Journal of High School Science

Predicting stack overflow question quality through the use of deep learning techniques

Manchanda R[1] Klotzman VI[2]

Submitted: September 7, 2022, revised: version 1, September 16, 2022
Accepted: September 24, 2022

Abstract

Stack Overflow is one of the most important and widely-used Q/A Community Platforms for software developers today, with over 8,000 posts on a typical weekday. Since any user is permitted to post questions and answers, in order to reduce clutter and optimize efficiency for users, inappropriate or inadequate posts must be judged and removed as per the community guidelines. Hence, a method for users to predict the quality of their question before posting can allow them to improve their posts and increase the chances of it getting answered. This paper proposes a novel Deep Learning-based Natural Language Processing approach to classify Stack Overflow questions as 'High Quality', 'Low Quality Edit', and 'Low Quality Close'. A Neural Network model was developed using a Distilled version of the BERT encoder, and this network was trained and tested using sample datasets. The experiment resulted in a training accuracy of 96.6% and a test accuracy of 93.5% with previously unseen data. Therefore, it was concluded that the model proposed can successfully predict the quality of Stack Overflow questions and help in the automated moderation of the platform. Future research should explore hyper-parameter tuning for this model and applying this work's output to Quality Prediction on other platforms.

Keywords

Natural language processing, Stack overflow, Deep learning, Neural networks, Bidirectional Encoder Representations from Transformers, Question quality prediction, Bidirectional transformers, Community platform, Data analysis, Natural language toolkit library

---

[1]Corresponding Author: Riya Manchanda, Lancers International School, DLF Phase-V, Sector-53, Gurgaon - 122001, Haryana, India. riyamanchanda10@gmail.com .
[2]Vanessa Ilana Klotzman, University of California, Irvine, California - 92617, USA. vklotzma@uci.edu .

Introduction

Stack Overflow is a community question and answer platform, which helps computer scientists find solutions to various technical challenges by posting questions, or by reading the questions posted by other members (1). Users who post questions receive answers from other members, out of which the user can accept an answer which they found helpful, for other users to refer to. Stack Overflow contains 18 million questions, 27 million answers, 75 million comments, and 55 thousand tags; every month, over 50 million people visit this invaluable platform (2). Being a user-centric community, an appropriate set of guidelines and a method for question moderation becomes crucial.

The community guidelines on Stack Overflow predominantly require users to post questions which are specific and unambiguous – meaning that every question must refer to a particular technology, tool, algorithm, language or API – in order to be considered (3). Generic and casual questions must be avoided to reduce clutter and prevent a drop in efficiency. Questions which do not adhere to these guidelines – or are duplicates – will be closed, or even deleted (3).

Over the years, Stack Overflow has exponentially gained popularity, with programmers and software developers increasingly turning to this forum for technical support. Given that over 8,000 questions are posted on an average day on Stack Overflow and 700 of them are closed, conserving optimal quality – according to the community guidelines – of the content posted becomes imperative, ensuring that everyone benefits (4). This research, which focuses on question quality in relation to Stack Overflow community guidelines, aims to develop a method for predicting the quality of a question before posting, determining whether it is likely to be deleted.

Researchers have explored this topic previously, making particular use of Natural Language Processing and Deep Learning techniques to accurately predict the quality of large datasets of questions and categorize them based on content. The success rate of such research has been high, and methods have varied from analyzing the semantic fields of posts to evaluating the importance of features such as user reputation, presence of code snippets and the relationship between questions and answers (2, 4-7).

The field of Natural Language Processing has made significant progress over the last few years. This discipline of machine learning deals with processing and analyzing Natural Human Language, which can often be rather ambiguous. Newly discovered training and learning techniques have shown promising results in terms of text mining and classification, motivated by the exponential surge in unstructured text data over the last two decades (8). Text classification refers to dividing a set of input text from a document into predefined classes. Various techniques have proven to be effective for this task, including Naive Bayes, Support Vector Machines, and Neural Networks. The general process followed for text classification is as follows: Defining a Dataset, Data Preprocessing, Exploratory Data Analysis, Feature Extraction, Classifier Model Development, and Performance Evaluation (9).

Neural Networks – first proposed in 1944 – are at the heart of Deep Learning today (12). Basic Neural Networks consist of input

layers, output layers, and hidden layers. They are composed of nodes which are connected densely and process information based on the weights assigned to them. When provided with an input, these nodes then calculate the weight for that input and pass it onto the next layer, hence learning in the process. There exist different types of Neural Networks, including feedforward, recurrent, convolutional and transformer. Transformer Neural Networks use Encoder-Decoder Architecture and attention layers (13). These are particularly effective in processing sequential data as the dataset can be passed as input at once (in parallel) instead of one-by-one. Hence, this is the Neural Network chosen for this project.

