Roticha /
Sentiment-analysis

Code Pull requests Actions Projects Security Insights Settings

0 stars 4 forks 0 watching Branches Activity
Tags

Public repository · Forked from gtisnanga/GRP-13-SENTIMENT-ANALYSIS-APPLE-AND-GOOGLE-PRODUCT--CAPSTONE-PROJECT

1 Branch 0 Tags Go to file t Go to file + Add file Code

This branch is 5 commits ahead of `gtisnanga/GRP-13-SENTIMENT-ANALYSIS-APPLE-AND-GOOGLE-PRODUCT--CAPSTONE-PROJECT:master` .

Contribute Sync fork

Roticha Change file name 1cf56b1 · 1 minute ago

Sentiment_flow_app	final app changes	last week
.DS_Store	update readme	last week
FINAL.ipynb	Typo correction	2 days ago
Presentation.pdf	Presentatin in PDF	3 minutes ago
judge_tweet_product_compa...	initial commit	2 weeks ago
notebook.pdf	edit notebook pdf name	yesterday
readme.md	Update readme.md	4 days ago

README

Sentiment Flow – Understanding Twitter Sentiment on Apple and Google Products

Business Understanding

Introduction

In today's fast-paced digital world, public opinion on products plays a significant role in shaping brand perception. Companies increasingly rely on **Natural Language Processing (NLP)** to analyze real-time customer feedback. This project applies NLP techniques to classify Twitter sentiment related to Apple and Google products, addressing the need for understanding public sentiment in a rapidly evolving market. By using sentiment polarity classification, we provide actionable insights into customer satisfaction and emerging issues. These insights enable companies, marketing teams, and decision-makers to make data-driven decisions, helping brands like Apple and Google improve their products, refine customer support strategies, and optimize marketing efforts based on social media sentiment.

Problem Statement

The primary challenge is to accurately classify the sentiment of tweets related to **Apple** and **Google products**. The goal is to determine whether a tweet expresses **positive**, **negative**, or **neutral** sentiment. This classification will help companies gauge customer satisfaction, identify potential issues, and tailor their responses accordingly.

Stakeholders

- **Apple & Google**: As the companies most affected by sentiment, it is crucial for them to understand public perception of their products in order to identify areas for improvement.
- **Marketing Teams**: Sentiment analysis can help marketing teams respond to negative feedback, adjust campaigns, and emphasize positive aspects of their products.
- **Customer Support Teams & Decision Makers**: Sentiment analysis will enable these teams to improve product development, customer support, and brand reputation management.

Business Value

By accurately classifying tweets, our NLP model provides actionable insights for stakeholders, such as:

- **Identifying negative sentiment**: This allows companies to address issues promptly.
- **Recognizing positive sentiment**: This guides marketing efforts and helps reinforce successful strategies.
- **Understanding neutral sentiment**: This provides context and balance for decision-making.

Objectives

Main Objective

The goal is to develop an **NLP (Natural Language Processing)** multiclass classification model for sentiment analysis, aiming to achieve an **accuracy of 80%** and a **recall score of 80%**. The model should categorize sentiments into three classes: **Positive**, **Negative**, and **Neutral**.

Specific Objectives

- Identify the most common words used in the dataset using a word cloud.

- Confirm which words are most commonly associated with positive and negative sentiment.
- Identify the products mentioned in user opinions.
- Analyze the distribution of sentiments in the dataset.

Conclusion

Our NLP model will provide valuable insights into Twitter sentiment regarding Apple and Google products. Stakeholders can leverage this information to make better decisions and improve customer satisfaction.

Data Understanding

Data source

The dataset is sourced from **CrowdFlower via data.world**, where contributors evaluated tweets related to various brands and products. Specifically:

- Each tweet was labeled to indicate whether it expressed **Positive**, **Negative**, or **Neutral** emotion toward a brand or product, or if the sentiment was unclear ("I can't tell").
- If emotion was expressed, contributors also identified the target brand or product.

Suitability of the Data

This dataset is well-suited for our project because:

- **Relevance**: The data aligns directly with the business problem of understanding Twitter sentiment for Apple and Google products.
- **Real-World Context**: The tweets represent real user opinions, making the analysis highly relevant.
- **Multiclass Labels**: We can build binary (positive/negative) and multiclass (positive/negative/neutral) classifiers.

Dataset Size

The dataset contains **over 9,000 labeled tweets**. We'll explore its features to gain insights.

Descriptive Statistics

- **tweet_text**: The content of each tweet.
- **is_there_an_emotion_directed_at_a_brand_or_product**: No emotion toward brand or product, Positive emotion, Negative emotion, I can't tell
- **emotion_in_tweet_is_directed_at**: The brand or product mentioned in the tweet.

Feature Selection

Tweet text is the primary feature. The emotion label and target brand/product are essential for classification.

