Final Project Report

Name:- Joshi Jay Vijaybhai
Student id:- 200485155
Project Name:- AzureSentiment: Sentiment Analysis Using Azure and Bing News API

Table of Contents

Introduction

The Main Idea of the Project:

In this project, we aimed to create a news analytics application that performs sentiment analysis on real-time news articles. Using Microsoft's Bing News Search API, Azure services and the Microsoft Fabric to measured news sentiment every day. The results are visualized on a live Power BI dashboard, exposing patterns in news and public sentiment. The tech aims to aid organizations, companies and the media with tracking public opinion to use it for strategic decision-making, crisis management or planning.

Research Problem:

It is a project to solve real-time the problem of sentiment analysis across news sources. It solves a problem of fetching news data by an API, and analyzing the data to get insights on how the public is feeling about any particular thing. The primary research challenge was to design a pipeline which should be capable of: News Data Ingestion;Data Transformation and Sentiment Classification on Data in Real-Time

Objective:

In essence, the goal of this project was to construct a data pipeline that could accurately predict sentiment over news data retrieved via Bing News Search API. This includes use of data engineering techniques and machine learning models (such as Synapse ML model) to gauge sentiment out of news articles.
The initiative lies in the study of sentiments from news articles , which is an attitude related to content concerning certain events, subjects, or entities through which the attitude of the news articles is either positive, negative, or neutral. Although analysis does not measure public opinion, media sentiment patterns can become a gauge for how stories are shaped and may affect public discourse. For instance, an increase in negative articles regarding a specific policy may indicate difficulties in public acceptance, despite the fact that the sentiment arises from the content of the news itself.

- **Input:** The input is the daily news articles from Bing News Search API stored in Microsoft Data Lake. These articles are from Politics, Business, Technology along with the rest.
- **Output:** Results are presented as a PowerBi dynamic dashboard highlighting daily changes in sentiment, summary tables for key metrics, and visual analytics (eg. graphs). We also set up an alert system so we can be notified via Microsoft Teams when sentiment goes above or below a threshold.
- **Usefulness:** Media attitude and trends on key issues can be monitored in real-time by organizations. Such knowledge helps in strategic planning and pre-emptive crisis response, which is critical for decision-making. Businesses and politicians can determine the tone of media narratives and predict what their potential impact may be on public perception and market dynamics by looking at sentiment patterns.

In the subsequent paragraphs we discussed about data sources and the methods followed by presenting an example for showing proposed approach. We also depicted the implementation for providing significant results with respect to it. Then we reported all the findings and the resulting output together with appropriate conclusion and suggestions for possible extensions of this work into recommendations for further research directions. To guarantee credibility as well as traceability in all details of this work, appropriate references is reported as well.

Background

Previous Related Projects:

During the CS714 (big data) course, I worked on a project that requires analyzing you-tube data collected using AWS. We were required to build an ETL pipeline and design a visualization in AWS QuickSight for the data gathered. This project itself was missing any alerting system or machine learning models for data analysis unlike my current one. In this project, we are planning on building a functional Sentiment Analysis model and real-time notifications in Microsoft Teams using machine learning. On top of that, I have barely touched the Azure ecosystem before so this will help me to broaden my cloud computing expertise into worlds outside of AWS.

Relevant Research Papers:

These are five important papers as of 2018–2024 that are related to sentiment analysis in real time especially for news articles by both machine learning and data engineering.

- **Sentiment Analysis Using Deep Learning Techniques: A Comprehensive Review[1]**
  This Paper shows some of the difficulties with the use of deep learning in sentiment analysis are large well-labeled datasets and models that generalize to different domains. For instance, it goes into details about the kinds of models used in sentiment analysis and focuses on ways such as regularization and data augmentation to avoid overfitting while being critical if we are to analyze news articles across different subjects in a real-time scenario.
- **Real-Time Sentiment Analysis of Natural Language Using Multimedia Input[2]**
  This study is based on the proposal of a model regarding real-time sentiment analysis of textual and multimedia data. Considering the integration of text, audio, and video data for the sentiments of the authors in this research will inspire new avenues towards news-given that a sentiment analysis system would work on systems that operate on multimedia data inside the news, be it video or podcast format.
- **Challenges and Future in Deep Learning for Sentiment Analysis: A Comprehensive Review and a Proposed Novel Hybrid Approach[3]**
  This paper will specifically dwell on multimodal sentiment analysis by using the integration of vision, audio, and textual features that are derived from user-generated videos. Using various sources of data such as social media, video news, and textual articles in building a real-time news analytic tool will support this with more robust and accurate sentiment analysis.

- **An Analysis of Sentiment: Methods, Applications, and Challenges[4]**
  This paper discusses the various approaches to sentiment analysis with an emphasis on machine learning and their hybridized techniques. It addresses pertinent issues like feature extraction, classification strategies, and domain adaptability, which are convoluted but crucial for sentiment analysis of heterogeneous news topics.
- **Enhanced Video Analytics for Sentiment Analysis Based on Fusing Textual, Auditory, and Visual Information[5]**
  This research clarifies the advantages of combining textual, auditory, and visual sources for the task of emotion detection. The form of emotion detection studied for this research was video analytics. Still, the general knowledge about fusion that can inform and expand emotion detection functionality beyond text, as applied here to include guidelines for using visual reporting sources.

This research is a timely review of important developments, challenges, and trends in sentiment analysis including the ongoing research that expands the scope of sentiment analysis to include different modalities like embedded social media content and images, both which are crucial in creating an overall, real-time application for news sentiment analysis.

Existing Techniques Relevant to the Project:

- **API Integration:** The Bing News Search API is employed to ingest fresh news articles into the solution on a daily basis. This involves integrating APIs as a data engineer working on data engineering projects.
- **Data Ingestion and Transformation:** The raw news articles are all consumed as a data and stored in Azure Data Lake, and I've developed a pipeline to do incremental data load through Azure Pipelines, using PySpark for data transformation.
- **Sentiment Analysis:** Synapse ML models are leveraged to perform sentiment analysis on the ingested news data to label articles as positive, negative, or neutral based on their emotional tone.
- **Reporting and Dashboard:** Power BI is used as the visualization tool to get the dashboard of sentiment trends and insights in a manner that refreshes on an ongoing daily basis of ingestion of the new news data.
- **Alerting System:** There is an alerting system using teams to flag when the sentiment of new articles surpasses thresholds for alerting (for example, when the articles have high negative or high positive sentiment using Data Activator).

