Moodboard Analytics

Shyam Sunder Kumar ()

February 18, 2021

1. Problem Statement : Sentiment Analysis

Understanding sentiments have been very central and crucial for business now a days. Here we want to build sentiment analysis model which consist of three classes POSITIVE, NEGATIVE and NEUTRAL and serve the model using an API . Before jumping on to the solution I will point out key challenges faced in sentiment Analysis:

- **Challenge 1**: Data Cleaning and pre-processing.
- **Challenge 2**: Ambiguity is central to the language. We want to build something that better captures the sentiment with higher accuracy.
- **Challenge 3**: Deployment and Serving the model .

Let's see step by step method how we can approach this problem:

- Step 1: **Data Cleaning and Extraction**

  Here I was provided with a 'dataset.csv' file which contained 923 product and respective reviews stored as a json dump. Our goal was to get all the reviews and their respective labels. For more details see the notebook 'notebooks/DataExtraction.ipynb' on how to extract the final data for training. Also our labels are marked as 1,2,3,4,5 we convert it to three labels positive, negative and neutral .

  Now we have our cleaned data. Now we can move to model building part. Exciting !!

  Note : We have considered label 1,2 as NEGATIVE, 3 as NEUTRAL and 4,5 as POSITIVE.

- Step 2: **Feature Extraction and Model Building**
  - 2.1. **Textual Data** :
    1. Remove all irrelevant characters such as any non alphanumeric characters
    2. Tokenize text by separating it into individual words
    3. Convert all characters to lowercase
    4. Removing stopwords (such as a, an, the, be)etc
  - B. **Feature Extraction**

    To extract features from text we can use multiple methods like bag-of-words, TF-IDF, word2vec embeddings or dynamic word embeddings as well. For our project we have used TFIDF feature extractor and BERT based embeddings.

    For more details go through "notebooks/BERTforSentimentAnalysis.ipynb notebook for more details.

  - C. **Models** We have used Naive Baye's as a baseline model and BERT based pre-trained model. Let's see how our model is performing on our dataset. After fine tuning we see a 10 percent improvment in accuracy.

- Step 3: **Deploying the model using Pytorch Serve**.

  Now we have created our model, how to serve it as an API . Here we have used Pytorch serve. For more details on How to serve and make predictions open 'MLOPs/ModelDeployment.ipynb'. 'MLOPs' folder contains all things needed for deployment.

Further improvements and analysis.

Interpretation of mis-classified data may give where our model is performing poorly based on which we can change our model further. The best part is we don't need a lot of data to fine tune our model. Interpretation of our model can further improve our insights.

We achieved 10 percent better accuracy than Naive Bayes based baseline model.