Text Classification Using IMDb Review Dataset

by
Adam Abdellatif Mokhtar

Table of Contents

Project scope

The project scope encompasses the development and implementation of a text classification model to accurately categorize IMDb movie reviews as either positive or negative.

Project steps

Dataset

The purpose of this dataset is to facilitate sentiment analysis tasks, particularly in the context of movie reviews. Researchers, data scientists, and machine learning practitioners can use this dataset to train and evaluate sentiment classification models. By analyzing the textual content of reviews and their associated sentiment labels, one can develop algorithms capable of automatically identifying the sentiment conveyed in movie reviews.

Dataset link : Kaggle

Data preprocessing

- Removing HTML tags
  - Removed by using BeautifulSoup to detect this tags and extract texts from it
- Removing special characters
  - Removed by using re library to replace all these special characters by space
- Removing stopwords
  - First download English stop words from nltk
  - Second using re method to replace this stop words by space
- Convert all text to lowercase
- Tokenization : split text into tokens
  - Using word_tokenize method from nltk library
- Stemming or Lemmatization : two ways to return each word to it's base
  - Porter stemmer: a process for removing suffixes from words in English
  - Word lemmatization: to reduce a word to its root form

Clean_data function

Args: text (string): single sentence or paragraph . Stemming (bool): True for stemming False for lemma.

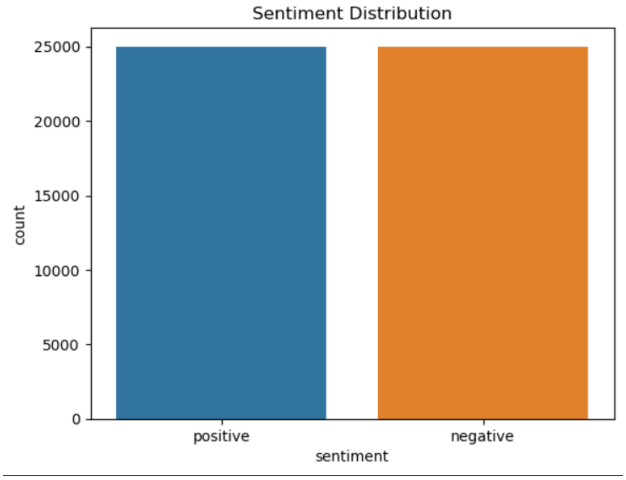
Returns: text: cleaned single sentence or paragraph

Examples: >>> my_function('it is test sentace for the doumentation', True) →

'test sentac doument'

Analysis sentiments

Compare count of the two class in dataset



Each class contains 25000 row so handling misbalancing is not necessary

Feature extraction

- Bag of words (BOW)
  - Using Countvectroizer method from sklearn
  - Num of features extracted is 5000
- Term frequency inverse document frequency (TFIDF)
  - Using TfidfVectorizer from sklearn
  - Num of features extracted is 5000
- Word embedding
  - Word2vec

    Trained the word2vec model in our dataset and used It to map the dataset

    create_word_embeddings(text,model) function

    Input: text (string) : single sentence or paragraph , default parameter for model

    Returns: array represents input text into numbers
  - GloVe

    Used pretrained model to map the dataset

    text_to_vector(text,embedding_index,embedding) function

    Input: text (string): single sentence or paragraph,

    embedding_index (dict)

    embedding_dim (int):number of features extracts

    Returns: array represents input text into numbers

Model Training

Training Machine learning models

1. Encoding sentiments by LabelEncoder() assign 1 for positive and 0 for negative
2. Split dataset into train and test datasets
    a. By using train_test_split method
    b. Test size = 20% (10000 row) and train size = 80% (40000 row) from the main dataset
3. Chose ML models for training
    a. Decision Tree: Simple and interpretable, useful for both classification and regression.
    b. Random Forests: An ensemble of decision trees, good for handling overfitting and improving accuracy.
    c. Logistic Regression: Commonly used for binary classification problems.
    d. Support Vector Machines (SVM): Effective for classification tasks, especially with high-dimensional data.
4. Hyperparameter tuning
   the process of finding the optimal values for the parameters that are not learned by the machine learning model during training
    a. Grid search is a hyperparameter tuning technique that exhaustively searches through a predefined set of parameter values to find the optimal combination for a model.
    b. Using GridSearchCV method from sklearn.model_selecttion
    c. Applied grid search on Random forest algorithm to improve its performance
    d. Saved the best model I got in models folder and named it best_ML_model.pkl

