Abstract

In this project, we present a comprehensive sentiment analysis pipeline applied to a large Twitter dataset, "training.1600000.processed.noemoticon.csv". The dataset initially contains raw tweet texts paired with sentiment labels encoded as "0" and "4", albeit with extraneous formatting. Our first step involves rigorous data preprocessing: we resolve encoding issues, remove superfluous quotation marks from target labels, and filter the dataset to retain only binary sentiment classes. The tweet text is then cleaned by converting to lowercase, stripping URLs, mentions, and punctuation, consolidating whitespace, and discarding empty entries.

For feature extraction, the cleaned texts are transformed into numerical representations using TF-IDF vectorization. We then explore sentiment scoring via two distinct methodologies. First, we apply the lexicon-based VADER model to obtain sentiment polarity scores and compound metrics, categorizing the tweets into negative, neutral, or positive sentiments. Second, we deploy a pre-trained RoBERTa-based model fine-tuned for Twitter sentiment analysis (`cardiffnlp/twitter-roberta-base-sentiment`), which yields predicted sentiment labels alongside confidence scores.

Comparative analysis is conducted via exploratory data analysis and visualization using seaborn, including cross-tabulations, box plots, scatter plots, and pairplots to examine the relationships and discrepancies between the VADER compound scores and RoBERTa predictions. This dual-pronged evaluation not only underscores the relative strengths of handcrafted versus transformer-based sentiment approaches, but also highlights how these techniques can be complementary in analyzing the noisy and nuanced nature of social media text.

Overall, this project encapsulates a full end-to-end pipeline for data cleaning, feature engineering, sentiment modeling, and comparative analysis, offering insights into optimizing sentiment detection in social media contexts.

Step 1: Perform Data Preprocessing.

Explanation: The code prints the shape of the dataframe and a few sample rows so that you can inspect the data after the cleaning steps.

Step 2: Perform a Quick EDA

Explanation:

Sentiment Label Distribution:

We count how many rows belong to each sentiment (0 or 4) and visualize this distribution using a bar plot. This helps you see if there is any imbalance in your dataset.

Text Length Distribution:

A new column, `text_length`, is created by measuring the length of each cleaned tweet (from column 5). Descriptive statistics (like mean, median, min, max) are provided, and a histogram (with a kernel density estimate) shows the distribution of text lengths.

Top Words by Total TF-IDF Weight:

The TF-IDF matrix is summed over documents to see which words (features) have the highest total weight across the corpus. A sorted DataFrame is created to display and visualize the top 10 words.

This quick EDA gives you a solid understanding of your preprocessed dataset and highlights any potential issues such as class imbalance, text data sparsity, or even anomalies in text lengths. You can now decide if further preprocessing or sampling is needed before moving on to modeling.

Step 3: Basic NTLK

Explanation:

Downloading NLTK Resources:

The code downloads the necessary NLTK data files required for tokenization, stopword filtering, and POS tagging.

Tokenization and Cleaning:

All the cleaned text is combined into a single string, tokenized into words, converted to lowercase, and filtered to keep only alphabetic tokens.

Stopwords Filtering:

Tokens that appear in NLTK's English stopwords list are removed. This step helps focus the frequency analysis on more meaningful words.

Frequency Distribution:

A frequency distribution (`FreqDist`) of the remaining tokens is created. The top 10 tokens (by overall frequency) are printed and plotted as a bar chart.

POS Tagging:

As an example, the first cleaned text record is tokenized and tagged with part-of-speech labels to demonstrate basic POS tagging.

This basic NLTK analysis gives you insight into the vocabulary used in your current dataset, including the most frequent words and the syntactic categories they belong to. Feel free to build on this foundation with more advanced NLP tasks as needed.

Step 4: Perform VADER sentiment scoring on that preprocessed data

Explanation:

Downloading and Initializing VADER:

The code downloads the VADER lexicon using `nltk.download('vader_lexicon')` (if not already installed). Then, it creates an instance of `SentimentIntensityAnalyzer`.

