Reddit Topic Clustering and Sentiment Analysis

March 7, 2023 - MSDS 697 - Distributed Data Systems
Team: Panic at the DSCO
Team Members: Joren Libunao, Anirav Jain, Daniel Tinoco, Theo Kim (Byunghyun), Zemin Cai

Problem / Motivation Statement

The project aims to develop an automated scalable machine-learning pipeline to perform clustering and sentiment analysis of world news on Reddit. The main objective is to apply the concepts learned from Distributed Data Systems to collect, preprocess, and transform data into useful information. The project will produce word clouds and sentiment analysis, which will help understand the sentiment of the news.

Dataset

Data source: Reddit API https://www.reddit.com/dev/api

- Size of dataset: 113,000 comments in 12 files (1 MB each), 1,100 posts in 12 files (50 KB each)
- Data format: .csv files in GCP, JSON in MongoDB
- Data types: String, DateTime, integer
- Missing values: None
- Time range: 2/25/23 - 3/8/23

Analytic Goals

With the posts data set, the goal was to cluster posts by topics and analyze the sentiment of the comments. In order to do this, we had to tokenize the post titles, remove stop words, and vectorize post titles. Then we clustered the posts using k-means. We wanted to create word clouds for each cluster if they have distinct topics with relevant phrases.

With the comments dataset, we aimed to analyze the sentiments of select topics into positive, negative, or neutral. We would also alter the thresholds for sentiment to better align with comment sentiment. Overall, we hope to uncover some insights through clustering and sentiment analysis. For example, we want to find out if some subreddits are more positive or negative and what is the sentiment of trending posts. To communicate our findings, we want to visualize output using a word cloud to highlight frequent words or topics in the subreddits and how the sentiment changes over time.

Data Engineering Pipeline

The first few steps of the pipeline were all performed via a Directed Acyclic Graph (DAG) file that we pushed through Apache Airflow. The DAG file began the pipeline by extracting the relevant data from the selected subreddits using PRAW, or Python Reddit API Wrapper, a special tool we

used to facilitate interacting with the Reddit API. The data we scraped included posts, titles, related comments, and associated features such as karma, the subreddit of origin, and the publish date and time.

Once we had the relevant data, we aggregated them into a DataFrame format and wrote it as a CSV file into Google Cloud Platform, where we stored it in buckets for posts and comments, respectively. This was followed by using a Spark Submit Operator in the DAG file to reformat the data as JSON aggregates to move the data to MongoDB Atlas. This was all handled by the DAG file daily at midnight UTC (4 pm PST).

Once the automated portion of the pipeline was established, we imported the data into Databricks using an Apache Spark and MongoDB connector to import it from MongoDB Atlas. Here we have reached the final stage of the pipeline, where we cleaned the data wherever necessary and processed it to be input into our Word2Vec model.

Preprocessing Goals, Algorithms, Time Efficiency

Using PySpark DataFrames, Spark ML, and Databricks, we preprocessed our data to be input into the Word2Vec model. We began by tokenizing all the posts and comments using the Tokenizer function in Spark ML, which splits all of the text into individual words, removes the punctuation of certain words (such as “won’t” and “can’t”), and makes them all lowercase to make it easier to process. This was followed by filtering the words using Spark ML’s StopWordsRemover to remove common, unimportant words such as “and”, “you”, and “but” in order to make our analysis more meaningful. This allowed us to focus on the keywords we needed to cluster our topics.

Once the words were all tokenized and stop words were removed, the data was ready to be input into Spark ML’s Word2Vec model. Word2Vec receives all of the words in the corpus, or in our case, the posts, and vectorizes each word in a specified dimension. We went with the default vector length of 100 values in our case. Using these vectors, we input them all into Spark ML’s KMeans algorithm with a selected K = 15, which we chose using the elbow method. This allowed us to group the data into 15 separate clusters. Finally, we pushed the clusters individually through the WordCloud library to generate a visual cluster of the most common words in that cluster. This gave us a better sense of the most common topics in the posts.

This entire process took 724.52 seconds, or roughly 12 minutes, for the posts. The longest portion of the report involved implementing the elbow method to select an ideal value for K when using the KMeans algorithm, as this alone took 10 minutes.

For comments, we primarily pushed the comment text through Natural Language Toolkit’s (NLTK) SentimentIntensityAnalyzer model to obtain their “compound” scores. These compound scores represent a comment's overall predicted sentiment and range between -1 and 1, where -1 represents negative comments, and 1 represents positive comments. We used this value to group

the comments into negative, neutral, and positive categories. This process took about 44.8 seconds, with the longest portion attributed to NLTK's SentimentIntensityAnalyzer model.

Cluster specification details:

- Databricks Runtime version: 12.1 ML (includes Apache Spark 3.3.1, GPU, Scala 2.12)
- Worker types: 32-80 GB Memory, 8-20 Cores
- Number of workers: 2-5
- Driver type: 16 GB Memory, 4 Cores

ML Goals, Outcomes, Execution Time Efficiency

The main outcomes of this project were to collect, preprocess and engineer features using Spacy, cluster topics and stories by similarity across various subreddits, build a word cloud using topic-segregated data, and conduct sentiment analysis of the topics, organize them into positive negative, and controversial lists. These objectives serve as the foundation for the ML goals and outcomes of the report. Data collection, preprocessing, and feature engineering are critical in preparing the dataset for analysis, while clustering enables the identification of patterns and relationships between subreddits. The word cloud visualizes the most common words within each topic, and sentiment analysis allows for a better understanding of the emotions and opinions expressed in each subreddit. Overall, this report's ML goals and outcomes aim to provide insights into the topics, sentiments, and relationships between subreddits to help make informed decisions.

Lessons Learned

Here are some of the lessons learned:

- Data processing and transformation: Understanding how to process and transform data using tools like Apache Airflow, Databricks, and Spark ML is critical in building an efficient and effective data engineering pipeline.
- Cloud computing: Using GCP for cloud computing can help reduce infrastructure costs and improve scalability.
- Data storage: Storing data in MongoDB Atlas can provide a scalable and highly available data store. It is important to understand how to set up and configure a MongoDB Atlas cluster and how to use it in a data engineering pipeline.
- Workflow management: Apache Airflow is an essential tool for workflow management in a data engineering pipeline. Understanding how to design and configure workflows is critical to ensuring the pipeline functions as expected.
- Machine learning: Spark ML is a powerful tool for machine learning. Understanding how to use Spark ML for tasks like classification, regression, and clustering can be an asset in building a data engineering pipeline incorporating machine learning.
- Collaboration: Collaboration and communication skills are essential when building a data engineering pipeline on a team. It is important to understand how to effectively

communicate with stakeholders and team members to ensure everyone is on the same page and the pipeline meets the project's objectives.

Conclusion

In this project, we were able to construct an Apache Airflow pipeline extracting Reddit posts and comments from the Reddit API and moving them into Google Cloud Storage and MongoDB Atlas, using Databricks and Spark ML to extract the data and complete our machine learning objectives.

As for our objectives, we were able to cluster and identify key topics and phrases that are trending in the news daily during the time that data was collected for 11 subreddits of our choosing, reflecting politics, environment, the economy, global health, global news, local news, and more. We were also able to identify the sentiments surrounding the news we captured to observe the atmosphere of the news' impact and the general sentiment of Reddit users.