Research suggests that Deep Learning classifiers are able to classify text appropriately within their larger context (8). A Bidirectional Transform Classifier uses layers, allowing it to hold short-term memory. This information allows these models to predict and model the subsequent data more efficiently, as is the requirement for depicting data as unpredictable and unstructured as Natural Language Text. Because it can be used to process sequential data such as words and sentences, the Bidirectional Transform Deep Learning classifier is well-suited for this research.

This paper describes a novel method of analyzing raw text from a dataset of Stack Overflow Questions, drawn from the open source platform Kaggle (10). The dataset contains questions from 2016 to 2020. A Deep Learning approach for Text Classification will be used, as the interpretation of the words in a text rely heavily on the preceding words. Initially, extractions of raw text from the dataset will be made using Text Normalization, to prepare the data for Machine Learning-based Text Classification. Using Python, a Neural Network will be trained and fitted using a distilled version of the Bidirectional Encoder Representations from Transformers (DistilBERT) Model with a single additional layer, embedding the raw text from the test data as vectors using the DistilBERT Encoder (11). The model will be tested using accuracy, precision, recall, and F1 score metrics to improve its performance.

This research adds value in allowing for more effective moderation of the posts and questions on Stack Overflow through automating the process, rather than requiring the community to manually review each question posted. This is especially significant when considering that the number of questions posted to Stack Overflow each day is between 8,000 and 10,000, with 29% questions remaining unanswered (1). Due to an increasing number of visits to Stack Overflow, an overall improvement of question quality will increase efficiency on the programmers' end, saving them time. This will facilitate community guideline reinforcement by discouraging vague or duplicate questions, as well as ensure that answers are appropriate and helpful to users, thus providing a significant contribution to the programming community overall.

Literature Review

Significant research has been devoted to the use of different Machine Learning-based Text Classification methods for classifying Stack Overflow questions based on quality, with promising results, some of which are described below.

Toth et al. (4) utilized a deep learning approach for text classification and used a binary classifier to analyze the content of

questions based on the lexical fields used by the author. This classifier was developed using the Keras Library, and contained one GRU layer, five Dense Layers, and one Flatten Layer. However, the temporary memory properties of the Classifier were not used in this case. Three primary metrics were used for evaluation, namely the recall, the accuracy, and the precision. The experiment was performed twice, the second time with a larger dataset, which yielded a more accurate result. The research successfully predicted whether a Stack Overflow question would be closed with 74% accuracy (4). The inspiration for utilizing a Neural Network classifier for the task was taken from this research and its promising results.

Baltadzhieva et al. (5) and Yazdaninia et al. (2) analyzed the different semantic and textual variables that can influence the quality of a Stack Overflow question, including the presence of questions tags, length of title and body, presence of code snippets, terminology used, and the reputation of the author (5-6). This research inspired the Data Processing methodology and semantic analysis chosen in this paper. Further context on each study individually is provided below.

Baltadzhieva et al. (5) estimated Ridge Regression models, and used the Natural Language Toolkit (NLTK) library for evaluating the effect of different Parts of Speech used. Two different sets of multiple regression models were applied during this research for predicting question quality: first, one which predicted the score of the question; and second, one which predicted the number of answers. By splitting the dataset into training, validation, and test, the authors concluded that question title length, body length, and code snippet negatively affected question quality, whereas the reputation of the author affected it positively (5). These results also influenced the decision to consider only the body of Stack Overflow questions for predicting quality in this paper.

Similarly, Yazdaninia et al. evaluated the quality of unresolved questions on Stack Overflow. They analyzed 18 million questions on Stack overflow, 47% of which were currently unresolved. Classifiers were trained independently based on the two sets of features, using XGBoost, and performance was measured using the Area Under the Curve Receiver Operating Characteristic (AUC-ROC). A 10-fold cross validation was used to validate trained models to prevent overfitting. The results indicated that the model trained on the novel features had an AUC of 0.7, as compared to the 0.66 achieved by the model trained with previous features (6). Hence, the inspiration for using Cross Validation for Future Work was taken from this research as well.

