Realtime Socket Streaming on Yelp Dataset

Aishwarya Gulab Thorat, Sheetal Patnaik, Lincy Rebello, Dhruv Shetty

Abstract

Our project proposes a comprehensive data processing pipeline leveraging AWS cloud services for analyzing Yelp business, reviews and tip data. The system integrates AWS S3, AWS Glue, AWS Athena, and AWS Sage Maker to perform sentiment analysis on user-generated content. The pipeline transforms the JSON Data into the paraquat format and employs multiple machine learning approaches including Logistic Regression, and TF IDF Vectorization to extract meaningful business insights through complex window function queries. These highlight the effectiveness of integrating cloud-based ETL processes with machine learning algorithms for large-scale customer feedback analysis. It gives a highly scalable architecture for businesses in need of actionable insights from unstructured review data without losing processing efficiency and data integrity.

Keywords: AWS Cloud Services, Data Processing Pipeline, Sentiment Analysis, ETL, JSON to Parquet Transformation, Machine Learning, AWS S3, AWS Glue, AWS Athena, AWS SageMaker, BERT, TF-IDF Vectorization, Logistic Regression, Window Functions, Business Analytics, Streaming Algorithms, PySpark, Unstructured Data Processing.

1. INTRODUCTION

The exponential growth of user-generated reviews on platforms like Yelp has created unprecedented opportunities for businesses to understand customer sentiment and behavior patterns. However, this also poses a lot of problems as we are processing and analysing large-scale unstructured data. The project addresses these challenges with a scalable and hybrid cloud-based architecture, by efficiently using ETL pipeline with AWS Glue to transform raw JSON review data into Parquet format. It starts with raw JSON files in AWS S3 bucket which has business reviews and tips of data. The methodology enables SQL analysis via AWS Athena and sentiment analysis through AWS SageMaker. The system is optimal in data processing and storage and the same is achieved by integrating traditional machine learning models like Logistic Regression with TF-IDF and optimizing with methods such as Locality Sensitive Hashing and Bloom Filters. The analysis is done with window functions that helped us provide valuable insights. Our architecture implements an efficient method for data processing and storage. The projects architecture will combine both cloud computing and machine learning models for efficient processing and analyzing customer reviews.

2. MOTIVATION

The primary motivations for this project are

- Addressing the growing need for Real-Time data processing and Analysis in today's data-driven business environment.
- To demonstrate the practical application of Cutting Edge Technologies in building a scalable and efficient data pipeline.
- To explore the potential of integrating artificial intelligence specifically large language models into the data streaming workflows for advice analytics like sentiment analysis
- To provide a comprehensive solution that covers data in ingestion, processing, analysis, distribution and Storage which addresses the full life cycle of data streaming.
- Also to create a flexible and extensible architecture that can be adapted to various use cases and industries requiring real time Data Insights.

3. LITERATURE REVIEW

Throughout the project's investigation, we've thoroughly reviewed a number of relevant studies and research articles in the area.

[1] This research reveals That spark streaming enables scalable, High throughput, fault tolerant stream processing of live data streams. however it can cause latency issues due to its micro batching model which may not be suitable for applications which require millisecond level responsiveness. our project aims to address this limitation by exploring optimizations and alternative

approaches to reduce latency while maintaining fall tolerant.

[2] In this study it highlights that there are advancements in LLM's which provides substantial improvements for doing sentiment analysis the integration with real time streaming Frameworks Still Remains an area that requires for the exploration. To address this in our project we aimed to bridge the gap by seamlessly incorporating llm based sentiment analysis into a high performance streaming pipeline

[3] In this study they gave more importance to search and analytics capabilities but they did not focus on the integration challenges which can occur when using elastic search for Real Time data streaming. we intend to address this by developing efficient model for Real Time indexing and querying of streaming data on AWS.

[4]This project highlights Kafka's capability for managing large-scale, high-throughput, fault-tolerant streams in distributed systems, making it relevant for applications like our project. But in the paper they haven't incorporated Apache Spark for processing and AWS tools for storage, analysis, and visualization which we did do in our project.

[5] The paper is pointing out on Kafka's architecture for distributed messaging and fault-tolerant data streams. But here there is no mention of advanced streaming algorithms which we have incorporated in our project like reservoir sampling and bloom filter.

[6] AWS Glue Best Practices states the key strategies on how to build secure and reliable data pipelines, emphasizing data encryption, access control, and error handling. Our project will utilize these best practices in order to further scale and robustious batch data processing. In this way, compliance with security standards meets with preserving operational efficiency in large-scale data workflows. .

[7] This paper focuses on providing high-level APIs for developers to process and analyze large-scale data streams in real-time.But in our project we tried to incorporate advanced techniques such as sentiment analysis, DGIM, LSH, and explainable AI for enhanced data processing, analysis, and visualization.

[8] AWS Glue Best Practices states the key strategies on how to build secure and reliable data pipelines, emphasizing data encryption, access control, and error handling. It also emphasizes schema enforcement and Glue workflows, ensuring data integrity.

4. SYSTEM ARCHITECTURE AND METHODOLOGY

Our project went through two major architectural changes due to various implementation challenges and optimization. The details of both are as follows:

There were several critical factors and technical challenges that led to transitioning our initial streaming architecture onto an AWS-based solution. Though our initial architecture looked theoretically valid, there were some practical obstacles to its viability.

4.1 Initial Architecture

In this approach we had planned and built an end-to-end data engineering pipeline shown below in the architecture using TCP/IP Socket, Apache Spark, OpenAI LLM, Kafka and Elasticsearch

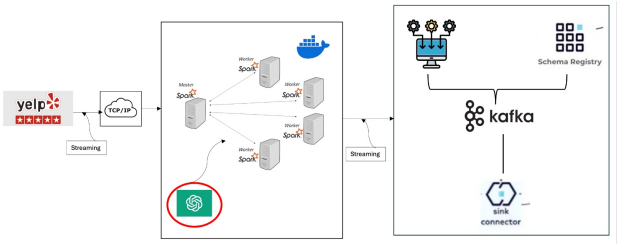


Figure 1: Data Processing Architecture

1. We started working on this. At first we got an API key for the yelp reviews. We started streaming the data from this API into our network in chunks.
2. But we noticed that the data incoming was quite less and hence we started searching for more datasets. Then we came across this data from yelp.com and started streaming data from there.
3. **Spark Master-Worker Architecture with Docker:** We have successfully executed a Docker-based Apache Spark setup with a master and worker nodes. This has allowed us to deploy a distributed data processing system.

