Real-Time Sentiment Analysis Pipeline for Reddit Data Using Kafka

Maxime APPERT

All code and Figures are available at:

https://github.com/Maximus4321/reddit-project-kafka/tree/main

INSTITUT
POLYTECHNIQUE
DE PARIS

Introduction

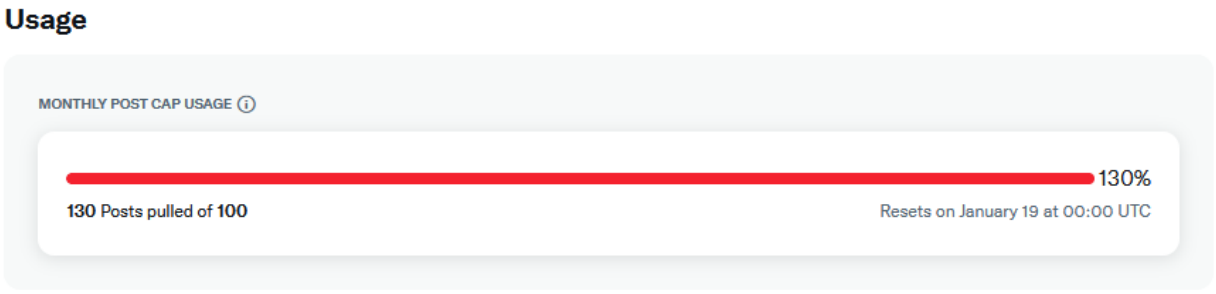
This report provides a detailed overview of my project, which consisted in creating a sentiment analysis pipeline using Kafka. Sentiment analysis is a powerful tool for understanding public opinion and identifying trends in large datasets of textual information. It enables businesses, researchers, and organizations to gauge customer satisfaction, track changes in sentiment over time, and even predict behaviors based on emotional responses. By analyzing the emotions and attitudes expressed in Reddit posts, this project aims to provide valuable insights into user opinions and facilitate data-driven decision-making.

My system processes Reddit data in real-time, performs sentiment classification, stores the created data, creates graphs and word clouds, and allows user feedback for model improvement.

Overview of Reddit and its API

Reddit is a very popular social media platform with 365.4 million weekly active users. The website is organised around communities, these communities are named subreddits. For example, there is a subreddit around the popular series breaking bad for fans to discuss it, or a subreddit called python for anyone who would need help writing a program could ask.

This made Reddit a very interesting alternative to twitter, the proposed social media for this project. Indeed, sentiment analysis could be much more easily narrowed by subreddits or keywords, something that would be harder to do with Twitter. Additionally, the Reddit API was chosen over Twitter due to limitations with Twitter's free tier, which restricts access to historical data and imposes strict rate limits, namely limiting the scraping of tweets to around 100 posts a month, which is much too small for any useful analysis.



Reddit, meanwhile, provides a more flexible API for searching and retrieving posts, while not implementing any strict limits to the amount of collected posts, making it a better fit for this project's requirements.

Overview of the Pipeline

The pipeline consists of multiple interconnected components, each responsible for a specific task:

1. **Producer Script**: Fetches Reddit posts and sends them to a Kafka topic.
2. **Consumer Script**: Processes the posts, classifies their sentiment using a machine learning model, collects user feedback on uncertain classifications, and forwards the results.
3. **Archiver Script**: Archives posts based on their sentiment.
4. **Graph Script**: Creates bar graphs visualizing sentiment counts.
5. **WordCloud Script**: Generates word clouds for textual insights.
6. **Training Script**: Trains the machine learning model used for sentiment classification.
7. **Configuration file**: Configuration file with global variables used across the scripts.

Below is a detailed explanation of each script and its role.

1. Producer Script

File: kafka_reddit_producer.py

The producer script fetches Reddit posts based on keywords and subreddits using the Reddit API and sends them to the Kafka topic raw_reddit_posts.

It first loads configuration from config.json to find the user credentials and access the Reddit API, as well as knowing what keyword and what subreddit the user wants to search through.

Then, it uses the python package PRAW (Python Reddit API Wrapper) in order to scrape recent posts made on reddit, with a specified keyword if defined, and in the specified subreddit, or globally if no subreddit is defined. The python package PRAW simplifies the search on reddit taking into account rate limiting in order to avoid overloading servers as well as simplifying the use of keywords.

The producer then sends each post as a JSON object (made up of the title of the post, the id, the text, the date of creation and the subreddit) to the Kafka topic raw_reddit_posts.

2. Consumer Script

File: kafka_reddit_consumer.py

The consumer script processes Reddit posts by classifying their sentiment and forwarding them for storage and visualization.

It first loads configuration from config.json to find the uncertainty threshold, as well as loading the model created in the sent_analysis.py file.

