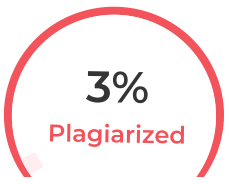
Apr 28, 2023

Plagiarism Scan Report



Characters:5118	Words:771
Sentences:39	Speak Time: 7 Min

Excluded URL	None

Content Checked for Plagiarism

4. Implementation In this section, we have presented the steps followed for implementation of the system architecture during the project development in terms of methodology, testing and result analysis. 4.1. Methodology The methodology used in this project can be divided into 3 major subsections: 4.1.1. Preprocessing Dataset (using pandas, numpy and re) Steps followed while preprocessing the custom dataset we curated: ● Removing/parsing HTML tags and links/URLs: Many mails were in raw format, containing all sorts of HTML tags and links/URLs. We don’t want our model to learn any features based on that. So, we used the BeautifulSoup library for parsing the HTML tags and pulling data out of them. Links starting with http and www were removed using regular expressions. ● Removing punctuation and non-alphabets: They affect the results of any text processing approach, especially what depends on the occurrence frequencies of words and phrases, since they are used frequently in text. Punctuation and non-alphabets were removed using string functions and regular expressions respectively. ● Converting to lowercase and removing stop words: The messages in the dataset were converted to lowercase using string function. By removing stop words (very commonly used frequent words which generally don’t provide any additional information/learning features) from text, the model can focus on the important words and their sequence/order instead. 4.1.2. Encoding Data for the Model (using Keras API of TensorFlow package) Steps followed for encoding the data for the model: ● Creating tokenizer (num_words=30000) ● Defining function for sequencing and padding/truncating (maxlen=150) ● Defining function to perform train-test split and fit the tokenizer on train set to generate sequences of word indexes (train_size=0.8) 124.1.3. Building, Training, Evaluating and Optimizing the Model (Using Keras) In this Project, We have trained using three different neural networks models, such as Simple neural networks (SNN), Convolutional Neural Network(CNN),Recurrent Neural Network (LSTM). Simple neural networks: we defined the model architecture using the Keras Sequential API. In this case, the model consists of an embedding layer, a flatten layer, and a dense layer with a sigmoid activation function.The Embedding layer has input_dim=30000(the total number of tokens in the word index dictionary as specified while creating the tokenizer) and output_dim=64, the flatten layer converts the 2D output of the embedding layer to a 1D vector, and the dense layer applies a sigmoid activation function to produce a binary output.The

model is compiled using adam optimizer with a learning_rate=0.0001,binary cross-entropy is used as the loss function, and accuracy is used as the evaluation metric.The model was trained for 15 epochs with a batch_size of 128 and validated on a validation_data of 0.10 validation split. For optimizing the model, we trained it multiple times and manually tuned its hyperparameters each time, until we achieved the best results Convolutional Neural Network(CNN) In this case, the model consists of an embedding layer, a 1D convolutional layer, a global max pooling layer, and a dense layer with a sigmoid activation function.The convolutional layer applies 128 filters of size 5 to the input data, which is a sequence of words represented by word embeddings. The ReLU activation function is used to introduce non-linearity into the layer output.Like our previous model, we kept the learning rate same. Similarly the number of epochs remained same so that we can do a comparative analysis with one another. Since this is a more complex model architecture than the simple neural network architecture, It took longer to train and required more computational resources. Recurrent Neural Network (LSTM). Here,we again set the input dimension to 30000, output dimension to 64, and input length to the length of your padded sequences equivalently to our earlier models.Added an LSTM layer with 128 units and set return_sequences to False, as we only want the final output of the LSTM. post that, Compiling and training sections remained same.For optimizing the model, manually tuned its hyperparameters each time, until we achieved the best results. The model's hyperparameters we tuned were the number of layers,output embedding dimension, learning rate, batch size, etc. 4.2. Testing/Verification All the models produced an average Test Accuracy and Test Accuracy which is satisfactorily high. After training the models around three times and best results have been chosen for the comparative analysis . That being said, the data set we used to train our model is quite large. Thus it took longer time. After the project development was complete, we performed live prediction on a sample data set to predict movie reviews either it is positive or negative.We verified the model's prediction and the results were quite satisfactory. The comparative analysis, classification report, confusion matrix and accuracy graphs of each models are given below.

Sources