# **Comp90024:** Cluster and Cloud Computing


## Assignment 2 Report

## Global Tariff Discourse Analysis: A Cloud-Based Social Media Sentiment Mapping System

### Group 72


| Full name | Student _ID |
|---|---|
| Jiangyi Yang | 1659893 |
| Dongping Lou | 1551703 |
| Shanshan Li | 1582867 |
| Yadi Chen | 1451572 |
| Gefei Zhao | 1435714 |


**May 18 , 2025**

# Contents

# 1 Introduction

## 1.1 Background Information

On April 2, 2025, U.S. President Donald Trump announced a new round of tariff measures, dubbed the “Liberation Day Tariffs,” marking a renewed intensification of his “America First” economic strategy. The policy introduced a flat 10% tariff on all imported goods and imposed punitive tariffs of up to 145% on imports from China. The stated goal of this initiative was to reduce the trade deficit, revive domestic manufacturing, and pressure trading partners into renegotiating trade agreements.

The announcement of this policy sent immediate shockwaves through international markets. Global supply chains faced widespread disruption, operating costs for multinational corporations surged, and the principles of free trade championed by the World Trade Organization (WTO) came under serious threat. In particular, the escalation of trade tensions between the U.S. and China sparked renewed concerns over economic instability and geopolitical risk.

Although Australia was not a direct target of these tariffs, it remains significantly affected due to its heavy reliance on global trade, particularly exports to China. If China's economy slows in response to the high U.S. tariffs, demand for Australian resources such as iron ore, liquefied natural gas, and agricultural products may decline. At the same time, some U.S. companies may turn to Australia as an alternative supplier, potentially creating short-term opportunities. However, the overall rise in uncertainty is likely to put pressure on Australia’s external trade and increase currency volatility.

Following over a month of rising tensions, the U.S. and China reached a temporary agreement on May 12, 2025, during negotiations in Geneva. This deal established a 90-day “tariff truce,” under which the U.S. reduced tariffs on Chinese goods from 145% to 30%, and China reciprocated by lowering its tariffs on U.S. goods and removing some non-tariff barriers. The agreement brought temporary relief to global markets.

These events sparked widespread discussion across various social media platforms. Posts related to the tariff policy surged sharply in early April, quickly becoming one of the most discussed topics online. The volume and sentiment of public discourse surrounding the issue reflect the depth of public concern and engagement, making it a valuable area of analysis for understanding global and regional reactions.

## 1.2 Objectives

### 1.2.1 Design and Implementation of a Cloud-Based Distributed Data Collection and Processing System

Build a distributed platform based on cloud-native technologies, integrating OpenStack cloud infrastructure, Kubernetes container orchestration, and Fission serverless functions to enable automated collection, real-time processing, and persistent storage of posts and user information related to tariff policies on the Mastodon social media platform since April 2025. The system emphasizes high availability and elastic scalability to ensure stable and reliable operation under large-scale data conditions.

### 1.2.2 Construction of an Efficient and Stable Data Pipeline to Ensure Real-Time Data Flow and Processing

Utilize Redis as a message queue middleware combined with Fission's serverless architecture to achieve full-process decoupling and automation—from collecting tariff-related posts and associated user data on Mastodon, through data formatting and cleansing, to storage. A multi-topic message triggering mechanism is employed to enhance system responsiveness and processing accuracy, ensuring timely and reliable data flow.

### 1.2.3 Application of Natural Language Processing Techniques for Multi-Dimensional Text Analysis

Leverage pre-trained Transformer models alongside mature NLP libraries within the Python ecosystem to conduct stance sentiment analysis, keyword extraction, and topic mining on the collected tariff-related textual data from Mastodon. This enables in-depth exploration of users' attitudes toward U.S. tariff policies and dynamic trends in public opinion.

### 1.2.4 Implementation of Geospatial Visualization and Multi-Dimensional Data Presentation

Utilize the Folium mapping visualization library and the Kibana platform to develop interactive maps and multi-dimensional charts that intuitively display the spatiotemporal distribution and dynamic evolution of tariff-related public opinion on Mastodon. This aids decision-makers in understanding regional variations and temporal trends in policy impact.

### 1.2.5 Validation of System Scalability and Fault Tolerance

Design flexible message triggers and topic mechanisms to support rapid integration and parallel processing of multiple data sources. Employ Elasticsearch replica strategies and persistent storage to safeguard data integrity and service continuity under node failures or network instability, ensuring robust scalability and fault tolerance of the overall platform.

# 2 System Architecture and Design

## 2.1 System Design Overview

To implement an architecture that can collect, process, store, analyse and visualize data, many technologies are used. Figure 2.1 is the system architecture and technologies used in this assignment. Following the order from collecting data to visualizing data.

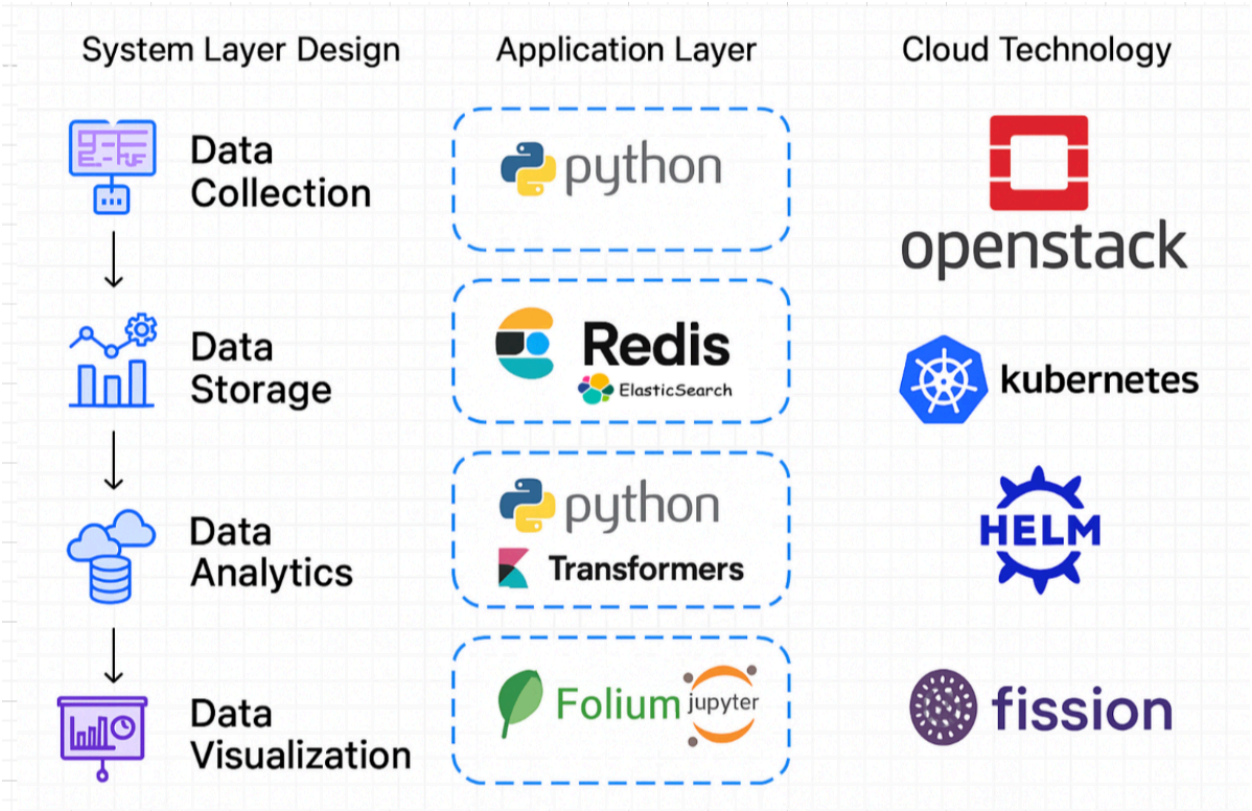



**Figure 2.1** System architecture and technologies

Technologies such as python, transformers and folium are used to collect data from target webpages, analyse data through NLP skill and generate geology graph to visualize data in geology scale. Redis is used to act as a message queue to transport data from harvester application to elastic search. And Elasticsearch is used to store data and analyse data.

For cloud technology implementation in this assignment, Kubernetes (k8s) served as our primary orchestration platform, which helped us arrange and manage containerized applications efficiently throughout the development lifecycle. While the other three technologies (Helm,

OpenStack, and Fission) served as crucial assistants to enhance Kubernetes' capabilities in our cloud infrastructure.