Learning Resources:

In order to gain a solid understanding of the skills and knowledge base that arises from implementation of these techniques, I referred :

- Official Azure documentation.
- Guides for using Microsoft Fabric.
- API documentation for guidance on integration.
- Previous coursework on cloud computing and machine learning.

Advantages and Disadvantages of Techniques:

API Integration:

• Advantage: Enables real-time data ingestion; simple access to a variety of news sources.

• Disadvantage: Dependent upon stable and reachable API; possible rate limits can postpone data ingestion.

Data Ingestion and Transformation:

• Advantage: Azure Data Lake provides performant storage and scalable data processing; PySpark offers opprobrious data transformation opportunities with its extensive libraries.

• Disadvantage: Steeper learning curve when assessing Azure services; complex pipeline can be a challenge to manage.

Sentiment Analysis:

• Advantage: Synapse ML models provide advanced machine learning capabilities for sentiment analysis; finds the patterns between news articles.

• Disadvantage: Models are biased; can be influenced by the quality of underlying data.

Reporting and Dashboarding:

• Advantage: Power BI offers easy to use visualization tools; offers dashboards active reports for real-time data insight.

• Disadvantage: Data sources need to be maintained on a continual basis to ensure accuracy; or when source complexity increases visualization the benefit will diminish in exchange for Institutional knowledge.

Alerting System:

• Advantage: Near real-time notification of a shift in sentiment increases responsiveness; integrates with Microsoft Teams in the program.

• Disadvantage: Alert fatigue can happen if a threshold is not determined; close to near real-time data flow or current sentiment enabled alerts.

Approach

DATA:

Data sources for this project come from the Bing News Search API. The Bing News Search API has granted us access to real news stories across categories falling within news-related themes and genres, such as politics, tech, entertainment, and so forth. For each news article, metadata include its title, description, date published, and URL; all these are ingested into the platform. It is a well-documented, trusted source which can be integrated without any hitch into Azure services and, therefore, is a sound source for the continuous ingestion of news data. In order to get more information about the API use the URL link : `https://learn.microsoft.com/en-us/bing/search-apis/bing-news-search/overview`.This API allows developers to retrieve news articles by specifying keywords, date ranges, or categories. The F1 tier permits up to 3 API calls per second and a maximum of 1,000 transactions per month. The paid plans have limits much greater, supporting as many as 50 calls per second and varying transaction caps with each plan. With every request capable of returning up to 100 articles, these are limited by the `count` parameter. As an added advantage, it's fully scalable to cater to small tasks like small-scale sentiment analysis or massive applications with huge enterprise undertakings.

METHOD:

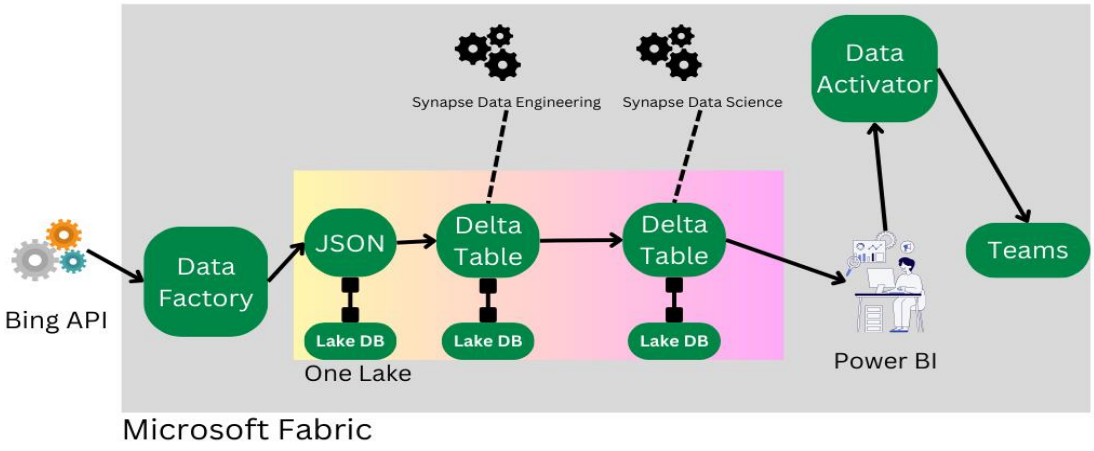

Fig:-Project Architecture

The project is executed through several processing stages, leveraging Microsoft Fabric's capabilities and integrating various Azure tools to achieve the goal of real-time sentiment analysis for news articles. The method consists of the following stages:

- **API Setup and Configuration:** We set up the Bing News Search API in Azure that can tap real-time data of news. It delivered fresh news articles every day and is one of the very important pieces of the data source for the project.
- **Data Ingestion Using Microsoft Fabric:** We connected Microsoft Fabric's data tool to the API so that we can pull data into it. The API is full of data in JSON file, which we copied and stored to OneLake, a data storage service by Fabric. OneLake has enabled the creation of many databases, and one of them is Lake Database, which we used here to store raw JSON files.
- **Data Transformation:** Then, raw JSON is transformed and structured to a proper format. The transformation in question is then carried out using Synapse Data Engineering; that is by creating Spark Notebooks that clean the JSON data into the tabular format. Clean data then be stored within the Lake Database, in a pre-defined schema Delta Table. Incremental loading is also included in the process; this mean new data loading continuously and without extra overhead.
- **Sentiment Analysis:** The cleaned and formatted data then run for sentiment analysis through Synapse Data Science. It is thereby derive the emotional tone of the news articles or classify them as positive, negative,mixed, or neutral. This processed data with sentiment scores are again warehoused in the form of a Delta Table in Lake Database.
- **Dashboard Creation Using Power BI:** From this analysis, I built a comprehensive news sentiment dashboard using Power BI. This includes depicting the key insights for the customer, such as general trends over time news sentiment, top stories, and even the distribution of sentiment to different categories.
- **Alerting System Using Data Activator:** The system is created alerting using the Data Activator, configured to be able to monitor the mood thresholds over time and to produce alerts in real-time using Microsoft Teams. It is illustrative to send out signals at times when overall sentiment of news articles exceeds certain predefined thresholds, highly negative or positive.