Data Limitations

- **Label Noise**: Subjectivity in human ratings may introduce some noise into the labels.

- **Imbalanced Classes**: Class imbalance might exist, which will need to be addressed during model training.
- **Contextual Challenges**: Tweets are often short and context-dependent, making sentiment analysis more complex.
- **Missing Data**: Some missing or incomplete data could impact model performance.

4.Data Cleaning & Feature Engineering

Data Cleaning

- **Corrupted records**: We identified and removed corrupted records using the `is_corrupted` function, which filtered out non-ASCII characters.
- **Neutral sentiment adjustment**: We replaced "No emotion toward brand or product" with "Neutral emotion" for consistency.
- **Dropped irrelevant records**: We removed tweets labeled as "I can't tell" from the dataset.
- **Missing values**: We dropped rows with missing `tweet_text` and filled missing values in the `emotion_in_tweet_is_directed_at` column by identifying products mentioned in the tweets.
- **Duplicates**: Duplicates were removed, and the dataset was reset for consistency.

Data Completeness & Consistency:

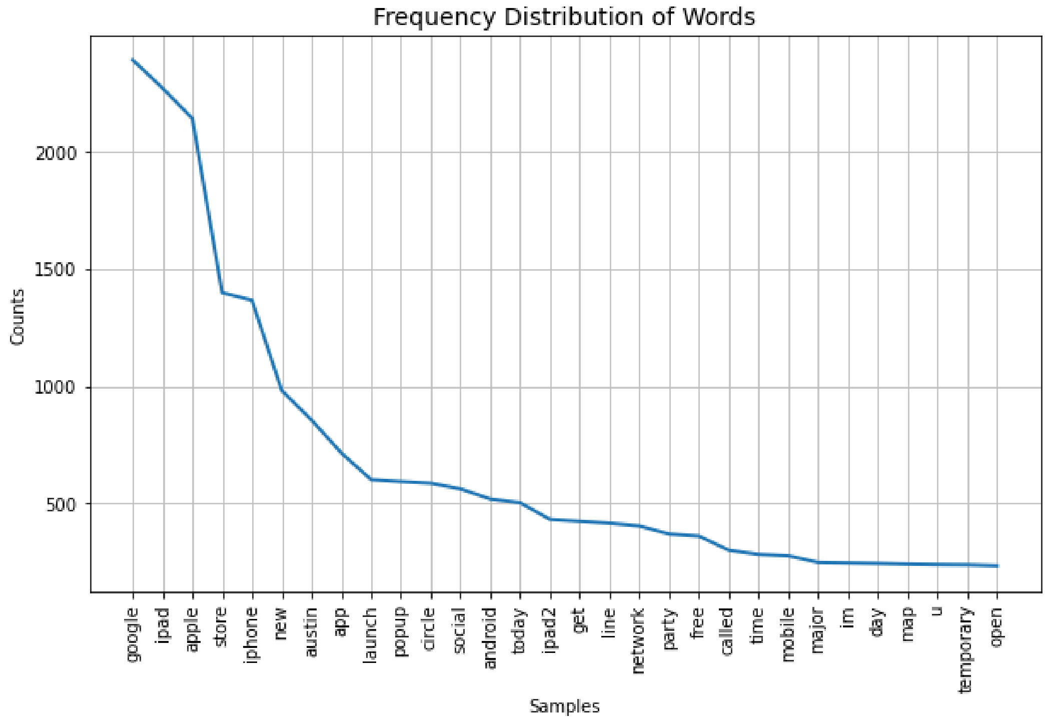
- The final dataset contains **8,439 rows**, with no missing values or duplicates. All columns have consistent naming and content.

