Data Analysis of Pushshift Reddit Comments using Apache Spark

CSC1142 Cloud Technologies

Contribution Table

Student Details	Contribution
Yu Yu Lwin A00053026 yuyu.lwin2@mail.dcu.ie	**Data Extraction & Transformation** • Implemented streaming data extraction pipeline. • Designed and configured Spark session with optimization. • Developed data quality tracking system. • Implemented core data cleaning and validation logic in transformation module • Developed temporal and text feature engineering. • Created 2 analytical aggregations: hourly_activity and sentiment_distribution • Set up logging infrastructure and error handling throughout pipeline
Reeha Althaf A00050820 reeha.althaf2@mail.dcu.ie	**Advanced Analytics & Output Generation** • Developed schema validation and column selection logic in transformation • Implemented advanced feature engineering: engagement metrics, sentiment proxy, controversy indicators, boolean flags • Created 2 analytical aggregations: top_subreddits and engagement metrics • Built a complete persistence layer. • Implemented multi-format output system • Generated 3 data visualizations using matplotlib • Developed main pipeline orchestration.

Link to the Dataset – Hugging Face fddemarco/pushshift-reddit-comments
Link to Gitlab – csc1142-assignment-pushshift-reddit-comments-using-apache-spark
Link to the Video – Cloud Computing Video Presentation_Reeha & Yu Yu

Table of Contents

Introduction and Motivation

There are lots of text data generated on social media platforms every day. Among them, reddit is one of the largest American proprietary social news aggregation and forum social media platforms [1]. Users can write comments and read news on different kinds of topics in thousands of subreddits. It is very interesting to research this data for data analysts, researchers, and corporations because everyone can analyse the trends, ideas, and behaviors for communications. However, the data size is huge, and it is difficult to download and analyse the comments using simple tools like Excel.

The aim of this project's pipeline is designed and developed to analyse and to download raw comments from Reddit dataset, clean it, create a structured dataset and convert it to CSV file for further analysis and reporting. This pipeline is focused on collecting basic and useful number of comments per subreddit per day, the average score, and the average comment length. This will be useful for further analysis to understand which subreddits are more active or which communities receive higher scores.

Designing and developing this pipeline is helpful and useful for both education and practice. It helps to learn how to use modern data tools in real-world processes such as data extraction process, data cleaning, data transformation and data output. Creating this pipeline is not only about using the methods and resources on the system, but also for data engineers to work on real-world projects. It also shows how to connect to real-world projects, cloud-based datasets and develop tools that can analyse them using Jupyter Notebook, or Power BI dashboards, Excel and more.

This project includes three technologies: the Hugging Face library for the reddit comments dataset, Python, and Apache Spark via PySpark. The dataset of Reddit comments used in this project comes from the "**fddemarco/pushshift-reddit-comments**" hosted on Hugging Face. Instead of downloading the full 292 gigabyte file by using these datasets directly from the website by streaming the dataset. Streaming is essential for this pipeline to run the entire process because the full dataset is huge and can demonstrate the pipeline with just a sample of 600,000 comments.

Python is used in this project because of its readability and simplicity of its ecosystem for data work, such as working with pandas for handling DataFrame, creating a virtual environment in local machines. Python libraries allow data analysts to convert numbers into pie charts, graphics, histograms, etc. to make it easier for data analysts to make their data driven visually appealing and comprehensible. This also makes the virtual environment reproducible and doesn't conflict with Python packages. Apache Spark from PySpark is used as a data processing engine due to its widespread usage in the big data processing industry. Spark can be scaled to very large datasets if required, but in this project, run it in local [*] mode on the laptop. Using Spark also demonstrates good practices for data cleaning and filtering invalid values, handling missing data and data aggregation, such as aggregating data by subreddit and date. The final output is converted to CSV files because CSV is a simple and widely supported format that can be opened in many tools, which fits the assignment requirement of producing a dataset suitable for analysis and reporting.