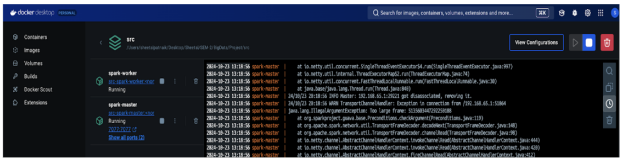


Figure 2: Docker Desktop

4. Sockets were the entry point for raw data for us. It allowed us to capture the data in real-time.We used spark to process the streaming data.To implement this we created

python files as streaming-socket.py and spark-streaming.py. These jobs streamed data into our local machine.

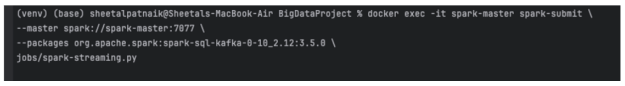


Figure 3: Job Streaming

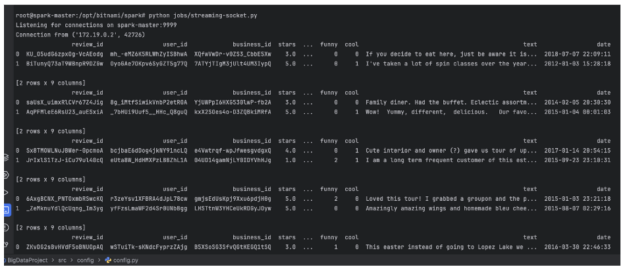


Figure 4: Job Streaming 2

5. Now we used kafka which acts as a buffer and manages the flow of the real time data. To set this up we used a kafka cluster on confluent cloud. Kafka acts as a message broker which ensures that the data is ingested and is processed in a smoothly.
6. We created a enviornment in confluent cloud console dashboard first and then created a cluster in it. We choose GCP as cloud provider and the selected region and availability. After setting up all the configurations correct we launched the cluster.
7. After the cluster was created, we noted down all the important information like the broker endpoints,cluster API keys, etc.
8. Next we created a topic and also set up a schema in the schema registry to ensure data consistency and compatibility is met.

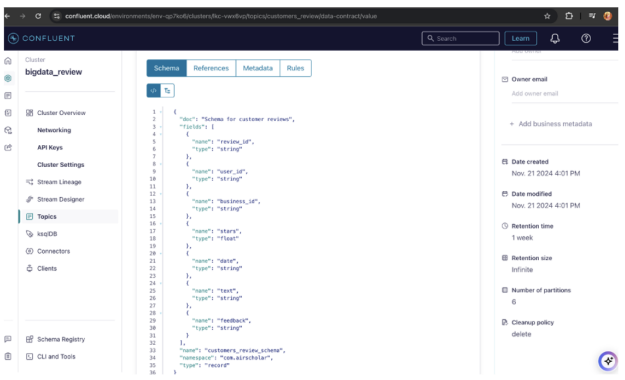


Figure 5: Confluent Topic

9. Once the cluster was set up properly and the API keys were generated we connected our application,producers, and consumers to our Kafka cluster using the provided broker endpoints and credentials.
10. Next we set up spark in spark-streaming.py to consume data from the streaming-socket and stream it to kafka. This was successful and we could see the data streaming into our topic in confluent cloud.

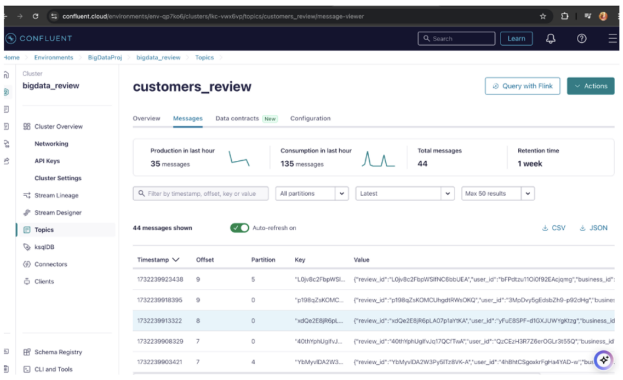


Figure 6: Streaming Data in Confluent Cloud

11. 1 In order to integrate OPEN AI's API for sentiment analysis, we purchased a API key and integrated it in our config.py.

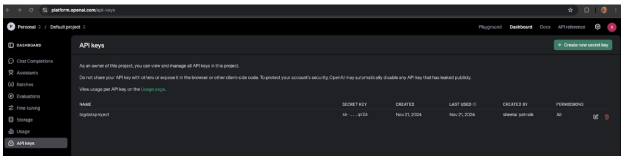


Figure 7: Open AI's API Key

Upon integrating this API for sentiment analysis, the expected result wasn't met. The expected result was that we receive the messages along with the feedback if the sentiment of the customer was positive, negative or neutral. But since there was an issue with the API the output was just showing ERROR.

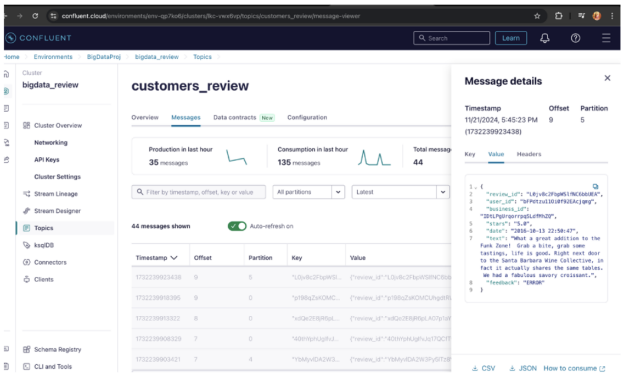


Figure 8: Open AI's API Feedback Error

Multiple attempts were made with the OPEN AI's API but received the same error message as feedback. As every API we used had a cost, so decided to look for other options in order to get the expected output.

5. REVISED AWS-BASED ARCHITECTURE

The decision to migrate to an AWS-based architecture was because of several compelling reasons. AWS's managed Services eliminated many of the operation complexities we encountered in an initial architecture. The new architecture leverages AWS S3 for robust data storage, AWS glue for efficient ETL processes, and AWS SageMaker for advanced Analytics, creating a more cohesive and manageable system.