Helm streamlined application deployment through standardized packaging and templating; OpenStack provided the foundational infrastructure with scalable compute and storage resources; and Fission extended functionality by enabling serverless workloads. Together, these technologies formed a synergistic ecosystem - Helm simplified Kubernetes deployments, OpenStack delivered reliable infrastructure, and Fission complemented containerized workloads, all seamlessly integrated under Kubernetes' orchestration to create a robust, full-featured cloud environment. This multi-tool approach allowed each component to focus on its specialty while maintaining tight integration with our Kubernetes-centric architecture.

## 2.2 Melbourne Research Cloud

The Melbourne Research Cloud (MRC) is a comprehensive cloud computing platform operated by the University of Melbourne's Research Computing Services (RCS), designed to provide researchers with scalable, secure, and flexible infrastructure for academic projects.

As an institutional node within Australia's national NeCTAR Research Cloud ecosystem, MRC leverages OpenStack technology to deliver seamless cloud services while maintaining its specialized focus on supporting University of Melbourne research initiatives. This dual affiliation enables researchers to benefit from both local institutional resources and, when required, expanded national computational capabilities through the NeCTAR network.

For this assignment, MRC provides us with a reliable platform to deploy all of our applications with its resources. As shown in figure 2.2, 4 instances with 8 cpus are used in the project, along with 210 GB of data storage space. Within this cluster configuration, we implemented a distributed computing architecture where three nodes were designated as worker nodes to handle computational workloads, while the remaining instance served as the master node to orchestrate and manage the cluster operations.

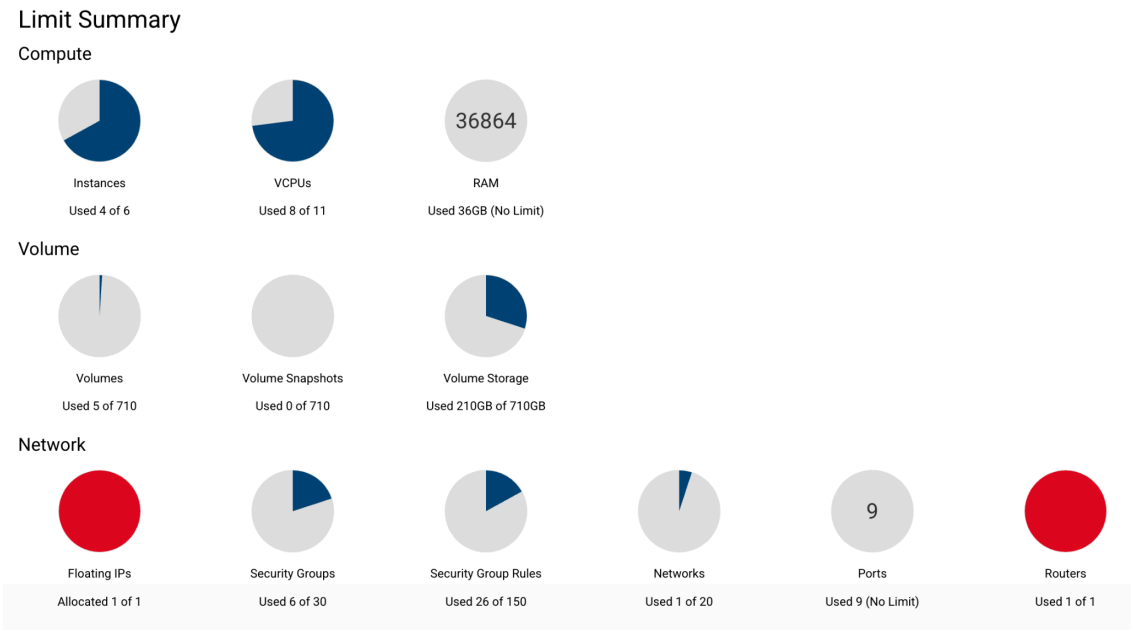



**Figure 2.2** resource usage in MRC

## 2.3 Pros and Cons

During the project's implementation and after gaining extensive practical experience with various aspects of MRC, we identified the following benefits and shortcomings:

Advantages

- **Easy to use**: It is easy and efficient for us to create our own cluster through openstack on MRC, we can apply for nodes, cpus, storage spaces easily by some short commands. Also we can get the detail of resource allocation situations clearly by checking the dashboard on the webpage or through the command line.
- **Stability:** Once set up, MRC VMs are generally stable. We have never experienced a clash or disconnection since deployment.
- **Kernel-based Virtual Machine**: MRC uses KVM to manage virtual machines, which allows the system to directly use hardware for memory tasks. This makes virtual machines run faster and easier to set up, giving researchers better performance with less effort.

Disadvantages:

- **No GPU Instances**: ML/DL projects (e.g., TensorFlow/PyTorch) required external cloud credits (AWS/GCP). We find that it's hard to deploy functions using pytorch and transformer packages on MRC.
- **Limited On-Demand Scalability**: MRC has restricted scalability. Users are assigned a fixed resource quota (CPU, RAM, storage), and exceeding this requires a manual

request and approval process,which can take lots of time.In contrast, **AWS** provides virtually unlimited resources, allowing users to spin up dozens or hundreds of instances instantly without approval.

- **Not elastic on resources:** MRC does not support automatic scaling of resources. All virtual machines must be manually created, configured, and managed. on the other hand, it offers services like Auto Scaling Groups and AWS Lambda that can dynamically scale resources up or down based on demand or usage patterns.

# 3 Application Stack

## 3.1 Data pipeline

To introduce the application stack we have used in the assignment, data pipeline should also be introduced. Figure 3.1 is the data pipeline and structure for the assignment.

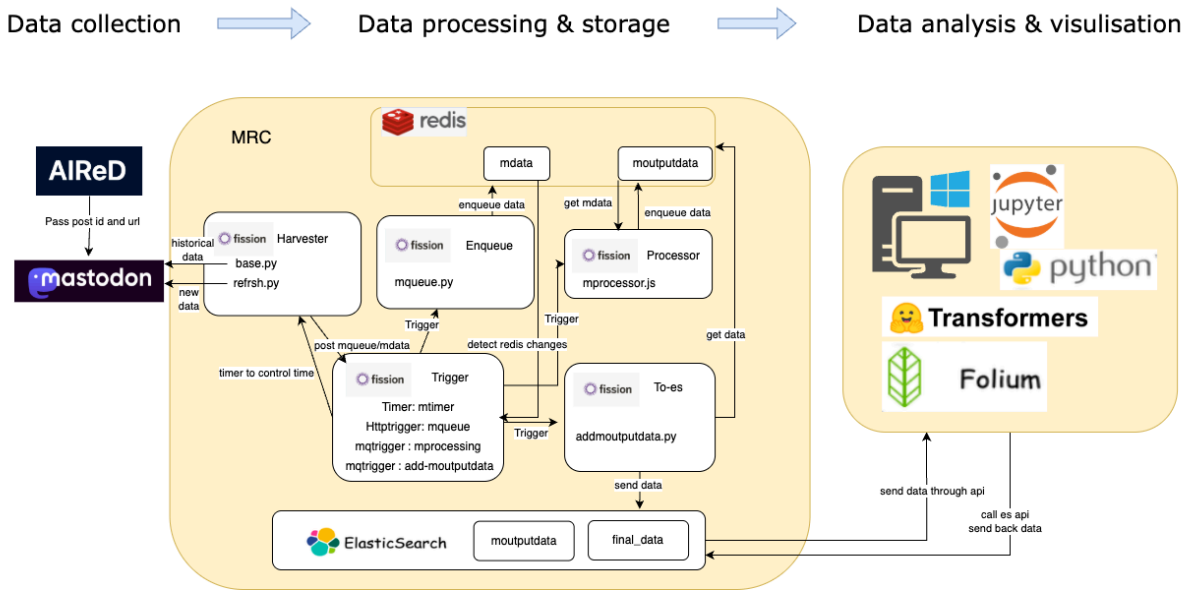



**Figure 3.1** Data pipeline and structure

In the data collecting part, mastodon post urls are collected from AIReD, then all posts are collected by two fission functions. One is base function, collecting data since 1st april,which is the date tariff was put forward. It only runs once to collect historical data. Another one is the refresh function, collecting data from 5 days ago, which would run each 2 hours. So that we can refresh new data about tariffs.

After data is collected, the function will call a url 'http://router.fission/mqueue/{topic}' to activate httptrigger. Then heeptrigger will trigger an enqueue function, which can send data to redis list. And the topic will follow the topic in the url. In this assignment, the topic is mdata.

While data is sent to redis, corresponding mqtrigger will also be activated. It would track specific topic and if data is sent to the topic. It will trigger the next function. In this project, the function is mprocessor, which processes the data to another format and sends data to another topic, topic moutputdata. The processor mainly adjusts some formats such as time of the original data.

After topic moutputdata receives data, another mqttrigger is activated. It will trigger the final function, addmoutputdata function, which is used for receiving data from moutputdata topic and send them to elastic search.