Materials and Methods

I. Dataset Description

For this research, we used the 2020 dataset titled, '60k Stack Overflow Questions with Quality Rating' from Kaggle Inc., posted by Moore (10). This dataset provided us with 60,000 questions posted on Stack Overflow over a span of 4 years, from 2016 to 2020. The body of the questions, which is what will be analyzed, is output in HTML format, which is convenient to clean and normalize. The questions included in the dataset were taken directly from Stack Overflow – including closed, deleted, unanswered, and answered questions. The dataset contained two separate csv files, one for test data and another for validation data. The questions in the dataset – organized based on question ID

– contained the following features: title; body; tags; creation date; and type. The types of questions were already labeled and classified into ‘Low Quality Closed’ (LQC), ‘Low Quality Edited’ (LQE), and ‘High Quality’ (HQ), which served as the three categories used for classification. The dataset authors defined each of the categories as follows:

1. HQ: Posts that do not need to go through a single edit
2. LQ-Edit: Low Quality posts that have a negative score, but still remain open after multiple edits
3. LQ-Close: Low Quality posts that are deleted without a single edit

Figure 1 shows some sample entries from the dataset.

	Title	Body	Y	label_enc
16406	how to remove several positions in an array	p i want remove specific positions in an arra...	LQ_CLOSE	1
16231	how do i update element in json string in java	p i want to update value of status and amo...	LQ_CLOSE	1
30281	how too summary values in specyfic range	srv foreach (result as ...	LQ_EDIT	2
55333	google ads solution for pwa spa	p i ve made a single page app via angular an...	HQ	0
17807	how to check if app user is in google play bet...	p i understand that the google play beta feat...	HQ	0
...	...	...	...	...
52034	regex numbers like k	p in regex code d code match all numbers...	LQ_CLOSE	1
44112	how to check if a link is a file or document w...	i am making a web crawler using selenium in py...	LQ_EDIT	2
23827	how does flex wrap work with align self align...	h code align self code h p in the fo...	HQ	0
38389	how to create dynamic array in c which will ...	p i found below solution but still user h...	LQ_CLOSE	1
19816	ssl tls certificates for lightsail	p aws certificate manager (acm) provides ssl ...	HQ	0

Figure 1. Sample Dataset Entries

II. Data Preprocessing

After defining and describing the dataset came Data Preprocessing, to prepare the data before feeding it into the model. Data Preprocessing becomes even more important when dealing with text, converting human natural language into a machine-readable format (9). Due to this project’s focus on the ‘body’ of the Stack Overflow, preprocessing focused only on the text data in the body as well. Beginning with Data Preprocessing, we checked for any null values in the dataset. Null Values are undesirable in a dataset because they are entirely independent of genuine values, which can lead to developmental inaccuracies while training the model. Since imputation is infeasible for raw text data, any null values found need to be removed. No Null Values were found in the test data or in the validation data.

Next, we converted the categorical data types – which were High Quality or ‘HQ’, Low Quality Close or ‘LQ_CLOSE’, and Low Quality Edit or ‘LQ_EDIT’ – into integers. These values were converted into the integers 0, 1, and 2, respectively. These integer labels were encoded onto the dataset using the Scikit Learn library. The next step was cleaning the text data in the body of the Stack

Overflow questions working with Text Normalization (14). The Regular Expression library and the Natural Language Toolkit (NLTK) library in Python were used to achieve this (15). Cleaning the data and removing the non-essential data elements included the following steps:

1. Removing the digits and numerical values from the body.
2. Converting the entire body text into lowercase lettering for uniformity.
3. Removing the URLs from the text, as they hold no semantic significance.
4. Performing Mention Detection and Removal using the Regular Expression library.
5. Normalizing the text to consist of exclusively 'a-z' characters.
6. Removing HTML tags from the body of the text to reduce clutter.
7. Tokenizing the test data, which is the process of splitting the body of the questions into individual words.
8. Identifying 'stop words' in the text using the NLTK library, and removing them because they add no significance to the data.
9. Finally, "rejoining" the words to form sentences, restoring the questions in the dataset

Stemming or Lemmatization of the text was not required for our purpose, as the Neural Network is bidirectional. This preprocessed data would now allow for better accuracy in the training of the classifier, as well as faster computation. Now the data was ready to be analyzed.

III. Exploratory Data Analysis

Data Analysis involved exploring the ratio of question types found within each dataset. Both datasets included 33% of each type of Stack Overflow question, creating an even distribution of question quality types. The question distribution is illustrated in Table 1.

Table 1: Distribution of Questions

Question Type	Number of Questions in Training Data	Number of Questions in Test Data
LQ_EDIT	15 000	5 000
LQ_CLOSE	15 000	5 000
HQ	15 000	5 000

This suggests that our dataset was balanced, which is ideal. A balanced dataset would ensure that the training of the Neural Network is not biased toward any particular question type. Additionally, an annual yearly distribution of the types of questions in the dataset was assessed, as is depicted graphically in Figure 2.

