A Project Report
on

Stack Overflow Question Rating Classification

Submitted by

Deshmukh, Prathmesh
Lewis, Gavin Henry
Katapally, Manognya

The report is intended for the project requirement completion for the course
Data Mining (CSCI-B 565 Spring 2022)

Table Of Contents

1 Abstract

The Stack Overflow question rating classification is a project where we classify the stack overflow questions in the time frame 2016-2020 to the categories of High Quality, Low-Quality with negative score and Low-quality posts that were closed by the community. The main classification model is based on the question title and the question body. This provides us with reliable predictor model. Successfully implemented exploratory data analysis, data filtering, modifying HTML tags in data, and model implementation for classification. Successfully demonstrated the working of ALBERT model which is a lite version of BERT for Self-Supervised Learning of Language Representation. The version of ALBERT used is deployed using the transformers package from Huggingface. ALBERT used has 12 million parameters as opposed to the BERT-base which has 110 million parameters.

2 Keywords

Stack Overflow, Natural Language Processing, Self-Supervised Learning, BERT, BERT-base, ALBERT, AlbertTokenizer, Classification, CUDA, TPU.

3 Introduction

The project covers the entire pipeline implementation of the machine learning for classification on the “60k Stack Overflow Questions with Quality Rating” data set that is available on Kaggle. We have explored the data and have performed various visualization on it which are covered in depth in the subsequent chapters. The dataset is cleaned, and we have employed two distinct strategies to do so. The ‘Title’ and the ‘Body’ attributes of the data are combined as a single feature for the inference in the model predictor. The classification model that we Focus on is ALBERT[2] which is a lite version of BERT[1]. The benefits of full network pre-training provide us with an optimal model for training the code with the limited training data. The BERT [3] algorithm being a heavy model with 110M nodes makes training slow and uses high memory. These are solved by using the ALBERT model which can also use the benefit of parallel computing using CUDA and TPUs in the transformer modules.

4 Methods

4.1 Understanding the Data

The dataset used in the project is “60k Stack Overflow Questions with Quality Rating” which is a dataset that is hosted on Kaggle (https://www.kaggle.com/datasets/imoore/60k-stack-overflow-questions-with-quality-rate). The data obtained is first explored to understand the further steps that is needed to be done for exploratory data analysis.
First, we try to infer from the ‘tags’ that are present in the data:

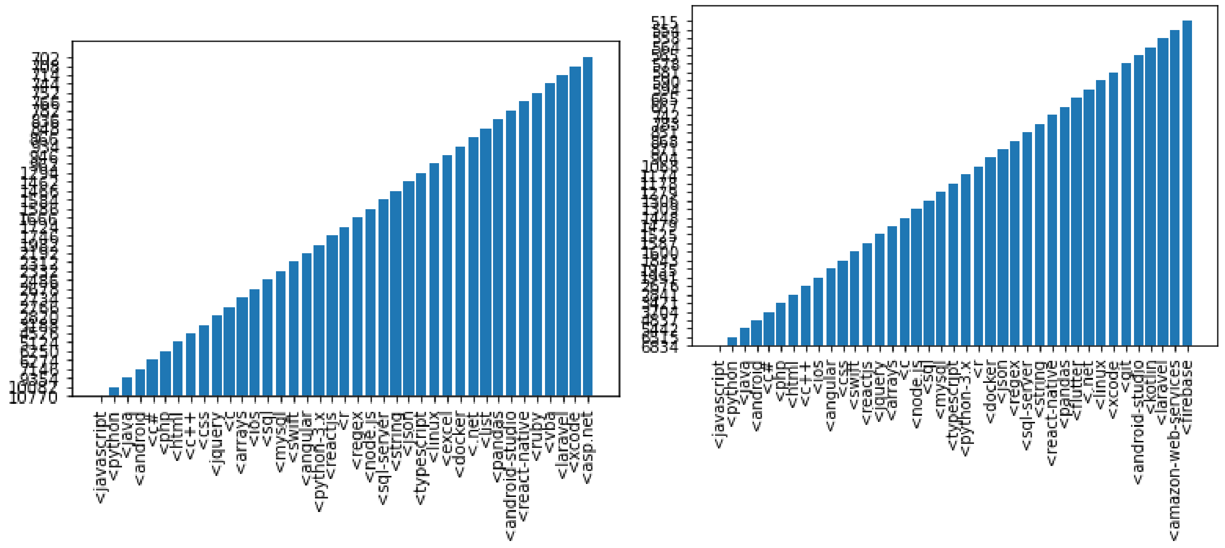
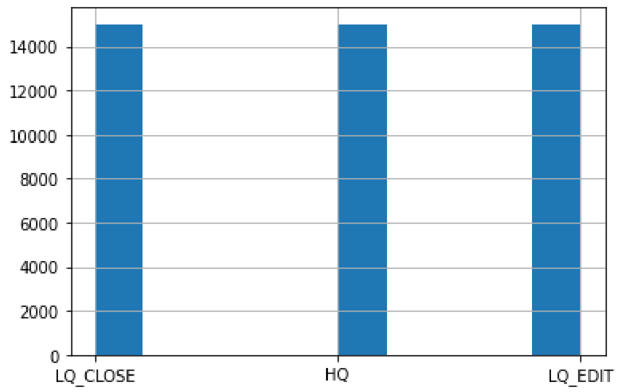
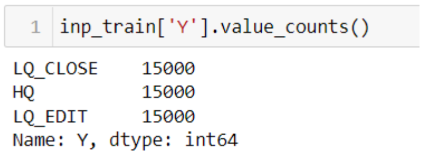