Related work

There are many applications and systems that are already implemented using Apache Spark which is like an approach. Three different related systems / applications are as follows:

Advanced Data Analysis of a Retail Store using Apache Spark (PySpark)

The application in [2] focuses on PySpark and performs structured transactional data analysis and identifies trends in revenue, product sales, and customer behavior. This application loads retail datasets, calculates summary statistics, and visualizes insights using Spark within an environment. As a comparison, this project's solution differs both in the type of data and in the analytical task. The retail project deals with numerical sales transactions, whereas a pipeline processes unstructured data of comments on Reddit and then converts this data into daily aggregated metrics at the subreddit level. Moreover, the retail data analytics application provides data visualisation into it, where this project provides CSV files suitable for viewing and further analysis outside of the system.

Exploratory data analysis of Airbnb data for New York City from inside Airbnb

[3] is also relevant to Spark is Airbnb Exploratory Analysis with PySpark, and it is performing exploratory data analysis on large datasets of Airbnb. It uses SparkSQL, DataFrames, joins and aggregation functions to explore distributions of prices, neighborhood trends, and occupancy. The system provides several interactive visual outputs showing regional variations in both listing behavior and customer preferences. Though both systems use Spark to carry out large-scale transformations, the Airbnb application focuses on structured numerical and categorical attributes related to places, whereas my pipeline deals with user-generated textual data. The solution focuses on temporal aggregation and subreddit-level activity metrics rather than exploratory visual analysis within a project's interface.

YouTube trending videos analytics using PySpark

[4] is the YouTube-Big-data-Analytics-using-Spark, which works on high volumes of metadata to find trends in video categories, creator performance, and regional engagement. ETL operations, data cleaning, grouping, ranking, and time-series trend detection with Spark SQL are done. The system's analytical objectives include identifying multimedia content trends and viewer behavior. The pipeline uses many of these same distributed operations-filtering, grouping, and aggregation but operates on social-media text interactions instead of multimedia metadata. Whereas the analytics system for YouTube finds patterns of popularity in content, the solution provides a summary of comment activity and engagement over time across subreddits.

Description of the dataset

Source of the Data

The dataset used in this project is the Pushshift Reddit Comments Dataset, hosted on HuggingFace at https://huggingface.co/datasets/fddemarco/pushshift-reddit-comments. This dataset contains 1.85 billion Reddit comments spanning from 2005 to 2023, with a total size of 292GB. The original data was systematically collected by the Pushshift project, created by researcher Jason Baumgartner, through continuous polling of Reddit's public API endpoints.

Dataset Schema:

The dataset contains the following key fields for each Reddit comment:

Field	Type	Example
id	string	Unique comment identifier
body	string	Text content of the comment
score	integer	Net upvotes (upvotes - downvotes)
created_utc	integer	Unix timestamp of comment creation
subreddit	string	Community where comment was posted
subreddit_id	string	Unique subreddit identifier
author	string	Reddit username of commenter
controversiality	integer	Flag for controversial comments (0 or 1)

Process of extraction

Implementation Overview:

Our extraction pipeline is implemented in the extract_stream_to_jsonl() function within Cloud Computing Project.py. We employ a streaming approach using the HuggingFace datasets library to process the 292GB dataset without requiring full download.

Streaming Strategy Justification:

The full HuggingFace dataset contains 1.85 billion records totaling 292GB. Streaming extraction offers critical advantages:

1. Processes records incrementally without loading the entire dataset into memory
2. Enables flexible sampling for development and testing without multi-hour downloads
3. Allows immediate pipeline iteration and experimentation
4. Reduces local storage requirements by writing only selected fields to disk.

The pipeline connects to HuggingFace using load_dataset() with streaming=True parameter, which returns an iterator rather than downloading data.

Description of data processing

Main Processing File: Cloud Computing Project.py

Processing Architecture and Data Cleaning

Our data processing pipeline uses Apache Spark's DataFrame API implemented in the clean_transform_and_aggregate() function. The pipeline reads the extracted JSONL file (3 million records by default) into a distributed Spark DataFrame configured with 8GB driver/executor memory, adaptive query execution, and Kryo serialization. Processing begins with spark.read.json() for automatic schema inference, selecting eight columns: id, body, score, created_utc, subreddit, subreddit_id, author, and controversiality.