This approach relies on integration of data engineering, data science, and visualization tools in Microsoft Fabric with the result in a robust pipeline for real-time sentiment analysis and reporting over news data.

EXAMPLE:

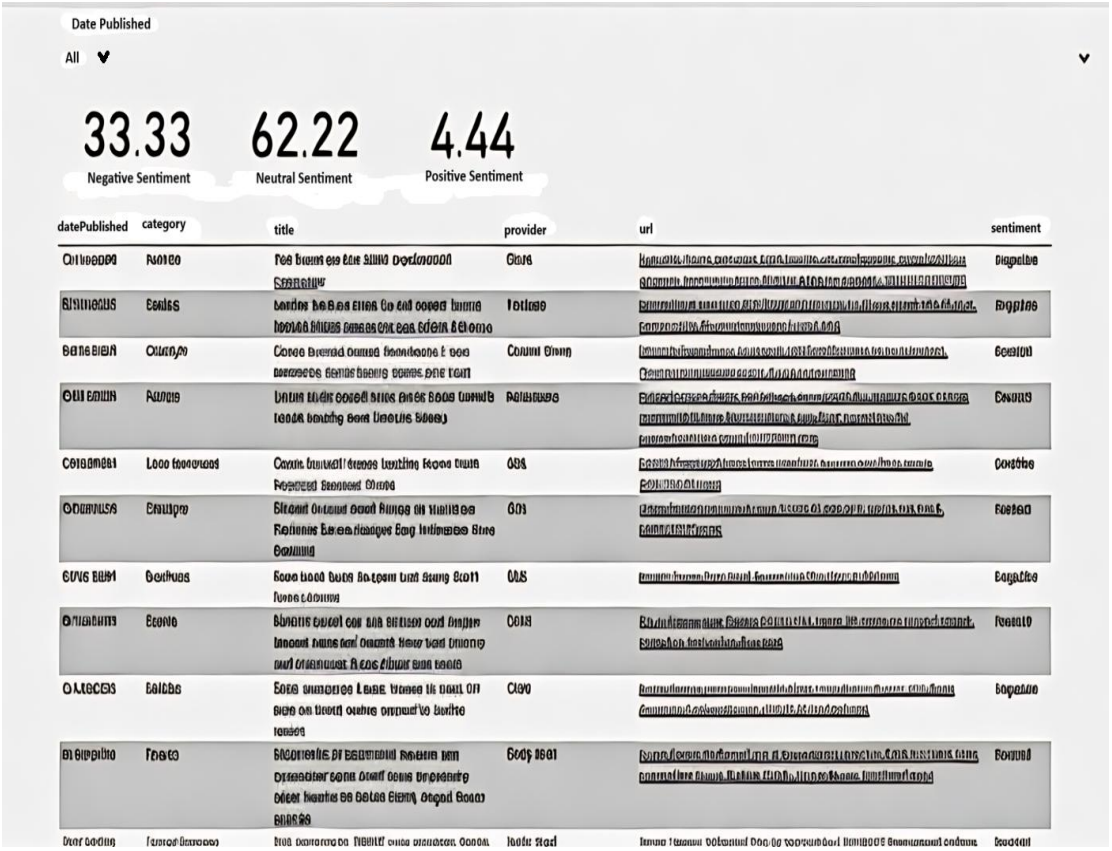

Fig:-Project Example

Above is an example of a dashboard that I presented at the completion of the project, including alert notifications integrated through Teams. This dashboard has a table showing article URLs and other information as well as the sentiment analysis for each articles. Measures such as negative, positive, and neutral sentiments are also included on the dashboard as shown on a Power BI data card. The dashboard can choose a particular date for sentiment analysis through a dropdown menu; otherwise, it will automatically show the sentiment analysis for the latest date's data. The content is blurred and less readable because it was generated by AI through a prompt. However, as compared to the final result, it is clear that we have not only obtained what is represented in the example but have also improved upon it by adding an additional page and new measures to further enhance the functionality of the dashboard.

Implementation

1. **Data Collection and Storage**

The first step of the project was to gather news articles by using the Bing News Search API, configured to bring in daily articles that came from specific keywords and categories including Politics, Business, and Technology. The articles and associated metadata such as title, description, date, and source were ingested into a Microsoft Data Lake for an adequate storage model that could handle any volume. This ensured a trusted, structured, and resourceful data analysis infrastructure.

- **Tools used included** Python for API integration and Azure SDKs for storage.
- **Key steps included** configuring API calls with parameters like freshness and query terms, parsing and cleaning JSON responses to extract relevant fields, and storing the data as structured files (Delta tables) in the Microsoft Data Lake for downstream processing. It retrieved only new or updated articles with the incremental loading approach, thereby reducing redundancies to leverage a timestamp or freshness parameter to track the last ingested data. Its workflow ensured efficient storage, processing, and consistency.