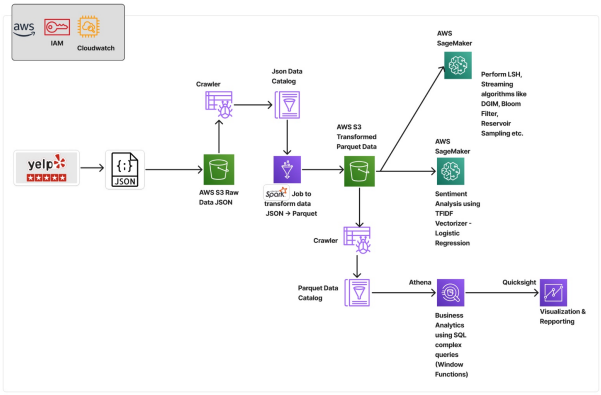


Figure 9: Revised AWS Architecture

1. We streamed our data from a Confluent topic to an Amazon S3 bucket using a sink connector.
2. Next, we connected our Amazon s3 with AWS Glue. This managed the main ETL service which is extracting the raw data from the s3 bucket, transforming it and again loading it back to the s3 bucket. So here we transformed our raw data which was in JSON format into a structured format which is parquet files.
3. In the s3 bucket, we have 5 large data files in JSON format. Upon examining we decided to focus mainly on business data, reviews data, and tips data.
4. Now in order to access data from the s3 bucket to GLUE we used AWS Crawler. This crawler infers the schema and creates or updates metadata in the **AWS Glue Data Catalog**.
5. Figure.10. shows the GLUE scripts we wrote in order to transform the data into clean parquet files and store them back in the S3 bucket in the output folder.
6. Then upon running the above scripts we transformed the data and loaded it back to the s3 bucket.

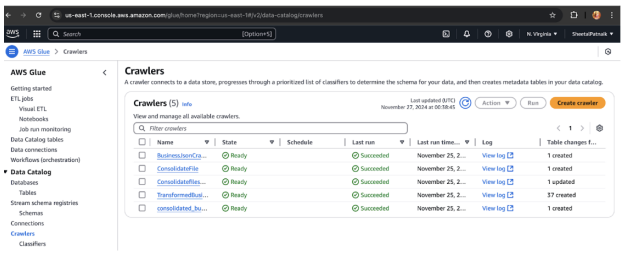


Figure 10: AWS Crawlers

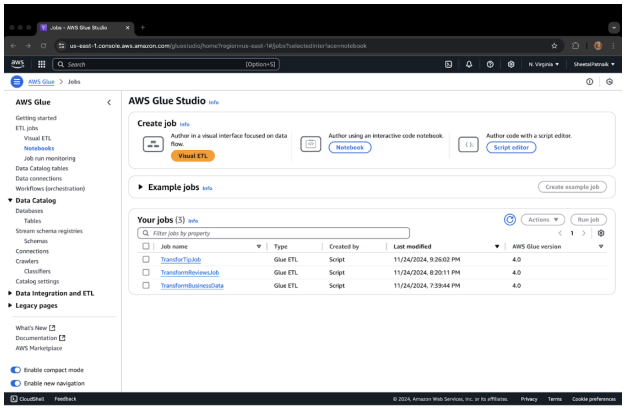


Figure 11: AWS Glue Studio with three transformation scripts

7. Next taking these consolidated. parquet files we did analysis on AWS Athena by taking the data directly from s3. Here we created a database and 3 tables for our parquet files and then ran complex queries on them. Athena was highly efficient for querying large datasets.
8. Figure 12. and Figure 13. shows a few queries that we ran along with their results in Athena.

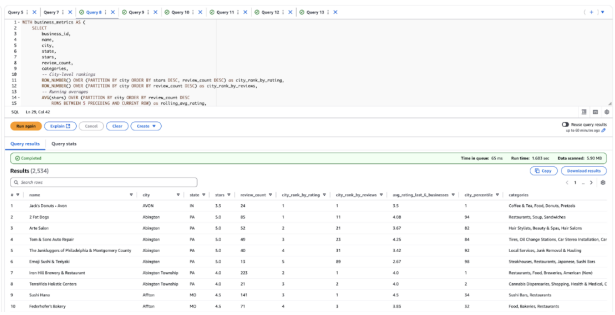


Figure 12: Query Analysis using AWS Athena

Figure.12. shows query :

- Top-rated businesses in each city
- Their ranking by both ratings and review count
- Rolling average of ratings
- Percentile ranking within their city

Figure.13. shows query analysis:

- Direct competitors in same category/city

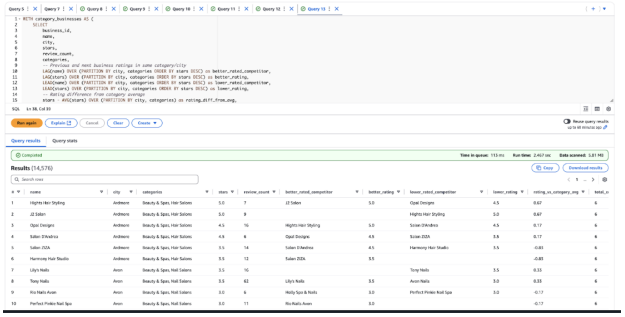


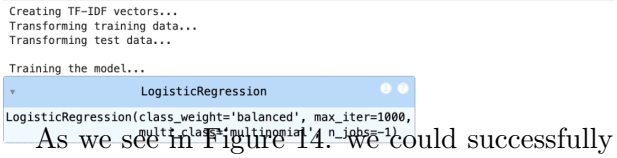
Figure 13: Query Analysis using AWS Athena

- Rating comparison with competitors
- Market position and total competition

Performance vs category average

9. Next in order to perform sentiment analysis we took the transformed parquet data from our s3 bucket to AWS SageMaker. SageMaker service from AWS enabled us to build and train TFIDF vectorizer and logistic regression model for sentiment analysis.