Data cleaning applies three filters: null body removal using F.col("body").isNotNull(), exclusion of deleted/removed comments via ~F.lower(F.col("body")).isin([F.lit("[deleted]"), F.lit("[removed]")]), and type validation. Type conversion uses safe casting where created_utc is cast to long and score to double, with subsequent null filtering to remove records where casting failed. This approach handles inconsistent source data types without pipeline failure. The DataQualityMetrics class tracks records at each stage: initial count, post-cleaning count, and records removed.

Feature Engineering

Twelve analytical features are engineered from raw data. Text features include comment_length (character count via F.length()) and word_count (word count using F.split() on whitespace and F.size()). Temporal features extract created_ts (human-readable timestamp from Unix epoch using F.from_unixtime()), plus year, month, day, hour via Spark SQL date functions, and day_of_week using F.dayofweek() returning 1-7. Boolean indicators include is_long_comment (1 if length exceeds 500 characters) and is_controversial (derived from controversiality field). Engagement metrics calculate engagement_rate as score divided by comment length plus one, and sentiment_proxyclassifies comments as positive (score > 5), negative (score < -5), or neutral. The analytical DataFrame is repartitioned into 20 partitions by year and month using repartition(20, "year", "month") to optimize partitioned storage and date-filtered queries.

Aggregations and Output

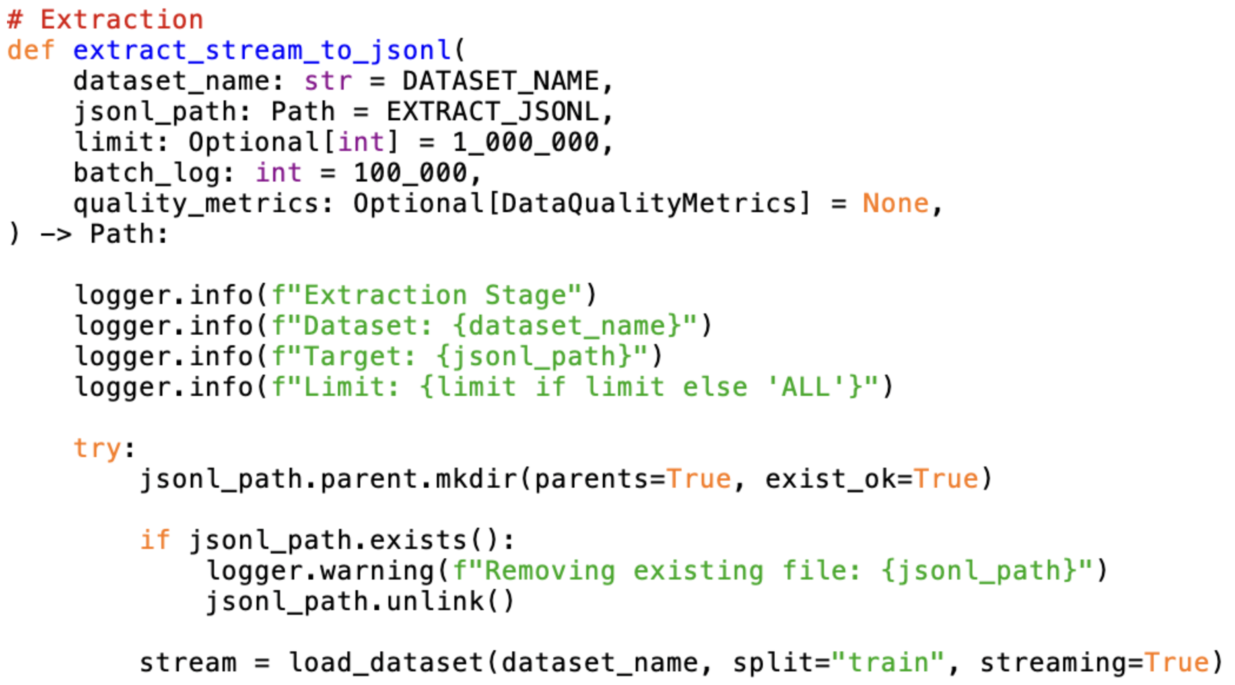
Four analytical aggregations support dashboard visualizations: top_subreddits groups by subreddit with comment counts, average score, average comment length, total score, and average engagement rate; hourly_activity groups by hour (0-23) with comment counts and average scores to reveal daily patterns; sentiment_distribution groups by sentiment classification with counts; engagement groups by subreddit with average engagement rates, controversial comment counts, and average word counts. All aggregations use Spark's distributed functions (F.count(), F.avg(), F.sum()) optimized by the Catalyst query optimizer.

