Submitted by: Pratyush Kumar Singh
LinkedIn, Email, GitHub

Document Classifier

Artificial Intelligence – Assignment

The implemented project involves a detailed pipeline for text classification, incorporating preprocessing, feature extraction, and model training using both traditional machine learning and deep learning approaches. The workflow begins with data exploration, where the dataset is loaded from a CSV file containing three columns: `ID`, `Text`, and `Category`. The dataset comprises 3,000 entries distributed across five categories: Scientific, Legal, E-commerce, News, and Blog. A bar plot visualizes the category distribution, indicating that the dataset is relatively balanced. Furthermore, the dataset is clean, with no missing values in any column, and basic statistics regarding the length of the text in terms of characters and words are computed. This exploratory analysis reveals that the average text length is approximately 205.39 characters or 30.56 words, with a minimum of 57 characters and a maximum of 414 characters.

The next phase of the project focuses on data cleaning, where the raw text is preprocessed to ensure uniformity and remove unnecessary noise. The text is converted to lowercase, and special characters, punctuation, and numbers are stripped. Additionally, extra whitespaces are removed to standardize the format of the text. This cleaning process is essential for preparing the text for tokenization and subsequent natural language processing tasks. Following cleaning, the text is tokenized by splitting it into individual words. This process facilitates the removal of stopwords, such as "and," "the," and other common words, using the NLTK library. Stopword removal reduces noise in the data and ensures that the remaining tokens are meaningful for analysis.

Stemming and lemmatization are applied to the tokenized text to further process the data. Stemming reduces words to their root forms by removing suffixes, for example, converting "running" to "run." However, stemming may produce words that are not linguistically meaningful. To address this limitation, lemmatization is employed, which considers the context and reduces words to their base or dictionary form. This dual approach ensures that word variations are treated consistently, enhancing the quality of the text for feature extraction. Text normalization is then performed to remove non-alphanumeric tokens, retaining only valid tokens that contribute to the text's semantic meaning. The normalized tokens are saved in a structured format for reuse.

Feature extraction is a critical component of the pipeline, as it transforms text data into numerical representations suitable for machine learning models. Two feature extraction techniques are used. The first is Term Frequency-Inverse Document Frequency (TF-IDF), which converts the text into numerical vectors based on the importance of words within a document relative to the entire dataset. The resulting sparse matrix captures the significance of each word in the context of the dataset, with up to 5,000 features. The second technique involves generating word embeddings using the Universal Sentence Encoder (USE). USE produces 512-dimensional dense

Submitted by: Pratyush Kumar Singh
LinkedIn, Email, GitHub

vectors for each text, capturing semantic and contextual information. These embeddings are particularly useful for deep learning models.

The labels in the dataset, represented by the `Category` column, are encoded into numerical values using the LabelEncoder. This step ensures compatibility with machine learning algorithms that require numerical inputs.

Once the data is prepared, a Random Forest classifier is implemented as the traditional machine learning approach. The dataset is split into training and testing sets, with 80% of the data used for training and 20% for testing. The model is trained using the TF-IDF features and achieves an accuracy of 99.50% on the test set. A detailed classification report shows near-perfect precision, recall, and F1-scores across all categories, demonstrating the model's ability to handle balanced data effectively. The trained Random Forest model is saved using Joblib for future use.

For the deep learning approach, a neural network is built to leverage the USE embeddings. The network consists of an input layer matching the 512-dimensional embedding size, followed by two dense layers with 128 and 64 units, respectively, and ReLU activation functions. Dropout layers are included to prevent overfitting by randomly deactivating a fraction of neurons during training. The output layer uses a softmax activation function to perform multi-class classification. The model is compiled with the Adam optimizer and categorical cross-entropy loss function, and training is conducted over 10 epochs with a batch size of 32. The deep learning model achieves a validation accuracy of 94.83%. The classification report highlights high precision and recall for most categories, though minor variations are observed in specific classes, such as News. The trained deep learning model is saved in HDF5 format for future use.

Both models demonstrate strong performance. The Random Forest classifier excels in handling high-dimensional sparse data produced by TF-IDF, achieving nearly perfect classification accuracy. The deep learning model, while slightly less accurate in this case, offers the advantage of capturing contextual and semantic nuances through the USE embeddings. The results underscore the complementary strengths of traditional machine learning and deep learning approaches for text classification tasks. Overall, this project successfully implements a robust text classification pipeline, showcasing the integration of preprocessing, feature extraction, and model training to achieve high accuracy and scalability. The saved models and features provide a solid foundation for future development, such as hyperparameter tuning or deployment in real-world applications.