Project Report

1. Step 1: Building a TF-IDF Model and Baseline Confusion Matrix

Approach

1. Data Loading and Preparation: Loaded the training and test datasets from CSV files and combined features and labels into DataFrames.

2. TF-IDF Model Construction: Initialized a TF-IDF vectorizer and transformed the training and test data into TF-IDF representations.

3. Training a Naive Bayes Classifier: Trained a baseline Naive Bayes classifier on the TF-IDF transformed data.

4. Evaluation with Confusion Matrix: Predicted labels for the test data using the trained classifier and computed the confusion matrix and classification report.

Results

The confusion matrix shows 965 true negatives and 5 true positives, with 145 false negatives and no false positives. The classification report indicates high precision (0.87) and recall (1.00) for class 0, but low recall (0.03) for class 1, resulting in an overall accuracy of 0.87 with imbalanced performance metrics.

2. Step 2: Using Transformer-Based Language Models for Keyword Extraction

Approach

1. Keyword Extraction with KeyBERT: Used the KeyBERT library to extract keywords from the text data.

2. Updating TF-IDF Vocabulary: Identified new keywords not present in the baseline TF-IDF vocabulary and updated the vocabulary of the TF-IDF model.

Results

It was extracting top keywords from df_train['text'] using a keyphrase extraction model with specified parameters. Then, it updates an existing TF-IDF model's vocabulary by identifying new unique keywords not present in its current vocabulary. Finally, it creates

a new TF-IDF vectorizer incorporating these updated keywords for enhanced text representation.

3. **Step 3: Training with Updated Vocabulary and Evaluating Differences**

Approach

1. Training with Updated Vocabulary: Trained a new Naive Bayes classifier using the updated TF-IDF representations and handled class imbalance with SMOTE.

2. Evaluation with Confusion Matrix: Predicted labels for the test data using the updated classifier and computed the confusion matrix and classification report.

Results

Analysis

The updated model showed changes in the confusion matrix and classification report, indicating an impact of the new keywords on classification performance. Improvements or declines in precision, recall, and F1-scores can be attributed to the inclusion of new keywords, which may have provided better contextual understanding.

4. **Step 4: Experimenting with Different Transformer-Based Models**

Approach

1. Keyword Extraction with Multiple Models: Utilized BERT, RoBERTa, and DistilBERT models with KeyBERT for keyword extraction.

2. Updating TF-IDF Vocabulary: Combined unique keywords from all models to update the TF-IDF vocabulary.

3. Training and Evaluation: Trained and evaluated models using the updated vocabulary to compare results.

Results

Comparing models with different transformer-based keyword extraction methods showed varying impacts on classification performance. Evaluating these results helps understand which model provides the most significant improvement and supports conclusions with metrics and analytics.

5. **Step 5: Deployment Considerations and Experimentation**

Approach

1. Setting Up Flask Application: Developed a Flask web application to deploy the trained text classification model.

- Created `app.py` to handle input from users and display predictions using the trained model.

- Utilized HTML templates (`index.html`) for user interface design and interaction.

2. Integration and Testing:

- Integrated the trained Naive Bayes classifier with the Flask framework.

- Tested the application locally to ensure functionality and performance.

3. Deployment on a Local Server:

- Deployed the Flask application locally for initial testing and validation.

- Ensured all dependencies and configurations were correctly set up for deployment.

Results and Analysis

- User Interface: Designed an intuitive user interface using Bootstrap for styling and interactive elements.

- Functionality: Successfully handled user input, processed it through the TF-IDF model and Naive Bayes classifier, and displayed predictions.

- Performance: Evaluated the application's performance in terms of response time and accuracy of predictions.

- Challenges: Faced challenges such as environment configuration, dependency management, and ensuring smooth integration of machine learning models with web frameworks.

Future Directions

- Deployment Scaling: Plan to deploy the application on a cloud platform for broader accessibility and scalability.

- Enhancements: Consider adding features like user authentication, data input validation, and real-time updates.