Detail project report of Sentiment analysis

1. Data Acquisition

Source:

The dataset was acquired from a Google Sheets document Click here.

Description:

After accessing the Google Sheets link and subsequently downloading the dataset, it was observed that the dataset comprises various columns. A detailed examination revealed that these columns primarily contain textual data entries alongside their associated classifications.

Each row in the dataset represents a unique text entry, and it's corresponding classification signifies the category or sentiment associated with that text. The primary goal of acquiring this data is to further process it, clean it, and eventually utilize it for modeling purposes.

2. Understanding Data through Visualization

Introduction:

Understanding the dataset is crucial in sentiment analysis tasks. Before delving deep into model training, it's pivotal to get acquainted with the unique characteristics of the data.

Here's a guided breakdown of the essential data understanding steps that I have performed :

1. Dataset Overview:

Begin by loading your dataset and inspecting the first few rows. This will provide an initial glimpse into the kind of data you're dealing with.

2. Dataset Dimension:

Identify the shape of the dataset, i.e., the number of rows and columns. This will help in understanding the volume of data and features present.

```
print(data.shape)
```

Output :

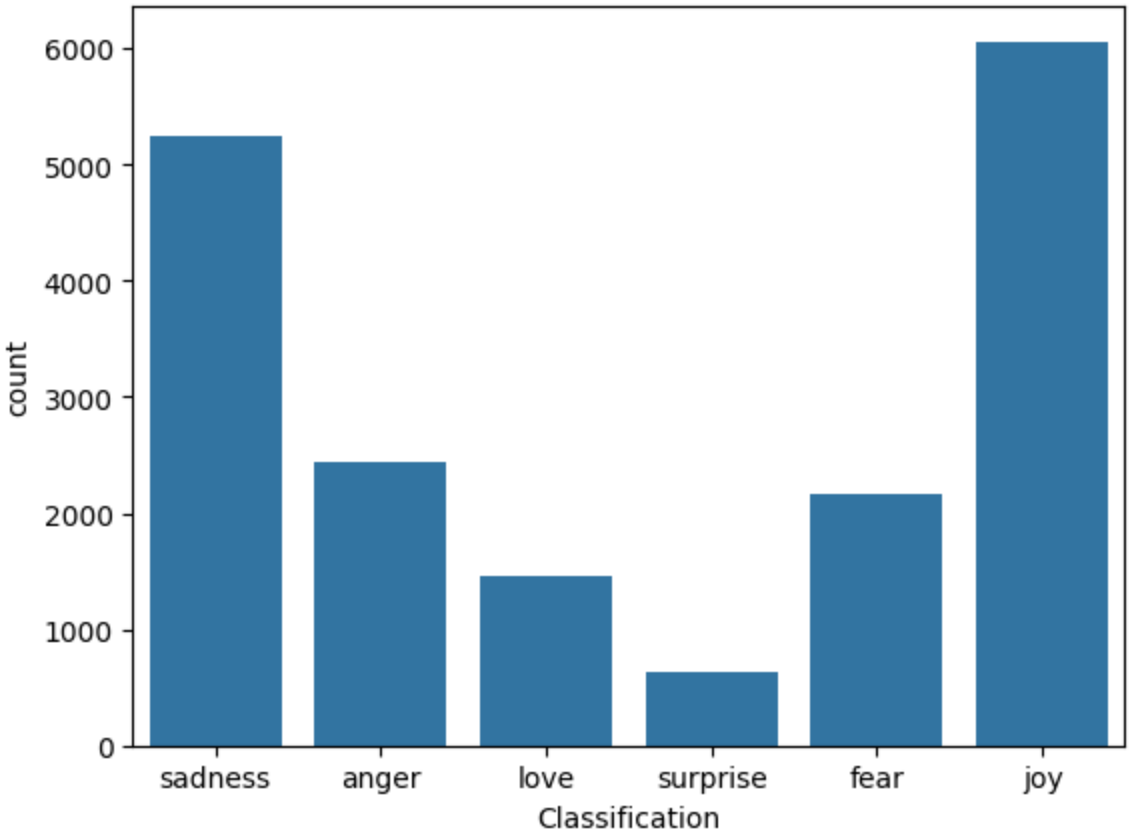
(18001, 2)

3. Missing Value Analysis:

Check for any missing values in the dataset. Missing data can impact the model's performance and hence needs addressing.