Figure 14: AWS S3 Connection to access Parquet Files



As we see in Figure 14. we could successfully do the sentiment analysis using logistic regression and got an accuracy of 79 percent.

reviews_df.head(20)

	text	stars	sentiment
0	If you decide to eat here, just be aware it is...	3.0	Neutral
1	I've taken a lot of spin classes over the year...	5.0	Positive
2	Family diner. Had the buffet. Eclectic assortm...	3.0	Neutral
3	Wow! Yummy, different, delicious. Our favo...	5.0	Positive
4	Cute interior and owner (?) gave us tour of up...	4.0	Positive
5	I am a long term frequent customer of this est...	1.0	Negative
6	Loved this tour! I grabbed a groupon and the p...	5.0	Positive
7	Amazingly amazing wings and homemade bleu chee...	5.0	Positive
8	This easter instead of going to Lopez Lake we ...	3.0	Neutral
9	Had a party of 6 here for hibachi. Our waitres...	3.0	Neutral

Figure 16: Sentiment Analysis

Figure 16 shows that the sentiment analysis is getting properly done on our data by giving output as positive, negative, or neutral as the sentiment.

5.1 VISUALZSATION:

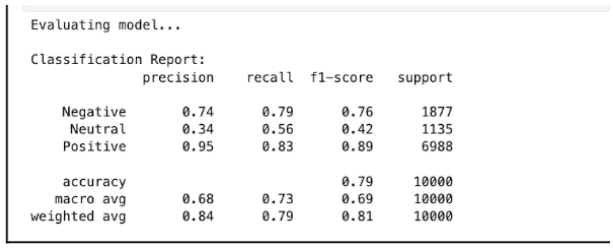


Figure 15: Model Evulation

In order to visualize and get reports from our analyzed data from AWS Athena we did visualisation on AWS Quicksight. The following is one of the visualizations that we did in our quicksight

Figure 17. shows shops that Philadelphia dominates with over 130K reviews, followed by New Orleans with about 60K. There's a steep drop-off after the top cities, with smaller cities having fewer than 10K reviews.

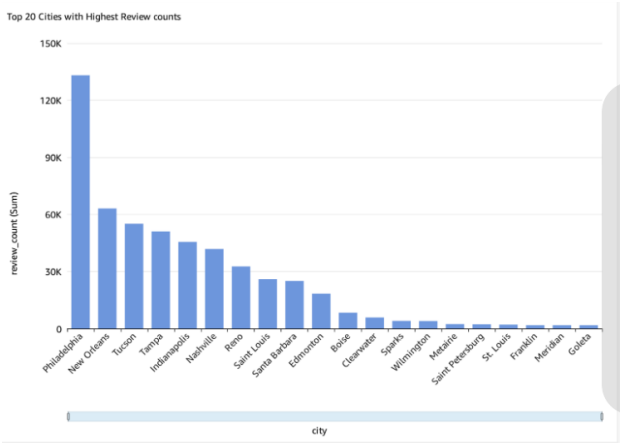


Figure 17: Top cities with Highest Review Count

Figure 18. shows that cafes and restaurants have the highest review counts (around 70), followed by Diners and Cinema categories (around 50). However, all categories maintain similar star ratings between 3-4 stars, regardless of review volume.

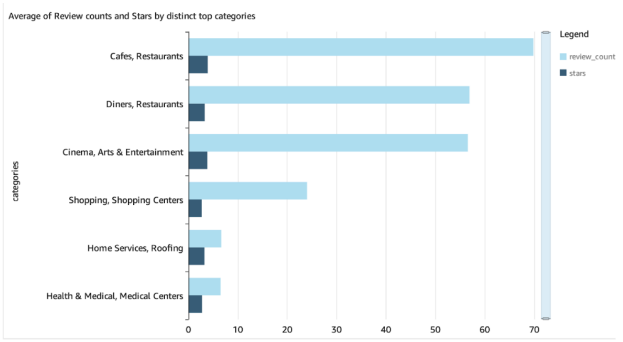


Figure 18: Avg of Reviews and Counts

Figure 19 shows restaurants dominate market competition, with Pizza, Chinese, and Mexican cuisines having the highest number of competi-

tors. Beauty services (nail and hair salons) form the second most competitive sector. The average number of competitors across all categories is 21.03 businesses.

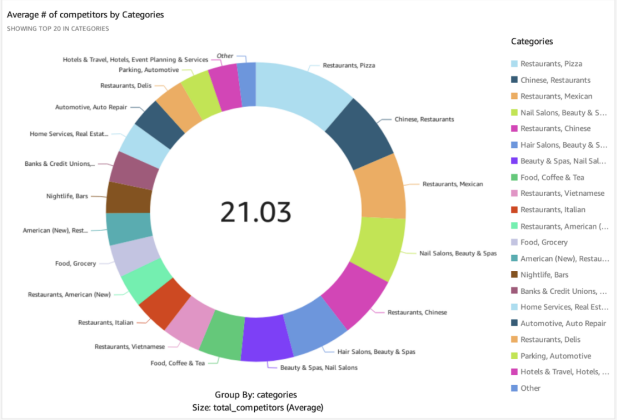


Figure 19: Avg competitors by Categories

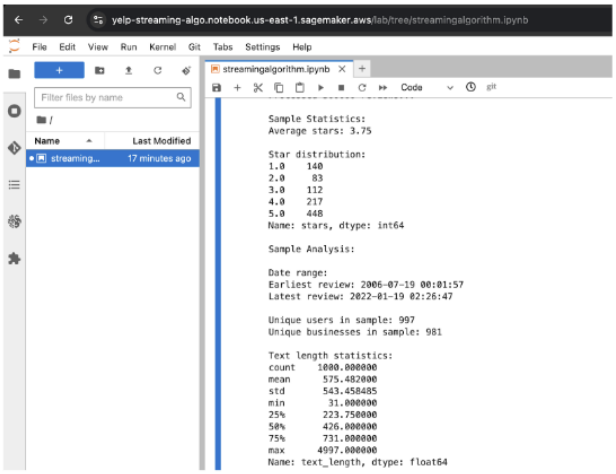


Figure 20: Reservoir Sampling Results

5.2 Streaming Algorithms: After analyzing our transformed parquet files in AWS Athena, we implemented a few streaming algorithms on our data by leveraging the AWS SageMaker service. Here we took the athena results which were stored in our s3 bucket and connected to AWS SageMaker.