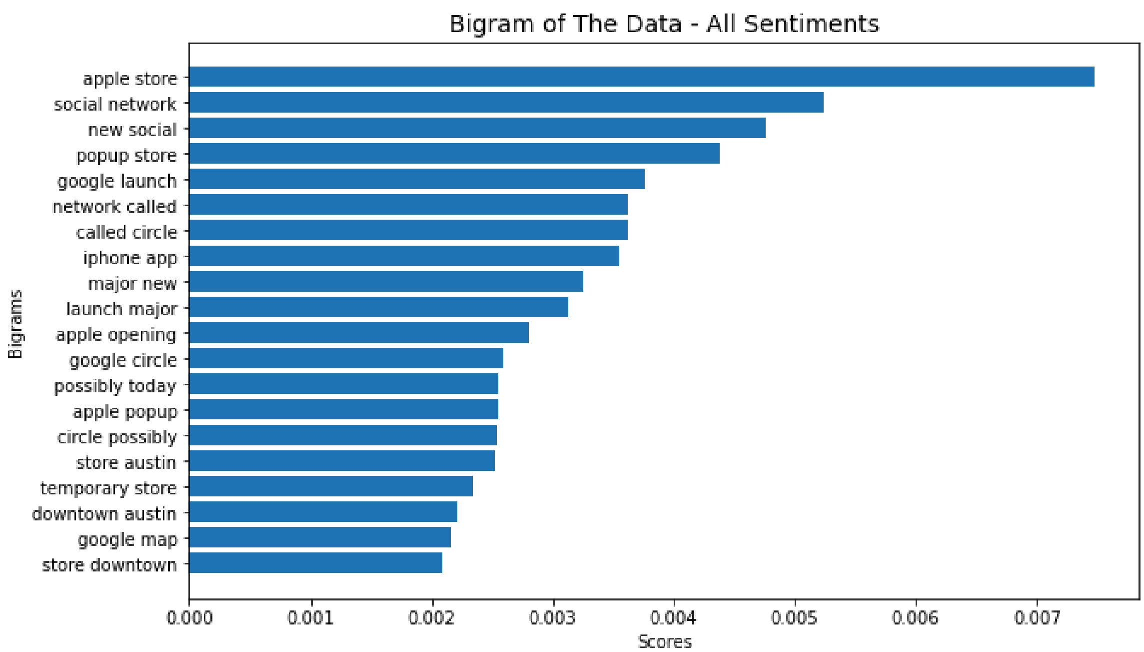
Text Preprocessing:

- We applied preprocessing steps including **lemmatization**, **stop word removal**, **tokenization**, and **part-of-speech tagging**.
- Cleaned tweets were stored as lemmatized tokens in a new column, with the final cleaned text saved in the `clean_tweet` column.

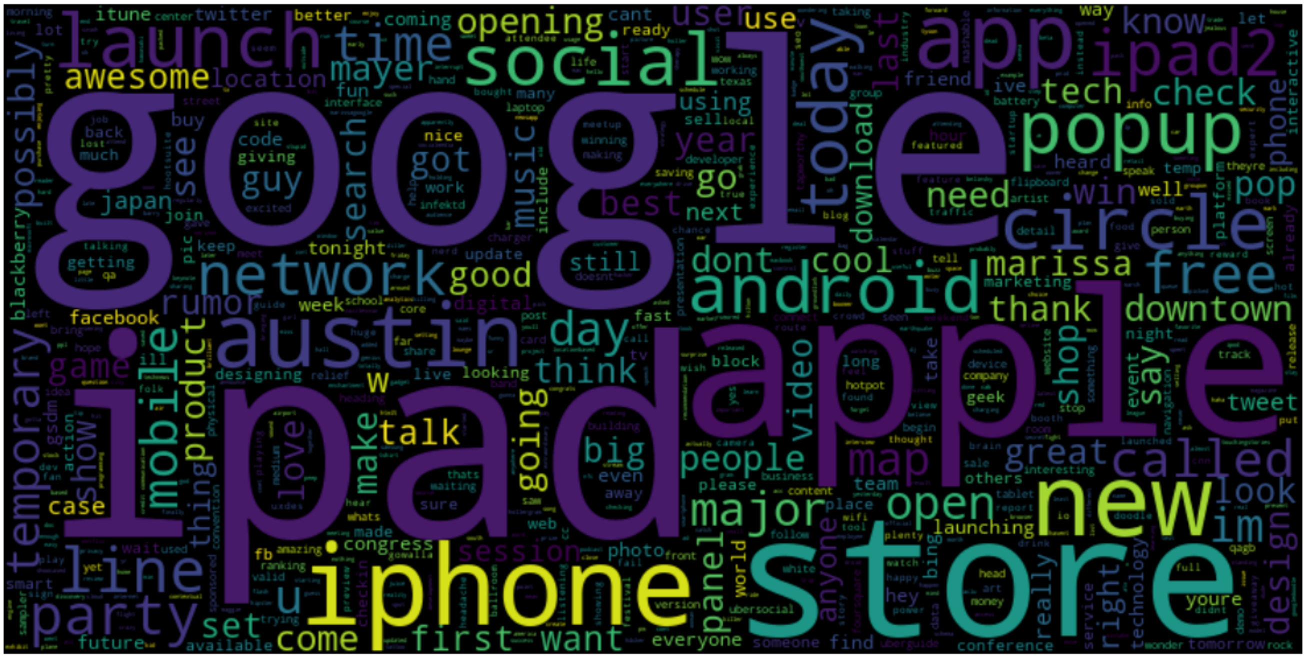
Visualizations

- Frequent terms in the lemmatized tweets were visualized using frequency distributions and bigrams, highlighting product-related terms such as "Google", "iPad", and "Apple.





- Wordcloud visualizations captured the overall trends and prominent words in the dataset



Modelling

These steps facilitate machine learning algorithms in processing the emotion variable, converting text into a numerical format for better analysis, ensuring the model is not biased towards the majority class, and providing clear metrics to evaluate performance on unseen data.