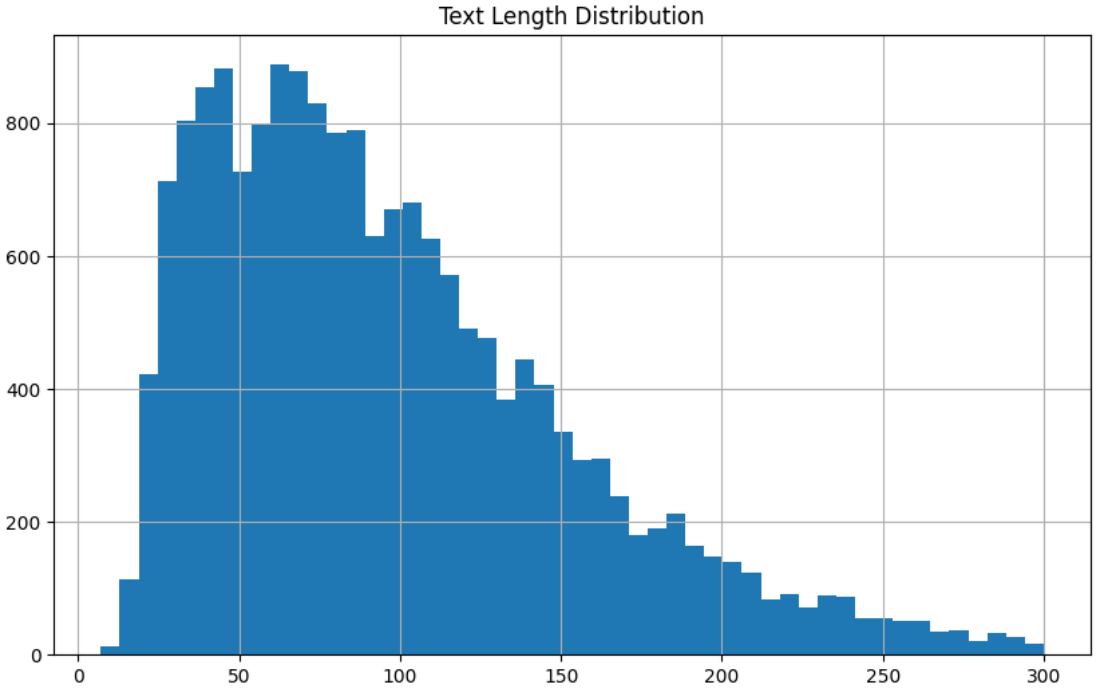
4. Class Distribution:

Investigate the distribution of different sentiment classes. This step will reveal if your dataset is balanced or if there's a skew towards specific classes.



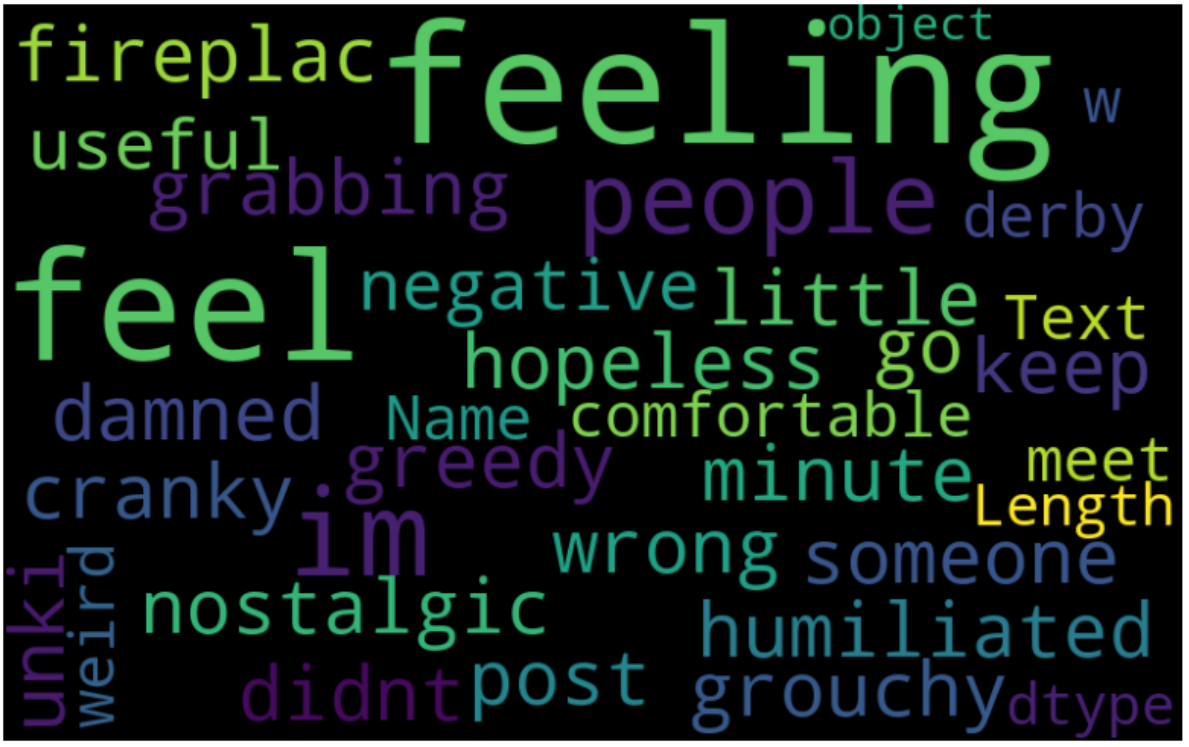
5. Text Length Analysis:

