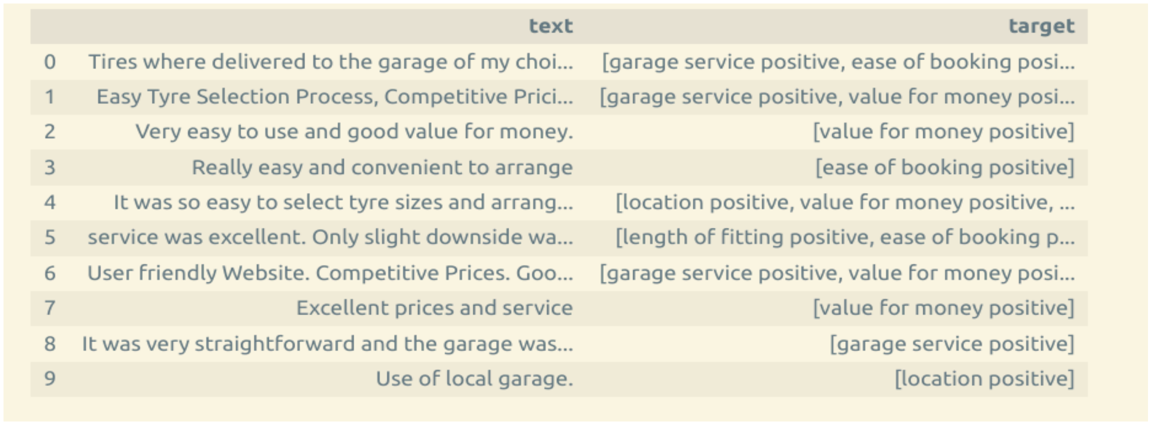
Problem Statement - The task is to develop a deep learning model that given a sample will identify the sub themes along with their respective sentiments.

Approach -We employed a fine-tuning approach to adapt the pre-trained Electra model to our sentiment analysis task. Fine-tuning involves updating the parameters of the pre-trained model using our labeled dataset while retaining the knowledge learned during pre-training. We modified the architecture of the Electra model to include a classification head and a dropout layer that predicts the sentiment label of input text samples. And next we have used Google's BERT model and added a dropout layer and linear layer.

Data Preprocessing- The dataset consists of samples having sub themes associated with it along with their sentiments. But each row doesn't have the same number of subthemes. Ultimately we want to have a dataset in which the first column has a sample and the after that N (number of unique labels, in our case it is 24) columns containing binary values.