Finally, data is sent to elastic search. And through the put-forward command(port 9200), data can be requested by the local computer using restful api. Then the following data analysis and data visualization jobs can be implemented.

## 3.2 Back-end

### 3.2.1 K8S

Kubernetes, commonly referred to as K8s, is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. Originally developed by Google, Kubernetes is now maintained by the Cloud Native Computing Foundation (CNCF) and is widely adopted across the industry for building and managing scalable, resilient, and cloud-native systems.

In this project, K8s is used mainly in the backend part. Through openstack, a K8s cluster is created on the MRC platform with 4 nodes and 9 cores,As shown in the figure 3.2.

```
● (base) ➜  comp90024 git:(master) ✗ kubectl get nodes
  NAME                                          STATUS   ROLES           AGE   VERSION
  comp90024-r2jjseyzbmlo-control-plane-zjtvt    Ready    control-plane   33d   v1.31.1
  comp90024-r2jjseyzbmlo-default-worker-h4hvk-jjk7s   Ready    <none>          33d   v1.31.1
  comp90024-r2jjseyzbmlo-default-worker-h4hvk-lbzb8   Ready    <none>          33d   v1.31.1
  comp90024-r2jjseyzbmlo-default-worker-h4hvk-lzf89   Ready    <none>          33d   v1.31.1
```

Figure 3.2 Nodes status

Then we can install kibana, elastic search, fission and some other applications through helm. Their namespace, version and other config can be adjusted by command line and config file.

There are many advantages about K8s.

- **Scalability:** Kubernetes allows horizontal scaling of applications, either manually or automatically based on resource usage metrics. This makes it ideal for dynamic workloads.

- **High Availability and Resilience:** Kubernetes automatically detects and replaces failed containers, reschedules workloads when nodes fail, and supports multi-node, multi-region deployments for fault tolerance.

- **Portability and Multi-Cloud Support:** As a platform-agnostic system, Kubernetes supports deployment across various cloud providers (e.g., MRC, AWS, GCP, Azure) as well as on-premise environments, enhancing flexibility and avoiding vendor lock-in.

### 3.2.2 Fission

Fission is an open-source serverless framework designed specifically for Kubernetes. It enables developers to run short-lived, event-driven functions without managing the underlying infrastructure. Fission abstracts away much of the complexity of Kubernetes, allowing developers to focus purely on writing and deploying code. Fission is developed and maintained by Platform9 and supports several programming languages, including Python, Node.js, Go, Java, and more, which makes it easy to use.

In this project, fission is mainly used in spec ways. Through initializing the spec folder and maintaining yaml files in it. Fission packages, functions and triggers can be easily and clearly deployed on K8s. As shown in Figure 3.3, all details about fission can be clearly tracked through spec.

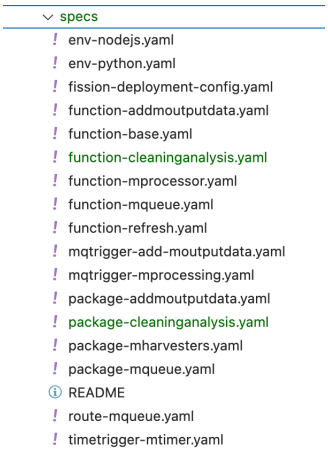



Figure 3.3 Specs details

Many fission components are used in the assignment. The first is fission env. It can define the basic environment for fission functions. Another is the fission package. Through packaging, functions can use some other libraries which are not covered in the basic environment. Thirdly, with basic env and specific package, fission function can be deployed and run on K8s. Finally, there are many kinds of triggers to help trigger those functions. For example, a timer trigger can trigger functions timely. A httptrigger can trigger functions when a specific url is called. And a mqtrigger can trigger functions when a specific topic in the message queue is updated. Through utilizing these fission components, we can realize some complex applications.

There are many advantages about fission.

- **Easy to use:** Fission function can be generated in many languages such as python, java and node.js.
- **Serverless Function Execution:** Run functions in response to HTTP requests, timers, or Kubernetes events — without writing boilerplate code.
- **Kubernetes Native:** Leverages Kubernetes-native components such as pods, deployments, services, and CRDs (Custom Resource Definitions), allowing tight integration with the Kubernetes ecosystem.

### 3.2.3 Redis

Redis (Remote Dictionary Server) is an open-source, in-memory data store primarily used as a database, cache, and message broker. It is known for its high performance, low latency, and rich data structure support. Unlike traditional relational databases that store data on disk, Redis keeps the entire dataset in memory, enabling extremely fast read and write operations. This makes Redis ideal for real-time applications such as caching layers, analytics engines, gaming leaderboards, and session management systems.

In this project, redis is deployed on port 6379. And functions can send or receive data through calling url ‘redis://redis-headless.redis.svc.cluster.local:6379’. Also it can be directly visualized by redis-insight. Two topics, mdata and moutputdata are created to receive and send data. The mdata topic is used for saving raw data collected from mastodon. While the moutputdata is used for saving data after processing. If these topic are updated, correspond mqtriggers will be activated and trigger specific functions.

There are many advantages about fission.

- **In-Memory Storage:** Redis stores all data in memory, enabling extremely fast read and write operations.
- **Persistence Support:** Although Redis is memory-first, it supports persistent storage using RDB snapshots or AOF (Append-Only File) logging.
- **Rich Data Types:** Redis supports a variety of data structures, including list, set, sorted set and so on.

### 3.2.4 Elastic search

Elasticsearch (ES) is a distributed, open-source search and analytics engine designed for horizontal scalability, reliability, and real-time performance. Initially released in 2010 by Elastic NV, it is built on top of Apache Lucene and is commonly used for full-text search, log and event data analysis, and complex analytics at scale. Elasticsearch is a core component of the Elastic Stack (ELK Stack), which includes Logstash (data ingestion), Beats (lightweight shippers), and Kibana (data visualization).

In this project, elastic search is deployed on port 9200.And functions can send or receive data through calling url '*kubectl port-forward service/kibana-kibana -n elastic 5601:5601*'. Two index,moutputdata and final_data, are created to store data. The moutputdata is used for store processed data sent from redis. And final_data is used for store analysed data from a local computer, which contains sentiment label and sentiment score. When local computer need to access elastic search. The 9200 port should be put forwarded. Then it can be accessed through restful api. Kibana is deployed in port 5601 to help pre-visualize data, as shown in the figure 3.4. We can visualize data by creating specific data view.

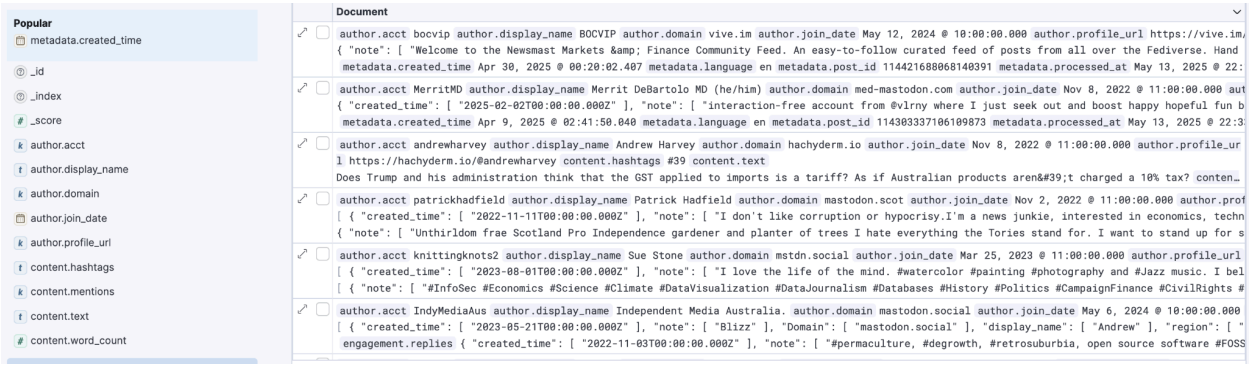



**Figure 3.4** Kibana data view

There are many advantages about elastic search.

- **High-Performance Full-Text Search:** Elasticsearch provides extremely fast full-text search capabilities, powered by Apache Lucene. It supports natural language queries, stemming, tokenization, and relevance scoring.
- **Scalability and Distributed Architecture:** Elasticsearch is designed to scale horizontally. It can distribute indices across multiple nodes, making it suitable for handling terabytes of data and high query throughput.
- **RESTful API and Language Clients:** Elasticsearch provides an easy-to-use HTTP/JSON interface, along with official clients for Python, Java, JavaScript, and more, making it integration-friendly.