Here the below code snippets extracts required data from a JSON file into lists and converts them into a structured Spark DataFrame for easier processing and analysis.

```
# Initialize lists
article_title = []
article_description = []
article_category = []
article_url = []
article_image = []
article_provider = []
article_date_published = []

for item in listofJson:
    try:
        # Parse the object
        data = json.loads(item)
        json_obj = data.get("json_object", {})

        # Check if required fields exist before accessing them
        if json_obj.get("category") and json_obj.get("provider", [{}])[0].get("image", {}).get("thumbnail", {}).get("contentUrl"):
            article_title.append(json_obj.get("name", ""))
            article_description.append(json_obj.get("description", ""))
            article_category.append(json_obj.get("category", ""))
            article_url.append(json_obj.get("url", ""))
            article_image.append(json_obj["provider"][0]["image"]["thumbnail"].get("contentUrl", ""))
            article_provider.append(json_obj["provider"][0].get("name", ""))
            article_date_published.append(json_obj.get("datePublished", ""))

    except Exception as exc:
        print(f"Unable to process this JSON object due to: {exc}")

[9]  - Command executed in 323 ms by JAY JOSHI on 12:06:36 AM, 12/02/24
```

Fig:-Processing JSON Data to Create a Spark DataFrame

This code is processing the list of JSON objects named `listofJson` extracting certain fields and returning data in the form needed to create a Spark DataFrame. It initially defines several lists in which the extracted data are placed: title, description, category, url, images, providers, and publication dates. It goes through all JSON objects from the list and, one at a time, parses every JSON object, and conditions will depend on whether a given field of data is present within that JSON object. For instance, it checks for the existence of a `category` and an image URL before adding data to the respective lists. Any errors encountered during processing are logged with an explanatory message.

```
# Combine data
info = list(
    zip(
        article_title,
        article_description,
        article_category,
        article_url,
        article_image,
        article_provider,
        article_date_published
    )
)

# Define the schema
framework = StructType([
    StructField("title", StringType(), True),
    StructField("description", StringType(), True),
    StructField("category", StringType(), True),
    StructField("url", StringType(), True),
    StructField("image", StringType(), True),
    StructField("provider", StringType(), True),
    StructField("datePublished", StringType(), True)
])

# Create the DataFrame
cleaned_dataframe = spark.createDataFrame(info, schema=framework)

[10] ✓ - Command executed in 242 ms by JAY JOSHI on 12:06:37 AM, 12/02/24
```

Fig:-Creating a Spark DataFrame from Extracted Data

The extracted lists are then combined using `zip` in Python. The data is mapped with a defined schema that explicitly outlines the data types along with column names of the actual DataFrame to be generated on the Spark. Finally, this all data and schema combined set up is used to generate the DataFrame on Spark and allows for further data analysis and processing in a big data system.

2. **Data Processing and Preparation**

The data was then stored and it underwent preprocessing to prepare it for sentiment analysis.

The preprocessing tasks included: removal of irrelevant contents or noise, tokenizing of text fields like title and description, and formats to make date and time field format consistent. Handeling the missing values to keep Data integrity. It also makes an incremental load to have the efficiency without redundancy. Once processed, it was stored in a structured format (Delta tables) in Microsoft Data Lake with the capability for scalability and automation for the subsequent analysis.

This code snippet, provided below, covers saving or updating(incremental loading) the Delta table ‘bing_api_db.tbl_API_sentiment’ using PySpark:

```
try:
    tabletag = 'bing_api_db.tbl_API_sentiment'
    finalsentiment.write.format("delta").saveAsTable(tabletag)

except AnalysisException:

    print(" Table already available")

    finalsentiment.createOrReplaceTempView("vw_finaldataframesentiment")

    spark.sql(f"""  MERGE INTO {tabletag} aimedtable
                    USING vw_finaldataframesentiment originview
                    ON originview.url = aimedtable.url
                    WHEN MATCHED AND
                    originview.title <> aimedtable.title OR
                    originview.description <> aimedtable.description OR
                    originview.category <> aimedtable.category OR
                    originview.image <> aimedtable.image OR
                    originview.provider <> aimedtable.provider OR
                    originview.datePublished <> aimedtable.datePublished

                    THEN UPDATE SET *

                    WHEN NOT MATCHED THEN INSERT *

                """)
```

✓ - Command executed in 8 sec 334 ms by JAY JOSHI on 12:26:37 AM, 12/02/24

```
Table already available
```

Fig:-Updating and Synchronizing Delta Table with New Data

First, it tries to save finalsentiment DataFrame as a Delta table under the given name: `bing_api_db.tbl_API_sentiment`. If the table doesn't exist, it gets successfully created. If it exists, an `AnalysisException` is thrown, which then this code catches and goes into updating the table via MERGE statement:.

This ensures that the table with existing data gets updated with new or modified data from the `finalsentiment` DataFrame. The view 'vw_finaldataframesentiment' is created as a temporary one based on the DataFrame, and then the MERGE compares the records in the view with the table, (`aimedtable`), using the matching condition (`ON originview.url = aimedtable.url`). If there is a match, and certain columns differ, such as `title`, `description`, `category`, etc., then the table gets updated with the new data: `THEN UPDATE SET *`. If there is no match, the new records are inserted into the table: `WHEN NOT MATCHED THEN INSERT *`. This ensures the Delta table is always synchronized with the latest data.

3. **Sentiment Analysis**

Sentiment of news articles was predicted employing machine learning models. Open-source SynapseML is the library in Azure Synapse Analytics, which was used for this task. This pipeline consisted of the following components:

- **Model Selection:** For the given data, the SynapseML sentiment analysis model called AnalyzeText was used.
- **Processing:** Model assigned sentiment scores to the article, such as positive, neutral, negative, and mixed.
- **Storage of Results:** Sentiment scores, along with article's metadata, were stored in the Microsoft Data Lake.

4. Visualization and Reporting

The processed data was connected to Power BI through Azure Data Factory, where a real time dynamic dashboard was built in order to present the results. The dashboard included:

- **Daily Sentiment Trends**: Charts indicating the changes in sentiment over time.
- **Power BI Data Cards**: Displays Key measures such as the percentage of positive or negative sentiments for the articles by category.Below is the one example of these crucial measures in our semantic data model. It gives us an idea about which measures are used in this piece of code for analysis.