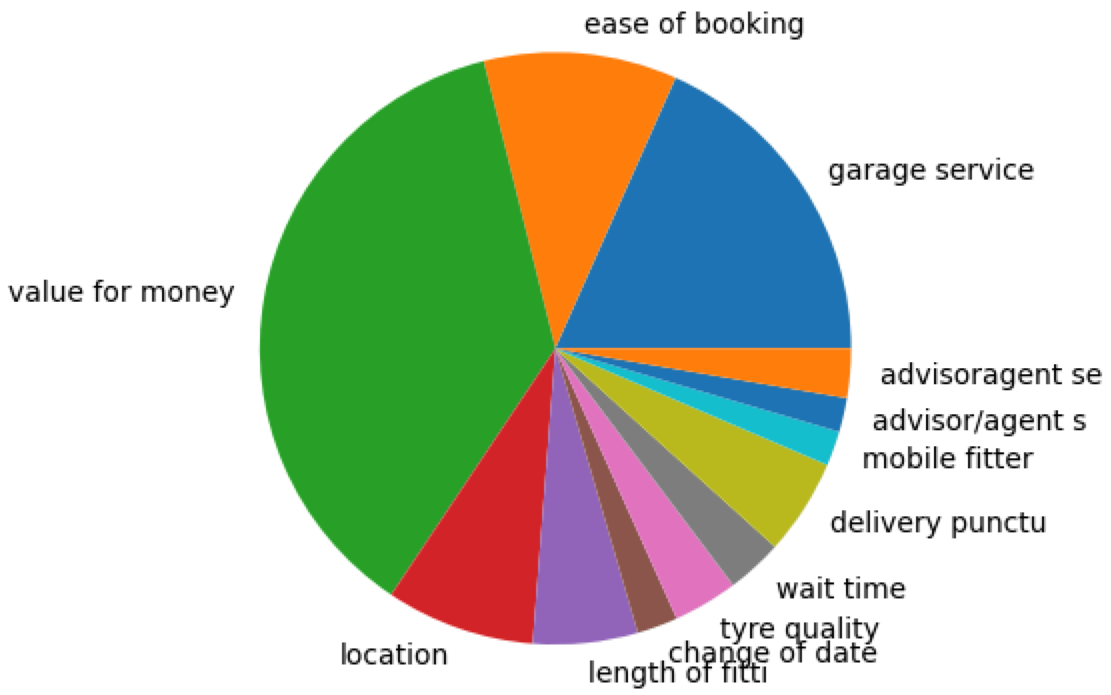
Procedure

1. Replace the np.Nan Values with empty strings.
2. Construct a Deep Copy for the Zeroth Column of the dataframe and then drop the zeroth row for easier processing of the whole data frame.
3. At most we have 14 labels, but some of the rows have one valid column value and some have more than two. So now we need to bring uniformity into the columns making it all one column. For that I have merged all the values of the column into a single list and assigned it to a new column named target.
4. Then I removed all the non-empty strings from all the lists in the "target" column using pandas vectorized operations and merged the previously constructed deep copy of the zeroth column and took only the text column and target column and dropped every other column.
5. After all these operations dataframe looks like this



	text	target
0	Tires where delivered to the garage of my choi...	[garage service positive, ease of booking posi...
1	Easy Tyre Selection Process, Competitive Prici...	[garage service positive, value for money posi...
2	Very easy to use and good value for money.	[value for money positive]
3	Really easy and convenient to arrange	[ease of booking positive]
4	It was so easy to select tyre sizes and arrang...	[location positive, value for money positive, ...
5	service was excellent. Only slight downside wa...	[length of fitting positive, ease of booking p...
6	User friendly Website. Competitive Prices. Goo...	[garage service positive, value for money posi...
7	Excellent prices and service	[value for money positive]
8	It was very straightforward and the garage was...	[garage service positive]
9	Use of local garage.	[location positive]

6. Below figure is the distribution of labels.

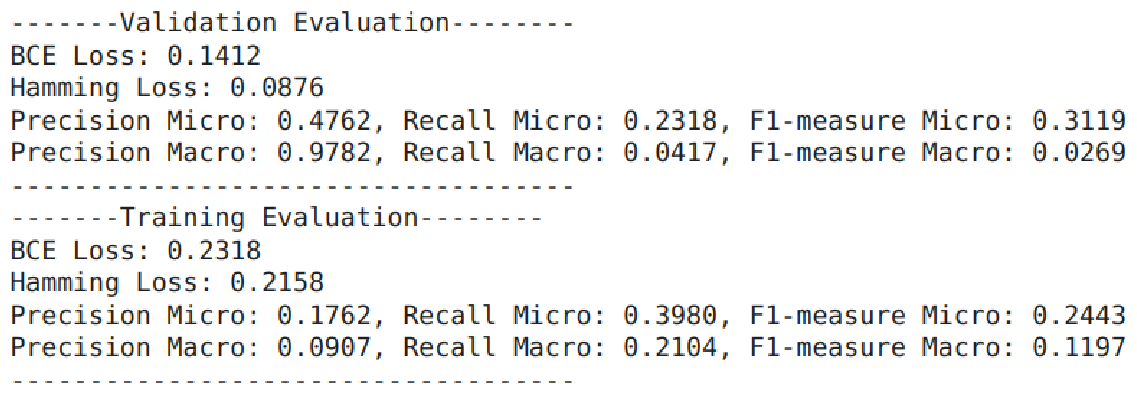


7. There are also some labels that don't have positive and negative in it, let’s call them unstructured labels or noisy labels. So we need to find these labels. These labels are “really good price”, “hassle free”, “good price”, etc. I categorize these as default_positive and default_negative. Instead of dropping them I’ve used sentiment analysis to predict whether these labels are positive or negative. For this I have used a hugging face pretrained model.
8. Since "value for money positive" and "garage service positive" are present in very high frequency, I have undersample it.
9. Then I encoded the labels using MultiLabelBinarizer.
10. Finally, split the data frame into train and test and store them on a filesystem using the pandas ‘to_pickle’ method for training purposes of the model.