In our project, we implemented reservoir sampling to efficiently process and sample from a large Yelp business dataset stored in AWS. Our algorithm fetches data in chunks of 1000 businesses at a time from a pool of up to 500,000 businesses, maintaining a reservoir of fixed size 1000. For each business in every chunk, the algorithm decides whether to include it in the reservoir based on random probability. Basically, each business has a fair chance of being in our final sample regardless of when it appears in the stream. The result is a statistically representative sample of 1000 businesses drawn from a much larger dataset, which we then used for following analysis:

1. Finds the earliest and latest review dates. 2. Counts how many unique users and businesses are in the sample. 3. Measures the length of each review to understand its distribution. **Visualization:** 1. Star rating distribution. 2. Length of review text 3. Review activity over time

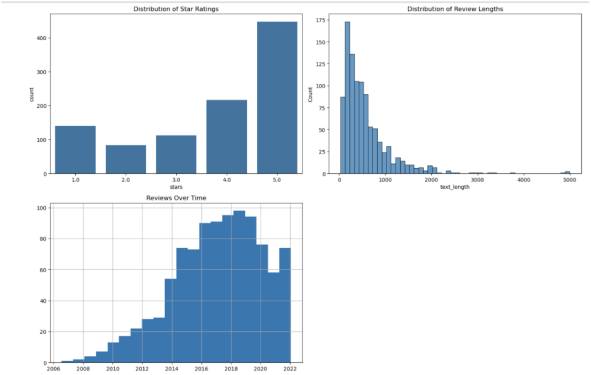


Figure 21: Reservoir Sampling

Next we used Bloom filter to detect potential duplicate businesses in our large Yelp dataset. Our implementation processed 150,346 businesses using a bit array of 1,441,075 bits and 6 hash functions to create unique fingerprints of each business based on their name and location coordinates. The Bloom filter successfully identified 551 potential duplicate businesses, with the highest concentration in Philadelphia (77 duplicates), followed by Tucson and New Orleans (32 each).

```
Starting duplicate business detection...
Fetching business data...
Loaded 150346 businesses
Initialized Bloom Filter with:
Size: 1441075 bits
Hash functions: 6

Checking for similar businesses...
100%|██████████| 150346/150346 [00:09<00:00, 15521.17it/s]

Found 551 potential duplicate businesses

Sample of potential duplicates:
              business_id                                name  \
0  7sdXYWXJMN3rXMSV8jjXRA  The Eye Institute Of West Florida
1  kg8N1PNJM5cJmSkafUZ0Ug             Eden Korean Restaurant
2  IxKNYZCTMRPnOcb2QLUmLg                      Jack In the Box
3  w8Bmb4xGD0mnv5o5GRv6_w                       Crust N' Fire
4  4xaiGq1Skr9JKImzJ8Nypg           American Legion Post 273

                 address          city state  \
0          1225 W Bay Dr         Largo    FL
1    1428 Marlton Pike E    Cherry Hill    NJ
2        1780 Gilsinn Ln        Fenton    MO
3            175 Rte 70       Medford    NJ
4  600 American Legion Dr  Madeira Beach    FL

                                      categories
0  Optometrists, Ophthalmologists, Health & Medic...
1                            Korean, Restaurants
2                          Burgers, Restaurants
3                            Pizza, Restaurants
4      Local Services, Community Service/Non-Profit

Duplicate distribution by city:
Philadelphia    77
Tucson          32
New Orleans     32
Tampa           28
Reno            25
Name: city, dtype: int64
```

Figure 22: Bloom Filter

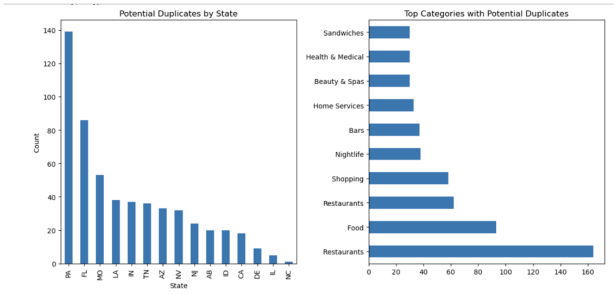


Figure 23: Output of Bloom Filter

Next we implemented the DGIM algorithm.By using this technique we connected our athena results of reviews data from s3 bucket to sagemaker and then monitored the streaming review counts over the time windows.

The following are the results of it.

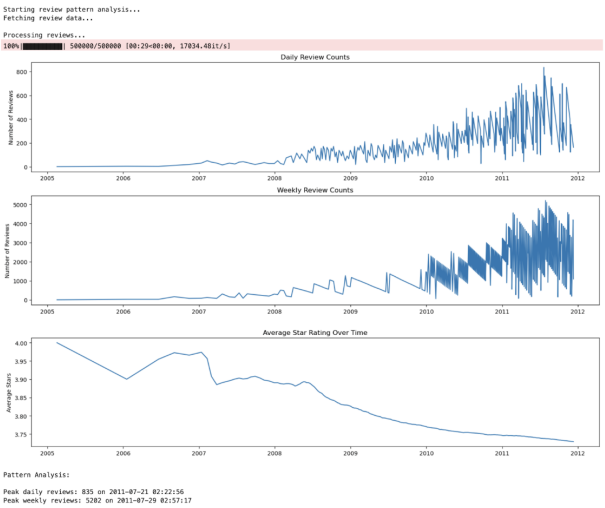


Figure 24: DGIM Algorithm

Use of Stream Processing Frameworks such as Spark and Kafka

In our project we have used spark and kafka in both our approaches.

In the first approach we have used Spark Streaming to process data from Yelp in real-time. Then the data was streamed through Confluent Cloud (Kafka) which acts as a message broker. Therefore here the streaming architecture enables continuous data flow from Yelp → Kafka → Spark for processing. Now in the second approach on AWS we used AWS Glue with PySpark to clean and transform the JSON data to parquet format.