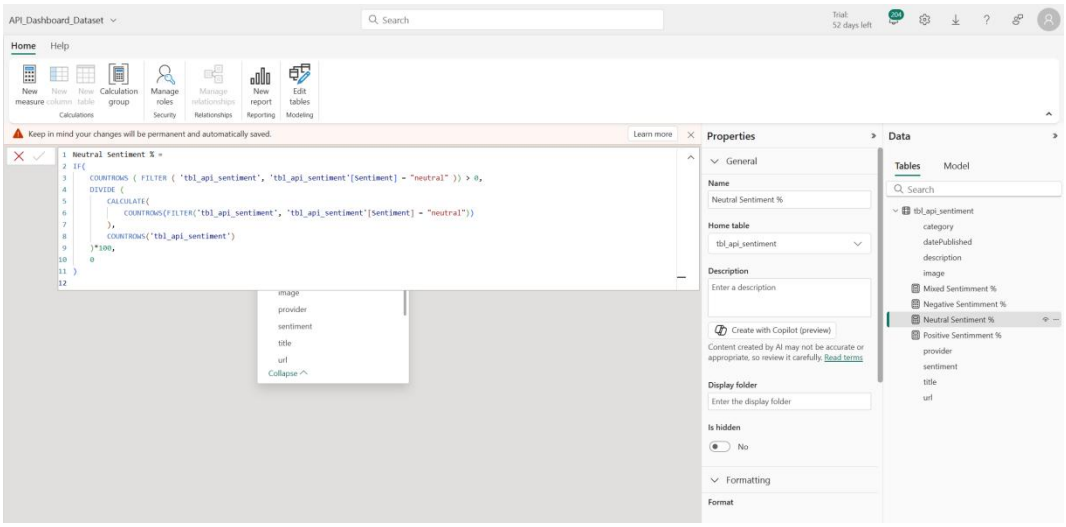

Fig:-Calculating Sentiment Percentages using DAX

The above code is in DAX. The code takes the percentage of the neutral sentiments in a dataset with the table name being `tbl_api_sentiment`. First, it checks rows within the table that has been labeled as "neutral" regarding the sentiment. When this happens, it calculates the percentage by dividing the row count that has the "neutral" sentiment by the total count of rows in the table. This way, the real proportion of neutral sentiment will be calculated accurately. Using the `DIVIDE` function prevents arithmetic errors because of a zero total row count for the given formula. In case no "neutral" sentiments are found, the formula returns 0.

This also calculates the percentage of positive, negative, and mixed sentiments. This condition is altered to select "positive," "negative," or "mixed" in order to extract the respective percentage of sentiment, and this happens in the very same manner, so it is really easy to draw conclusions on the distribution of sentiments over the dataset.

- **Table View**: Detailed descriptions along with sentiment analysis of articles.
- **Date Filtering**: The specific dates could be selected with a drop-down option otherwise the latest articles and their sentiment could be automatically displayed by filtering processes.

An alerting system was developed using Data Activator, which sends notifications to Microsoft Teams whenever sentiment scores crossed predefined thresholds.

5. Automation and Monitoring
As automated, the pipeline is smoothed out to ensure smooth handling, and **Azure Data Factory** is used to orchestrate data ingestion and processing. **Monitoring tools** such as Azure Monitor are also used, which track pipeline performance so that failures can be captured and data quality assessed and ensured for reliability and efficacy along the process.

6. Scalability and Maintenance
The system was designed to scale with increasing data volumes through the cloud infrastructure of Azure. Periodic updates to ingestion query parameters or keywords ensured relevance and accuracy in changing scenarios. Also, the processing speed of the pipeline was optimized for handling large datasets by using parameterized queries instead of hardcoded ones, ensuring flexibility and efficiency.

Technology Stack

The project implemented a robust and scalable sentiment analysis system that has integrated a number of tools to offer actionable insights nearly in real-time.

- This data retrieval was done via the Bing News Search API and uploaded to Microsoft Data Lake
- While the whole data pipeline was automated via Azure Data Factory. Hence, the dashboard keeps getting updated in real-time.
- The analysis of the sentiments was done using SynapseML, and Power BI was utilized for data visualization.
- Moreover, Data Activator and Microsoft Teams enabled establishing alert mechanisms for early warning and quick decision-making.

Results of Running Software

The data for this project is the real-time news articles from Bing News Search API. In that, topics vary like politics, business, technology, and entertainment, etc. The titles and descriptions along with metadata like date published, source of article, URL and categories topics are included. The sentiment scores were created using SynapseML. The data was scaled up to 100 articles per request with historical data being accumulated for the analysis over time. Raw data retrieved as JSON files were further processed and stored as structured Delta Tables in Microsoft Data Lake for further analysis. In the pre-processing phase, text fields were tokenized and cleaned, date formats standardized, and missing values were handled, maintaining consistency and reducing noise in the data. The incoming articles were subjected to real-time sentiment analysis, and the data of the API was handled through incremental data loads. Then, trend analysis would be conducted based on accumulated data from latest day,previous days or weeks to trace the overall patterns in the sentiment with time.

