1 Experiment

Used different Machine learning models and Deep learning models. Observation and experiment performed while building the model are mainly preprocessing the data set, finding any pattern in the data set, and finding what is essential in the Dataset affecting sentiment classification, then making a pipeline of the model followed by setting metrics for the model according to the Dataset given.

1.1 Preprocessing and Observation in Dataset

The Dataset consists of a string, and the sentiment of that string is either positive or negative Distribution of sentiment is 9178 negative sentiment and 2363 positive sentiment, which clarify that the Dataset is imbalanced. So we need a metric for testing that better defines the imbalanced Dataset.
For testing purposes, I have taken 30% of the whole Dataset and the remaining for training and validation. Since Dataset is imbalanced, I have used "stratify." while splitting the Dataset, which splits the minority class and the majority class by equal Distribution.
Now preprocessing part, so there were HTML tags, Websites, @ mentions, hashtags, numbers, stopwords, emoticons, and noise characters, i.e., any non-word character.
So, basically, I have tried removing stopwords and with stopwords, and the result is with stop words I am getting better results, so basically there were two reasons; first in, stopwords include negations words as well, which directly shows a "Negative" sentiment and other while using Bi-direction LSTM I want to more syntactic meaning from the text.
Similarly, Hashtags are essential, so I haven't removed these patterns.
Remove Website markup, remove @ mentions, and Save emoticons for later appending; emoticons help to find the sentiment; remove any non-word character and append the emoticons, removing the noise character for standardization.
Convert to lowercase
Used TfidfVectorizer, CountVectorizer, and TfidfTransformer in model pipelines for mapping string to vectors, and Pipeline is defined better in Model Sections.
For tokenization, we used two different methods default tokenizer, i.e., splitting by spaces, and other is porter stemmer for tokenization in the Pipeline.

1.2 Models Used

1.2.1 Clasical ML model

Used Logistic Regression, Support Vector Machine, and Multinomial Naive Bayes with three different variants used undersampling, OverSampling, and the simple model. The best result is obtained in the Simple model without Under Sampling and Under-Sampling. Used Grid search with cross-validation(5,10,15) and used accuracy as a measure during cross-validation.
So, for Logistic Regression and SVM(Kernal tested on RBF and linear, linear gives better results), Pipeline used TF-IDF vectorization and then a model for training. And while fitting, the model used different n-grams, but the best result I got was with 1 gram; while grid searching, I tried with or without a preprocessor. Similarly for tokenizer used a basic tokenizer and porter stemmer.
But for Multinomial Naive Bayes, I have changed the pipeline by adding a vectorizer before TFIDF. Pipeline was vectorizer, then Tfidf Transformer, then classifier.

1.2.2 Deep Learning model

I have tested LSTM, Bi-directional LSTM, and Stacked Bi-directional LSTM. The best result I have got in Stacked Bi-directional LSTM while using LSTM I have tried manually, and LSTM is giving "Positive sentiment" for "@airline bad," but while using Bi-LSTM, this false output is solved. While using Stacked Bi-directional LSTM, I am getting a better result. The Deep Learning model is implemented using TensorFlow. Pipeline for Stacked Bi-LSTM followed:
Used Keras preprocessing text for Tokenizer, Keras preprocessing sequence for padding sequences(because of different length string), used SpatialDropout1D because it drops entire 1D feature maps instead of individual elements. stacked two Bidirectional LSTM layers with dropout as 0.2, and used softmax as activation function, loss function as categorical cross-entropy, adam as an optimizer, and accuracy as metric for training.

1.3 Metric

While training, I used accuracy as a metric, and I used precision, recall, and F1 scores while testing. The scores of each and every model are saved code as well.

1.4 Implementing API

Build 2 API; app API using pyramid and integrated swagger in it, and the other is sentiment_api using Fast API.
In Fast API, I have integrated the classical machine learning model as well as the Deep Learning model and app API; I have integrated the Deep learning model