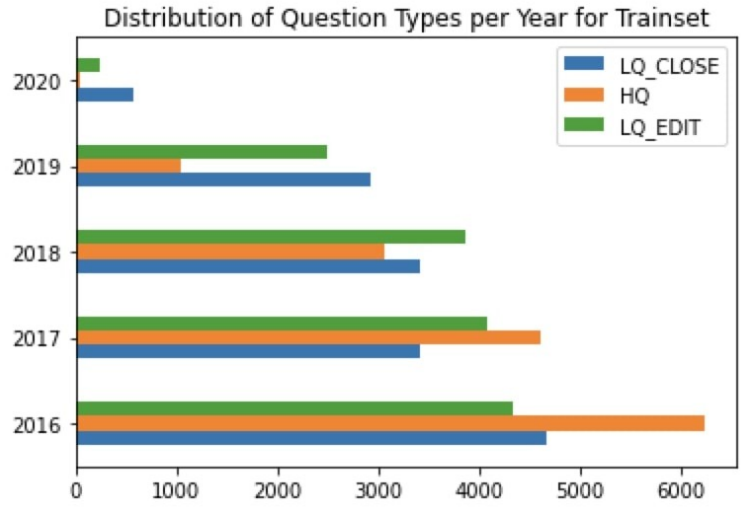

Figure 2a. Annual Quality-based Question Distribution in the Training Set

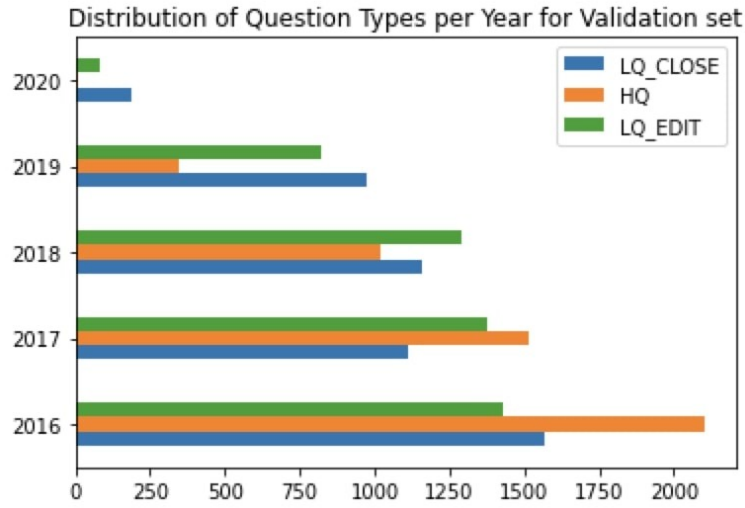

Figure 2b. Annual Quality-based Question Distribution in the Validation Set

This indicates that the dataset contains more older questions than new ones, and that a larger proportion of the older questions were high quality. This implies that the quality of questions has been deteriorating over the years, emphasizing again the importance of this research.

The average length of questions before and after the Data Preprocessing were also explored, to see how this contributed to

quality. The question length before Preprocessing is depicted in the histograms in Figure 3.