Determine the average length of text entries. Understanding text lengths can be useful in certain preprocessing steps like padding for deep learning models.



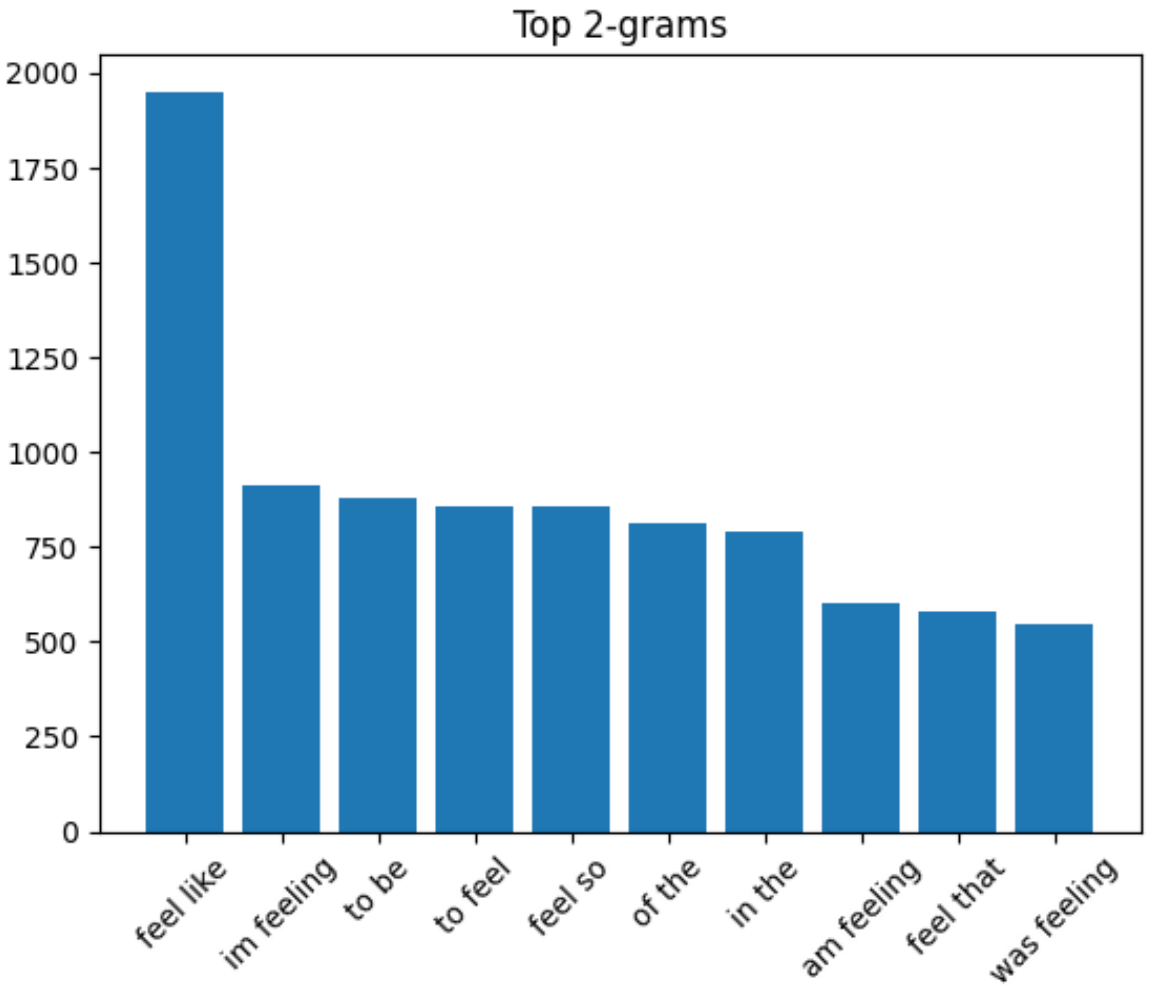
6. Common Words:

Identify the most frequently occurring words in your dataset. This can provide insights into the predominant themes or topics in the data.



7. N-gram Analysis:

N-grams represent continuous sequences of 'n' items from a given text. Analyzing N-grams, especially bigrams (2-grams) or trigrams (3-grams), can help in understanding common phrases or combinations of words that frequently appear together.


Top 2-grams
2000
1750
1500
1250
1000
750
500
250
0
feel like
im feeling
to be
to feel
feel so
of the
in the
am feeling
feel that
was feeling

3. Data Cleaning and Pre-processing

1. Duplicate Record Removal:

During the data preprocessing phase, it's essential to ensure that the data is clean, consistent, and void of any duplications that might skew the results of the model**.**

Process:

- The dataset was examined for records that had duplicated 'Text' entries.
- These duplicated entries were then removed from the main dataset to ensure the uniqueness of each text entry.

List of Removed Duplicate Records:

Text	Sentiment
i feel like some of you have pains and you cannot imagine becoming passionate about the group or the idea that is causing pain	love
i tend to stop breathing when i m feeling stressed	sadness
i loved the feeling i got during an amazing slalom run whether it was in training or in a race	surprise
i feel on the verge of tears from weariness i look at your sweet face and cant help but tenderly kiss your cheeks	love
i was intensely conscious of how much cash i had left in my gas and food envelope and i still have what i intended to save for next week which helps me not feel so stressed and scared	anger
i feel cared for and accepted	joy
i still feel completely accepted	Joy
im still not sure why reilly feels the need to be so weird	Fear
i shy away from songs that talk about how i feel toward god or that maybe even talk about my faithful response toward god	love

Note:

These records are a subset of the entire dataset. For a full list of duplicate records, please refer to the **"duplicates.csv"** file.

2. Profanity Check and Removal:

To maintain the quality and integrity of the dataset, it's important to ensure that the data does not contain any inappropriate or profane content.

Process:

- Each text entry in the dataset was scanned for profanities or inappropriate content using a predefined list of profane words/phrases.
- Records containing such content were then removed from the dataset.

List of Removed Records with Profanities:

Text	Classification
i am feeling horny so i ask her that lets go home	love
i called it god because i d seen god in a book and figured god was the right name for feeling so utterly affirmed and accepted without question	love
legs would feel shitty for a few miles but would come around like they always do	Sadness
i love the porn industry and i feel satisfied and fulfilled working in it i have to say that it doesn t really bring in the big bucks	joy
i feel that horrible helplessness to make things better for them and that feels like it will kill me inside	sad
im feeling every bit the spiteful vindictive bitch i can be at times	anger
i have been highly critical of dennis covingtons book in this article i must admit that he did say something that has merit in this discussion when he noted in his closing chapters this feeling after god is a dangerous business	anger

Note:

These records are a subset of the entire dataset. For a full list of duplicate records, please refer to the **" profane_rows.csv"** file.

3. Identification and Marking of Incoherent or Grammatically Incorrect Records:

Ensuring the textual data is coherent and grammatically correct is crucial, especially in tasks related to Natural Language Processing. Erroneous sentences can mislead models and degrade their performance.

Process:

Automated Grammar Check: The dataset was subjected to automated grammar check tools to identify sentences with glaring grammatical errors.

Sentence Completion Check: Text entries were analyzed to detect abrupt endings or incomplete sentences. The general approach was to identify sentences ending with conjunctions or sentences that do not seem to convey a full idea.

Semantic Coherence: While this is harder to automate, a sample of the dataset was manually checked to ensure that the texts make sense semantically. Entries that appeared to be random collections of words or didn't convey a clear thought were flagged.