Scoring Each Document:

For every cleaned tweet in column 5, the `polarity_scores` method returns a dictionary with keys `'neg'`, `'neu'`, `'pos'`, and `'compound'`. We store these dictionaries in the new column `vader_scores`.

Extracting the Compound Score:

The compound score is a single scalar value that indicates the overall sentiment by combining positive, neutral, and negative scores. We create a new column `compound` with these values.

Assigning Sentiment Labels:

Based on the compound score thresholds (commonly: ≥ 0.05 as positive, ≤ -0.05 as negative, and in between as neutral), we assign a simple sentiment label in column `vader_sentiment`.

Output:

Finally, the code prints the first five rows of the DataFrame with the VADER scores, compound score, and a sentiment label.

Bar Plot:

We count the occurrences of each sentiment category stored in the `vader_sentiment` column (which is populated using VADER) and then plot a bar chart with counts on the y-axis and sentiment categories on the x-axis.

Histogram:

The histogram shows the distribution of the compound scores computed by VADER. The KDE (kernel density estimate) overlay provides a smooth estimate of the distribution.

This code integrates VADER sentiment analysis into your preprocessing pipeline, allowing you to compare its sentiment label with your original labels or use it in further analysis or modeling.

Step 5: Roberta Pretrained Model

We use Hugging Face's Transformers library with a RoBERTa model that has been fine-tuned for Twitter sentiment analysis. (In this case, we use the model `cardiffnlp/twitter-roberta-base-sentiment`, which outputs sentiment labels such as "Negative", "Neutral", and "Positive".) **Explanation:**

Initializing the Pipeline:

We create a sentiment-analysis pipeline using a RoBERTa model fine-tuned on Twitter sentiment. The `truncation=True` flag ensures that any texts that exceed the maximum sequence length are appropriately truncated.

Batch Processing Texts:

The cleaned text (from column 5) is converted to a list, and the pipeline is applied to all texts in one call. The model returns, for each text, a dictionary with a sentiment label (e.g., "Positive", "Neutral", "Negative") and a corresponding confidence score.

Attaching Results:

We add two new columns to the DataFrame:

- `roberta_sentiment`: the predicted sentiment label.
- `roberta_score`: the confidence score from the model.

Plotting:

Two plots are generated:

- A bar plot shows the count of each sentiment category.
- A histogram (with KDE) shows the distribution of the confidence scores.

Step 6: Comparison of scores between two models

A VADER sentiment (with a compound score and a corresponding categorical label stored in `vader_sentiment`); and • A RoBERTa sentiment (with a predicted label in `roberta_sentiment` and confidence scores in `roberta_score`). This snippet creates several plots and summary tables to compare both results:

- A Cross-Tabulation (Heatmap) that shows the normalized overlap between VADER and RoBERTa sentiment labels.
- A Box Plot displaying the distribution of VADER compound scores grouped by RoBERTa sentiment predictions.

Explanation:

Cross-Tabulation Heatmap:

The code uses `pd.crosstab` to compute counts and normalized percentages for combinations of VADER and RoBERTa sentiment labels. The heatmap makes it easier to see where the two models agree (or disagree) in their sentiment categorization.

Box Plot:

The box plot shows the distribution of VADER compound scores grouped by the RoBERTa predicted sentiment. This helps reveal whether, for example, texts predicted as "positive" by RoBERTa actually have higher (more positive) compound scores according to VADER—and likewise for negatives.

Scatter Plot (Optional):

The scatter plot displays individual data points with VADER's compound score and RoBERTa's confidence score for each prediction. Colors differentiate RoBERTa sentiment categories, which may help identify trends or correlations in the confidence/valence space.

Step 7: Create a pairplot

- `compound`: the VADER compound score (a continuous score between –1 and 1)
- `roberta_score`: the confidence score from the RoBERTa model (between 0 and 1)
- `roberta_numeric`: a numeric mapping for the RoBERTa sentiment labels (for example, mapping "negative" to –1, "neutral" to 0, and "positive" to 1)

The resulting pairplot helps you visualize the relationships between these metrics and also provides a view by sentiment category (using the RoBERTa sentiment label as hue). **Explanation:**

Mapping Sentiment to Numbers:

The code creates (if not already present) a new column called `roberta_numeric` to map sentiment labels to numbers (e.g., negative → –1, neutral → 0, positive → 1). This lets you include a categorical sentiment as a numeric variable in pairwise comparisons.