The analytical DataFrame (19 columns) is written to Parquet format partitioned by year/month using .write.partitionBy("year", "month").parquet(), providing 10x compression and enabling partition pruning. Summary aggregations are coalesced to single CSV files per table using .coalesce(1). Dashboard JSON files are generated by converting top 50 results to Pandas DataFrames via .toPandas() and writing with .to_json() for fast dashboard loading. Static visualizations (200 DPI PNG) are generated using Matplotlib: horizontal bar chart for top 20 subreddits, line chart for hourly activity, and pie chart for sentiment distribution. Quality metrics tracking initial/final record counts, features created, and processing duration are saved to quality_reports/pipeline_quality.json with ISO timestamps, enabling reproducibility verification across different sample sizes.

Development of the Pipeline

Data Extraction Tool

Our extraction pipeline uses the HuggingFace datasets library, which provides a unified interface for accessing large-scale datasets with streaming capabilities. The library was chosen for its native support for streaming extraction, which enables processing datasets larger than available memory by fetching records incrementally rather than downloading the entire dataset upfront.



```python
# Extraction
def extract_stream_to_jsonl(
    dataset_name: str = DATASET_NAME,
    jsonl_path: Path = EXTRACT_JSONL,
    limit: Optional[int] = 1_000_000,
    batch_log: int = 100_000,
    quality_metrics: Optional[DataQualityMetrics] = None,
) -> Path:

    logger.info(f"Extraction Stage")
    logger.info(f"Dataset: {dataset_name}")
    logger.info(f"Target: {jsonl_path}")
    logger.info(f"Limit: {limit if limit else 'ALL'}")

    try:
        jsonl_path.parent.mkdir(parents=True, exist_ok=True)

        if jsonl_path.exists():
            logger.warning(f"Removing existing file: {jsonl_path}")
            jsonl_path.unlink()

        stream = load_dataset(dataset_name, split="train", streaming=True)
```

Fig 1: Extraction of the dataset from HuggingFace

Data Processing Tool

Apache Spark was selected as the core processing engine for distributed computation on large-scale Reddit data. Spark provides significant advantages over alternative technologies for this use case.

We used Apache Spark (PySpark DataFrame API) as the processing engine for distributed computation on our reddit dataset. Spark's in-memory computation delivers faster performance than MapReduce for iterative operations. It maintains data in memory across transformations and provides a batch-oriented design that is much easier to implement.