Build and Training Deep learning Model

Long short-Term memory (LSTM)

Used TensorFlow framework for build and train my model

Preprocessing steps

1. Tokenize the text with TensorFlow Tokenizer with number of features 3000
2. Converts the text data into sequences of integers, where each integer represents a word based on the word index created in the previous step.
3. Pads the sequences to a fixed length of 300 words.
4. Split dataset into three small data sets Train, validation, and test

Building

1. Define the model as Sequential model
2. Add embedding layer

3. Add regularization technique like SpatialDropout1D layer for drops entire feature maps.
4. Add LSTM layer
5. Add Dense layer
6. Compile the model
    a. Using loss function ‘binary_crossentropy’
    b. Using optimizer ‘adam’
    c. Using metrics ‘accuracy’

Training the model

1. Number of epochs = 3
2. Patch size = 256

I got training accuracy = 0.90 and Validation accuracy = 0.87

Evaluation

For the evaluation I used

1. Accuracy
2. Precision
3. Recall
4. F1
5. AUC-ROC
6. Classification report
7. Confusion matrix

All this matrices used from sklean.matrices

evaluate_model(y_test,y_pred) function

Input: y_test (list): the True values for each row,

y_pred (list): the model’s predicted value for each row

output: the function return nothing but visualize

1. Classification report
2. Value for each performance measure matrix mentioned before

Results for Evaluate each model

1. Decision tree

```
Result of: Decision Tree
              precision    recall  f1-score   support

    positive       0.72      0.72      0.72      5035
    Negative       0.72      0.71      0.71      4965

    accuracy                           0.72     10000
   macro avg       0.72      0.72      0.72     10000
weighted avg       0.72      0.72      0.72     10000

Confusion Matrix
[[3637 1398]
 [1432 3533]]
Accuracy: 0.717, Precision: 0.7164875278848104, Recall: 0.7115810674723061, F1 Score: 0.7140258690379951
ROC_AUC: 0.7169623311542266
```

2. Random forest

```
Result of: Random Forest
              precision    recall  f1-score   support

    positive       0.84      0.85      0.84      5035
    Negative       0.85      0.83      0.84      4965

    accuracy                           0.84     10000
   macro avg       0.84      0.84      0.84     10000
weighted avg       0.84      0.84      0.84     10000

Confusion Matrix
[[4284  751]
 [ 823 4142]]
Accuracy: 0.8426, Precision: 0.8465154302064173, Recall: 0.8342396777442095, F1 Score: 0.8403327246906067
ROC_AUC: 0.842541884552343
```

3. Logistic regression

```
Result of: Logistic regression
              precision    recall  f1-score   support

    positive       0.87      0.86      0.87      5035
    Negative       0.86      0.87      0.87      4965

    accuracy                           0.87     10000
   macro avg       0.87      0.87      0.87     10000
weighted avg       0.87      0.87      0.87     10000

Confusion Matrix
[[4348  687]
 [ 638 4327]]
Accuracy: 0.8675, Precision: 0.862983645791783, Recall: 0.8715005035246727, F1 Score: 0.867221164445335
ROC_AUC: 0.8675278088626343
----------------------------------------------------------------
```

4. Random forest with Grid search

```
Best parameters found:  {'max_depth': None, 'min_samples_split': 10, 'n_estimators': 200}
              precision    recall  f1-score   support

    positive       0.85      0.85      0.85      5035
    Negative       0.85      0.85      0.85      4965

    accuracy                           0.85     10000
   macro avg       0.85      0.85      0.85     10000
weighted avg       0.85      0.85      0.85     10000

Confusion Matrix
[[4283  752]
 [ 752 4213]]
Accuracy: 0.8496, Precision: 0.848539778449144, Recall: 0.848539778449144, F1 Score: 0.848539778449144
```

5. Deep learning model (LSTM)

```
LSTM
              precision    recall  f1-score   support

    positive       0.89      0.87      0.88      5073
    Negative       0.87      0.88      0.87      4927

    accuracy                           0.88     10000
   macro avg       0.88      0.88      0.88     10000
weighted avg       0.88      0.88      0.88     10000

Confusion Matrix
[[4394  679]
 [ 567 4360]]
Accuracy: 0.8754, Precision: 0.8652510418733875, Recall: 0.8849198295108586, F1 Score: 0.874974914710014
ROC_AUC: 0.8755369894646743
------------------------------------------------------------
```