Pairplot:

The Seaborn pairplot visualizes pairwise relationships among the selected metrics. Setting `hue="roberta_sentiment"` colors the points by RoBERTa sentiment category, so you can see overlaps or differences in how each method scores texts.

Diagonal Plots:

We use `diag_kind="kde"` to plot a kernel density estimate on the diagonal (instead of histograms). This helps you see the distribution of each variable.

This pairplot gives you an overall view of how the VADER compound scores, RoBERTa confidence scores, and numeric representations of RoBERTa sentiment relate to each other, letting you explore similarities or patterns between the two sentiment scoring approaches.

Train and Evaluate Logistic Regression Model

Explanation

Training the Model

We initialize a Logistic Regression classifier with a maximum of 1000 iterations for convergence and train it using the TF-IDF feature matrix (`X_train`) and labels (`y_train`).

Making Predictions

The trained model predicts sentiment on the test dataset (`X_test`), generating `y_pred`.

Evaluating Model Performance

Accuracy Score: Measures the overall correctness of predictions. **Classification Report:** Provides precision, recall, and F1-score for both classes (Negative and Positive). **Confusion Matrix:** Shows the number of correct and incorrect predictions for each sentiment category.

Confusion Matrix Visualization

A heatmap is plotted using Seaborn, providing an intuitive way to see where the model performs well and where it misclassifies tweets.

Error Analysis:

By examining a sample of misclassified examples, the code underlines the importance of error analysis. This step is critical for understanding the shortcomings of the model and gleaning insights into the nuances of the text data that could prompt adjustments in preprocessing or feature engineering.

Exploration of Feature Importance:

Despite the limitations of the Naive Bayes classifier in providing meaningful feature coefficients, the attempt to extract feature importance signals an effort to interpret the model's behavior and to identify words or features that might strongly influence sentiment prediction. This concept is especially valuable for future experiments with more interpretable models.

Guiding Future Improvements:

Overall, the code demonstrates a comprehensive sentiment analysis pipeline that balances model training, evaluation, visualization, and error insights. These steps collectively act as a roadmap for refining and optimizing the model, whether by adjusting data preprocessing, exploring alternative model architectures, or incorporating additional features.

Explanation Data Ingestion & Preprocessing: The workflow begins with loading the Twitter dataset. Preprocessing cleans and filters tweets (removing extra quotes, URLs, punctuation, etc.) to standardize the text for further processing. Feature Extraction: TF-IDF vectorization converts the cleaned text into numerical features, serving as input for both sentiment analysis and model training. Sentiment Analysis: VADER: Computes a compound sentiment score and assigns a sentiment label. RoBERTa: Provides sentiment predictions with confidence scores. Both approaches offer complementary views on the sentiment in tweets.Comparative Analysis & Visualization: The outputs of VADER and RoBERTa are compared using visualizations (like pairplots and