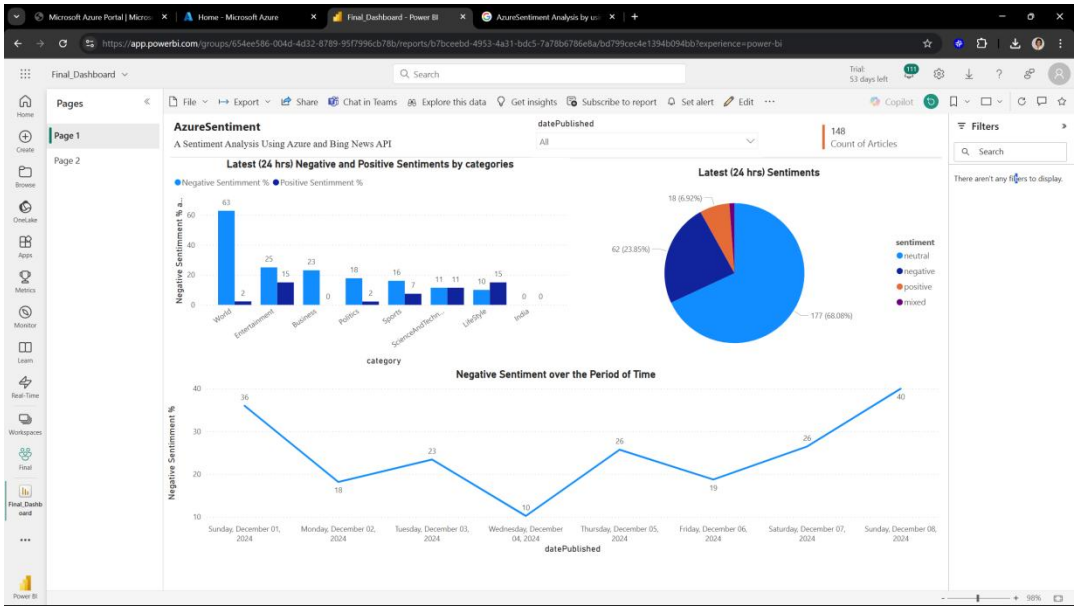

Fig:-Screenshot of Real Time Dashboard's page:-1

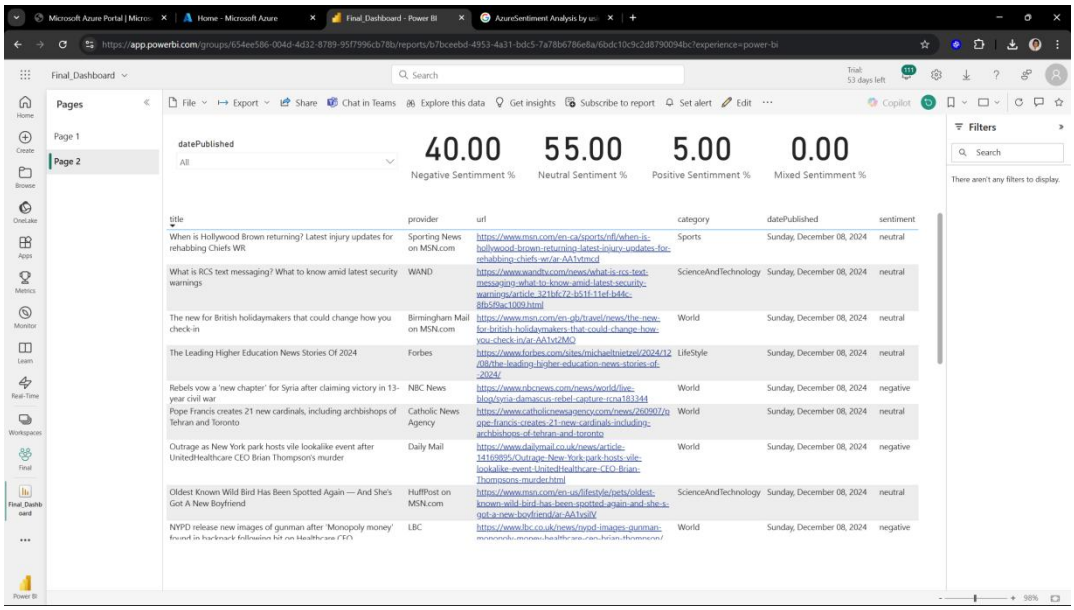

Fig:-Screenshot of Real Time Dashboard's page:-2

The screenshots above are two actual real-time dashboard pages refreshed daily with newly ingested data. These dashboards are designed with dynamic filters, which means graphs will automatically render the sentiment analysis of the latest data ingested and the latest day. For historical analysis, the user can choose specific dates from the `datePublished` drop-down to have graphs and measures reflect results based on the selected date. Moreover, users can visit the original articles through supplied URLs for further details.

Page 1 contains a line graph that shows a trend of negative sentiment by time, based on accumulated data.This visualization will help to identify patterns and trends in sentiment. Companies and PR firms can consider such graphs and insights when keeping track of how the news media portrays their organization. Given that media has a tremendous role in shaping public sentiment, these findings will allow organizations to manage their brand effectively and be proactive against some problems beforehand.

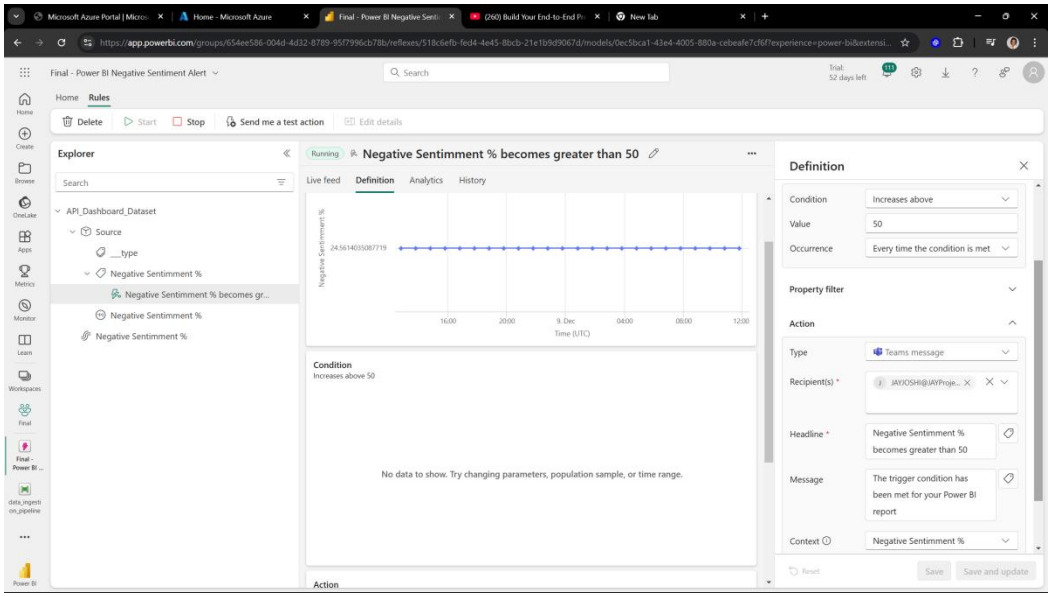

Fig:-Screenshot of Real Time Alert for Microsoft Teams

The abbove screenshot explains the real-time alert setting for the dashboard. The "action" section of the screenshot shows the message which will be displayed when the negative sentiment goes over 50%. It triggers an alert with a headline and message, which would give due time to be informed of the change in sentiment. These alerts are set up to automatically check for an update every hour. As can be seen in the screenshot, no such event has occurred yet where negativity has crossed above 50%. However, if it does it will send a direct notification to Teams where one can get instant notifications.

