Sentiment Analysis on Twitter Data using Apache Spark

Professor : GIANCARLO SPERLI

Author:

Mohammadhossein Darabi

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

1. Introduction and Overview

People share their thoughts and emotions on social media platforms like Twitter every day. These short texts are often filled with opinions, reactions, or feelings about products, events, and experiences. Being able to automatically understand the tone or sentiment behind such messages is valuable for businesses, researchers, and developers alike. This task is known as sentiment analysis — the process of determining whether a piece of text expresses a positive or negative emotion.

This project focuses on classifying tweets into two categories: positive and negative. We use the Sentiment140 dataset, which includes 1.6 million tweets already labeled with sentiment. This makes it possible to train machine learning models to recognize patterns and make predictions on new tweets.

Because the dataset is large, we use Apache Spark — a powerful big data processing framework — to handle everything efficiently. The entire workflow is implemented on Databricks, a platform that supports working with Spark in the cloud.

We start by cleaning the tweet text, turning it into numbers using a technique called TF-IDF, and then train two models: Logistic Regression and Naive Bayes. We measure how well these models work using common metrics like accuracy and F1 score.

This report explains each step clearly, from preparing the data to evaluating the models, showing how Spark can help with real-world language problems.

2. Task Definition

The main goal of this project is to understand what people feel based on what they write on Twitter. We take a large number of tweets and try to figure out if the person who wrote each one is feeling positive (happy, excited) or negative (sad, frustrated).

To do this, we use machine learning. We first clean the tweet text to remove things like links and special characters. Then we turn the words into numbers using a method called TF-IDF, which helps the computer understand which words are important.

After that, we train two models: one using Logistic Regression and the other using Naive Bayes. These models learn from a part of the tweets, and then we test how well they can predict the sentiment of tweets they haven't seen before.

The task is to accurately classify each tweet into one of two categories:

- Positive (label 1)
- Negative (label 0)

By comparing the results of the two models, we can see which one does a better job at this classification task.

---

3. Data Description

At the beginning of the analysis, we renamed the columns of the dataset to more meaningful names to make the data easier to understand and work with. This step helped us access and reference the data more clearly throughout the project. We also checked the dataset for missing or empty values in the key columns, especially the tweet text and sentiment label. Tweets with null or blank text were removed to ensure clean and complete inputs for further processing.

We used the Sentiment140 dataset, which contains 1.6 million tweets. Each tweet is labeled to show whether the sentiment is positive or negative. The dataset includes the following information:

- `target`: Sentiment label (0 = negative, 4 = positive)
- `id`: Unique tweet ID
- `date`: The date the tweet was posted
- `query`: The search term used (can be blank)
- `user`: The username of the person who tweeted
- `text`: The full tweet text

For our task, we only used the `target` and `text` columns. All other columns were kept but not used for training. We noticed that the dataset was perfectly balanced, with:

- 800,000 tweets labeled as negative (0)
- 800,000 tweets labeled as positive (4)

There were some tweets with empty or null values in the text field, which we removed before continuing. After filtering, we continued with the cleaned and complete records.

---

4. Text Cleaning and Preprocessing

Text preprocessing is a crucial step in preparing the text for machine learning models. It ensures that the input is clean, consistent, and meaningful. In our project, we included several core preprocessing techniques, and we also expanded it to include additional steps often used in natural language processing tasks. It included both traditional and expanded steps suitable for large-scale sentiment analysis. Below is the exact transformation pipeline used in the project:

Before training any models, we cleaned the tweet text to remove noisy and irrelevant information that could affect the learning process. This step is crucial in Natural Language Processing (NLP) because real-world text often includes unstructured elements such as symbols, links, and emojis.

We applied the following preprocessing steps:

- **Lowercasing**: All tweet text was converted to lowercase to avoid treating the same words in different cases as separate tokens.
- **URL and Mention Removal**: We removed links (e.g., http://...) and Twitter mentions (@username), which do not carry sentiment value.
- **Hashtag and Punctuation Removal**: Hashtags and punctuation were stripped using regular expressions.
- **Whitespace Trimming**: Extra and irregular spaces were replaced with single spaces.
- **Tokenization**: Text was split into words using whitespace-based tokenization.
- **Stopword Removal**: Common stopwords (e.g., 'and', 'the', 'is') were removed using PySpark's `StopWordsRemover` to reduce noise.
- **Vectorization (TF-IDF)**: We used Term Frequency-Inverse Document Frequency to transform the tokenized text into numeric features.

These steps were implemented directly in the Spark pipeline, ensuring they were scalable and compatible with the Databricks environment.

This preprocessing step helped ensure that the input to the models was clean, consistent, and in a format suitable for numerical analysis.

4. Analysis of the Classification Process

After cleaning the tweets and transforming the text into numerical features using TF-IDF, we trained and evaluated two different classifiers: Logistic Regression and Naive Bayes.

We split the dataset into training and testing sets with an 80/20 ratio:

- Training set size: 1,280,209 tweets
- Testing set size: 319,791 tweets

Each model was evaluated using the following performance metrics:

Logistic Regression Results:

- Accuracy: 0.7796
- F1 Score: 0.7795
- Precision: 0.7797
- Recall: 0.7796

Naive Bayes Results:

- Accuracy: 0.7637
- F1 Score: 0.7637
- Precision: 0.7638
- Recall: 0.7637

To better understand the model's predictions, we also looked at the confusion matrix for each classifier, which showed how many tweets were correctly and incorrectly classified as positive or negative. Additionally, we tested the model on a few custom tweets and observed the predicted sentiment.

Both models performed well, with Logistic Regression achieving slightly higher overall accuracy and balance across all metrics. Naive Bayes was faster to train and can still be useful for simpler or resource-constrained scenarios.

This comparison provides insight into how different machine learning models can be used to interpret the sentiment behind millions of short texts like tweets.

5. Model Comparison

To get a clearer picture of how both models performed, we compared their results side-by-side across four main metrics: accuracy, F1 score, precision, and recall. This helped us understand not just which model was more accurate, but also which one handled class balance better.

Metric	Logistic Regression	Naive Bayes
Accuracy	0.7796	0.7637
F1 Score	0.7795	0.7637
Precision	0.7797	0.7638
Recall	0.7796	0.7637

From this comparison, we can see that Logistic Regression slightly outperforms Naive Bayes in every metric. However, Naive Bayes still produces solid results and is much faster to train, which can be helpful when speed or simplicity is a priority.

A bar chart was also created to visualize this comparison and make the differences easier to spot.

---

6. Conclusion

In this project, we successfully built a working sentiment analysis system using Apache Spark and the Sentiment140 dataset. We went through the full pipeline: loading the data, cleaning the tweets, turning the text into numbers, training models, and evaluating how well they performed.

We tested two different models: Logistic Regression and Naive Bayes. Logistic Regression gave us the best results across all the key metrics — accuracy, precision, recall, and F1 score — while Naive Bayes was a bit faster and simpler to run. This shows that even basic machine learning models can give good results if the data is clean and well-prepared.

By doing everything in Spark and using Databricks, we were able to process a large dataset smoothly. This approach can easily scale to more data or more complex models if needed.

Overall, this project shows that it's possible to analyze millions of tweets and figure out what people are feeling — all with just some thoughtful preprocessing, simple models, and the power of distributed computing.