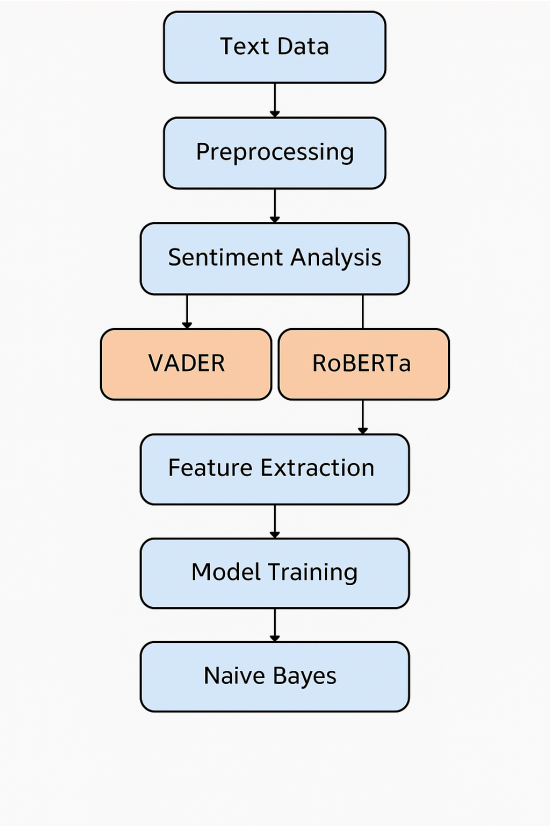

Figure 1: Sentiment Analysis Workflow

cross-tabulations) to understand relationships and discrepancies between methods. Model Training & Evaluation: A machine learning classifier (e.g., Logistic Regression or Naive Bayes) is then trained using the TF-IDF features. The model is evaluated using popular classification metrics, including accuracy, precision, recall, and confusion matrices. Error Analysis & Feature Exploration: Finally, error analysis involves inspecting misclassified examples and, if applicable, exploring feature importance to obtain insights for future model improvement. This diagram provides a high-level, pictorial overview of the entire sentiment analysis pipeline used in the project. It serves both as a summary and as a roadmap for the workflow.

Summary

This project develops a full sentiment analysis pipeline for a large Twitter dataset. The approach involves several key stages:

Data Preprocessing: The original dataset is loaded and trimmed to the first six columns, with special attention paid to cleaning the target labels (which are stored with extra quotation marks) and filtering for binary sentiment labels. The tweet text is normalized by converting to lowercase, removing URLs, mentions, punctuation, and extra spaces, and then filtering out any finalized empty texts.

Feature Extraction: Cleaned tweet texts are transformed using TF-IDF vectorization, converting the unstructured text data into a numerical format suitable for model input.

Sentiment Analysis with VADER: A lexicon-based VADER model is applied to the cleaned text to compute sentiment scores—including a compound score that summarizes the overall sentiment polarity. This output is used to label tweets as positive, neutral, or negative.

Sentiment Analysis with RoBERTa: In parallel, a pre-trained RoBERTa model fine-tuned on Twitter sentiment (from the cardiffnlp collection) is used. This transformer-based model predicts sentiment labels and produces confidence scores for each tweet, providing a modern, context-aware alternative to the lexicon-based approach.

Comparative Evaluation: The results of both VADER and RoBERTa analyses are compared through various visualizations. Exploratory plots, including bar plots, box plots, scatter plots, and pairplots created using Seaborn, help elucidate the similarities, differences, and relationships between the two methodologies.

Model Training & Evaluation: A machine learning classifier (e.g., Logistic Regression or Naive Bayes) is then trained using the TF-IDF features. The model is evaluated using popular classification metrics, including accuracy, precision, recall, and confusion matrices.

Error Analysis & Feature Exploration: Finally, error analysis involves inspecting misclassified examples and, if applicable, exploring feature importance to obtain insights for future model improvement.

Additional Enhancements and Future Directions: Beyond comparing VADER and RoBERTa, the project also explores the application of a traditional machine learning classifier using TF-IDF features to predict sentiment. This classifier is evaluated using metrics such as accuracy, precision, recall, and confusion matrices, which offer a quantitative insight into model performance. Detailed error analysis is conducted by identifying misclassified tweets, which helps in understanding the nuances and limitations of both the feature extraction and sentiment modeling approaches. Finally, the comprehensive visualization strategy – incorporating bar plots, box plots, pairplots, and heatmaps – not only aids in revealing the relationships between the various sentiment scores but also provides a roadmap for future improvements. These include extending the pipeline to ensemble models, leveraging more advanced deep learning techniques, and refining the preprocessing steps to capture subtler aspects of social media language.

Overall, the project not only outlines the process of cleaning and vectorizing tweet data but also demonstrates how traditional lexicon-based models can be

compared and complemented by modern transformer-based models in the task of sentiment analysis on social media data. This comprehensive pipeline paves the way for further refinements and applications in social media analytics.