Sentiment Analysis of Customer Feedback

IITMOP

Niveath A
Shankar Balajee
Srinivasan Kidambi
Shiva Surya

Indian Institute of Technology Madras

28 January 2023

Overview

Analysis of the Problem Statement

Objective of the problem

Sentiment analysis is the process of analysing textual data and inferring the general mood of the text.

- Our task involves the follows
  - We are required to develop a robust model that can analyse verbatim feedback from customers and label them as per their sentiment.
  - The model must be **quick, efficient** and also **easily extendable** to different languages and domains.

Analysis of the Problem Statement

Analysing the dataset

- The data provided to us has positive labels and negative labels.
- There are some outliers in the data as well , which have no sort of tone whatsoever.

For example ”Convolve 2023” was a feedback given, and yes that has no relevance to the product or service .

Approach

To solve this problem, we have tried a variety of approaches from different angles .

In this section we will have a brief look at our various attempts, and the best solution to the problem with a justification for the same.

Data Processing

In order to judge anything we need some things as a baseline, so at first, we tried approaching the problem by traditional ML methods. First, we split our data into train set and validation set with a ratio of **85: 15**

Approach

Text Processing

- One must first process the text into a vector of numbers. These are called **embeddings** and can be done using a variety of approaches.
- We tried out methods such as **Bag-of-Words and TFIDF vectorization**. We could not achieve best results using the above methodologies. The final approach will be described in a while.

Approach

Baseline Models Implemented

- We tokenised our text using the techniques described in the previous slide
- Training of the vectors was done on linear models and **Ensemble** Techniques. Upon observation, TFIDF vectorisation was clearly better.
- Post this, models like **RandomForest, XGB** were trained. XGBoost outperformed all ensembling techniques like Decision Trees and CatBoost. Logistic Regression could not improve the accuracy.
- Best score using the TFIDF embeddings were obtained using Random Forest , which was about 60% in terms of validation F_1 score.

Approach

Deep Learning Approach

- Now in order to improve the accuracy, we tried taking a Deep Learning Approach using **LSTM**(Long-Short-Term-Memory) on pre-trained word-embeddings like **glove-embeddings** and spacy embeddings.
- These were chosen as they have been trained with billions of words and often produce rich-quality embeddings. Fresh embeddings were also created by us but as expected , our embeddings were of much lower quality than the above mentioned.
- Using Glove-Embeddings and LSTM , we were able to achieve an F_1 score of about 80%

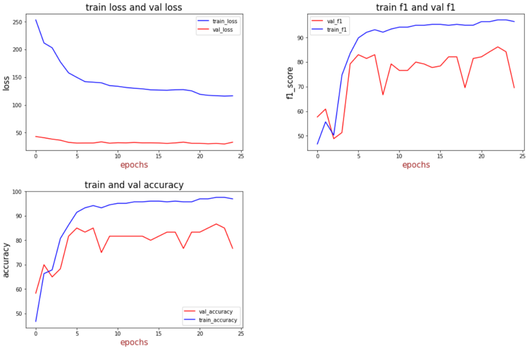
Approach

Transformers

- **Transformers** have recently caused a buzz in the world of NLP and are regarded as the best language model with high parellisability, long-range dependency capturing and ease of training.
- As many pre-trained models exist on Hugging face, **Mini LM-6, distill-roberta** were tried on the data with **mean pooling** and fine tuning of the final layer.

Approach
Plots

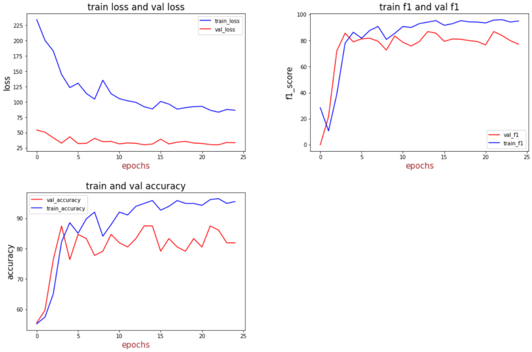
Figure: *distill-roberta training curve*