Fig 1 This is the bar plot to the left is that that we obtained for the count of the tags that are available in the tags attribute. And the one to the right is the relation of the attribute to the target that we obtain.

We have also checked if the target variable that we have (attribute ‘Y’) is balanced or not. For that we have counted the distribution of the target variable as well as plotted a histogram.





```
inp_train['Y'].value_counts()
```

```
LQ_CLOSE     15000
HQ           15000
LQ_EDIT      15000
Name: Y, dtype: int64
```

Fig 2 The above set of images we can conculde that the target variable is balanced and we can proceed with the data cleaning phase of the model creation.

We have also Plotted the histogram for the length of the body string removing the outliers in the data. The following is the distribution of the length of the body string.

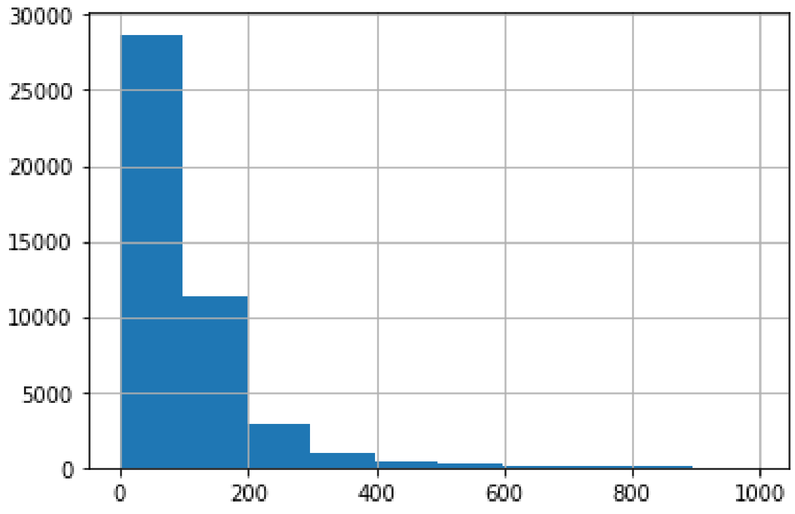
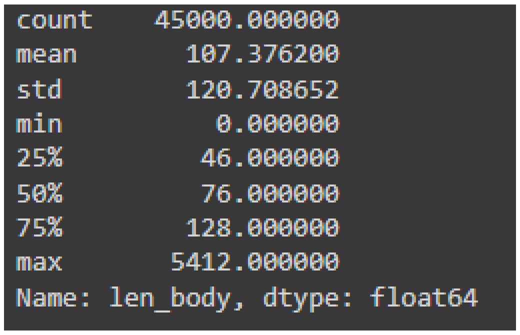