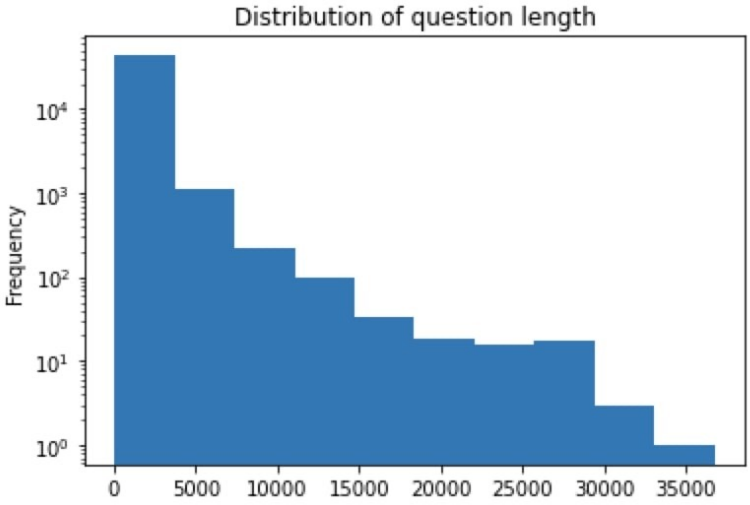

Figure 3a. Training Set question length distribution before Preprocessing

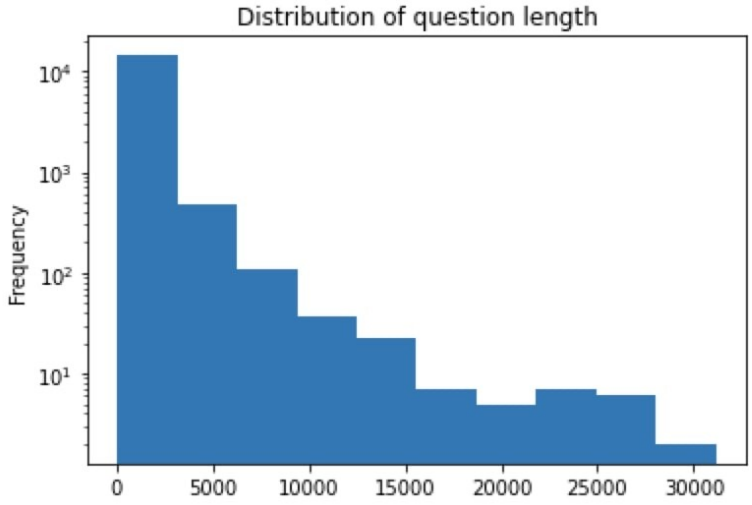

Figure 3b. Validation Set question length distribution before Preprocessing

This was compared with the question length after Preprocessing, as is shown in Figure 4.

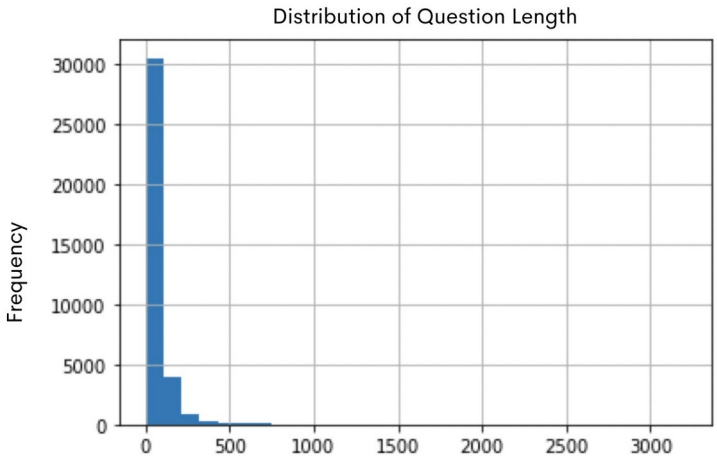

Figure 4a. Training Set question length distribution after Preprocessing

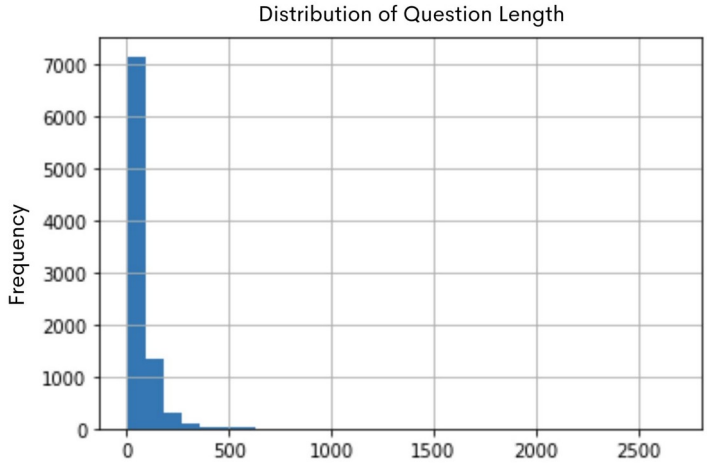

Figure 4b. Validation Set question length distribution after Preprocessing

As can be seen, the average length of the questions decreased significantly after Text Normalization and removing the stop words. This implied that the data was much more suitable for training the model efficiently, since the redundant tokens in the dataset were removed, shifting the focus to only the relevant components of the Stack Overflow question, which may lead to better classifier performance.

Semantic Analysis of the Data

A semantic analysis was performed on the data, by creating 'word clouds' of the most frequent words in the questions of each type in the dataset, with the intent to determine the linguistic features that characterize the quality of a question, and distinguish 'high' from 'low' quality. These word clouds are shown in Figure 5.

