NLP Exploratory Data Analysis (EDA)

Rujul Dwivedi

February 12, 2025

1 Introduction

Exploratory Data Analysis (EDA) in Natural Language Processing (NLP) is crucial for understanding text data before building models. It helps in:

- Identifying dominant words and phrases.
- Understanding word relationships and frequencies.
- Detecting data imbalances and distribution anomalies.
- Feature engineering for model training.

2 Word Clouds and Frequency Analysis

A word cloud is a graphical representation where word size corresponds to frequency. It helps in identifying dominant themes in text data.

2.1 Mathematical Representation

Given a set of words $W = \{w_1, w_2, ..., w_n\}$ with frequencies $f(w_i)$, the probability of a word appearing is:

$$P(w_i) = \frac{f(w_i)}{\sum_j f(w_j)}$$

where $\sum_j f(w_j)$ is the total word count.

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

text_data = "Natural language processing (NLP) is a fascinating
    field."
wordcloud = WordCloud(width=800, height=400, background_color='
    white').generate(text_data)

plt.figure(figsize=(10,5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

3 Tokenization: Breaking Text into Words

Tokenization is the process of splitting text into smaller units called tokens.

```
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize

sentence = "Tokenization is an important step in NLP."
tokens = word_tokenize(sentence)
print(tokens)
```

4 Named Entity Recognition (NER)

Named Entity Recognition (NER) identifies proper names, locations, and organizations within text.

```
import spacy
nlp = spacy.load("en_core_web_sm")

doc = nlp("Google was founded in Mountain View, California.")
for ent in doc.ents:
    print(ent.text, ent.label_)
```

5 Handling Data Imbalance

Class imbalance occurs when certain categories appear more frequently than others. Solutions include:

- **Undersampling**: Removing excess samples from dominant classes.
- **Oversampling**: Duplicating minority class samples.
- **SMOTE (Synthetic Minority Over-sampling Technique)**: Generating synthetic examples.

```
from imblearn.over_sampling import SMOTE
from collections import Counter
from sklearn.model_selection import train_test_split

X, y = ["data1", "data2", "data3"], [0, 1, 1]
X_train, X_test, y_train, y_test = train_test_split(X, y,
   test_size=0.3)

smote = SMOTE()
X_resampled, y_resampled = smote.fit_resample([[i] for i in
   X_train], y_train)
print(Counter(y_resampled))
```