## 3.3 Front-end

Front-end part is mainly realized by jupyter notebook. Through using restful api and put forward port 9200, we can fetch and post json data easily. To generate beautiful graphs and charts many packages are also utilized.
In the data analysis part, the packages used are:

- **Elasticsearch** provides a Python client for interacting with an Elasticsearch cluster.
- **emoji** is used to detect, filter, or process emoji characters in text.
- **nltk** provides common stopwords to help remove irrelevant words during text preprocessing.
- **transformers** loads pre-trained NLP models for tasks like summarization, translation, or question answering.

In the data visualisation part, the packages used are:

- **Matplotlib** is used for statistical graphing, including bar charts, line charts, pie charts, and scatterplots.
- **Folium** is used for interactive visualisation of geospatial data.

## 3.4 Scalability

For now, data is only collected for AIReD and Mastodon. But our project also contains good scalability. Through fission, redis, and es, we can quickly scale up and collect data from other websites.

For the fission part, if we need to collect data on a new website such as twitter, we can write the harvester to get data from twitter and deploy it as a fission function. Then through adding new triggers, we can trigger the new harvester timely. Then we can add a new topic in redis and add a new enqueue function and new httptrigger. When the harvester posts data to the specific url, data can be sent to the new redis topic.

After that, we can add a new processor and mqtrigger to process data. If the processed data structure is the same as original data stored in moutoutdata topic. We can directly send data to this topic with the fission function. Or we can create a new topic and add a new topic and es index to fit the new data structure. Finally we can add a new fission mqtrigger and fission function to send data to es. Then we can analyse new data again.

### 3.5 Fault-tolerance

In the development and implementation of all the components used in the project on MRC. We have also taken some measures to ensure fault-tolerance. So that the project would not be influenced by some errors.

For the elastic search, We have set the replica to 2. So if one node is broken, another replica can still work. Besides, we use a custom perfretain storage class with the Retain reclaim policy to ensure that even if Elasticsearch is uninstalled, the persistent volumes will not be automatically deleted, thereby preventing data loss.

For the fission part,we have added lots of logging output so that errors can be tracked quickly. If error occurs, we can check the log of problematic functions through command fission function log -f --name functionname. Then we can fix the function. It is easy to redeploy the fixed function. What we need is use command *fission spec apply --specdir ./specs --wait*. Also, the *--set persistence.storageClass='perfretain'* configuration in Fission provides persistent storage, preventing the loss of function or configuration data in case of component failures.

Also, we used unit tests to test two main harvesters, the base harvester and the refresh harvester. For base harvester, it contains these functions：

- get_jwt,
- fetch_all_ids,
- infer_region,
- infer_region_from_note,
- extract_text,
- get_domain_from_url,
- process_url

And test functions are wrote for all these functions to ensure that all parts of the harvester are functional, as shown in the figure 3.5.

```
(unimelb) ➜  fission git:(feature/backend) ✗ python -m unittest harvester/test_base.py -v
test_extract_text (harvester.test_base.TestBase) ... ok
test_fetch_all_ids (harvester.test_base.TestBase) ... ok
test_get_domain_from_url (harvester.test_base.TestBase) ... ok
test_get_jwt (harvester.test_base.TestBase) ... ok
test_infer_region (harvester.test_base.TestBase) ... ok
test_infer_region_from_note (harvester.test_base.TestBase) ... ok
test_process_url (harvester.test_base.TestBase) ... ok

----------------------------------------------------------------------
Ran 7 tests in 3.396s
```

**Figure 3.5** testunit output for base harvester

Similarly, for the refresh harvester, it contains these functions:

- infer_region,
- infer_region_from_note,
- extract_text,
- get_domain_from_url,
- process_url

All these functions are tested by corresponding functions. So they are all ensured to be functional.

# 4 Data Collection

The data collection process for this project is divided into two main stages: Initial Data Collection and Daily Incremental Updates. Data is sourced from the Mastodon social media platform through a combination of a custom ElasticSearch-based API and the official Mastodon API.

## 4.1 Initial Data Collection

The system first utilizes the ElasticSearch API provided by *https://api.aio.eresearch.unimelb.edu.au* to retrieve posts related to a specific keyword — tariff in this case — within a defined time range (April 1 to May 10, 2025). The process involves the following steps:

- Authenticate via JWT (JSON Web Token) using a provided API key.
- Since the API returns a maximum of 200 results per page, a pagination mechanism is implemented to iteratively request all matching post IDs (i.e., Mastodon post URLs).
- As the search results may include URLs from non-Mastodon domains, a regular expression is used to filter out only valid Mastodon post links, avoiding invalid or irrelevant data.
- For each valid post URL, the Mastodon Python API client is used to retrieve detailed post information, including the content, author metadata, language, timestamp, number of favourites and their users, boosts and their users, and replies along with corresponding users.
- A geo-inference function is applied to estimate the geographical region of the post author (e.g., Australia, China, Japan), based on post language, the author's profile description, and the instance domain name, using a pre-defined set of keyword-based rules.
- All collected data is saved in JSON format, providing structured data for downstream analysis.

This stage establishes a large and well-structured dataset of posts containing the target keyword, forming the basis for further temporal and user behavior analysis.

## 4.2 Daily Incremental Data Updates

To maintain data freshness, an automated incremental data collection mechanism is implemented with the following workflow:

- Since the ElasticSearch platform updates data up to the previous day (not in real time), the system queries for all posts containing the keyword *tariff* posted "yesterday."
- Using JWT authentication, the system queries the ElasticSearch API for new post IDs.
- A regular expression is used again to validate Mastodon URLs, ensuring data integrity and avoiding duplicates.
- For each new valid post, the Mastodon API retrieves the full post details, and the same geo-inference process is applied to extract the author's regional information.
- The newly collected data is saved in separate JSON files for daily updates.
- To prevent API rate limiting, a 1-second interval is applied between data fetches.
- The update mechanism runs periodically every few hours. While ElasticSearch data is updated daily, Mastodon post data retrieved via URLs is updated in real time, thus allowing near-real-time insights through multiple scheduled runs.

## 4.3 Data Cleaning and Geo-inference

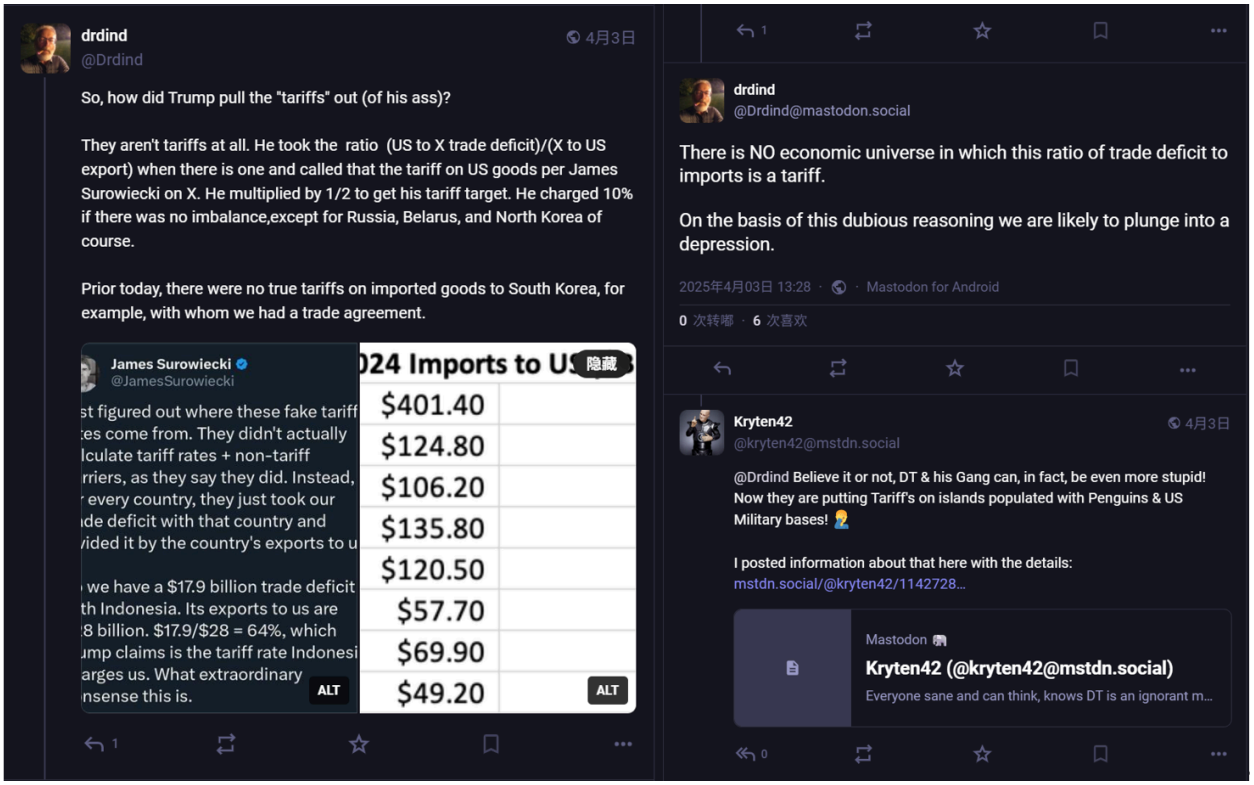
Since Mastodon post content is in HTML format, a regular expression-based HTML tag remover is implemented to extract clean text. This process also removes unnecessary line breaks and optimizes text formatting to better match the actual display on Mastodon, including preserving links, emojis, and other visible elements.

