### Lab Manual: Sentiment Analysis on Public Feedback

**Lab 1: Exploring AI Tools for Government Use**

### 1. Objectives

* **Analyze Public Sentiment**: Utilize AI to process large-scale public feedback on government initiatives.
* **Understand Sentiment Analysis Models**: Explore the use of pre-trained AI models for sentiment classification.
* **Build Scalable Solutions**: Develop workflows that scale sentiment analysis tasks efficiently.
* **Visualize Results**: Generate and interpret graphical outputs to aid in data-driven decision-making.

**Learning Goals**:

1. Understand how sentiment analysis can be used to improve public services.
2. Learn how to apply pre-trained AI models like **BERT** to classify sentiment in text datasets.
3. Interpret sentiment analysis results to identify actionable insights.

**Estimated Time**:

* **Reading**: 5 minutes.
* **Execution**: 30–45 minutes.

### 2. Overview

#### **Sentiment Analysis in the Public Sector**

Government agencies often collect feedback through surveys, social media, and other platforms. AI-powered sentiment analysis enables scalable and accurate understanding of public opinion.

#### **Key Concepts**:

* **Sentiment Analysis**: The process of categorizing text data into positive, negative, or neutral sentiments.
* **Pre-trained Models**: AI models (e.g., **BERT**, **DistilBERT**) trained on large datasets to recognize sentiment patterns.
* **Real-world Relevance**: Use cases include public sentiment tracking for urban planning, healthcare, and policy announcements.

#### **Benefits of AI for Sentiment Analysis**:

1. Reduces manual effort in analyzing extensive feedback datasets.
2. Enhances decision-making through data-driven insights.
3. Identifies trends to gauge the impact of government policies.

**Example Applications**:

* Analyzing responses to new transportation initiatives.
* Monitoring satisfaction levels in healthcare programs.
* Identifying dissatisfaction trends in public service delivery.

**Reading Time**: 5–10 minutes.

### 3. Environment Setup

1. **Requirements**:
   * Access to **Google Colab**: A free, cloud-based platform for Python programming.
   * Pre-trained models like **BERT** or **DistilBERT** via Hugging Face.
   * Dataset: Public feedback data downloaded using the following code snippet:
   * import requests  
       
     url = "https://data.cityofnewyork.us/api/views/erm2-nwe9/rows.csv?accessType=DOWNLOAD"  
     response = requests.get(url)  
       
     with open("nyc\_311\_comments.csv", "wb") as file:  
      file.write(response.content)
2. **Prerequisites**:
   * Familiarity with Python programming.
   * Understanding of basic libraries like Pandas and Matplotlib.
3. **Installation Steps**:  
   a. Open [Google Colab](https://colab.research.google.com).  
   b. Install necessary libraries:

* !pip install transformers pandas matplotlib wordcloud
* c. Upload the dataset to Colab by navigating to the **Files** section and using the **Upload** button.

1. **Verification**: Ensure all installations are successful:

* import transformers  
  import pandas as pd  
  import matplotlib.pyplot as plt  
  print("Environment set up successfully!")

**Setup Time**: 10–15 minutes.

### 4. Step-by-Step Instructions

#### **Part 1: Load and Explore the Dataset**

1. **Load Dataset**:
   * Import the dataset into a Pandas DataFrame:
   * data = pd.read\_csv('nyc\_311\_comments.csv')  
     print(data.head())
2. **Check Data Structure**: Ensure the dataset includes a column with text feedback data.
3. **Handle Missing Data**: Remove incomplete entries:

* data.dropna(subset=['Complaint Type'], inplace=True)

1. **Preview Dataset**:
   * Display a summary of the dataset:
   * print(data.info())
2. **Enhance Dataset Understanding**: Analyze the frequency of complaint types:

* print(data['Complaint Type'].value\_counts())

1. **Advanced Data Exploration**: Check distribution of complaints across boroughs:

* print(data['Borough'].value\_counts())

#### **Part 2: Perform Sentiment Analysis**

1. **Load Pre-trained Sentiment Model**:
   * Use Hugging Face Transformers to initialize the sentiment analysis pipeline:
   * from transformers import pipeline  
     sentiment\_pipeline = pipeline("sentiment-analysis")
2. **Test the Model**: Input a single text example:

* result = sentiment\_pipeline("I am thrilled with the improvements in public transport!")  
  print(result)

1. **Apply Model to Entire Dataset**:
   * Add a new column for sentiment classification:
   * data['sentiment'] = data['Complaint Type'].apply(lambda x: sentiment\_pipeline(x)[0]['label'])  
     print(data.head())
2. **Inspect Results**: Analyze the sentiment labels for accuracy.
3. **Explore Sentiment Distribution**: Count each sentiment category:

* print(data['sentiment'].value\_counts())

1. **Filter Specific Sentiments**: Extract and analyze positive feedback:

* positive\_feedback = data[data['sentiment'] == 'POSITIVE']  
  print(positive\_feedback.head())