Fig 3 Plot of the histogram for the length of the body string removing the outliers in the data



```
count    45000.000000
mean       107.376200
std        120.708652
min          0.000000
25%         46.000000
50%         76.000000
75%        128.000000
max       5412.000000
Name: len_body, dtype: float64
```

Fig 4 The following is the distribution of the length of the body string.
Since the mean is 107, we have taken the embedding size as 128.

4.2 Data Cleaning

For data cleaning we have used two seaprate approaches:

1. Using html2text: Used html2text package to extract the html elements from the ‘body’ attribute that is present in the data. This function will remove the other html tags that are present in the data. This approach is non-optimal as it will remove the tags that are present in the dataset question as well

2. Using Regular Expression and contractions: The regular expression that are used can filter the html tags that are present in the data of the ‘body’ attribute. The ‘title’ and the ‘body’ attribute are merged as a single entity. Python contractions are used to expand the contractions that are present in the sentenses in the data, eg. “Don’t” is replaced with “Do not”

The second approach of data cleaning is futher used in the pipleine of this report.

4.3 Implementation

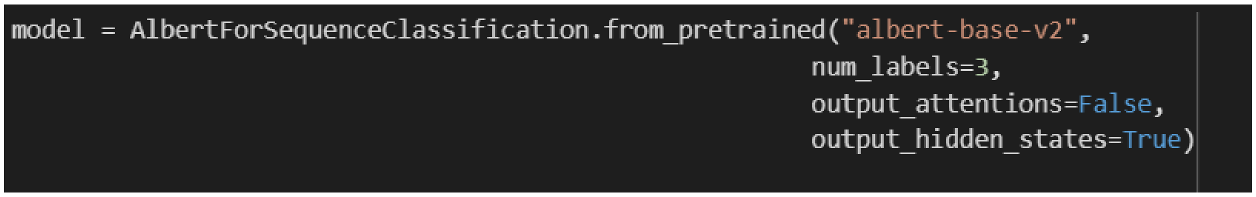
For the implementation part, we implemented the ALBERT [4] model and also explored the implementation of the BERT model as well for the classification problem. We have decided to use the ALBERT model as it is computationally more optimal to use this.The ALBERT model has far much lower trainable parameters and thus would train faster.

The ALBERT [6] model that we have used has the embedding size of 128. The model has the out features or the features at the n-1 layer of the model as 768. The bias of the model is set as true.

For the ALBERT model we use 3 classification label as output as we need to classify the data into 3 output classes - High Quality, Low-Quality with negative score and Low-quality. The albert tokenizer and the albert model for sequence classification is taken from the transformer package on python which is based on the pytorch library.

First the tokens are generated from the albert tokenizer and then the model is generated using the tokens from the output of this tokenizer.

We have used the following model format for creation of the model.:



```
model = AlbertForSequenceClassification.from_pretrained("albert-base-v2",
                                                         num_labels=3,
                                                         output_attentions=False,
                                                         output_hidden_states=True)
```

Fig 5 code can be seen in the notebook section of the code.
We are currently using the ALBERT BASE V2 model.

5 Results

ALBERT model performed best on the configuration of 20 epoch for training with learning rate of 3e-5 and weight decay of 5e-6. We have also conducted tests with the BERT model but due to compute resource limitation we had to switch to a light weight BERT model.[5]

All the training done is using the GPU Tesla P100 16GB Vram which is available on both Kaggle as well as Colab.

The model had an average loss of 0.0201 on the train data and a minimum loss of 0.3538 on the test data. Best model is chosen using the metric of minimum loss on the test data.

5.1 Results - Accuracy

Class	Model Evaluation on Test	Accuracy
LQ_CLOSE	2686/3340	0.8042
LQ_EDIT	3065/3335	0.9190
HQ	2980/3375	0.8830
Total Accuracy on Test		0.8688

The above is the table depicting the accuracy of the model that we get that we get for each class of the output as well the accuracy of the model overall.