A composite region inference function estimates user locations by combining data from the user's bio (`note`)(we use nlp tech to do it), post language codes, and Mastodon instance domains. This enhances the accuracy of location-based analysis.

Additionally, each post’s metadata — including the number of favourites, boosts, replies, and the associated user information — is collected to support rich behavioral and interaction analysis.

## 4.4 JSON Data Structure Description

When we click into the URL of a relevant post, the interface looks like this:




It displays the post content, author, number of reblog or favourite, and any replies. We can get the users related to this post and their information we can get from the mastodon api.

The collected data is stored in structured JSON format, with each file representing a list of Mastodon posts that match the target keyword (e.g., “tariff”). Each entry in the JSON corresponds to a single post and contains detailed metadata about the post itself, the author, and interactions (likes, reblogs, replies). The structure is organized as follows:

- **post_id**: Unique identifier of the post on Mastodon.
- **created_time**: Timestamp when the post was published, in UTC format.
- **content**: Plain-text content of the post, with HTML tags removed.
- **url**: Direct URL link to the post on the Mastodon instance.
- **language**: Detected language of the post content (ISO 639-1 code).
- **region**: Inferred geographic region of the post author based on instance domain, user bio, and language.
- **author**: A nested object containing metadata about the author, including:
    - **acct**: Author’s username (with instance domain if applicable).
    - **display_name**: Author’s display name.
    - **created_time**: Account creation date.

    - **url**: Profile URL of the user.
    - **Domain**: Mastodon instance the user belongs to.
    - **note**: User bio text.
- **favourited_by**: A list of users who liked the post, with each entry including the same structure as `author`, plus an inferred `region`.
- **reblogged_by**: A list of users who reblogged the post, with identical structure to `favourited_by`.
- **replies**: A list of reply objects, each containing:
    - User metadata (`acct`, `display_name`, `created_time`, `url`, `Domain`, `note`, `language`, `region`).
    - **content**: The reply text content.

This structure ensures that all relevant post metadata and user interactions are captured in a standardized and machine-readable format. And this is the JSON data we generate after the initial crawling and processing.

# 5 Data Analysis

From now on, we have obtained a structured dataset collected from Mastodon. Each record includes an original post related to the tariff policy, which is the main focus of this project. In addition to the post content, the dataset also collects rich user information including instance domains, regions, as well as interaction data for each post, for example, the users that replied, reblogged, or favorited it.

In this stage, we refined the raw data into a cleaner and more structured format. Then we applied natural language processing techniques to assess each user's stance toward the tariff topic. The generated sentiment labels and scores were added to the dataset. Finally, the processed data was indexed into Elasticsearch. This guarantees flexible, fast indexing and retrieval for following analysis and visualization.

## 5.1 Software Tools

### 5.1.1 Elasticsearch

Elasticsearch is a popular distributed DBMS that's especially useful for big data scenarios. Unlike other systems like CouchDB or PostgreSQL, Elasticsearch is document-oriented, making it easier to handle partitioning. While this means it is less strict about consistency and availability, but much stronger in terms of partition tolerance, which is key when working with large, distributed datasets.

Internally, Elasticsearch uses a two-phase commit method to replicate data from primary shards to secondary shards and employs Paxos-like behaviours for electing a master node. Because of this design, it might temporarily lose availability if there is a network issue. Also, it uses sequence numbers and a number that identifies the primary shard to keep document updates orderly.

For data analysis, it is a suitable choice because it easily handles large-scale searching and aggregation queries. Additionally, Elasticsearch offers straightforward interaction through RESTful APIs, and provides a graphical interface, Kibana that simplifies data exploration and visualization.

### 5.1.2 Other Tools (used in PyCharm)

- Transformers: Pretrained models used to analyze user posts, calculating sentiment scores and identifying sentiment labels based on the text content.
- Requests & urllib3: HTTP libraries for fetching new data and updating existing datasets via RESTful APIs.

- re & emoji: Used regular expressions and emoji processing tools to clean text data, removing unnecessary special characters and converting emojis into textual descriptions.
- datetime & timezone: Made sure timestamps and timezone stayed consistent across different datasets.

## 5.2 Data Structure

The Elasticsearch index is configured with 3 shards and 1 replica to support distributed storage and fault tolerance. The dataset was collected from Mastodon and indexed into Elasticsearch under the index name **moutput**. Each document captures initial information, including 4 parts:

- **author:** author.acc, author.display_name, author.join_date etc.
- **content:** content.text, content.hashtags, content.mentions, and content. word_count. The **text** field is analyzed using the built-in English analyzer, while the hashtags field uses a custom analyzer (hashtag_analyzer) that tokenizes hashtags in lowercase for keyword matching.
- **engagement:** includes three nested subfields **replies**, **favorites**, and **reblogs**. Every nested part also includes user metadata.
- **metadata:** records key information about each post, including post_id, created_time, processed_time and region etc. With appropriate field types like keyword and date, filtering and time-based aggregations can be easily done during query operations.

To support further analysis, we derived a new index, final_data by extending moutput with additional fields related to user attitude towards tariff and interaction information.

- **status:** added under the author field to store sentiment annotations, including status.label and status.score.
- **Interaction_counts:** introduced under the **engagement** field to record the number of **replies_counts**, **favorites_counts**, and **reblogs_counts**, which can be used for aggregation queries.

## 5.3 Sentiment Analysis

To examine public sentiment surrounding the tariff policy, a sentiment classification process was conducted on original Mastodon posts. By default, users who favorited a post were assumed to inherit the same sentiment label from the author, while rebloggers are marked as “unknown” due to their ambiguous stance.

### 5.3.1 Model Selection

During the data collection stage, we performed language detection and found that around 85% of the posts were in English. However, the dataset also includes content in French, Italian, and several other less common languages.

To deal with the multilingual nature of our dataset, we decided to use the joeddav/xlm-roberta-large-xnli model. It's a powerful tool for zero-shot classification that's been trained on the XNLI dataset, which covers 100 languages, so it works well even for non-English text.

### 5.3.2 preprocessing and classification

Text preprocessing includes removing URLs, HTML tags, markdown images, markdown links and emoji normalization. This helped produce cleaner and more meaningful inputs for the language model when generating embeddings and making predictions.

Classification part applied a zero-shot classification model (joeddav/xlm-roberta-large-xnli) to each cleaned post, using the user-defined labels: support tariff, oppose tariff, and neutral. The resulting label and score were added to the database.

## 5.4 Elasticsearch Query and Update Operations

### 5.4.1 Scroll Query

Given the structure and size of our dataset, traditional query methods were insufficient for large-scale document searching. Therefore, we adopted the scroll API, which allowed us to retrieve documents in batches and thus can load real-time data in parallel.

### 5.4.2 Bulk Update

After generating sentiment labels, scores and interaction counts for each post, we used the bulk API to update documents in Elasticsearch efficiently. By performing multiple indexing or update operations in a single API call, we significantly reduced the overhead and improved performance.

This was particularly useful when updating hundreds or thousands of documents after sentiment classification or user interaction enrichment. Batching multiple documents into a single request can optimize the use of the network and minimize the load on Elasticsearch cluster.

### 5.4.3 Aggregation Query

Our data structure made it easy to run a wide range of aggregation queries to extract useful insights from the dataset with both bucket and metric aggregations.

- **author.status.label** and **metadata.region** are mapped as keywords. This lets us easily count how many users support or oppose the tariff, or see which regions are more active in the discussion.
- **metadata.created_time** and **engagement.favorites.created_time** are defined as date, which allows us to track how user engagement and attitude towards tariff changed over time.
- **replies**, **favorites** and **reblogs** are all mapped as nested fields. For example, we could group replies by sentiment label or region inside each post.
- **interaction_counts** defined as integers, can be directly queried to get which regions or which domains have more active users.
- **content.hashtags** also supports precise keyword matching and efficient aggregation of popular hashtags.

Altogether, the mapping design allowed us to query the data flexibly from different angles—by time, by region, by sentiment type, by user, and also by interaction type.

## 5.5 Data Analysis Pipeline

The analysis pipeline described below was executed locally, with data processing scripts running in a local Python environment and connecting to the Elasticsearch cluster.