Dashboard to View Results

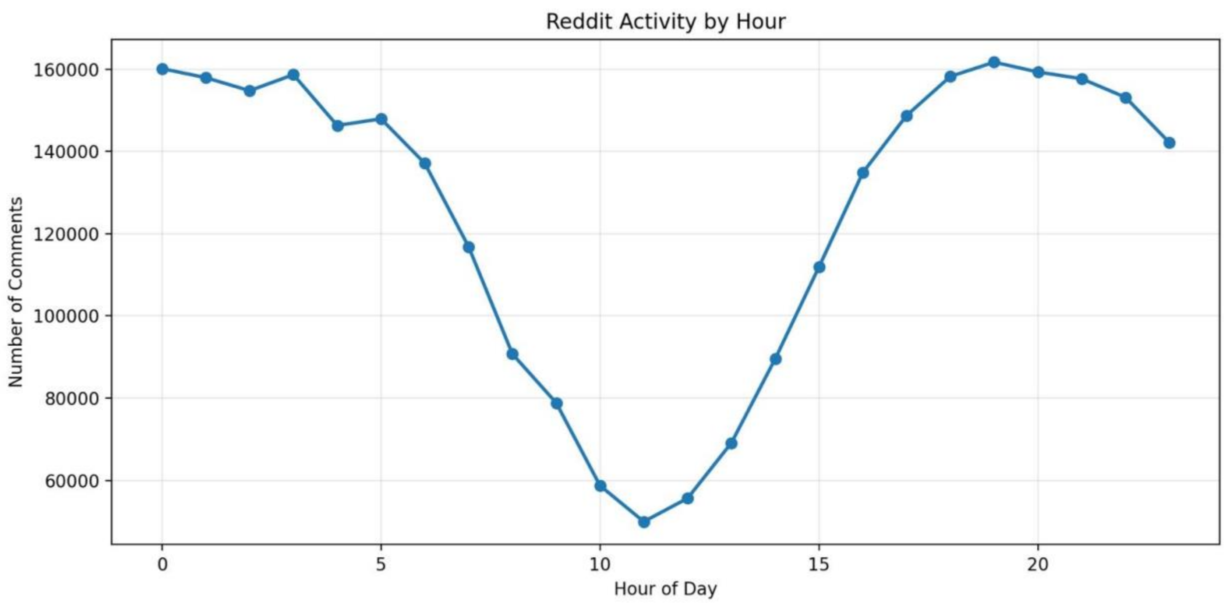


Fig 2: Reddit Activity by Hour

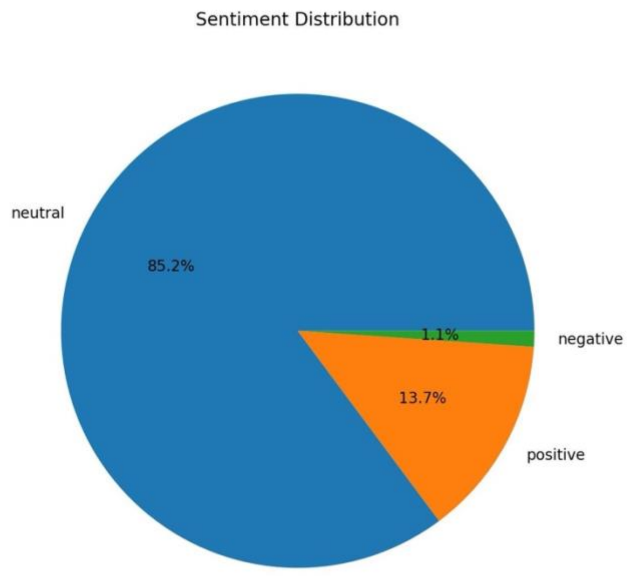


Fig 3: Sentiment distribution

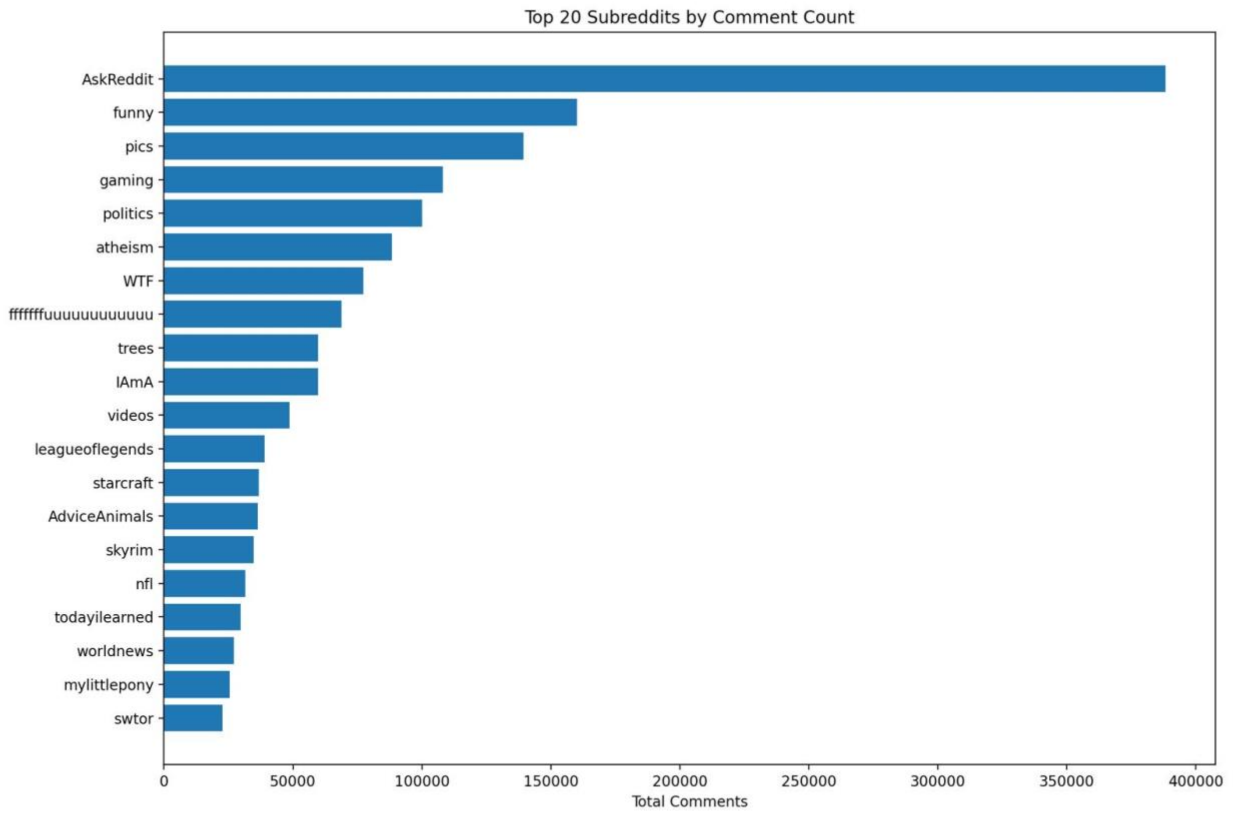


Fig 4: Top 20 Subreddits by Comment Count

Challenges and lessons learned

During the implementation of this project, some challenges are faced related to quality and quantity as dealing with real-world datasets and tools usage. These issues initially slowed, but they also helped us to understand how real-world data pipelines are built and maintained.

1. Environment and dependency issues in local machine
   The first challenge is the Python environment set up correctly. Python, managed by the operating system and Homebrew, is using and pip couldn't use it directly due to an error which is related to an externally managed environment. Some investigation is required because it is required to understand and learn how to set up and activate a virtual environment using the command **python -m venv .venv**. Then, a virtual environment is needed to install the required libraries rather than installing globally. Moreover, the latest version of Python (3.13) did not support this feature, which caused problems.
   Lesson Learned: Required to install stable latest version for Python and virtual environment need to use if needed. Matching the Python version to what the main libraries support is as important as writing code.
2. Operating on large cloud datasets