Approach

Plots

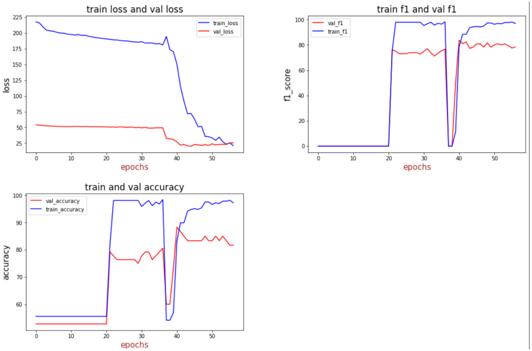
Figure: *mini-lm6 training curve*



Approach

Plots

Figure: *multi-qa-mpnet training curve*



Towards our final architecture

- Combining the word embeddings into **document embeddings**(which serve as a feature vector for the entire sentence(s)) required more detailing.
- Simple Mean Pooling of words would **not** serve well as the sequential context of words is lost.
- However RNNs serve as a good network to combine these feature vectors of words into a single dense representation of the text.
- An LSTM network was chosen over GRU due to the short length of text, and required lesser training.

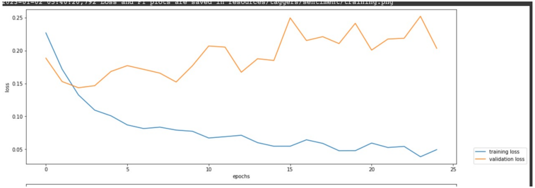
Classification

- The classifier we used is a simple **feed-forward neural network** with dropout layers for regularisation.
- **LayerNormalisation** is done in the RoBERTa framework for better generalisation.
- Ensemble methods like AdaBoostClassifier were not able to capture features correctly. We believe this is because AdaBoost uses **"stumps"** which are single heighted decision trees which are often used only for sparse encodings.

Hyperparameters

1. The training pipeline was created using **Flair** an open source library, which consists of various state-of-the-art NLP frameworks.
2. Various trials were conducted to determine the optimal parameters for our model.

Figure: ***overfitting*** *due to poor choice of parameters*



Hyperparameters

Optimal Hyperparameters

Learning Rate

Learning Rate $= 10^{-6}$

Scheduler

CosineAnnealing with $T_{max} = 30$

Cross-Entropy Loss was used as the loss function.

Optimizer

AdamW with *weight decay* $= 0.1$

Results & Inferences

F_1 Metrics

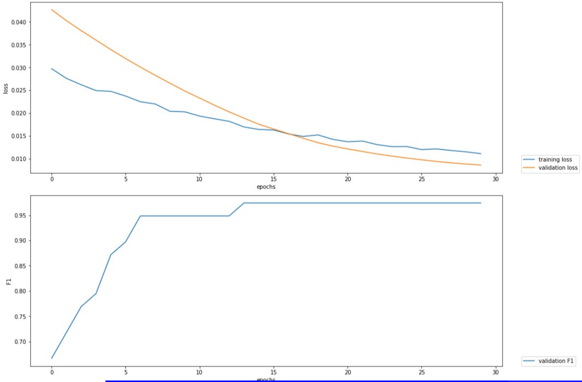
1. Train F_1 score $= 0.9799$
2. Validation F_1 score $= 0.9744$

twitter-RoBERTa-base was used as our transformer taking into consideration the short length of tweets and high correlation of social media posts with emotion.

Results & Inferences

Model Convergence

Figure: *Final Model*



What makes our model better?

- Large Language Models such as RoBERTa do **not** require end-to-end training for performing any particular task. Simply fine-tuning the last few layers is more than sufficient. This is commonly known as **Transfer Learning**.
- RoBERTa has been trained for **multiple languages** and for different domains. So, our model can be easily **extended** to work in various languages as per your needs.
- Our model has very **low inference latency**, order of tens of milliseconds($\sim 30ms$). Hence, the model can manage large volumes of data.

Github Repo

Solution Report, Code and this Presentation can be viewed here

Thank You