5.2.1Use of Locality Sensitive Hashing We used LSH to find similar reviews in our Yelp dataset by:

- Processed 10,000 random reviews
- Converted review text into TF-IDF vectors
- Used MinHash and LSH to efficiently find similar text pairs
- Set a similarity threshold = 70%

Results:

- Found 1,000 pairs of highly similar reviews
- Top matches had similarity scores = 0.717

Successfully identified nearly identical reviews like the carrot cake example where two customers gave the same review with similar words but gave different ratings (4.0 and 5.0 stars).We could detect similar reviews in different languages (as shown in the Spanish reviews example)

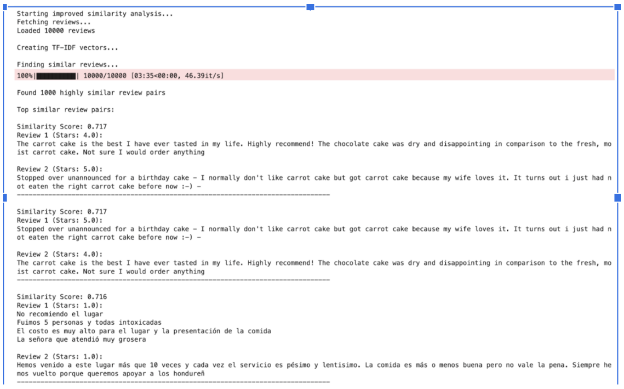


Figure 25: Local Sensitive Hashing

```
Similarity Analysis:
Average similarity score: 0.328
Star rating correlation: 0.247

Distribution of similarity scores:
count    52134.000000
mean         0.327930
std          0.028455
min          0.300002
25%          0.308110
50%          0.319373
75%          0.338284
max          0.716933
Name: similarity_score, dtype: float64
```

Figure 26: Similarity Analysis

5.2.2 Use of Privacy Techniques

1. K-Anonymity

To implement the privacy technique we have used K-anonymity with k=5 to protect user privacy in our Yelp dataset. We implement this in AWS sagemaker and here we have processed 50,000 records, and applied anonymization technique by:

- Reducing location precision (rounding latitude/longitude)
- Converting exact dates to month-year format
- Grouping star ratings into categories (1-2, 3, and 4-5 stars)
- Ensuring each group had at least 5 similar records

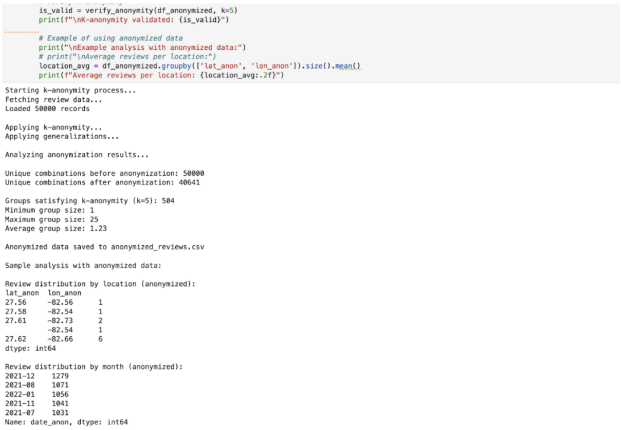


Figure 27: Privacy Technique

```
Review distribution by month (anonymized):
2021-12    1279
2021-08    1071
2022-01    1056
2021-11    1041
2021-07    1031
Name: date_anon, dtype: int64

Rating distribution (anonymized):
4-5    32710
1-2    12578
3       4712
Name: stars_anon, dtype: int64

K-anonymity verification (k=5):
Total groups: 40641
Groups violating k-anonymity: 40137

K-anonymity validated: False

Example analysis with anonymized data:
Average reviews per location: 10.70
```

Figure 28: Distribution

Figure 23. shows that we have got unique combinations reduced from 50,000 to 40,641 after anonymization which means that 504 groups were satisfied k-anonymity.

2. **Differential Privacy:** We added Differential Privacy to make our Yelp data even more private and secure. We added small random changes to business locations and review ratings. For example, slightly adjusted the location coordinates (by about 0.02 degrees) so you can't pinpoint exact business locations, and we modified the rating counts while keeping the overall patterns similar.

```
Starting Differential Privacy process...
Applying Differential Privacy...
Privatizing location data...
Privatizing rating statistics...

Privacy Analysis:
Average location perturbation: 0.020028 degrees

Generating summary statistics...

Analysis with Differentially Private Data:

Private Review Distribution by Location (Top 5):
lat_priv   lon_priv
27.526370  -82.530571    1
27.545127  -82.544640    1
27.553841  -82.584167    1
27.563987  -82.660095    1
27.595123  -82.694907    1
dtype: int64

Private Rating Distribution:
5.0    23268.0
4.0     9440.0
1.0     8828.0
3.0     4713.0
2.0     3746.0
dtype: float64

Utility Analysis:
Average rating distribution difference: 0.0000
Location distribution difference: 45328 unique locations
```

Figure 29: Differential Privacy

Our results showed that even after these privacy changes, the data remained useful for analysis - the rating patterns stayed almost the same, but individual business and reviewer information became more protected. We combined this with our earlier K-anonymity approach and created two layers of privacy protection while keeping the data valuable for research.

5.3 Any other tools and techniques covered in the course not included in the other criteria

5.3.1 Explainable AI: So here in our project we have implemented Explainable AI using the LIME (Local Interpretable Model-agnostic Explanations) framework in order to understand how our sentiment analysis model is making the predictions. So this helped us identify the words and phrases in Yelp reviews which have the most influence on the model's sentiment classifications. For each review, we generated detailed feature explanations that showed the positive or negative contribution. **LIME -** To show how individual words influence predictions **Feature contribution analysis -** To show how different words affect each sentiment class

Figure 30: Explainable AI

5.3.2 Machine Unlearning: Here our model forgets specific reviews or patterns when needed, without having to retrain the entire model from scratch. We do this by finding important words in the reviews we want to forget and making them less influential in the model's decisions. When we tested this, it worked well - for example, after making the model forget a negative review, it became less sure about similar negative reviews (its confidence dropped from about 72% to 57%), but it still kept its basic ability to understand review sentiments. It is like the model became more cautious about making strong judgments about reviews similar to the ones it was told to forget.

Figure 31: Machine Unlearning

6. KEY LEARNINGS

6.1 Phase 1: Initial Architecture

Tools and Workflow: Socket Streaming: Produce data ingested as a stream of customer reviews in real time.

Kafka: Utilized Kafka, a message broker capable of handling large-size streams efficiently.

Spark Streaming: Ingestion of real-time data from Kafka into Spark Streaming for its processing.

6.1.1 Key Learnings: Phase 1

Real-Time Data Processing: Consequently, understood the intricacies in the setup of a real-time data pipeline for dynamic analytics.Gained working experience with high-velocity data using Spark Streaming.Understood the importance of message brokers like Kafka for data flow.

Sentiment Analysis: Learned to apply OpenAI GPT models for sentiment classification by using the API considering the tradeoffs of latency versus accuracy.

6.2 Phase 2: AWS Architecture

Batch Processing: After exploration into this real-time pipeline, the focus shifted onto a batch processing approach due to its higher scalability and cost efficiency. The new architecture thereafter used the very same Yelp dataset but this time availed many AWS services.