It consumes posts from the Kafka topic raw_reddit_posts, and then uses our pre-trained logistic regression model and its TF-IDF vectorizer, and processes each post's title and body text to classify its sentiment as positive or negative, as well as noting how confident it is in the prediction.

If the model's confidence is lower than the uncertainty threshold defined by the user, it proposes its prediction to the user and collects feedback (Yes/No) to refine future training. It automatically appends user feedback to userfeedback.csv, which will then be used on the next training of the model. The user can also set the uncertainty threshold to 0 if he does not wish to give any feedback on the results.

The script then sends the post (title of the post, the id, the text, the date of creation and the subreddit) and its classified sentiment to the Kafka topic processed_reddit_posts for further processing.

3. Archiver Script

File: archive_reddit.py

The archiver script organizes processed Reddit posts by their sentiment into separate files for storage. It consumes posts from the Kafka topic processed_reddit_posts, separates posts based on their sentiment into positive_posts.json and negative_posts.json, and appends each post incrementally to the respective JSON file.

4. Graph Script

File: graph_reddit_sent_analysis.py

The graph script visualizes the sentiment distribution of Reddit posts by generating bar graphs, including the keyword and subreddit if they are used.

It first loads the keyword and subreddit from the configuration file if they are available, and then it consumes posts from the Kafka topic processed_reddit_posts and maintains a running count of positive and negative sentiments as posts are processed. At regular intervals, it generates and saves bar graphs illustrating the sentiment distribution in the graphs folder.

These graphs are useful for understanding the overall sentiment trends in certain subreddits. For example, communities centered around current issues (COVID, rant etc…) will have

significantly more negative posts than positive ones, but this can also be applied to more time centric events, for example, if a long standing series has a particularly good episode that has recently been released, this would be illustrated in these sentiment bar graphs as would have more positive posts.

5. WordCloud Script

File: kafka_wordcloud.py

The word cloud script provides textual insights by creating word clouds based on the content of Reddit posts.

It first loads the keyword and subreddit from the configuration file, and then it consumes posts from the Kafka topic processed_reddit_posts. It then segregates these posts based on the sentiment classification (positive or negative), and it generates a word cloud for each sentiment category, combining them into one single image and saving the generated word clouds as image files in the word clouds folder.

The word clouds provide insights by creating images that are particularly useful for identifying common themes and keywords in specific subreddits. For example, it can highlight the most frequently used words in positive posts, showing what users value the most in a specific community, and it can also reveal patterns in negative posts, which could be crucial for understanding user dissatisfaction or complaints. The word cloud offers an intuitive way to grasp the overarching topics without delving into individual posts.

6. Training Script

File: sent_analysis.py

This training script is used to build the sentiment classification model used in the consumer script. It first loads the Sentiment140 dataset, which is a dataset of labeled tweets for sentiment analysis, containing both positive and negative sentiments. It combines this dataset with the userfeedback.csv file, which includes any feedback provided by users along with the corresponding sentiments. This combination allows the model to adapt and improve based on user-provided corrections, making it more accurate over time.

The script then preprocesses the text data by performing tokenization to split sentences into words, stemming to reduce words to their base forms, and stop-word removal to eliminate common but uninformative words (e.g., "and," "the"). Next, it converts the cleaned text data into numerical feature vectors using the TF-IDF vectorizer, capturing the importance of each word relative to the dataset.

After preprocessing, the script trains a logistic regression model on the processed data. It evaluates the model's performance using the accuracy metric to ensure the model is robust and reliable. The model is designed to generalize well to unseen Reddit posts while incorporating user feedback for enhanced adaptability. Finally, the script saves the trained logistic regression model and the TF-IDF vectorizer to files, making them available for the consumer script..

