INTELLIGENT DOCUMENT CLASSIFIER

BY : SANYA UPPAL

JANUARY 12, 2025
CLOUD COUNSELAGE
Cloud Counselage Pvt.Ltd.

Model 1: Traditional Model

Model Used: Logistic Regression Model

1. Importing Libraries

The code begins by importing various libraries for data processing, visualization, natural language processing (NLP), and machine learning:

- **Pandas** for data manipulation.
- **Seaborn** and **Matplotlib** for creating visualizations.
- **Scikit-learn** for model building, encoding, and evaluation.
- **NLTK** for text processing tasks like tokenization, stopwords removal, and lemmatization.
- **Gradio** for building a user interface to interact with the machine learning model.

2. Data Exploration

- The dataset is loaded using pandas.read_csv(). The data is explored with methods like df.info(), df.isnull(), and df.shape to get an overview of the columns, missing values, and dataset dimensions.
- The categorical column 'Category' is analyzed to find unique categories, counts, and percentage distribution.

3. Data Visualization

Visualizations are created for better understanding:

- **Bar Plot** shows the count of each category.
- **Pie Chart** displays the proportion of each category.
- **Text Length Distribution** shows how text length varies across documents.
- **Word Cloud** visualizes the most frequent words in the text column.
- **Category-wise Average Text Length** and **Frequent Words by Category** highlight variations in text data across categories.

4. Data Preprocessing

- **Encoding**: The 'Category' column is encoded using both **Label Encoding** and **One-Hot Encoding** to make it suitable for machine learning algorithms.
- **Text Preprocessing** includes:
    - **Tokenization** to break text into words.
    - **Stopwords Removal** to filter out common words like "the" and "and".
    - **Lemmatization** to reduce words to their base form (e.g., "running" becomes "run").
    - **TF-IDF Vectorization** to convert the text data into numerical features.
    - **Latent Semantic Analysis (LSA)**, using **Truncated SVD**, to reduce dimensionality and uncover hidden structures.

5. Model Building and Evaluation

- The data is split into training and testing sets using **train_test_split**.

- **Logistic Regression** is chosen as the model to classify documents. The model's performance is evaluated using accuracy and classification metrics such as precision, recall, and F1-score.
- **Grid Search** is employed to tune hyperparameters for optimal performance.

6. Model Deployment (Gradio UI)

- **Gradio** is used to create a user interface for real-time interaction with the model. Users can input text and get predictions based on the trained model.

MODEL 2: DEEP LEARNING

MODEL USED: LSTM

Word2Vec Model

The second model involves:

- **Word2Vec**: Converts words into vector embeddings based on their context, using either **CBOW** or **Skip-gram** architectures.
- **Saving the Model**: After training, the model is saved for later use.
- **Embeddings Generation**: Converts words into vector representations.
- **Data Splitting and Scaling**: Data is split into training and testing sets, with scaling applied to ensure uniformity across features.

LSTM Model Architecture

- **Model Architecture**: The architecture includes a **Dense Layer** with 512 units, followed by **LeakyReLU** activation. Regularization techniques like **L2 regularization** are applied to avoid overfitting.
- **LSTM Layer**: Long Short-Term Memory (LSTM) networks are often used in NLP tasks, though not fully described here.

PERFORMANCE OF BOTH MODELS:

Logistic Regression Model:

- **Accuracy**: 99%
- **Precision**:
    - Blog: 0.98
    - E-commerce: 1.00
    - Legal: 1.00
    - News: 1.00
    - Scientific: 1.00
- **Recall**:
    - Blog: 1.00
    - E-commerce: 1.00
    - Legal: 1.00

    - News: 0.98
    - Scientific: 1.00
- **F1-Score**:
    - Blog: 0.99
    - E-commerce: 1.00
    - Legal: 1.00
    - News: 0.99
    - Scientific: 1.00
- **Overall Accuracy**: 99%
- **Macro Average**:
    - Precision: 1.00
    - Recall: 1.00
    - F1-Score: 1.00
- **Weighted Average**:
    - Precision: 1.00
    - Recall: 0.99
    - F1-Score: 1.00

LSTM Model:

- **Test Loss**: 1.072
- **Test Accuracy**: 50.8%

Logistic Regression model performs significantly better, with high accuracy, precision, recall, and F1-score across all categories.

The **LSTM model**, on the other hand, has a relatively low test accuracy (50.8%). I reject LSTM or any deep learning model because data doesn't have a sequence or temporal nature (i.e., the two features are independent or do not represent time-dependent relationships), using LSTM is unnecessary and over-complicated.

With a shape of (3000, 2), you have only two features. LSTM requires more data (both in terms of number of features and samples) to capture meaningful patterns from sequential data. In cases where data is not sequential, simpler models like **logistic regression**, **decision trees**, **SVMs**, or **fully connected feedforward neural networks (Dense layers)** would be more suitable.

Conclusion:

For the project "Intelligent Document Classifier," and considering the dataset provided, I have decided to use a traditional model, specifically **Logistic Regression**, due to its evaluation result of 99%, which significantly outperforms the **Deep Learning technique (LSTM)** in this case.