**Incremental Data Fetching:**
Data updated in the original index (moutput) is retrieved based on the timestamp field (metadata.created_time). Each process retrieves only new documents created since the last execution.

**Sentiment Analysis:**
For each document retrieved, an NLP model is applies to generate two parts:

- Sentiment Score: Numerical representation of sentiment intensity (e.g., from 0 to 1).
- Sentiment Label: Categorical sentiment classification (e.g., support_tariff, oppose_tariff, neutral).

The results (status.score and status.label) are periodically stored into the index named sentiment_status. Each sentiment entry in this index uses the same unique document ID from moutputdata.

**Real-Time Data Fetching:**
Real-time data is periodically fetched from the moutput index via Scroll API.

**Bulk Indexing to final_data:**

Sentiment information is matched by document ID against the sentiment_status index. Once matched, each document is enriched by appending the following fields: **author.status.score**, **author.status.label**, e**ngagement.favorites.status.label**, **engagement.reblogs.status**.

the following interaction metrics are also calculated and stored for analytics:
**engagement.interaction_counts.favorites_count**
**engagement.interaction_counts.replies_count**
**engagement.interaction_counts.reblogs_count**

Then documents are efficiently indexed into Elasticsearch using the Bulk API.

# 6 Data Visualization

## 6.1 Public opinion visualization

### 6.1.1 Weekly opinion distribution analysis

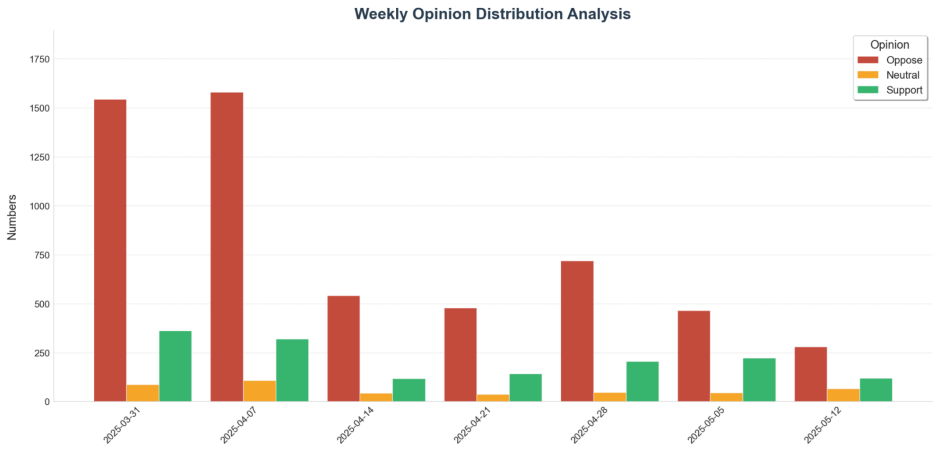



Figure 6.1.1 Weekly opinion distribution analysis

This bar chart shows the views of netizens on the US tariff policy from March 31, 2025 to May 19, 2025, and records the number of posts in support, neutrality and opposition. It can be seen that a large number of posts every week are against this policy. As time goes by, the volume of online discussions has gradually declined. Although opposing voices still dominate, the gap between supporters and opponents is narrowing, indicating that the policy shock effect is time-limited, with a cycle of two weeks.

### 6.1.2 Keywords analysis

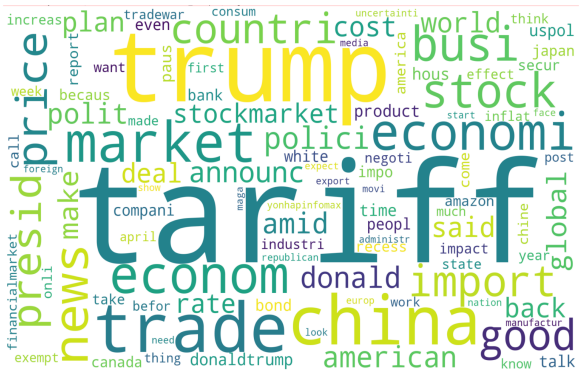



Figure 6.1.2 Word cloud of posts

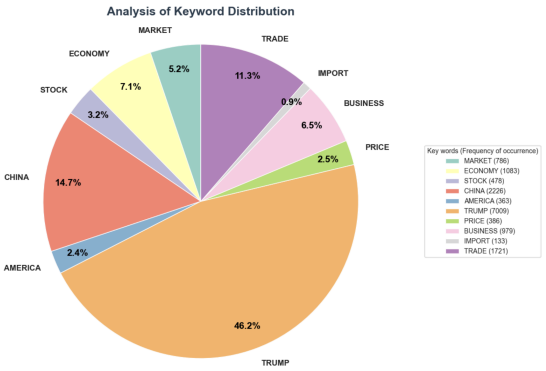



Figure 6.1.3 Analysis of keywords

From this cloud of words, it can be seen that netizens' discussions are mainly centered around trump,american tarrif, China, import, stockmarket, price, trade, indicating the deep connection between tariff policy and international trade and state relations. And it has been deconstructed by the public as a political and economic complex, with the focus of discussion shifting from simple trade measures to industrial security strategies.Also, product and cost could be noticed showing quite big. The public is worried about the impact of tariff policies on their daily living costs.

Among the selected ten keywords, trump formed an absolute focus with a proportion of 46.2%, far exceeding other keywords. In contrast, america only accounted for 2.4%, indicating that the public highly associate tariff policies with specific political figures. Prove that policy discussions are more inclined towards the narrative of individual political legacies rather than that of state subjects. Secondly, there is "china" and "trade", indicating that policies are regarded as strategic game tools between China and the United States. The public all believe that the tariff policies promulgated by the United States are closely related to Sino-US relations and profoundly affect international trade. economy, business and market follow closely. The public's discussions on the market economy tend to be macro rather than specific trade behaviors.

### 6.1.3 Trend of post and interaction

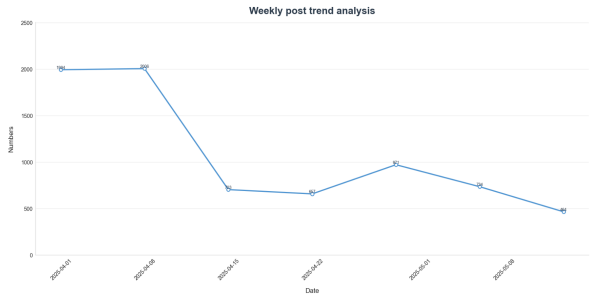



Figure 6.1.4 Weekly post trend

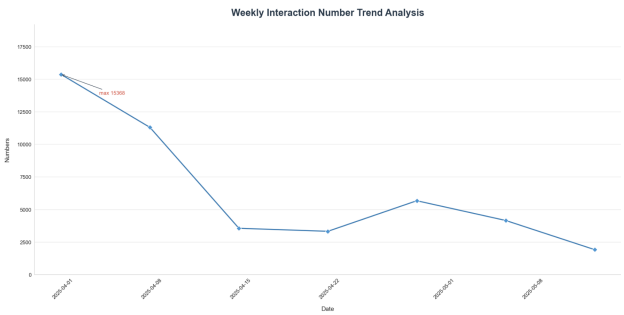



Figure 6.1.5 Weekly interaction trend

Weekly post trend analysis reveals that the number of posts has a two-stage attenuation, plummeting sharply from 1,994 to 91, and later fluctuating within the range of 464-657, reflecting the residual discussion stickiness of the core user group and forming a conduction time lag of "bursting of the interaction bubble → retention of low-quality content". As shown in weekly interaction number trend analysis, the weekly interaction volume presents a typical exponential attenuation feature. The initial peak of 15,368 was reached within 72 hours of the policy release, and then it attenuated at a rate of 41.7% per week. The abnormal rebound in 5/1 week was significantly associated with the announcement of the WTO interim meeting. The comparison of the two graphs shows that the interaction volume is driven by external events and shows a tendency to decay rapidly, while the attenuation of the posting volume is more dependent on the endogenous ecosystem. The ratio of posting volume to interaction volume reached 96.7% in 5/8 weeks, indicating that the discussion ecosystem has regressed to low-value circle communication.