7. Configuration File

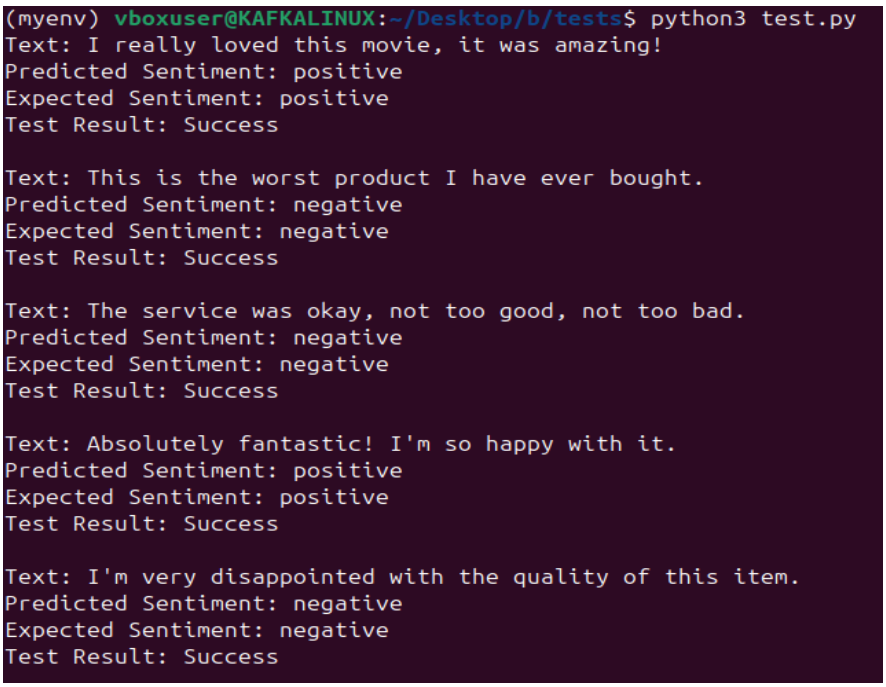
File: config.json

The configuration file centralizes parameter settings for all scripts across the pipeline. It specifies the keyword to search for in Reddit posts (reddit_keyword), the subreddit to search within (reddit_subreddit), and the threshold for determining whether a sentiment prediction is uncertain (uncertainty_threshold), which is used for user feedback. It also includes the authentication credentials (reddit_client_id, reddit_client_secret, reddit_user_agent) required to access the Reddit API.

Experiments

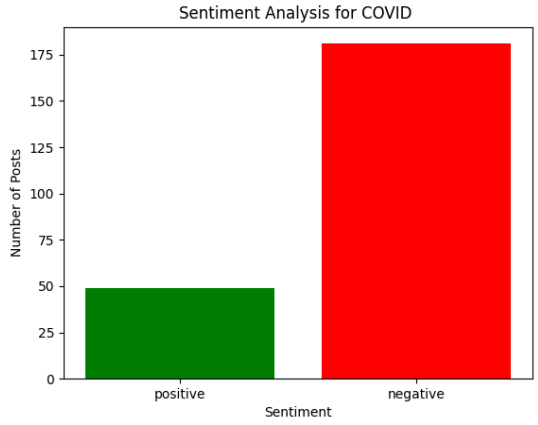
Initial Model Testing

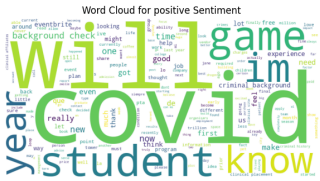
The first experiments focused on testing the sentiment classification model itself. Using the test.py script, I ran tests on various texts with obvious sentiments to verify that the model correctly identified positive and negative sentiments. These tests confirmed the model's functionality, as the outputs aligned with expectations.

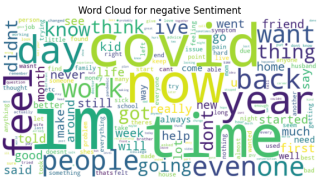


Experiment 1: COVID Keyword Across All Subreddits

I conducted an experiment searching for the keyword "COVID" across all subreddits. As expected, there was a significantly higher proportion of negative sentiments compared to positive ones. The word cloud for positive sentiments included terms like "student", potentially showing students who enjoyed time off from school. Conversely, the negative word cloud featured terms such as "year", "house", "friend", and "work", indicating frustrations or challenges faced during the pandemic.

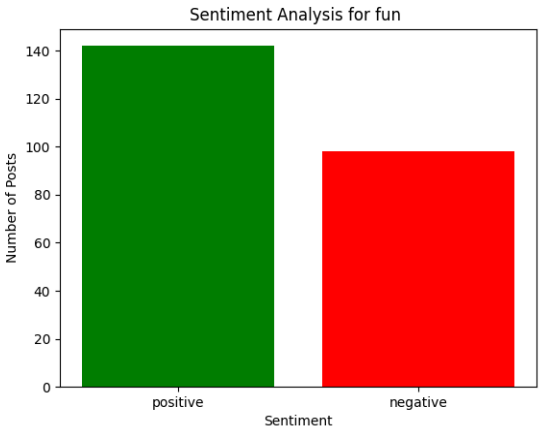


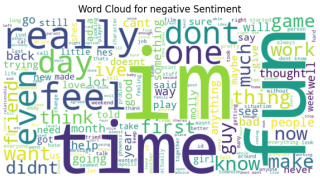




Experiment 2: Fun Keyword Across All Subreddits

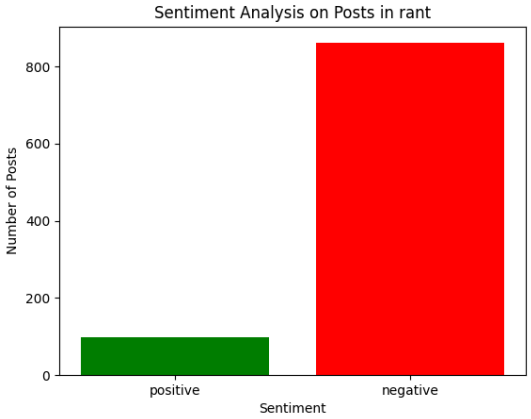
For the second experiment, I searched for the keyword "fun" across all subreddits. This search yielded more positive sentiments than negative ones, as expected. The positive word cloud included words like "game", "play", and "new", showing people enjoying positive activities. On the other hand, the negative word cloud had terms such as "don't", "feel", and "even", possibly reflecting people who are going through difficult periods.