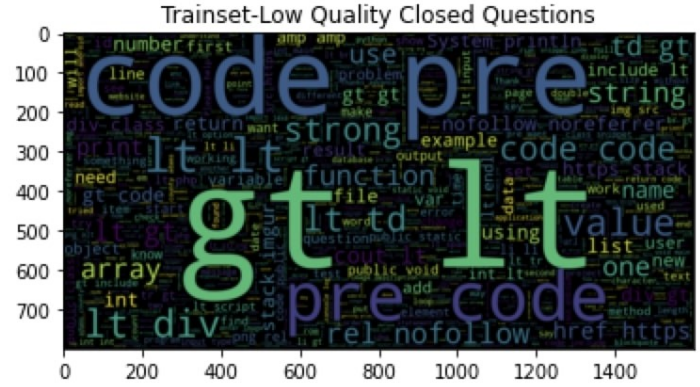

Figure 5a. Word Cloud for ‘LQ-Close’ Questions in the Training Set

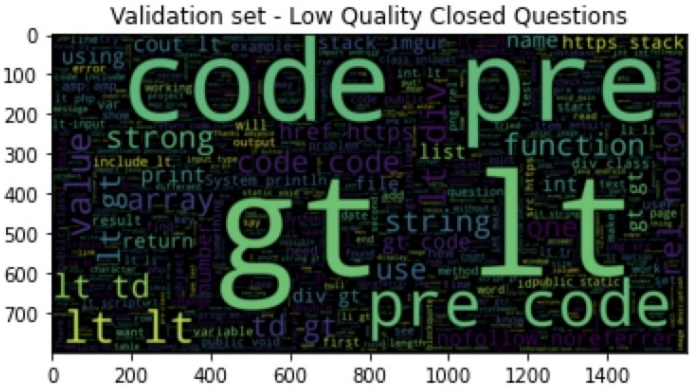

Figure 5b. Word Cloud for ‘LQ-Close’ Questions in the Validation Set

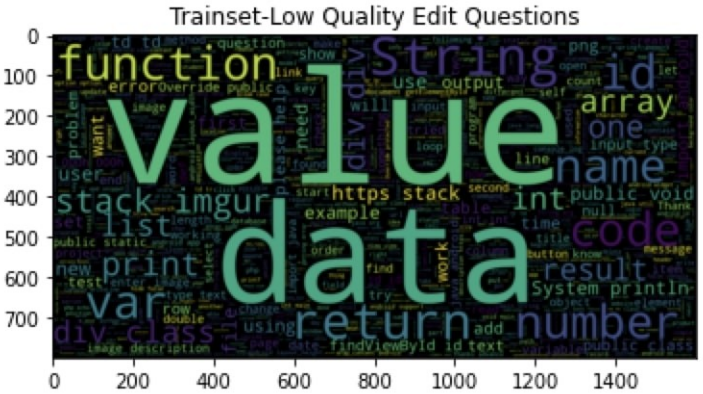

Figure 5c. Word Cloud for ‘LQ-Edit’ Questions in the Training Set

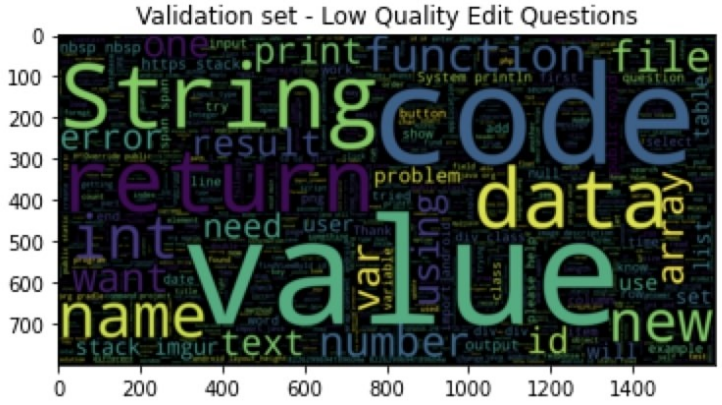

Figure 5d. Word Cloud for ‘LQ-Edit’ Questions in the Validation Set

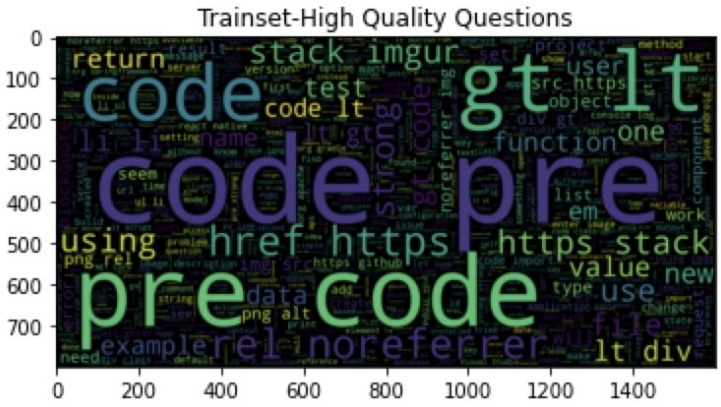

Figure 5e. Word Cloud for ‘HQ’ Questions in the Training Set

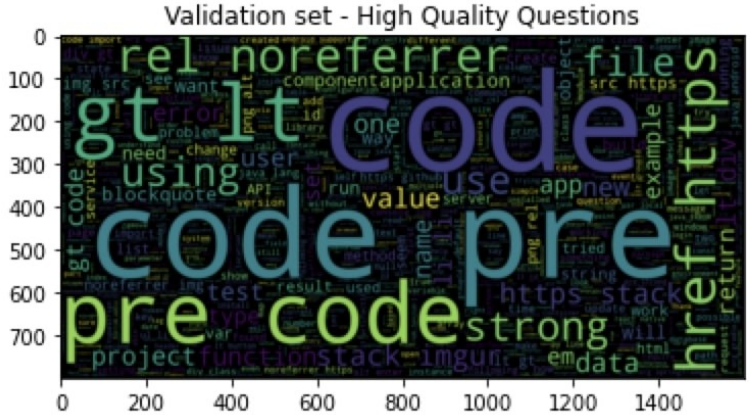

Figure 5f. Word Cloud for ‘HQ’ Questions in the Validation Set

Although the word clouds depict a high degree of commonality between the words used in the different question types in both datasets, words like ‘code’, ‘pre’, as well as specific terminology such as ‘href https’ and ‘rel noreferrer’, are used more often in high quality questions, whereas words like ‘value’, ‘data’, ‘gt’, and ‘lt’ appear with relatively higher frequency in low quality questions. Therefore, it becomes important to look at the composition of the entire question body as a whole during the model development, rather than just the individual word choices of the author when making predictions about quality. With this, the model was ready to be developed and trained.

IV. Model Development

Background Information on BERT