Tools and Workflow:

AWS S3: Business, reviews, and tips raw JSON data files would be kept in AWS S3.

AWS Glue :then converts the JSON data into a well-formatted, analysis-ready Parquet format for storing and querying.

AWS Athena: directly uses Parquet files from S3 to apply advanced SQL in implementing Window Functions for analytics.

AWS Sagemaker: logistic regression with TF-IDF vectorization is performed on the reviews data to classify sentiment. BERT was tried, but it needed GPU support.

AWS QuickSight: In order to get the visualizations and report we utilised the AWS Quicksight and generated report from the transformed tables from Athena.

6.2.1 Key Learnings: Phase 2

Scalable Batch Processing: Worked with large-scale data using PySpark in AWS Glue.Converted data into Parquet format to save on the storage cost and maintain/improve query velocities.

Advanced Querying and Analytics: Accomplished complicated SQL analysis through Athena for deriving insightful data that contained window functions.Improved query optimization and query based on schema.

Efficient Sentiment Analysis: TF-IDF with Logistic Regression: How to Efficiently Perform Sentiment Analysis using SageMaker. Recognized computational trade-offs while playing with more advanced models, such as BERT.

Cost Efficiency: It thus provided periodic data processing at much cheaper rates than dealing with real-time architecture. Integration and Modularity: Integrated AWS services into a modular architecture by ensuring maintainability and scalability.

7. TECHNICAL DIFFICULTIES

1. Cost Management

Services like SageMaker, which are always-on systems were extremely expensive. Hence, we had to use these resources in a limited and cost-optimized manner.

2. Schema Evolution

The problem of integration with the downstream tools, like AWS Athena were caused due to the dynamic nature of the data schema.

3. Learning Curve

There was a Huge Learning Curve involved in the integration of Kafka, Spark, the OpenAI API, AWS Glue and SageMaker.

4. Scalability

Scalability became difficult to handle, especially at the real-time streaming architecture. High-velocity ingestion of data stressed the infrastructure to cause latency and throughput issues.

5. Data Quality and Transformation:

The raw JSON handling had lots of problems regarding data quality and complexity because of the very large dataset, which was deeply nested, requiring advanced transformation techniques.

8. BEST PRACTICES

1. Utilize Scalable Cloud Architectures

It is extremely important to have scalable architectures that can adapt to the volume of data. In this project, scalability is attained by leveraging AWS Cloud Services.

2. Optimize Data Storing Format

Due to the form it takes, the storage of data in Parquet format is beneficial. This has been effectively shown by using Parquet over JSON files when we look at the efficiency and performance in AWS Athena queries.

3. Leverage Schema Management Tools

Well-defined, consistent schemas are crucial for the compatibility of different tools using them. Keeping the schema metadata consistent,

using the AWS Glue Data Catalog was very useful.

4. Balancing Real-Time vs. Batch Processing

It is important to choose the right processing pipeline depending on the application's needs. In this project, the batch architecture was more cost-effective and scalable, considering the use case.

5. Apply modularity to architecture

This ensures maintainability and flexibility because the pipeline will be divided into more modules for ingestion, transformation, storage and analysis. Each of these can be independently developed, tested, and scaled.

8.1 Innovation

1. Integration of Real-Time and Batch Processing Paradigms

The project works on the hybrid architecture defined below: Real-time processing is operated with Kafka, Spark Streaming, OpenAI APIs Batch processing uses AWS Glue, Athena, SageMaker pipelines, which are flexible, adapting to different user needs.

2. Optimizing Sentiment Analysis Pipelines

The project uses the OpenAI GPT API to perform a real-time sentiment analysis. It designed a batch pipeline with Logistic Regression using TF-IDF as the dataset is large. The approach tried using BERT to understand what is required for NLP models, too.

3. Efficient Use of AWS Services

AWS Glue was used for converting JSON files to Parquet files to handle the storage issues. AWS Athena allowed the project to run complex SQL queries, while AWS SageMaker helped us deploy Machine Learning models for batch processing.

4. Advanced Query Techniques for Insight into Business

The project makes an efficient use of SQL features in AWS Athena, where we tried to understand the hidden patterns and trends that would help in efficient decision-making.

5. Cost Optimization by Using Parquet and Batch Processing

The raw JSON files are first converted to Parquet, which helped reduce the cost and query time. The methodology has now moved to performing batch processing for periodic analysis that assists in reducing operational costs.

8.2 TEAM MEMBERS AND THEIR ROLES/ CRediT

API Integration - Aishwarya, Dhruv
Socket Consumer - Aishwarya, Sheetal
Data Processing Pipeline - Aishwarya, Sheetal, Lincy
Sentiment Analysis Module - Sheetal, Lincy
Kafka Integration - Lincy, Aishwarya
Processing Data in AWS Glue using Pyspark - Aishwarya, Lincy
Sentiment Analysis on Parquet Data - Sheetal, Aishwarya
Query Analysis in Athena - Sheetal, Aishwarya
Implementation of Streaming Algorithms, LSH and Privacy Techniques - Sheetal, Aishwarya, Lincy
Implementation Machine Unlearning and ExplainableAI - Dhruv, Aishwarya
Visualisation - Dhruv
Final Project Report - Aishwarya, Sheetal, Lincy, Dhruv

8.3 Prospects Of Winning Competition

The innovative value of the design, completeness of the implementation, and real-world applicability promise the competition some really good prospects. A review of the strong points against the most important judging criteria follows here.

A two-phase approach, real-time streaming architecture in combination with batch processing, showing adaptability for a wide gamut of data processing needs.

Integration of state-of-the-art tools like OpenAI APIs, AWS Glue, Athena, and SageMaker: this is a very innovative edge for the project, using cutting-edge technologies for sentiment analysis and business insights.

