# Technical Documentation: NewsBot Intelligence System

## 1. Project Overview

The **NewsBot Intelligence System** is a comprehensive Natural Language Processing (NLP) pipeline designed to analyze and classify news articles. The system performs text preprocessing, feature extraction, sentiment analysis, named entity recognition (NER), and classification to produce structured insights from unstructured news content.

This documentation explains the system's components, the logic behind each stage, the libraries used, and how to execute the notebook.

---

## 2. Objectives

- **Preprocess raw news articles** to clean and normalize text.
- **Extract features** such as TF-IDF vectors, POS tags, and sentiment scores.
- **Classify articles** into predefined categories using machine learning models.
- **Identify named entities** and analyze their patterns across categories.
- **Integrate all components** into a single, end-to-end analysis pipeline.

---

## 3. System Architecture

The NewsBot system consists of the following modules:

1. **Data Loading & Exploration**
2. **Text Preprocessing**
3. **Feature Extraction (TF-IDF)**
4. **POS & Syntax Analysis**
5. **Sentiment Analysis**
6. **Text Classification**
7. **Named Entity Recognition (NER)**
8. **Final System Integration & Reporting**

---

## 4. Dependencies

The project was implemented in **Python 3.11** and requires the following libraries:

- `pandas`, `numpy` – Data manipulation
- `scikit-learn` – TF-IDF vectorization, model training, evaluation
- `nltk` – Tokenization, stopwords, lemmatization, VADER sentiment
- `spacy` – Tokenization, POS tagging, dependency parsing, NER
- `matplotlib`, `seaborn` – Data visualization
- `wordcloud` – Word cloud generation
- `networkx` – Entity co-occurrence network

Setup:

```bash
pip install pandas numpy scikit-learn nltk spacy matplotlib seaborn wordcloud networkx
python -m spacy download en_core_web_sm
```

NLTK resources:

```python
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('vader_lexicon')
nltk.download('averaged_perceptron_tagger')
```

---

## 5. Detailed Component Walkthrough

### *5.1 Data Loading & Exploration*

- Loaded the **20 Newsgroups dataset** from `sklearn.datasets`.
- Merged train/test splits into a single DataFrame.
- Mapped category indices to category names.
- Visualized category distribution.

### *5.2 Text Preprocessing*

- **Lowercasing**
- **Removing HTML tags, URLs, emails, non-alphabetic characters**
- **Tokenization** with spaCy
- **Stopword removal** with NLTK
- **Lemmatization** with WordNetLemmatizer

### *5.3 Feature Extraction (TF-IDF)*

- Vectorized processed text using `TfidfVectorizer`.
- Parameters: `max_features=5000`, `ngram_range=(1,2)`.
- Extracted top terms per category and created word clouds.

### *5.4 POS & Syntax Analysis*

- Extracted POS tag frequencies per category.
- Used spaCy dependency parser to identify noun phrases, subjects, and objects.

### *5.5 Sentiment Analysis*

- Applied **NLTK's VADER** sentiment analyzer.
- Calculated compound sentiment scores and assigned polarity labels.
- Aggregated sentiment scores by category.

### *5.6 Text Classification*

- Models trained: **Multinomial Naive Bayes**, **Logistic Regression**, **SVM**.
- Features combined: TF-IDF, sentiment scores, length features.
- Best model: **Naive Bayes** (~57% accuracy).

### *5.7 Named Entity Recognition (NER)*

- Used spaCy NER to extract entities of types PERSON, ORG, GPE, DATE, etc.
- Counted entity occurrences per category.
- Built entity co-occurrence graphs with NetworkX.

### *5.8 Final Integration*

- Created a pipeline to process, classify, and analyze new articles.
- Generated a comprehensive report summarizing classification, sentiment, and entity findings.

---

## 6. Execution Instructions

### *Local Jupyter Notebook*

```bash
jupyter notebook Final_NewsBot.ipynb
```

### *Google Colab*

1. Upload the notebook to Colab.
2. Install dependencies as shown in section 4.
3. Run all cells sequentially.

---

## 7. Example Usage

```python
new_title = "Mars Rover Discovers New Rock Formation"
new_content = "NASA announced that the Curiosity rover found a surprising new rock formation on Mars..."
full_text = f"{new_title} {new_content}"

processed = preprocess_text(full_text)
predicted_category = best_model.predict(tfidf_vectorizer.transform([processed]))
entities = extract_entities(full_text)
sentiment = analyze_sentiment(full_text)

print("Predicted category:", predicted_category[0])
print("Entities:", [(e['text'], e['label']) for e in entities])
print("Sentiment:", sentiment['sentiment_label'], sentiment['compound'])
```

---

## 8. Results Summary

- **Best Classifier:** Naive Bayes (Accuracy ~57%)
- **Most Positive Category:** comp.graphics
- **Most Negative Category:** talk.politics.guns
- **Entities Extracted:** 310,810 total, 102,816 unique

---

## 9. Future Improvements

- Integrate a transformer model like **BERT** for classification.
- Use real-world news datasets for broader applicability.
- Deploy as a web application using **Flask** or **Streamlit**.

---

## 10. References


- scikit-learn Documentation – https://scikit-learn.org/
- spaCy Documentation – https://spacy.io/
- NLTK VADER – https://github.com/cjhutto/vaderSentiment
- 20 Newsgroups Dataset – http://qwone.com/~jason/20Newsgroups/