### 6.1.4 Sentiment analysis of every opinion

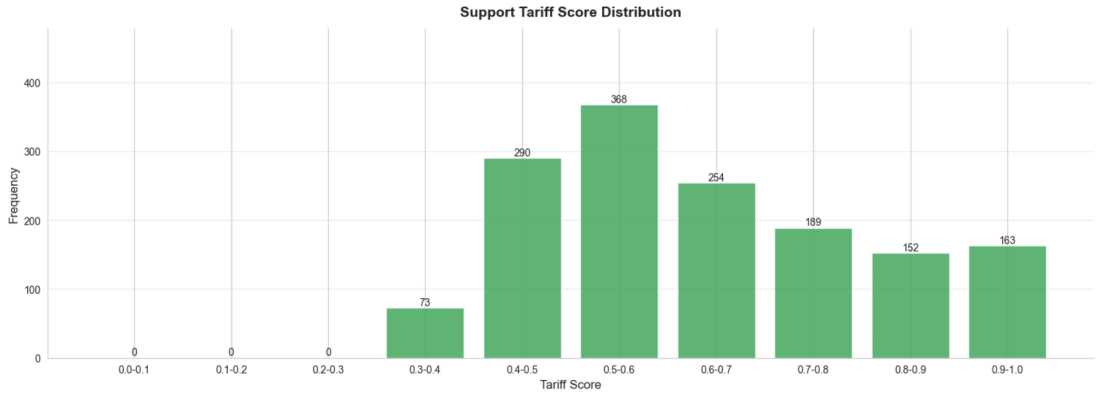



Figure 6.1.6 Support tariff score distribution

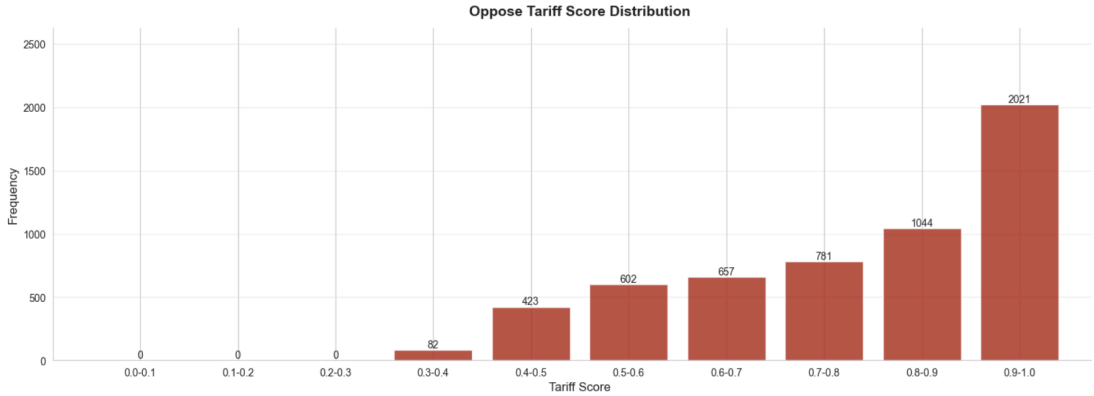



Figure 6.1.7 Oppose tariff score distribution

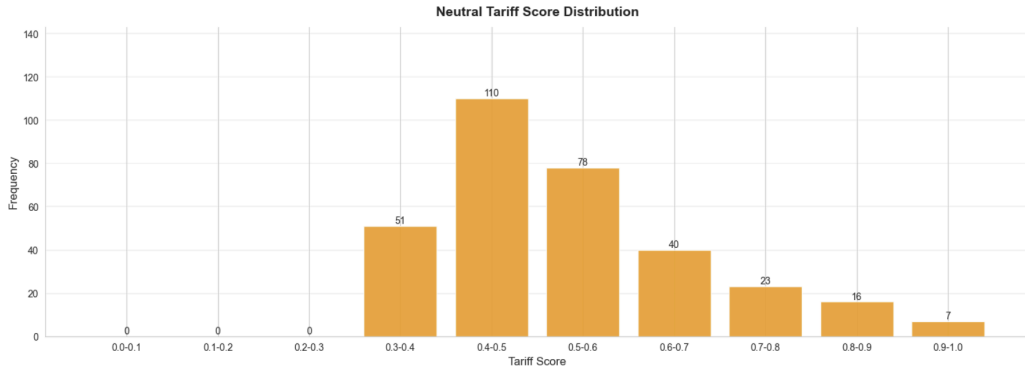



Figure 6.1.8 Neutral tariff score distribution

The supporter group shows a significant moderate-intensity support characteristic, with the core emotional intensity concentrated between 0.5 and 0.6 zones, indicating that the majority of supporters hold a rational stance on the tariff policy rather than extreme support. The proportion of supporters scoring above 0.7 points dropped sharply, and only 13.2% of the supporters had an emotional intensity close to the peak of 0.9-1.0 points.

The opposition group shows an extremely polarized phenomenon. The frequency between the 0.9-1.0 score ranges is as high as 2021, which is 12.4 times that of the control group. Moreover, the emotional intensity increases exponentially starting from the 0.3 score, indicating that the opposition sentiment has a social dissemination amplification effect.

The emotional intensity of the neutral group is concentrated within the 0.3-0.7 range, reflecting the high uncertainty of the impact on policies. There is a significant differentiation in transformation potential among this group: the group with a score of 0.4-0.5 May turn to support, while the group with a score of 0.5-0.6 is more likely to turn to oppose.

## 6.2 Post-Volume Visualization

### 6.2.1 Geographic Distribution of Post-Volume

To visualize the post-volume data, we first imported the Natural Earth “Admin 0 – Countries” shapefile to provide country boundaries, and overlaid it on basemap (© OpenStreetMap contributors, CC-BY-SA). A graduated green color scale highlights regions by post volume: dark green denotes high activity (Figure 6.2.1), while pale or neutral tones indicate low to zero posts. Many countries remain zero posts, indicating either no local instances harvested or low public activity on the servers.

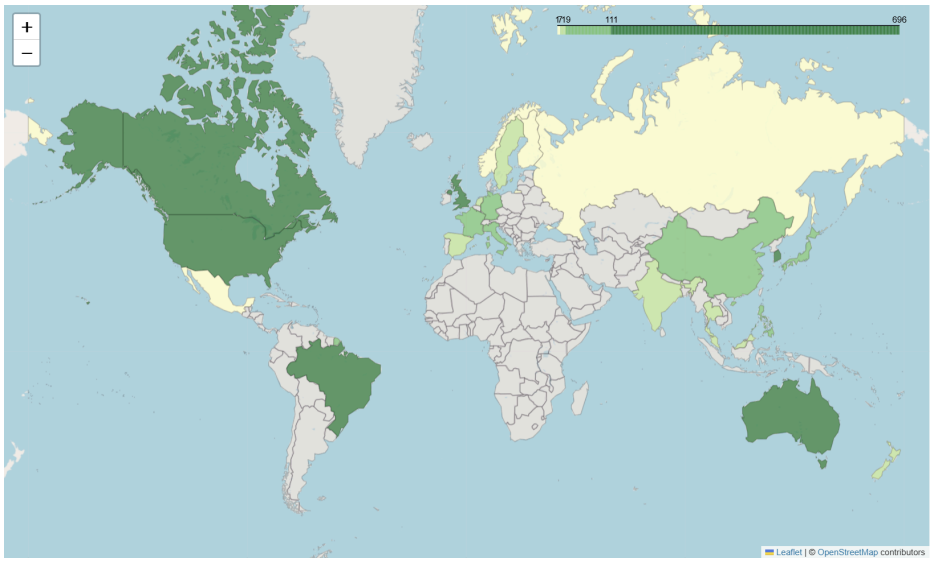



Figure 6.2.1 Choropleth of total posts by region

### 6.2.2 Top10 Region and Domain by Post Counts

We further summarized the top ten regional posts by region (Figure 6.2.2). Australia ranks fifth in the world with approximately 170 posts, second only to the US, UK, South Korea, and Brazil.

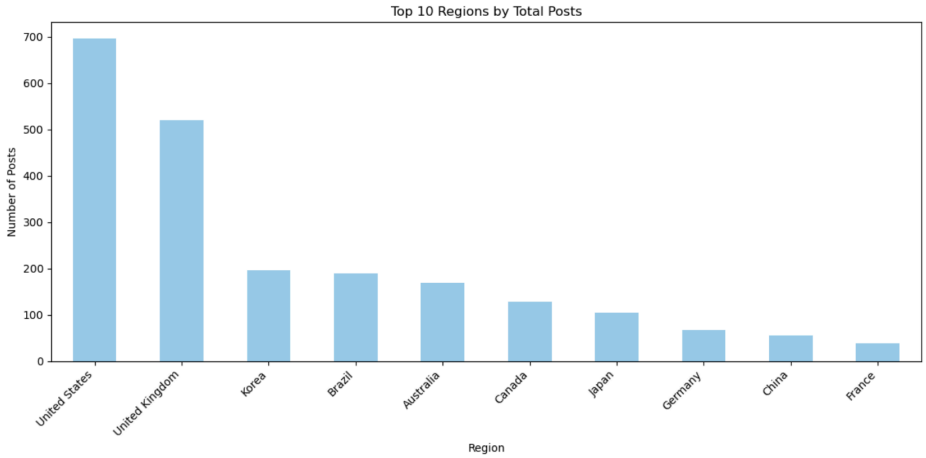



Figure 6.2.2 The top 10 regions by total posts

Then we examined which specific domains contributed most posts (Figure 6.2.3). Among the top top top-level domains, instances related to Australia contribute more than 75% of the total in Australia.

