Duplicate Question Detection Using Machine Learning

Project Description

This project aims to detect duplicate questions using machine learning techniques. The project processes pairs of questions, extracts various text-based features, and then applies machine learning models to predict whether the questions are duplicates.

Project Structure

The project consists of three main Jupyter notebooks and a script for training and evaluating models:

1. **initial_EDA.ipynb**:
   - Performs initial exploratory data analysis (EDA) on the dataset.
   - Analyzes the distribution of duplicate and non-duplicate questions.
   - Examines the frequency of unique and repeated questions.
2. **only-bow.ipynb**:
   - Implements a Bag-of-Words (BoW) approach for feature extraction.
   - Merges the text data, applies CountVectorizer to extract features, and splits the data into training and testing sets.
3. **bow-with-preprocessing-and-advanced-features.ipynb**:
   - Combines advanced preprocessing techniques and feature engineering with BoW.
   - Extracts additional features such as the length of the questions, the number of words, common words, and word share between pairs of questions.
   - Visualizes these features to understand their distributions and impact on duplicates.
   - Applies machine learning models such as Random Forest and XGBoost to classify duplicate questions.

Usage

1. **Data Preprocessing**:
    - Load the dataset using Pandas.
    - Remove rows with null values in `question1` and `question2` columns.
    - Sample the data for analysis and feature extraction.
    - Engineer features like question length, word count, common words, and word share.
2. **Feature Extraction**:
    - Apply Bag-of-Words using CountVectorizer to convert text data into numerical features.
    - Combine these features with engineered features for model training.
3. **Model Training**:
    - Split the data into training and testing sets.
    - Train models like Random Forest and XGBoost using the training set.
    - Evaluate the models on the test set and measure accuracy.

Results

- The Random Forest and XGBoost models were trained on the extracted features and achieved the following accuracy scores on the test set:
- **Random Forest**: [insert accuracy]
- **XGBoost**: [insert accuracy]

Future Work

- Experiment with additional text preprocessing techniques such as TF-IDF and word embeddings.
- Explore deep learning models like LSTM and BERT for improved performance.
- Fine-tune hyperparameters for better model accuracy.