In this work, we use the language representation model called DistilBERT. DistilBERT, is a small, faster, cheap, and light Transformer trained by the distilling BERT base (11). The acronym BERT stands for Bidirectional Encoder Representations from Transformers. It is a state-of-the-art embedding model published by Google in 2018 (16). BERT is open source and it has created a major breakthrough in the field of NLP by providing good results in many NLP tasks such as text generation, sentence classification, and many more. It is based on Transformers, a deep learning model in which every output element is connected to every input element, and the weightings between them are dynamically calculated based upon their connection.

The BERT Base model has 12 layers and over 100 million trainable parameters, which makes it very inefficient to train, hence this research used a distilled version of this BERT transformer (11). Knowledge Distillation is a technique that refers to compressing a ‘teacher’ model into a much more light-weight ‘student’ model. This technique was proposed in 2014 by Hinton and his fellow researchers (17). In this process, the ‘student’ model is trained to mimic the output probability distribution of the teacher. The process is illustrated in Figure 6 (18).

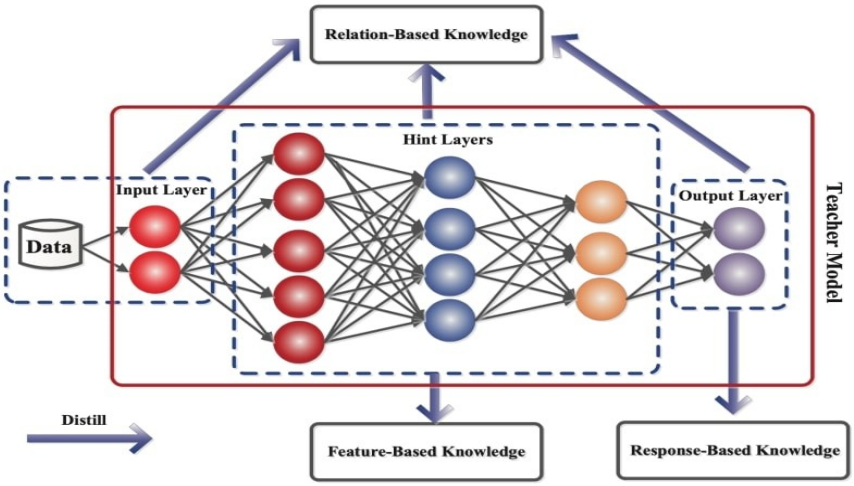

Figure 6. Knowledge Distillation Process. Figure taken from (18)

The novel DistilBERT model is 40% smaller in size than BERT, while retaining 97% of its language processing abilities and being 60% faster (11).

