n fission 9090:80` and updated documentation to resolve this confusion.

Function Cold Starts: Cold start latency (approximately 2 seconds) for Python functions impacted our tight per-topic schedule; we introduced a keep-alive ping to maintain warm pods.

5.6 Elasticsearch Limitations

Mapping Flexibility: Evolving requirements (adding new NLP fields) required dynamic mapping updates. Elasticsearch field conflicts caused index failures; we adopted a rolling alias and zero-downtime reindex strategy.

Bulk Ingestion Performance: We tuned Elasticsearch Bulk API to use 500-document batches, disabled refresh during import, and enabled asynchronous replication—avoiding memory spikes and throttling while monitoring request latencies.

Query Performance: Aggregations on free-text fields were slow; we created dedicated keyword subfields and precomputed term vectors with the `coerce` option to accelerate facet queries.

5.7 NLP Limitations

While the NLP pipeline effectively extracts basic geographic information, sentiment, and keywords, several limitations remain, particularly in handling the complexity of natural language and geographic references:

Geographic Matching Limitations: The city and state recognition module relies on exact string matches based on a predefined location dictionary. As a result, it fails to recognize informal abbreviations or misspellings. For example, it can correctly detect "Sydney" but not variations like "Syd" or "Sydneyiders". Since it is impractical to exhaustively include all possible nicknames or short forms, some locations remain unrecognized.

Single Location Extraction: When a post mentions more than one city or geographic reference, the current logic captures only the first detected city. This approach simplifies processing but may misrepresent the actual focus of the post, particularly in discussions comparing multiple regions (e.g., "Rents are rising in Melbourne and Sydney").

Language Subtleties: The sentiment analysis model may misinterpret sarcasm, idioms, or cultural expressions common on social media. For instance, sarcastic posts with positive wording but negative intent could be incorrectly labeled as positive.

These limitations affect the granularity and precision of region-based analysis, especially when interpreting spatial trends or local sentiments. Future work could incorporate fuzzy matching, context-aware location disambiguation, or external geocoding APIs to improve geographic inference accuracy.

5.8 Frontend & Visualization

Data Volume and Rendering: We aggregated raw records into time bins or SA3 regions server-side, sending only the summarized data (hundreds of points rather than thousands) to Plotly and Folium to prevent browser performance issues.

Interactive Filtering Performance: We exposed paginated, server-side filtering via REST endpoints, allowing Jupyter to fetch only needed data slices instead of full DataFrames, thus eliminating widget performance issues.

Cross-Platform Consistency: Our upstream normalization unified Reddit and Mastodon data into a common schema (id, resource, text, timestamp, like_count, repost_count), enabling uniform visualization logic and UI components.

6 Conclusion

This project successfully developed a scalable cloud-based system for analyzing public sentiment around Australia's housing and cost-of-living crisis. By integrating Reddit and Mastodon data, we built an end-to-end pipeline using Kubernetes, Fission, and Elasticsearch that supports automated data collection, NLP enrichment, and interactive visualizations.

Our system revealed consistent, mostly neutral sentiment trends across time and regions, despite fluctuations in post volume. The insights, delivered via modular REST APIs and visualized in Jupyter Notebooks, highlight the public’s steady emotional tone and the lack of strong regional polarity in sentiment.

Key technical strengths include serverless deployment, automated scheduling with CronJobs, fault-tolerant batch processing, and modular API design. These features ensured robust performance and responsiveness under varying workloads.

While effective overall, the system has limitations in handling nuanced language (e.g., sarcasm) and informal geographic references. Future improvements may focus on enhancing NLP precision and expanding location inference capabilities.

In summary, this project demonstrates a flexible and resilient architecture capable of delivering meaningful social insights from large-scale unstructured data, while providing valuable experience with modern cloud technologies and big data analytics.

7 Team Roles & Responsibilities

Ian Chi (1579898)

- Responsible for setting up the Kubernetes + Fission cluster and configuring the deployment environment
- Integrated all members’ code to be able to run on Kubernetes
- Set up environments and refined API and NLP-related data processing
- Collected and analyzed container monitoring data for system performance insights
- Handled Elasticsearch data ingestion
- Managed and organized all code in GitLab
- Debugged code throughout the entire project lifecycle

Ling Du (1675590)

- Implemented API-based data scraping workflows, including data source analysis, authentication handling, and scheduled crawling strategies for social media platforms
- Designed data pipelines to preprocess and structure raw JSON data for Elasticsearch indexing
- Developed visualizations
- Assisted with data quality checking and validation to ensure the reliability of scraped content

Yang Jin (1498220)

- Implemented the full NLP processing pipeline and integrated the NLP analysis with Elasticsearch

- Configured batch processing and error handling to ensure fault tolerance and scalability of the NLP module
- Designed the structure of the final report and presentation slides
- Managed team communication and progress tracking

Yuhong Wu (1507649)

- Designed and implemented Elasticsearch mappings for NLP-processed social media data
- Developed four modular Fission functions for analysis and aggregation
- Created API endpoints for frontend integration (/month, /state, etc.)

8 Appendix

8.1 Demo Video Link

<https://youtu.be/-i9v1pg3WtM>

8.2 GitLab Repository

https://gitlab.unimelb.edu.au/YUNGENC/ccc_a2_group28.git