Part B: News Article Classification

Introduction

In the modern digital era, news content is produced and consumed at an unprecedented scale across websites, apps, and social media platforms. As the volume of news articles grows, organizing and managing this information becomes increasingly challenging. Manual classification of articles into categories like **sports, politics, and technology** is not scalable or efficient.

This project addresses that challenge by applying **NLP** and **machine learning** techniques to automatically classify news articles based on their content. Automation improves content delivery, personalization, and searchability.

We use a labeled dataset of news articles to train classification models that predict the correct category. The process includes text preprocessing, feature extraction, model development, and performance evaluation.

Objective

1. To build a classification model that categorizes news articles into **sports, politics, and technology**.
2. To clean and preprocess raw text for better feature representation.
3. To apply methods like **TF-IDF**, **Word2Vec**, and **GloVe** for numerical transformation.
4. To develop classification models such as **Logistic Regression**, **Naive Bayes**, and **SVM**.
5. To evaluate model performance using metrics like **accuracy** and **F1-score**.
6. To present insights using **EDA and visualizations** in the final report.

Task 1: Data Collection and Preprocessing

The dataset contained labeled news articles from categories like politics, sports, and technology. We cleaned the text by removing punctuation, special characters, and stopwords. Then we applied tokenization and lemmatization to standardize the content.

Insights:
- Preprocessing ensures better separation between topic-specific words.
- Lemmatization improves semantic clarity.

What we learned:
- Proper preprocessing is the foundation for accurate classification models.

Task 2: Feature Extraction

TF-IDF vectorizer was used to convert the cleaned text into numerical format suitable for training. We performed Exploratory Data Analysis (EDA) to identify category distribution and top keywords.

Insights:
- TF-IDF gave a strong signal for frequent and unique words per category.
- EDA confirmed class balance and feature distribution.

What we learned:
- Feature extraction transforms qualitative text into quantitative format for ML.

Task 3: Model Development and Training

We built classification models using Logistic Regression, Naive Bayes, and SVM. Models were trained on the TF-IDF feature matrix, and cross-validation ensured stable performance.

Insights:
- SVM gave the highest classification accuracy.
- Naive Bayes performed well for speed and interpretability.

What we learned:
- Model selection should be based on both accuracy and interpretability.

Task 4: Model Evaluation

Models were evaluated using precision, recall, F1-score, and confusion matrices. We compared model results and analyzed misclassifications.

Insights:
- F1-score helped evaluate imbalanced categories.
- Confusion matrix showed clear strengths and weaknesses.

What we learned:
- Evaluating on multiple metrics helps us choose the most reliable model.

We Learned in This Project

- **Handling Multi-Class Classification**
    - While IMDb was binary, this task taught us how to work with **multi-class** problems (e.g., sports, politics, technology) and the challenges they bring.
- **TF-IDF for Topic Separation**
    - Learned how term importance differs between categories, and how vectorizers help in separating article types based on frequent keywords.
- **Cross-Validation and Hyperparameter Tuning**
    - Practiced splitting data effectively to ensure the model generalizes well across all categories.
- **Building Scalable Models**
    - This task emphasized how NLP can be applied in real-world scenarios, like classifying thousands of news articles automatically.

Pros:

1. **Multi-class Learning Experience**
    - Unlike binary sentiment analysis, this project introduced multi-class classification (e.g., sports, politics, tech), helping us understand how to handle multiple categories effectively.
2. **Domain-Specific Language Clarity**
    - News articles often use topic-specific vocabulary. This helped the model distinguish between classes using TF-IDF and word embeddings.
3. **Real-World Applicability**
    - Automating article classification is a valuable use case for content recommendation systems and news aggregation platforms.
4. **Scalable Pipeline**
    - The preprocessing and model-building steps can easily scale to thousands of articles and new categories with minimal rework.

Cons:

1. **Vocabulary Overlap Between Categories**
    - Some topics, such as politics and technology, may share similar words (e.g., "policy," "regulation"), leading to misclassifications.
2. **Data Imbalance Issues**
    - Some categories may be underrepresented, which can bias the model's predictions toward more dominant classes.
3. **Loss of Context in Bag-of-Words/TF-IDF**
    - These models ignore word order and context, making it difficult to capture complex article meanings or sarcasm.
4. **Interpretability Challenges**
    - In multi-class settings, it becomes harder to interpret *why* a misclassification happened without deeper semantic analysis or visualization tools.