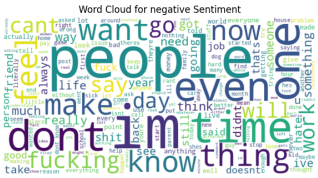




Experiment 3: Posts in the Rant Subreddit

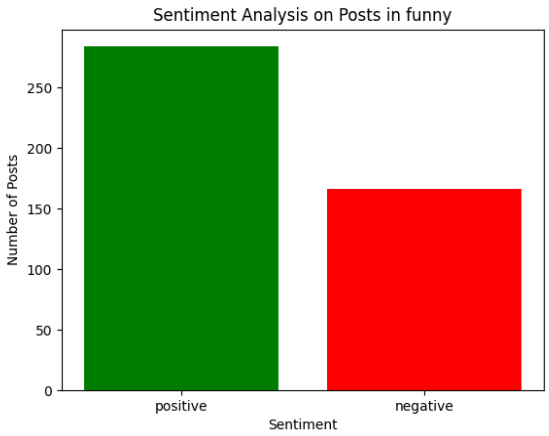
In the third experiment, I searched for all posts within the "rant" subreddit. As anticipated, the results showed a much higher proportion of negative sentiments compared to positive ones. Both positive and negative word clouds highlighted terms such as "people", "don't", "know", "time", and "thing", showing common themes of frustration and venting within this community.

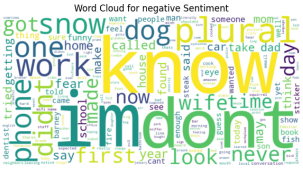




Experiment 4: Posts in the Funny Subreddit

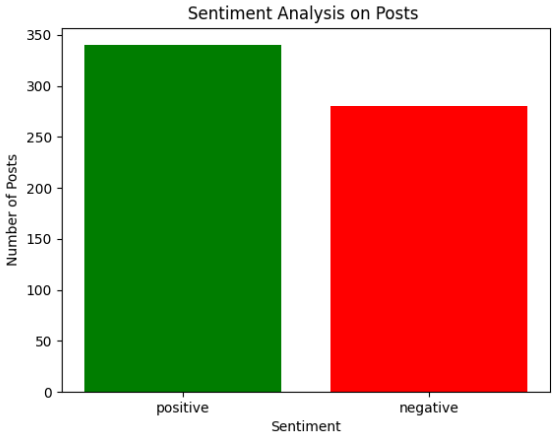
The fourth experiment involved searching for all posts in the "funny" subreddit. This subreddit, as expected, showed a much higher number of positive sentiments compared to negative ones. The bar graph supported this finding, and the word clouds provided additional insights. The positive word cloud featured terms like "dog", "kid", and "friend", showing themes of family and joy. On the other hand, the negative word cloud included words like "work", "never", "don't", and "know", which may reflect frustrations or moments of irony within an otherwise humorous context.

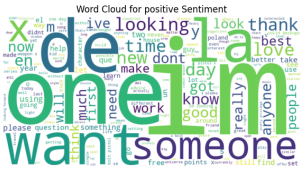




Experiment 5: All Posts Across All Subreddits

For the final experiment, I analyzed all posts across all subreddits without filtering by keywords. The results showed an almost equal distribution of positive and negative sentiments. The positive word cloud included terms like "new," "love," "want," and "thank," showing themes of gratitude and optimism. Conversely, the negative word cloud featured terms like "don't," "need," "year," and "work," highlighting themes of stress and dissatisfaction.





Conclusion

The sentiment analysis pipeline developed in this project has demonstrated strong performance across various use cases. The system effectively classifies Reddit posts in real-time, produces insightful visualizations such as graphs and word clouds, and incorporates user feedback to improve the model's accuracy over time. Additionally, the experiments confirmed that the model performs as expected in different contexts.

While the pipeline is functional and provides meaningful results, there is still room for improvement. For instance, enhancing the feedback mechanism with a more dynamic user interface could make it more intuitive and efficient. Additionally, further experimentation with advanced machine learning models, could improve sentiment classification accuracy, especially in edge cases, as the logistic regression model is a very simple one. Despite these areas for growth, the current implementation is a robust framework that performs well on large-scale textual data.