#### **Part 3: Visualize Results**

1. **Create a Bar Chart**:
   * Visualize sentiment distribution:
   * data['sentiment'].value\_counts().plot(kind='bar', color=['green', 'blue', 'red'])  
     plt.title('Sentiment Analysis Distribution')  
     plt.xlabel('Sentiment')  
     plt.ylabel('Frequency')  
     plt.show()
2. **Generate a Word Cloud**:
   * Extract frequent words from positive sentiments:
   * from wordcloud import WordCloud  
     positive\_text = ' '.join(data[data['sentiment'] == 'POSITIVE']['Complaint Type'])  
     wordcloud = WordCloud(width=800, height=400, background\_color='white').generate(positive\_text)  
     plt.imshow(wordcloud, interpolation='bilinear')  
     plt.axis('off')  
     plt.show()
3. **Advanced Visualization**: Create a pie chart to show sentiment proportions:

* data['sentiment'].value\_counts().plot(kind='pie', autopct='%1.1f%%', colors=['green', 'blue', 'red'])  
  plt.title('Sentiment Proportions')  
  plt.ylabel('')  
  plt.show()

1. **Time-Based Analysis**: Plot sentiment trends over time (if Created Date is available):

* data['Created Date'] = pd.to\_datetime(data['Created Date'])  
  sentiment\_over\_time = data.groupby(data['Created Date'].dt.to\_period('M'))['sentiment'].value\_counts().unstack()  
  sentiment\_over\_time.plot(kind='line', figsize=(12, 6))  
  plt.title('Sentiment Trends Over Time')  
  plt.xlabel('Month')  
  plt.ylabel('Frequency')  
  plt.show()

#### **Part 4: Save and Share**

1. **Export Processed Data**: Save the DataFrame with sentiment labels:

* data.to\_csv('processed\_feedback.csv', index=False)

1. **Share Insights**: Present your findings through visualizations and summaries.

**Execution Time**: 30–45 minutes.

### 5. Key Questions or Observations

1. **Sentiment Categorization**:
   * What proportion of the feedback is positive? Negative? Neutral?
   * Are there any outliers or surprising results?
2. **Model Performance**:
   * How well does the model handle ambiguous feedback?
   * Were there examples where the sentiment was misclassified?
3. **Visualization Analysis**:
   * What do the bar chart and word cloud reveal about public opinion?
   * How might these insights influence policy changes?
4. **Advanced Reflection**:
   * Could multilingual text processing improve sentiment accuracy?
   * What limitations exist in pre-trained models for government-specific tasks?
5. **Ethics in Sentiment Analysis**:
   * How can biases in AI models affect sentiment analysis outcomes?
6. **Temporal Insights**:
   * What trends were observed in sentiment over time?
   * How might seasonal changes or external events influence these trends?

**Reflection Time**: 15–20 minutes.

### 6. Exploration and Experimentation

1. **Test Additional Models**: Experiment with **RoBERTa** or fine-tune **BERT** for improved accuracy.
2. **Time-Based Sentiment Analysis**: Analyze trends over time to detect shifts in public opinion.
3. **Challenge Task**: Implement a sentiment scoring system (e.g., assigning values to positive, neutral, and negative sentiments) to quantify overall feedback.
4. **Cross-Analysis**: Compare sentiment trends across different complaint types or boroughs (if available in the dataset):

* borough\_sentiments = data.groupby('Borough')['sentiment'].value\_counts()  
  print(borough\_sentiments)

1. **Real-Time Analysis**: Integrate live feedback data from APIs (e.g., NYC Open Data API) to dynamically update sentiment results.

### 7. Results and Evaluation

#### **Expected Outcomes**:

* **Bar Chart**: A visual breakdown of sentiment categories.
* **Word Cloud**: Frequent words from positive and negative feedback.
* **Time-Series Graph**: Trends in sentiment over time.
* **Summary**: Key insights derived from the data analysis.

#### **Evaluation Questions**:

1. Were the results consistent with your expectations?
2. What actionable insights can be drawn from the sentiment distribution?
3. How could the analysis be expanded for larger datasets?
4. How might sentiment trends over time inform government decisions?

### 8. Additional Learning (Optional)

1. **Learn More**:
   * Explore fine-tuning techniques for custom sentiment labels.
   * Investigate applications of sentiment analysis in other domains (e.g., customer feedback).
2. **Advanced Challenge**:
   * Create a dashboard that updates sentiment analysis results in real-time.

**Time for Exploration**: 30–60 minutes.

### 9. Completion Confirmation

Congratulations on completing the Sentiment Analysis Lab! Share your results with stakeholders to discuss potential improvements in government initiatives.

### Embedded Questions to Test Knowledge

1. **Model Understanding**:
   * What is the difference between BERT and DistilBERT in sentiment analysis?
2. **Data Preprocessing**:
   * Why is it important to handle missing data before applying a model?
3. **Visualization**:
   * How does a word cloud enhance the interpretation of positive feedback?
4. **Insights**:
   * What policy changes could be suggested based on a majority of negative feedback?
5. **Experimentation**:
   * How could real-time sentiment analysis benefit urban planning projects?
6. **Temporal Insights**:
   * What patterns in sentiment over time were most surprising, and why?