   The Hugging Face Reddit dataset is huge and cannot be downloaded as a single file easily. It was not evident at first glance how to access it without downloading everything. After doing some research, streaming mode from the dataset's library can be used. The streaming mode enables us to read data one row at a time and sets access restrictions for data, like 3,000,000 comments.

Lesson Learned: Using the streaming method is more efficient handling large datasets, opting to stream processes one record at a time instead of loading all into memory. This will allow adding a limit parameter to the pipeline that makes it easier to test the system and debugging.

References

[1] Wikipedia Contributors, “Reddit,” Wikipedia, Mar. 06, 2019. https://en.wikipedia.org/wiki/Reddit

[2] “Advanced Data Analysis of a Retail Store using Apache Spark (PySpark) — Anindya’s Blog,” Github.io, 2016. https://anindya-saha.github.io/blog/data-science-with-spark/retail-database-analysis-python/retail-database-analysis-python.html

[3] chris-leonard, “GitHub - chris-leonard/airbnb_exploratory_analysis: Exploratory data analysis of Airbnb data for New York City from Inside Airbnb,” GitHub, 2025. https://github.com/chris-leonard/airbnb_exploratory_analysis

[4] saikrishnauppala, “GitHub - saikrishnauppala/Youtube-Big-data-Analytics-using-Spark: YouTube trending videos analytics using PySpark,” GitHub, 2025. https://github.com/saikrishnauppala/Youtube-Big-data-Analytics-using-Spark

[5] F. Demarco, “pushshift-reddit-comments,” Huggingface.co, 2025. https://huggingface.co/datasets/fddemarco/pushshift-reddit-comments