For model validation, a comparison was made on the sentiment results generated by SynapseML with an auxiliary dataset that had labeled data in the kaggle called Positive-Neutral-Negative Sentiment Analysis Data .

```
1  from pyspark.sql.functions import col
2  from pyspark.ml.evaluation import MulticlassClassificationEvaluator
3
4
5
6
7  accuracy_df = finalsentiment.withColumn("correct",
8                                          (col("predicted_sentiment") == col("sentiment")).cast("int"))
9
10 accuracy = accuracy_df.select("correct").agg({"correct": "avg"}).collect()[0][0]
11 print(f"Accuracy: {accuracy:.2f}")
```

✓ - Command executed in 8 sec 155 ms by JAY JOSHI on 11:26:14 AM, 12/09/24

> Diagnostics 1

```
Accuracy: 0.89
```

Fig:-Benchmarking

100 of the first entries in each of the sentiment labels, positive, neutral, and negative, were used for testing purposes and was found to be about 0.89 in accuracy. Since real-time ground truth is not provided as there is no true value available for freshly ingested data to compare our model's result with, the supplementary dataset does indeed provide for benchmarking on the performance of the models. Considering everything, datasets, preprocess, and experiment altogether proved that the totality

provided holistic evaluation both of the project pipeline as well of dependability related to insights regarding sentiment.

Initially, we had hard-coded SQL queries to ingest data from the API. We then switched to using parameters which helped us to dynamically ingest data for specific keywords from the back end. This added flexibility to the ingestion process. As seen in the images below, this change didn't affect runtime performance-it neither improved nor degraded it.