Preprocessing

Prepare data for modeling by:

- **Label Encoding**: Converted emotion labels into numerical values.
- **Vectorization**: Used **TF-IDF** and **CountVectorizer** to transform text data into numerical vectors.
- **SMOTE**: Applied **SMOTE** (Synthetic Minority Over-sampling Technique) to handle class imbalance.
- **Train-Test Split**: Split the dataset into training and testing sets for model evaluation.

Models

The machine learning algorithms used for this project are:

- **RandomForest**
- **Naive Bayes** (MultinomialNB)
- **Logistic Regression**
- **Decision Trees**

We will use the split data to predict which model achieves the highest accuracy and select the best deployment model.

Results

Random forest classifier

Count Vectorization Results

- **Best Random Forest Model (Count Vectorization)**: `RandomForestClassifier(n_estimators=200, random_state=42)`
- **Test Accuracy (Count Vectorization)**: 0.706
- **Test Recall (Count Vectorization)**: 0.705

TF-IDF Vectorization Results

- **Best Random Forest Model (TFIDF Vectorization)**: `RandomForestClassifier(random_state=42)`
- **Test Accuracy (TFIDF Vectorization)**: 0.837
- **Test Recall (TFIDF Vectorization)**: 0.836

- **Improvement in Performance**: With Count Vectorization, the model showed decent performance, but TF-IDF significantly boosted both accuracy and recall, reflecting better feature representation of the text.
- **Vectorization Impact**: TF-IDF's ability to down-weight common words while emphasizing rare but important terms helped the model achieve higher performance in both recall and accuracy.

Naive Bayes (MultinomialNB) Model

Count Vectorization Results

- **Best Naive Bayes Model (Count Vectorization)**: `MultinomialNB(alpha=0.01)`
- **Test Accuracy (Count Vectorization)**: 0.660
- **Test Recall (Count Vectorization)**: 0.659

TF-IDF Vectorization Results

- **Best Naive Bayes Model (TFIDF Vectorization)**: `MultinomialNB(alpha=0.01)`
- **Test Accuracy (TFIDF Vectorization)**: 0.795
- **Test Recall (TFIDF Vectorization)**: 0.795

- **Accuracy Improvement**: The accuracy increased substantially when using TF-IDF, showing that Naive Bayes benefits from a more refined text representation.
- **Impact of Smoothing**: With Count Vectorization, the model struggled to distinguish between sentiment classes, but TF-IDF's ability to capture important context led to better differentiation between the classes.

- **Recall Consistency**: Both Count and TF-IDF showed similar recall scores, however, the overall model's ability to identify positive or negative sentiments was stronger with TF-IDF, suggesting a better fit for the classification task.

Logistic Regression

Count Vectorization Results

- **Best Logistic Regression Model (Count Vectorization)**: `LogisticRegression(C=31.0)`
- **Test Accuracy (Count Vectorization)**: 0.707
- **Test Recall (Count Vectorization)**: 0.705

TF-IDF Vectorization Results

- **Best Logistic Regression Model (TFIDF Vectorization)**: `LogisticRegression(C=31.0, max_iter=150)`
- **Test Accuracy (TFIDF Vectorization)**: 0.831
- **Test Recall (TFIDF Vectorization)**: 0.830

- **Slight Improvement in Accuracy**: Count Vectorization gave a slight increase in accuracy, but TF-IDF significantly outperformed it, especially after hyperparameter tuning.
- **Recall Gain**: The TF-IDF did not only improve in accuracy but also led to a better recall, suggesting that Logistic Regression is more sensitive to the context provided by TF-IDF's word weighting scheme.
- **Model Sensitivity**: The results indicate that Logistic Regression benefits from the more nuanced features provided by TF-IDF, helping it better identify subtle sentiment changes in tweets.

Decision Tree

Count Vectorization Results

- **Best Decision Tree Model (Count Vectorization)**: `DecisionTreeClassifier(max_features=5, min_samples_split=5)`
- **Test Accuracy (Count Vectorization)**: 0.695
- **Test Recall (Count Vectorization)**: 0.693

TF-IDF Vectorization Results

- **Best Decision Tree Model (TFIDF Vectorization)**: `DecisionTreeClassifier(max_features=5, min_samples_split=4)`
- **Test Accuracy (TFIDF Vectorization)**: 0.758

- **Test Recall (TFIDF Vectorization)**: 0.757

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

- **Jupyter Notebook** 99.8%
- **Python** 0.2%

Suggested workflows

Based on your tech stack

SLSA Generic generator Configure

Generate SLSA3 provenance for your existing release workflows

Publish Python Package Configure

Publish a Python Package to PyPI on release.

Python package Configure

Create and test a Python package on multiple Python versions.

More workflows Dismiss suggestions