5.1 Results – TSNE [7]

To genrate the TSNE [7] we have used the n-1 layer that the model has which has 768 nodes. This is used to visalize the output that we get from the model. The T-sne is key as the data that we have is in a higher dimensional space and thus we have generated the tsne diagrams for the test as well as the train.

Train Data T-SNE

Test Data T-SNE

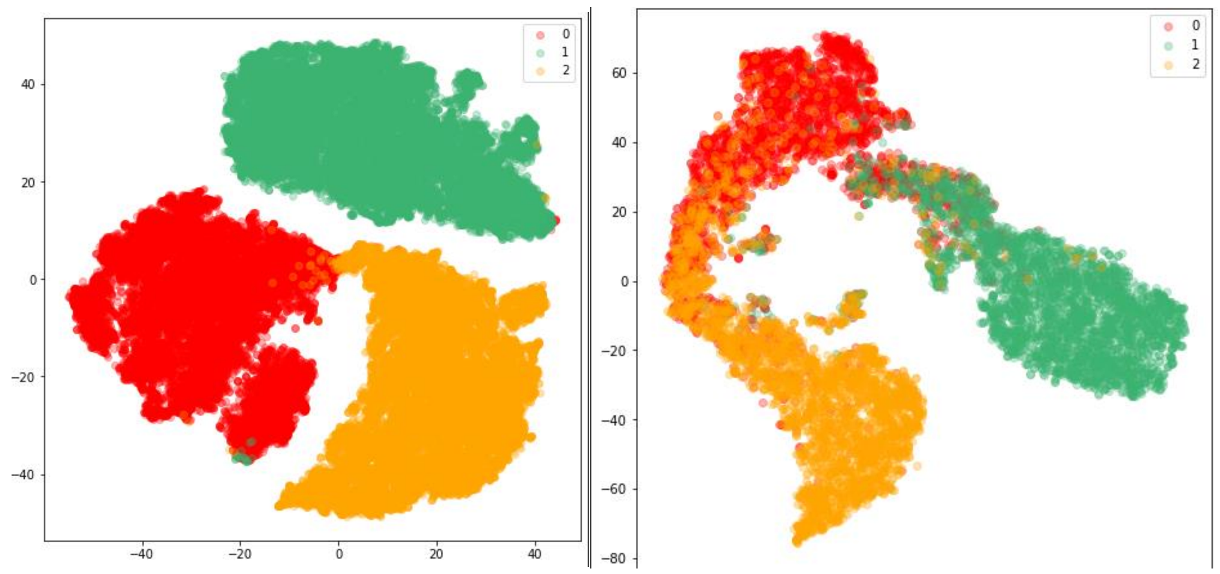


Fig 6 Plots for the T-SNE for test and train data

We can see that the T-SNE graphs that are generated for the test and the train are very uniform as we can see clear cluster separation from the model that we have trained.

6 Discussion

From the above results we can conculde that the ALBERT model is efficent for classifcation of the input strings that we feed. The stackoverflow dataset that we have gets effectively classified. The accuracy of the model thus obtained is 86.88%.

We have thus explored the implementation of the transformer model which is ALBERT. We have also implemented various exploratory data mining stretegies on the data that we have and thus obtained the graphs on the data. We have also successfully implemented the T-SNE results in a lower dimensional space

The T-SNE visualization also shows that the model perfoms effectively as we can see a clear fomation of clusters for each output class map.

7 References

[1] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova https://arxiv.org/abs/1810.04805
[2] ALBERT: A Lite BERT for Self-supervised Learning of Language Representations Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, Radu Soricut https://arxiv.org/abs/1909.11942
[3] BERT Implementation Support documentation: https://huggingface.co/docs/transformers/model_doc/bert
[4] https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270
[5] https://towardsdatascience.com/deep-dive-into-the-code-of-bert-model-9f618472353e
[6] ALBERT V2: https://huggingface.co/albert-base-v2
[7] T-SNE https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html