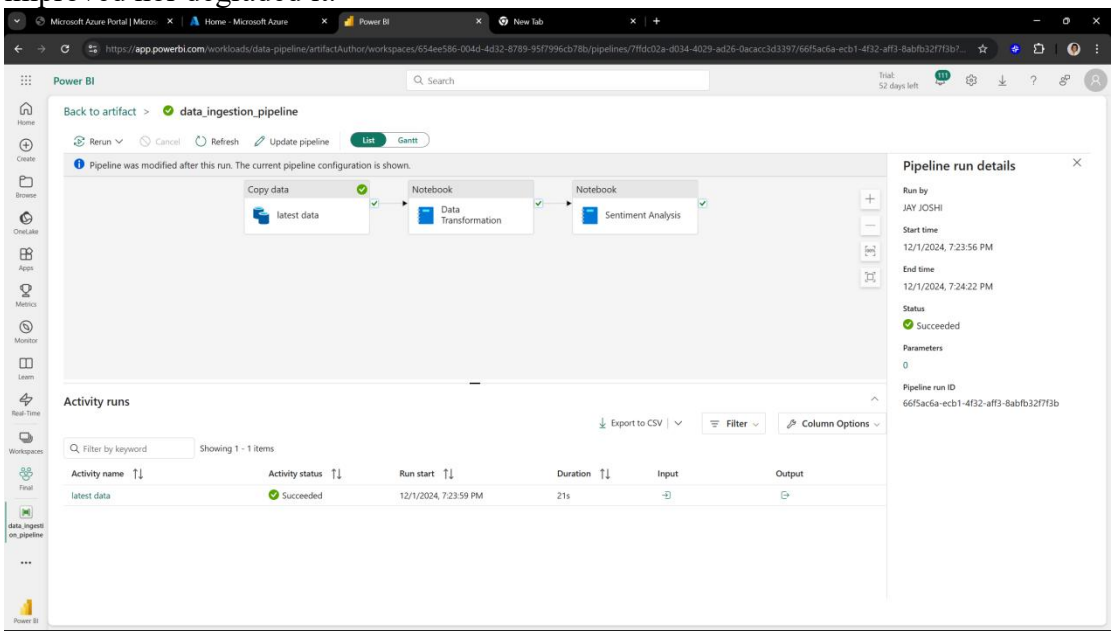

Fig:-Screenshot of Pipeline run without Parameters to get latest data

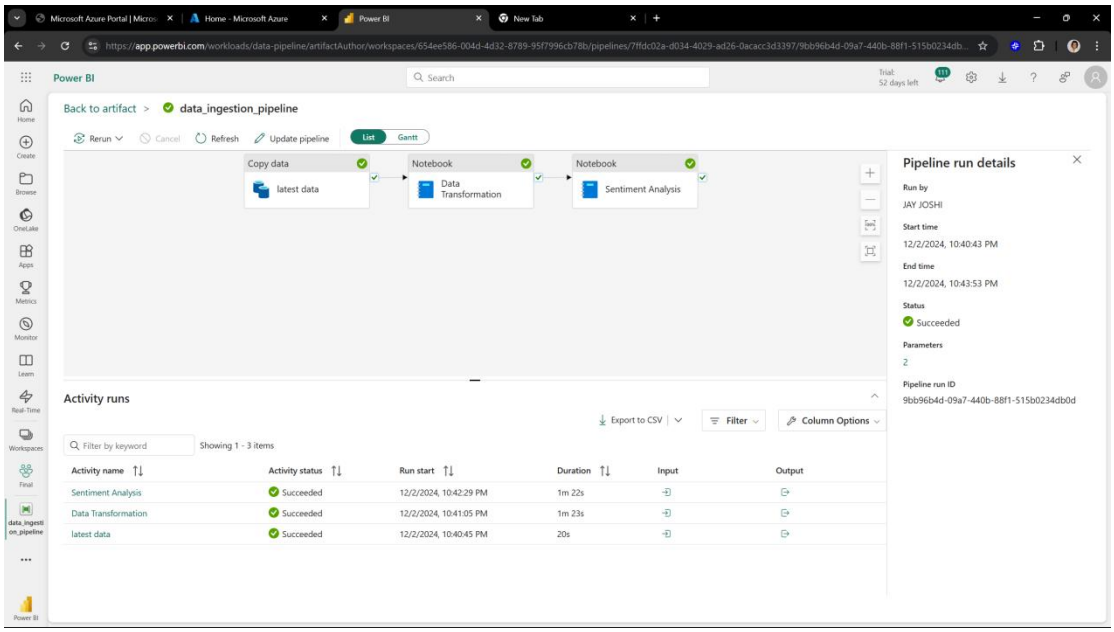

Fig:-Screenshot of Pipeline run with Parameters to get latest data

As illustrated in the picture, it takes about 20 seconds to ingest new data. The processing of this data and the subsequent doing sentiment analysis takes about 1 minute and 20 seconds, which is remarkably efficient.

Next, regarding storage, we made sure to implement incremental loading to remove duplicative data. As this implementation utilizes a cloud service, scale on storage is no problem at all. The cloud's intrinsically scalable nature ensures increased handling of growing data volumes without degrading the performance.

Conclusion

This project was, in reality, putting into practice an overall system for sentiment analysis of news articles on real-time bases using Bing News Search API, Azure SynapseML, and Power BI. This will track over time the sentiments on issues like politics, business, and technology and generate actionable insights for businesses, PR firms, and other organizations following public sentiment through time. Using the Bing News Search API, news articles were collected in real time, and metadata included title, description, category, and date of publication. Incremental loading into a Microsoft Data Lake meant it was scalable without duplicating data. Preprocessing included cleaning, tokenizing, and structuring raw data into Delta tables for analysis purposes on Azure Synapse.Using SynapseML, sentiment analysis was done, classifying articles as positive, neutral, negative, or mixed, and the results were stored for trend analysis. A dynamic Power BI dashboard visualized the trends of sentiments, providing metrics such as positive, negative, and neutral sentiment distributions, filterable by date, with access to original articles through URLs. There was also a real-time alerting system using Data Activator and Microsoft Teams that alerted the user of when the thresholds have been crossed in sentiments to be proactive against media shift. It used Azure Data Factory to automate data pipelines to ensure continuous efficient handling of growing data volumes while it is scalable enough to work with huge datasets over cloud infrastructure. In total, the project has been very powerful to deliver real-time insights via cloud-based tools, and such a platform is extremely beneficial to the organizations being monitored and managed in order to maintain their media presence. Automation, scalability, as well as the adaptability of the system to moving trends make this a robust and adaptive solution for real-time news media sentiment analysis. Providing the dimension of data engineering, machine learning, and data visualization, the project gives a scalable approach toward understanding and managing public sentiment effectively.Overall, the project was completed in alignment with the specific problems and goals outlined in the introduction.

Suggestions for Future Research

Limitations of the Existing Approach

- Some limitations hinder the current approach to data ingestion, limiting its efficiency and scalability.The most critical issue here is the dependency on a human to update queries, which is both time-consuming and prone to human mistakes, thereby making it difficult to quickly adapt to changing needs.
- Furthermore, non-technical users rely on developer to update the pipeline, which constrains access and extends the entire process.
- The system is also rigid in allowing real-time or ad-hoc query updates that prevent the users from doing dynamic analysis.
- As the number of required keywords grows, managing them directly in the pipeline becomes very cumbersome, causing scalability issues and performance bottlenecks.
- Keyword ingestion customization does not have a user interface; therefore, the process is less user-friendly and relies on backend changes.

Limitations of Existing Implementation or Experiments

- Predefined queries cannot be experimented on for testing hypotheses or pivoting analyses in an experiment environment.
- Without technical help, users cannot try out different combinations of keywords to iterate quicker.
- The system is prone to data overlap or the ingestion of irrelevant information because of its hardcoded queries and inability to perform real-time fine-tuning.
- Furthermore, although it is an extremely useful tool in data visualization, the dashboard still does not support keyword management, thereby reducing its usability as an all-inclusive tool.

Proposed Future Enhancement

We would be able to overcome these deficits by proposing a future update where the process of keyword management is made more user-friendly and scalable by being included in the dashboard itself.

- It means that users would be able to put a keyword or phrase in there without having to manually modify the pipeline, which increases the efficiency significantly.
- This would be an automation process where the entry of a keyword would begin a pipeline to fetch data in terms of date ranges, regions, or languages on the Bing API.
- **Some of the benefits that come with this approach are:**
    - it allows non-technical users to handle keywords on their own without involving developers, and makes the system much more flexible in rapid testing of new keywords.
    - The approach improves scalability since more users and use cases can be supported without having to manually intervene.

Overall, this proposed enhancements would streamline the process of ingesting data, reduce reliance on technical team, and bring more agile, real-time analyses that result in faster insight and greater operational efficiency.

References

- Sahoo C, Wankhade M, Singh BK. Sentiment analysis using deep learning techniques: a comprehensive review. International Journal of Multimedia Information Retrieval. 2023 Dec;12(2):41.[1]

- Jain R, Rai RS, Jain S, Ahluwalia R, Gupta J. Real time sentiment analysis of natural language using multimedia input. Multimedia Tools and Applications. 2023 Nov;82(26):41021-36.[2]

- Islam MS, Kabir MN, Ghani NA, Zamli KZ, Zulkifli NS, Rahman MM, Moni MA. Challenges and future in deep learning for sentiment analysis: a comprehensive review and a proposed novel hybrid approach. Artificial Intelligence Review. 2024 Mar 5;57(3):62.[3]

- Sharma HD, Goyal P. An Analysis of Sentiment: Methods, Applications, and Challenges. Engineering Proceedings. 2023 Dec 19;59(1):68.[4]

- Al-Azani S, El-Alfy ES. Enhanced video analytics for sentiment analysis based on fusing textual, auditory and visual information. IEEE Access. 2020 Jul 27;8:136843-57.[5]