Model Analysis-

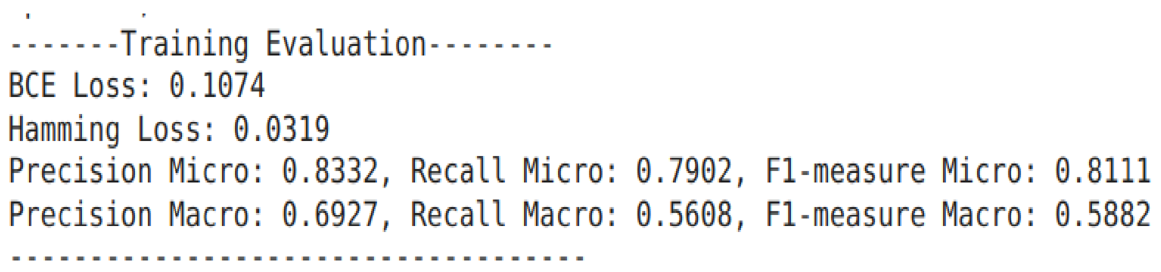
I have taken two models for the Subtheme sentiment classification. First one being google’s Electra and other one being also of Google name bert as the base model. I trained both models for 5 epoch using Binary Cross Entropy as loss function and Adam as optimizer. And here are the results.

ELECTRA - Analysis

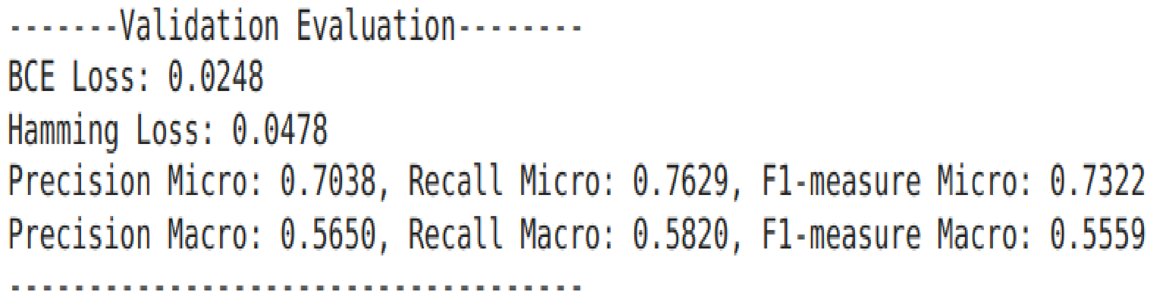


```
-------Validation Evaluation--------
BCE Loss: 0.1412
Hamming Loss: 0.0876
Precision Micro: 0.4762, Recall Micro: 0.2318, F1-measure Micro: 0.3119
Precision Macro: 0.9782, Recall Macro: 0.0417, F1-measure Macro: 0.0269
------------------------------------
-------Training Evaluation--------
BCE Loss: 0.2318
Hamming Loss: 0.2158
Precision Micro: 0.1762, Recall Micro: 0.3980, F1-measure Micro: 0.2443
Precision Macro: 0.0907, Recall Macro: 0.2104, F1-measure Macro: 0.1197
------------------------------------
```

Bert-Uncased - Analysis



```
-------Training Evaluation--------
BCE Loss: 0.1074
Hamming Loss: 0.0319
Precision Micro: 0.8332, Recall Micro: 0.7902, F1-measure Micro: 0.8111
Precision Macro: 0.6927, Recall Macro: 0.5608, F1-measure Macro: 0.5882
------------------------------------
```



```
-------Validation Evaluation--------
BCE Loss: 0.0248
Hamming Loss: 0.0478
Precision Micro: 0.7038, Recall Micro: 0.7629, F1-measure Micro: 0.7322
Precision Macro: 0.5650, Recall Macro: 0.5820, F1-measure Macro: 0.5559
------------------------------------
```

Comparison Of Models

BERT-Uncased performs way better than Electra. Both the models are of similar size but in case of BERT it has an additional linear layer appended which can account for its better accuracy.

Scope of Improvement

1. Instead of using BERT-base-uncased we can use a bigger model like BERT-large-uncased or cased on also to incorporate case of sample and hence a detailed features.
2. We can also fine tune a LLM on sentiment analysis datasets
3. We could also incorporate data augmentation techniques for NLP such as back-translation, synonym replacement, or word shuffling

How to Run the Scripts

1. Pip install -r requirements.txt
2. First run preprocess_dataset.ipynb. It will generate pickle files for model training and validation
3. Run fine_tune.py
4. Run fine_tune_bert.py
5. Run inference_electra.py —text “Replace with your text” to use electra model for inference
6. Run inference_bert.py –text “Replace with your text” to use bert model for inference
7. If only inference is to be done then here the link containing all the generated pickle file and pretrained models for inference.