CLOUD COUNSELAGE
IT & MANAGEMENT CONSULTING SERVICES

Project Report

Artificial Intelligence – Assignment

Submitted to: Cloud Counselage IT & Management Consulting Services

Submitted by: Mayuri Khatpe

Introduction

The Artificial Intelligence Assignment focuses on building an end-to-end solution for an intelligent document classifier. The objective is to classify a set of documents into predefined categories using Natural Language Processing (NLP) and deep learning techniques. This report highlights the execution, methodology, and findings from implementing two models: Logistic Regression and LSTM.

This assignment evaluates how well insights can be derived from data, implemented through code, and communicated effectively.

Project Objectives

You are tasked with building an end-to-end solution for an intelligent document classifier. The goal is to classify a set of documents into predefined categories using Natural Language Processing (NLP) and deep learning techniques.

1. Preprocessing Steps

The dataset used for this task consisted of 3000 text documents categorized into five classes: Scientific, Legal, E-commerce, and others. The following preprocessing steps were applied:

Step 1.1: Data Loading

- The dataset contained three columns: `ID`, `Text` (document content), and `Category` (labels).
- No missing values were present in the dataset, and data types were consistent.
- The dataset was split into 80% training and 20% testing data.
  - Training data shape: (2400, 1685)
  - Testing data shape: (600, 1685)

Step 1.2: Text Cleaning

- **Tokenization**: Text was split into individual words.
- **Stop-word Removal**: Common words (e.g., "the," "is") were removed using NLTK's English stop-word list.
- **Lemmatization**: Words were reduced to their base forms using the WordNet Lemmatizer.
- **Vectorization**: Text data was transformed into numerical format using **TF-IDF Vectorizer**.
  - Maximum features were limited to 5000 to reduce dimensionality.

2. Architecture and Methodology

Two models were implemented and evaluated for this classification task:

A. Logistic Regression (Traditional Machine Learning)

- **Model**: A linear classifier optimized using L2 regularization.
- **Implementation**:
  - Features derived from the TF-IDF representation of text were fed into the Logistic Regression model.
  - The model was trained for 100 iterations.
- **Hyperparameter Optimization**:
  - Grid Search was used to tune `C` (regularization strength) and `solver` (optimization algorithm).
  - Best hyperparameters: `C = 100`, `solver = 'liblinear'`.
  - **Best F1-Score**: 0.9983.

B. LSTM (Deep Learning)

- **Architecture**:
  - **Embedding Layer**: Mapped input words into a dense 128-dimensional vector space.
  - **LSTM Layer**: Captured sequential dependencies with 128 units and a 20% dropout rate.
  - **Dense Layer**: A softmax layer for multiclass classification.
- **Implementation**:
  - Features from the TF-IDF vectorizer were reshaped to fit the LSTM's input format.
  - The model was trained for 5 epochs with a batch size of 32.
  - Optimizer: Adam; Loss: Sparse Categorical Crossentropy.

3. Evaluation Results and Comparison

Both models were evaluated using Accuracy, Precision, Recall, and F1-Score. The results are summarized below:

Metric	Logistic Regression	LSTM
Accuracy	0.995	0.218
Precision	1.00	0.047
Recall	0.995	0.218
F1-Score	1.00	0.078

Observations:

- Logistic Regression significantly outperformed the LSTM model across all metrics, achieving near-perfect classification results.
- The LSTM model struggled, indicating a mismatch between the input representation (TF-IDF) and the sequential nature of LSTM.

4. Optimization

Step 4.1: Optimize Logistic Regression (Grid Search)

Grid Search was used to systematically evaluate combinations of hyperparameters to identify the best-performing configuration for Logistic Regression.

- **Hyperparameters Tuned**:
    1. `C`: Regularization strength. Values tested: [0.01, 0.1, 1, 10, 100].
    2. `solver`: Optimization algorithms. Values tested: `['liblinear', 'lbfgs']`.
- **Results**:
    - **Best Hyperparameters**: `C = 100`, `solver = 'liblinear'`.
    - **Best F1-Score**: **0.9983**.

The optimized model was used for predictions, further solidifying the performance of Logistic Regression.

5. Challenges Faced and Solutions

Challenge 1: Poor Performance of LSTM

- **Issue**: The LSTM model failed to capture meaningful patterns, resulting in low metrics.
- **Solution**: Future iterations could replace TF-IDF features with pre-trained embeddings (e.g., Word2Vec, GloVe, or BERT).

Challenge 2: High Dimensionality of TF-IDF

- **Issue**: The large feature space increased computational overhead.
- **Solution**: Limited the number of features to the top 5000 terms.

Challenge 3: Hyperparameter Tuning for Logistic Regression

- **Issue**: Identifying optimal hyperparameters for maximum performance.
- **Solution**: Used Grid Search with 3-fold cross-validation to systematically evaluate parameter combinations.

6. Conclusion

- The **Logistic Regression model** achieved outstanding performance, with an F1-Score of **1.00**, demonstrating its suitability for this classification task.
- The **LSTM model** underperformed, highlighting the importance of appropriate input representations for deep learning models.
- Future work should focus on incorporating embeddings or transformer-based architectures to enhance the deep learning model's performance.