Introduction

This reflective report describes my experience of training a machine learning model to classify news articles in five categories which are tech, business, sport, politics and entertainment. I used preprocessing techniques, feature extraction methods and machine learning algorithms to achieve this classification.

Procedure

Data Preprocessing

Data preprocessing refers to the technique used to convert the raw data into a clean meaningful data set, and it is essential that we train the machine learning models with structured data for achieving better results.

The first step in my procedure was preprocessing plain text data, by:

1. Converting text into lowercase text
2. Tokenization
3. Removing stop words
4. Stemming

Named Entity Recognition (NER)

NER is the form of **Natural Language Processing,** NER is the task of identifying and categorizing key information or entities in the text, entity can be any words or set of words that refers to same thing.

Sentiment Analysis

The process of analyzing the text to decide if its content is positive, negative or neutral.

Feature Extraction

1. **Word Embeddings**

Word embedding represents a word used in text analysis. Word embedding is in the form of vectors.

2. **Term Frequency - Inverse Document Frequency (TF-IDF) Vectorization**

Method measures the importance of terms in a document relative to a collection of documents.

Model Training and Evaluation

Random Forest is a supervised machine learning algorithm which has many decision trees that take the average to improve the predictive accuracy of that dataset. Based on the concept of ensemble learning, which combines multiple classifiers to solve a complex problem and improves the model's performance. I used a **RandomForestClassifier** for the classification task. To ensure robust evaluation, I used stratified k-fold cross-validation with 5 folds. Additionally, we split the training data further into training and development sets to fine-tune the model. I implemented the **grid search** using **GridSearchCV** from **sklearn.model_selection** to perform the **hyper parameter tuning with cross-validation** to find the best combination of hyperparameters for the RandomForestClassifier model. This optimizes the performance of the model.

Results

Performance Metrics

Accuracy: 97%

Macro-averaged Precision: 97%

Macro-averaged Recall: 97%

Macro-averaged F1-score: 97%

Python Code

Libraries used in the code.

1. Os

2. SpaCy
3. Numpy
4. Pandas
5. Stopwords
6. word_tokenize
7. PorterStemmer
8. SentimentIntensityAnalyzer
9. TfidfVectorizer
10. GridSearchCV
11. StratifiedKFold
12. train_test_split
13. Chi2
14. SelectKBest
15. classification_report
16. RandomForestClassifier

User defined function.

1. **preprocess_text(text)**

Description: Function performs preprocessing on the given text by converting it to lowercase, tokenizing it, removing stop words, applying stemming, and joining tokens back into a string.

Libraries: nltk, nltk.corpus.stopwords, nltk.tokenize.word_tokenize, nltk.stem.PorterStemmer

Returns: Preprocessed text (string)

2. **extract_named_entities(text)**

Description: Function extracts named entities from the given text.

Libraries: spacy

Returns: List of named entities found in the text

3. **get_sentiment(text)**

Description: Performs sentiment analysis on the given text. It analyzes the sentiment polarity of the text and classifies it as positive, negative, or neutral.

Libraries: nltk.sentiment.SentimentIntensityAnalyzer

Returns: Sentiment polarity value (-1 for negative, 0 for neutral, 1 for positive)

4. **get_word_embeddings(text)**

Description: Function Computes word embeddings for the given text. Word embeddings capture semantic information in the text.

Libraries: spacy

Returns: Vector representation of the text

Built in library functions.

1. **TfidfVectorizer()**

Description: This function converts a collection of raw documents to a matrix of TF-IDF features.

Libraries: sklearn.feature_extraction.text.TfidfVectorizer

Returns: TF-IDF matrix of features

2. **RandomForestClassifier()**

Description: Function implements a random forest classifier.

Libraries: sklearn.ensemble.RandomForestClassifier

Returns: Trained random forest classifier model

3. **SelectKBest(chi2, k=1000)**

Description: Function selects the top k features based on the chi-squared statistical test.

Libraries: sklearn.feature_selection.SelectKBest, sklearn.feature_selection.chi2

Returns: Transformed feature matrix with selected top k features

For loops in the code.

1. **Category and File Iteration**

Purpose: This loop iterates over each category in the dataset and then iterates over each file within that category, to read the text content of each file, preprocess the text, extract named entities, perform sentiment analysis, and compute word embeddings.

Libraries: os, pandas, nltk, spacy, SentimentIntensityAnalyzer from nltk.sentiment

Returns: Appends preprocessed data along with additional features like named entities, sentiment, and word embeddings to the preprocessed_data list, which contributes to building the dataset for model training and evaluation.

Contribution: This loop plays a crucial role in preparing the dataset by extracting relevant features from each document, which are essential for training and evaluating the classification model.

2. **Stratified K-Fold Cross-validation with hyperparameter tuning**

Purpose: This loop iteratively splits the data into training, validation, and test sets using stratified k-fold cross-validation. For each fold, it selects the best hyperparameters for the RandomForestClassifier model using grid search.
Libraries: skf_cross_validator.split, SelectKBest from sklearn.feature_selection, GridSearchCV from sklearn.model_selection, RandomForestClassifier from sklearn.ensembleclassification_report from sklearn.metrics
Returns: Prints a classification report for each fold.
Contribution: This loop contributes to optimization of the RandomForestClassifier model by finding the best combination of hyperparameters through grid search within each fold of cross-validation. It helps in ensuring that the model is well-tuned and generalizes well to unseen data by evaluating its performance on the development set. The feature selection step (SelectKBest) further enhances model performance by selecting the most relevant features for training the model. Overall, this loop is crucial for improving the predictive accuracy and robustness of the RandomForestClassifier model for the given dataset.

Variables in code.

1. **data_path**

Data Type: String

Purpose: Path to the dataset.

2. **preprocessed_data**

Data Type: List of Dictionaries

Purpose: Stores preprocessed data along with additional features for each document, such as preprocessed text, category, named entities, sentiment, and word embeddings.

3. **news_categories**

Data Type: List of Strings

Purpose: Stores the categories.

4. **dataframe**

Data Type: Pandas DataFrame

Purpose: Stores the processed data in tabular format.

5. **Vectorizer**

Data Type: TfidfVectorizer object

Purpose: Used for feature extraction using TF-IDF vectorization.

6. **X**

Data Type: Sparse Matrix (usually scipy.sparse.csr_matrix)

Purpose: Stores the input features (TF-IDF vectors) for training the classification model.

7. **y**

Data Type: Pandas Series or ndarray.

Purpose: Stores the target labels corresponding to the input features.

8. **randomForestClassifier**

Data Type: RandomForestClassifier object

Purpose: Represents the machine learning model (Random Forest Classifier).

9. **feature_selector**

Data Type: SelectKBest object

Purpose: Performs feature selection by selecting the top k features based on chi-square statistics.

10. **skf_cross_validator**

Data Type: StratifiedKFold object

Purpose: Used for stratified k-fold cross-validation.

Usage of Development Set

We introduced a development set to further fine-tune the model and prevent overfitting. This allowed us to evaluate the model's performance on unseen data during training.

Future Improvements and Biases

In future iterations, we could explore more advanced feature extraction techniques and experiment with different machine learning algorithms.

Conclusion

In conclusion, my approach successfully classified news articles into relevant categories using sentiment analysis. By employing rigorous preprocessing, feature extraction, and model training techniques, I achieved robust performance on the task. Moving forward,

continuous refinement and evaluation will be essential to maintain the model's effectiveness and mitigate biases.

GitHub repository link: https://github.com/ShubhamLolge/Applications-of-ML