Examples of Detected Issues:

Text	Classification	Has_Spelling_Errors	Is_Incomplete
i didnt feel humiliated	Sadness	True	True
im grabbing a minute to post i feel greedy wrong	anger	False	True
i am feeling grouchy	anger	false	true
i feel romantic too	love	false	True
i feel kinda appalled that she feels like she needs to explain in wide and lenghth her body measures etc pp	anger	True	True

Note:

These records are a subset of the entire dataset. For a full list of duplicate records, please refer to the **"data_with_errors.csv"** file.

4. Model Training and Evaluation

Task:

Train a model to classify the text.

Steps and Actions:

1. **Tokenization & Sequencing:**
   - Tokenized the text data.
   - Converted texts to sequences for model training.

2. **Data Split:**
   - Split the dataset into training and test sets to evaluate the model's performance.

3. **Model Architecture & Training:**
   - Defined a Convolutional Neural Network (CNN) model architecture.
   - Trained the model using the training set.
   - Evaluated the model's performance on the test set.

4. **Model Saving:**
   - Saved the trained model for deploying and future use.

5. **Tools/Techniques Used:** Python with TensorFlow and Keras libraries.

6. **Models Trained and Their Performance:**

   1. **CNN:**

      - **Precision:** 0.9303686974717207
      - **Recall**: 0.9301503094606542
      - **F1-Score:** 0.9300870665836237
      - **Accuracy:** 0.9301503094606542

   2. **Logistic Regression (LR):**

      - **Precision**: 0.8638974517335469
      - **Recall**: 0.8576480990274093

- **F1-Score**: 0.8518750401947963
- **Accuracy**: 0.8576480990274093

3. **Long Short-Term Memory (LSTM):**

   - **Precision:** 0.1209473327525707
   - **Recall**: 0.3477748305334512
   - **F1-Score:** 0.1794770610231675
   - **Accuracy**: 0.3477748305334512

4. **Multinomial Naive Bayes (MNB):**

   - **Precision**: 0.7726138007945849
   - **Recall**: 0.690834070144415
   - **F1-Score**: 0.6287310458091588
   - **Accuracy**: 0.690834070144415

5. **Random Forest (RF):**

   - **Precision**: 0.8792659883683545
   - **Recall**: 0.8779840848806366
   - **F1-Score**: 0.8767844109969946
   - **Accuracy**: 0.8779840848806366

5 Experiments & Tweaks

With the aim of achieving the best model accuracy, various experiments were conducted, primarily focusing on the CNN model.

Here's a highlight:

1. **CNN:** While the CNN performed the best among all models trained, an unidentified error prevented its direct use for API.

2. **Random Forests as an Alternative:** Due to the above-mentioned issue with CNN, the Random Forest (RF) model was chosen for deployment. Despite not being the highest performing model, its precision, recall, and F1-Score were deemed acceptable for this application.

6 API Development

Objective:

Develop a RESTful API to serve predictions from a trained Random Forest model.

Action:

1. Setting Up the Flask Application:

- Initiated a Flask application to facilitate the creation of web services.
- Structured the code to ensure modularization and ease of debugging.

2. Loading the Trained Model:

- Loaded the pre-trained Random Forest (RF) model. This was performed once during the startup of the API to ensure efficiency.
- Used the TensorFlow framework to manage the model operations, especially if the RF model was saved as a TensorFlow model.

3. Defining the API Endpoint:

- Created an endpoint named /classify that handles POST requests.
- This endpoint expects a JSON payload with a key 'text' containing the input text to be classified**.**

4. Input Validation:

- Before passing the input text to the model, performed validation checks:
- Checked if the text length is within the specified constraints (minimum 2 characters and maximum 150 characters).
- Returned appropriate error messages if the input doesn't meet the requirements.

5. Classification & Response Formation:

- If the input is valid, it's passed to the loaded RF model for classification.
- The model's prediction is then wrapped in a structured JSON response, indicating either the classification result or any error that might have occurred during the process.

6. Error Handling:

- **Implemented error handlers to catch potential issues, ensuring the API returns informative error messages rather than crashing.**

Tools/Techniques Used:

- **Flask:**

  Used as the primary web framework for creating the API. Flask is lightweight and suitable for creating simple to moderately complex web services.

- **TensorFlow:**

  - If the RF model was saved using TensorFlow, the TensorFlow library was used to load and make predictions using the model.
  - Offered utilities to handle model operations smoothly.

- **Random Forest (RF) Model:**

  The trained RF model, which is the backbone of this API, providing the classification capability.

- **Postman (for Testing):**

  Postman was an invaluable tool during the development phase, allowing for easy testing of the API endpoints, validation of responses, and debugging.