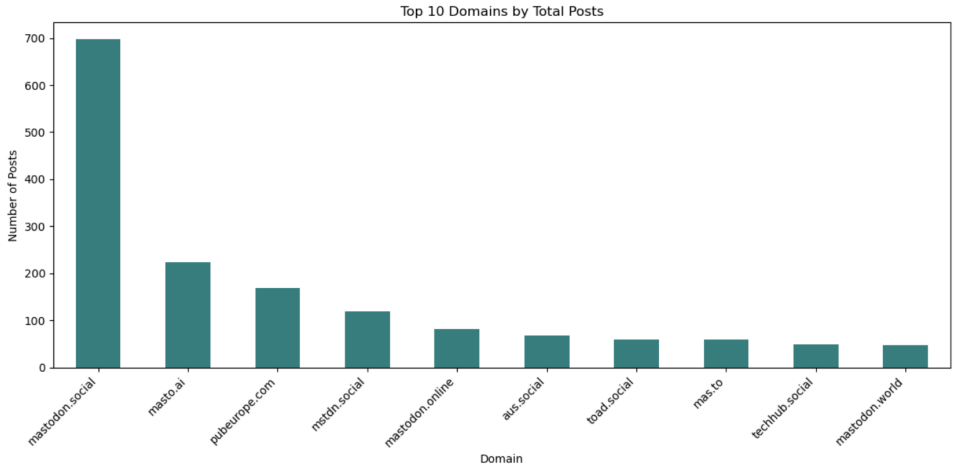



Figure 6.2.3 The top 10 domains by total posts

## 6.3 Interaction-Volume

### 6.3.1 Geographic Distribution of Interaction-Volume

To visualize the interaction-volume data, Figure 6.3.1 shows the global heat map of the total interaction volume of "comments+shares+likes" for each region. Dark green represents high mutual momentum, while light or gray represents low mutual momentum or no interaction.

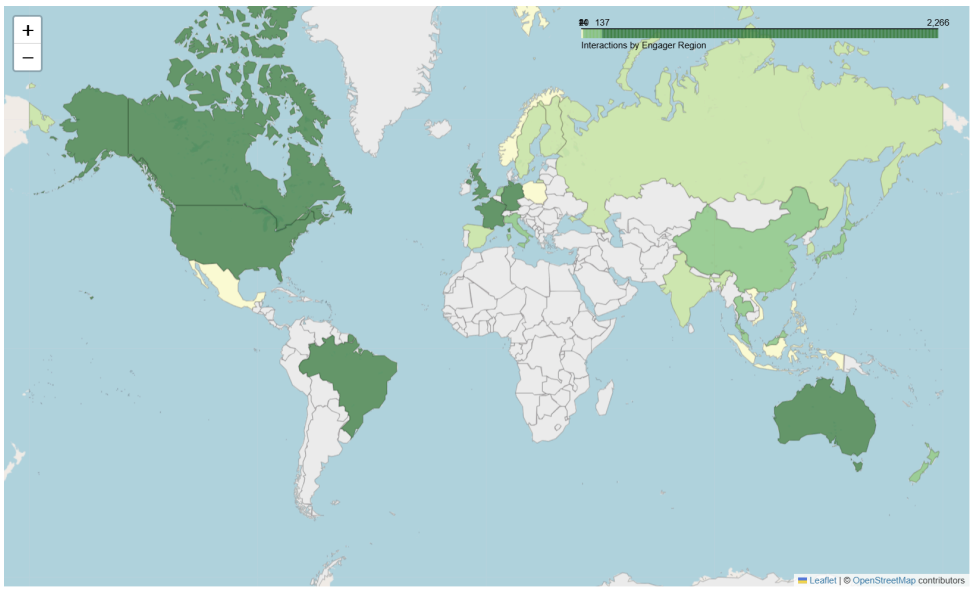



Figure 6.3.1 Choropleth of total interaction by region

6.3.2 Top10 Domain by Interaction Volume

Figure 6.3.2 shows the interaction volume in each region, which intuitively illustrates Australia's relative position. Australia, which ranks fifth in terms of posting volume but third in popularity, has surpassed Brazil, which ranks fourth in postin volume, demonstrating the strong feedback and desire for discussion on tariffs in the Australian community.

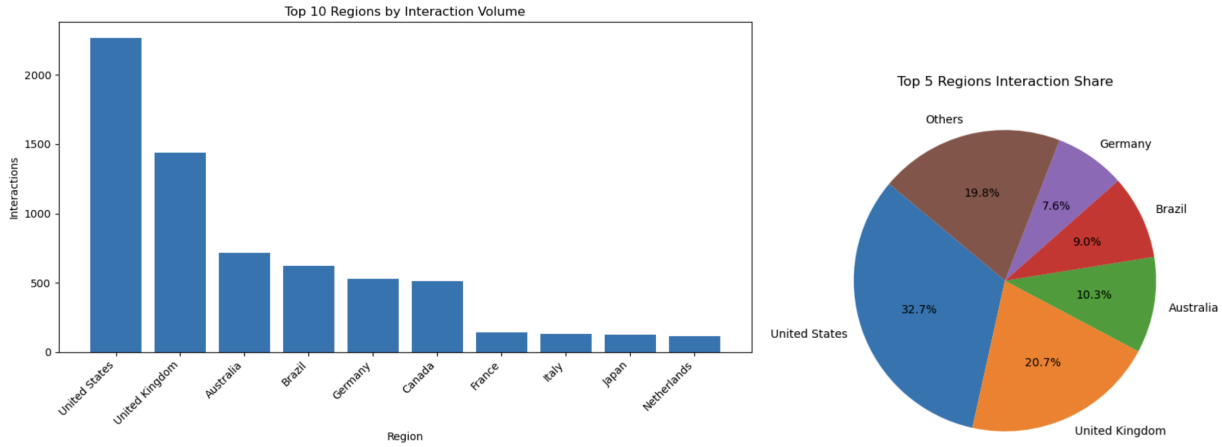



Figure 6.3.2 The top regions by total posts

# 7 Conclusion

In this project, we designed and implemented a cloud native distributed pipeline for collecting, processing, and visualizing public opinions related to tariffs. By utilizing the OpenStack infrastructure and Kubernetes cluster, we integrate Fission's serverless functionality with Redis as a message queue to automate data ingestion, cleaning, and storage in Elasticsearch. On this basis, we used Python-based NLP to perform sentiment analysis and topic extraction, and used Folium to build geospatial maps and dashboards.

Through this project, our team gained practical experience in cloud cluster deployment and Kubernetes orchestration, and elastic scalability. Learned how to use the Fission function and Redis to achieve real-time and reliable data flow. With expertise in Elasticsearch, it has achieved fast indexing, querying, and multi-dimensional search of a large number of social media posts. Practice text analysis to extract user emotions and key policy themes on a global and regional scale. Implemented visualizations, and displayed Australia's top three globally in terms of posting and engagement.

Ultimately, this project not only achieved our goal of real-time data collection and multi-dimensional presentation, but also provided us with practical skills in cloud native architecture.

# 8 Teamwork

To successfully deliver this project, we divide the project into different parts, ranging from data collection to data visualization. Each of us has different responsibilities, as shown as below.

Jiangyi Yang
• Deployment of the whole cluster, including all components such as Fission, Redis and ES
• Design and implement all backend jobs such as fission components and message queue and unit tests.
• Design and implements of the data pipeline and the system architecture
• All codes are placed in the backend folder

DongPing Lou
• Design and implementation of the data collection jobs
• Design and deployment of the tariff scenario
• All codes are placed in the dataCollection folder

Gefei Zhao
• Analysis about the sentiment and opinion scenario
• Cleaning data and send back to ES
• Implement of the sentiment score and sentiment opinion
• All codes are placed in the data_analysis folder

ShanShan Li
• Visualization of posts opinion and sentiment
• Visualization of engagements and posts trend
• Visualization of Keyword distribution
• All codes are placed in the data visualization folder

Yadi Chen
• Visualization of geological posts and engagement distribution
• Visualization of regional posts and engagements count
• Visualization of posts and engagements in different domains

# 9 External Link and resources

Gitlab Link:
https://gitlab.unimelb.edu.au/jiangyiyang/comp90024_team_72/-/tree/main
Youtube Link:
https://www.youtube.com/playlist?list=PLBGy6cO3iY4JJfzsyCUD9kTY9J9CzDUBh

# 10 References

Natural Earth. (2024). *Admin 0 – Countries* [Shapefile]. Version 4.1.0. Public Domain. Retrieved from https://www.naturalearthdata.com