8.4 Pair Programming For pair programming in the project, we relied on Visual Studio Code's Live Share extension and GitHub collaboration tools. Tools Used:

- VS Code Live Share for real-time collaboration
- GitHub for version control and code review

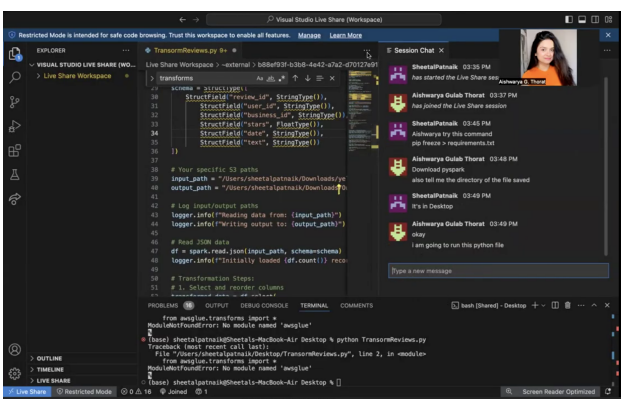


Figure 32: LivesShare

8.5 Trello,Grammarly, and LATEX

Throughout our project development, we used Trello in order to manage tasks efficiently and organize the workflow.

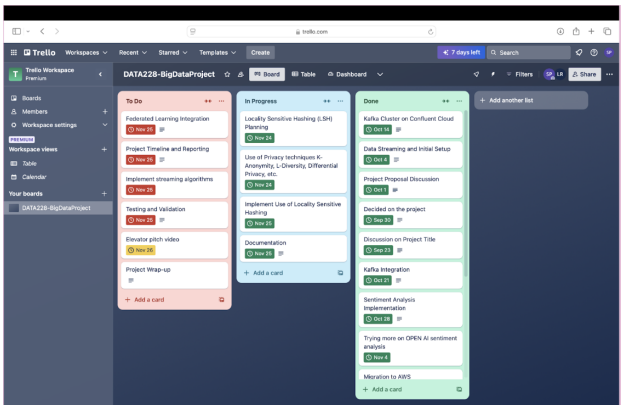


Figure 33: Trello Tasks

Additionally, Grammarly played an important role in maintaining professional documentation quality. As seen in Figure 29. the report is done entirely in latex.

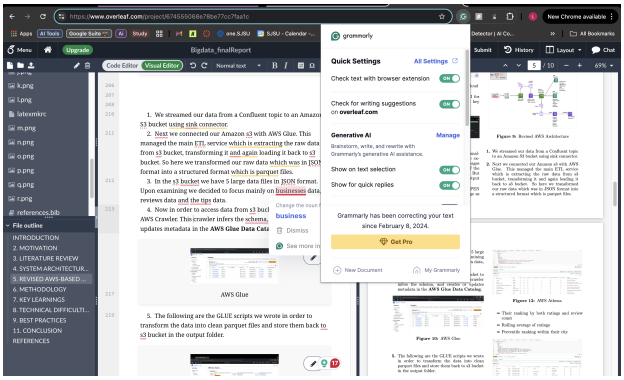


Figure 34: Grammarly

8.5.1 Canva Canva Was used to Create PowerPoint Presentation and architecture diagrams

8.6 Tools and Techniques

- Docker for containerization and service isolation
- Apache Kafka for real-time message streaming
- Spark Streaming with master-worker configuration
- TCP/IP protocol for data transmission
- Elasticsearch for data indexing and storage
- Kibana and Tableau for visualization
- Schema Registry for data format management

8.6.1 New Tools Used In our assignments we had never used AWS Glue, Athena, SageMaker, and also the usage of OPEN AI's Integration for real-time sentiment analysis was something that was new and did take a while for us to get through them and learn about them and practice on them. We have also used AWS Quicksight for visualisation which was a new tool for us.

- AWS Implementation Tools
- AWS S3 for data storage and management
- AWS Glue for ETL operations and PySpark transformations
- AWS Athena for SQL querying and analysis
- AWS SageMaker for machine learning implementation
- TF-IDF Vectorizer for extracting text features
- Window functions for temporal analysis, and Parquet format for optimized storage.

8.6.2 Mandatory question

Did you find or come across solutions to similar problems by using Generative AI or other sources?

Yes, during the course of our project, we did come through many problems for which we had to take help from Generative AI like ChatGPT, Gemini Ai, and Claude Ai. As specified in the report, the major problem was dealing with the FEEDBACK error for which we asked if there are other ways through which we can proceed with our project.

It did give us a few approaches that we tried in order to work on our project. We also took help from GEN Ai in order to learn about the new algorithms and implement them in our project for example machine unlearning, etc. Our other major usage was debugging errors which we encountered when we were building our project.

A few of the other prompts are as follows:

```
verify_athena_setup() for this code got
Error: An error occurred (
    AccessDeniedException) when calling the
    GetTables operation:
User: arn:aws:sts::767397690472:assumed-role
    /AmazonSageMakerServiceCatalogProducts
UseRole/SageMaker
is not authorized
to perform: glue:GetTables on resource:
arn:aws:glue:us-east-1:767397690472:database
    /yelp_db because no identity-based
    policy allows
the glue:GetTables actionFalse
```

How do I Resolve this error?

```
Error: An error occurred (
    AccessDeniedException) when calling the
    DeleteTable operation:
User: arn:aws:sts::767397690472:assumed-role
    /AmazonSageMakerServiceCatalogProducts
UseRole/SageMaker
is not authorized to perform:
glue:DeleteTable on resource:
arn:aws:glue:us-east-1:767397690472:database
    /yelp_db because no identity-based
    policy allows
the glue:DeleteTable action False again got
    the same error.
I added a policy but getting error as this.
    Resolve
```

9. CONCLUSION

The transition from streaming-based architecture to AWS Services was a big leap towards the efficiency and scalability of our system. AWS implementation provided better cost management, reduced operational complexity, and enhanced analytical capabilities. This migration highlighted the advantages of cloud-native services over traditional streaming architecture when it comes to large-scale data processing and analysis.This gave us an idea about how to choose on the pipeline and gave us a deep understanding about various AWS services that can be used to leverage large datasets.

REFERENCES

[1] https://people.csail.mit.edu/matei/papers/2013/sosp_spark_streaming.pdf

[2]https://arxiv.org/pdf/2005.14165

[3]http://stmarysguntur.com/wpcontent/uploads/2019/04/1021302647.pdf

[4]https://people.csail.mit.edu/matei/papers/2013/sosp_sparkstreaming.pdf

[5]https://ijrbat.in/upload_papers/02102017020520178.pdf

[6]https://www.vldb.org/pvldb/vol16/p3557-saxena.pdf

[7]https://arxiv.org/abs/2109.03285

[8]https://ieeexplore.ieee.org/document/10162810

[9]Performing tertiary analysis with data lakes using AWS Glue and Amazon Athena

[10]AWS Glue Best Practices: Building a Secure and Reliable Data Pipeline