The goal of any given NLP technique is to understand human language as it is spoken naturally. In the case of BERT it typically means predicting a missing word in a blank. Hence, BERT works by leveraging large amounts of training data, using a Masked Language Model (MLM), Next Sentence Prediction (NSP), and transformers. BERT is specifically trained on Wikipedia and Google Book Corpus (16). These large informational datasets contribute to BERT's deep knowledge. BERT, using a Masked Language Model, enforces bidirectional learning from text by hiding a word in a sentence and forcing BERT to bidirectionally use the words on either side of the covered word to predict the masked word (19). BERT uses next sentence prediction, as then BERT can learn about relationships between sentences by predicting if a given sentence follows the previous sentence or not. BERT uses Transformers as it makes it possible to parallelize machine learning training extremely efficiently. The Transformers uses an Encoder-Decoder architecture, however, the Decoder is not required in BERT (20). The BERT architecture is illustrated in Figure 7.

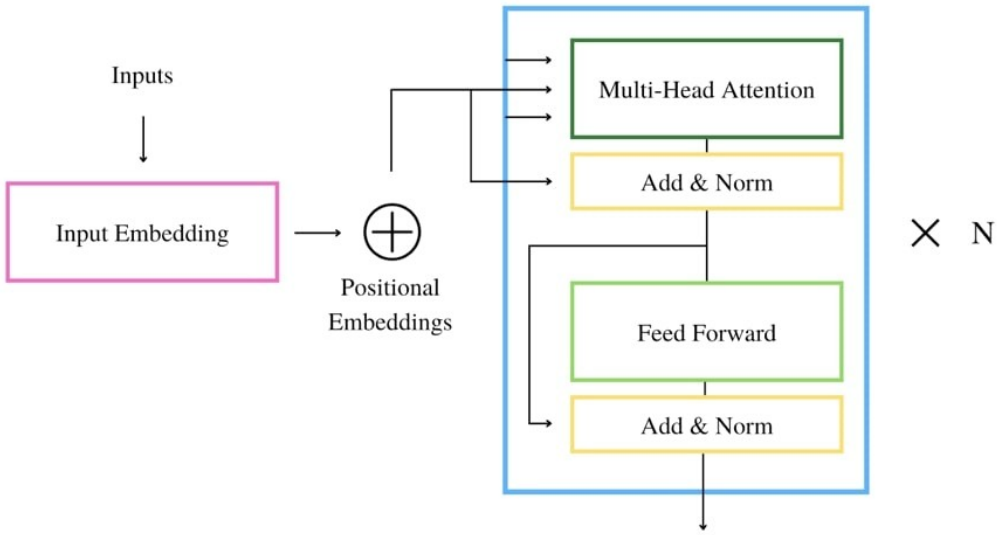

Figure 7. Transformer (Encoder) Architecture of the BERT Model

Transformers work by using 'Attention Mechanisms' for memory. The BERT encoder mechanism receives a series of vector-embedded tokens, with markers to distinguish between sentences, and positional embedding to indicate the position of words in sentences (21). An example is illustrated in Figure 8 (19).

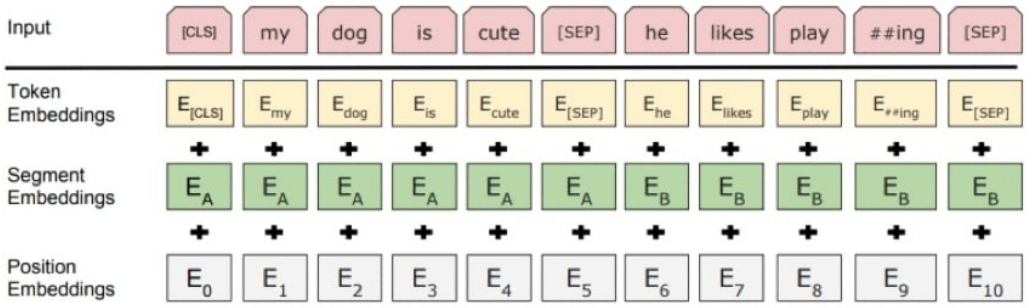

Figure 8. Embedding Inputs in BERT. Figure taken from (19)

The input is then processed in the hidden layers of the model. It is passed into the Multi-Head Attention Mechanism, where weights are added to the inputs by associating the different words with each other. This includes calculating a score matrix depicting how much 'attention' each word needs to put on another word. The output vector from this layer is added with the input embedding again (known as a Residual Connection), followed by Layer Normalization (22). The output is processed on a Feed Forward Neural Network (referring to a unidirectional Neural Network) with multiple layers, using the ReLu activation function. The ReLu activation function in Equation 1.

$$R(x) = max(0, x) \quad [1]$$

This is followed by another series of Layer Normalizations and Residual Connections. The output is hence generated for the MLM and NSP training tasks, allowing BERT to understand the relationship between the elements of a sentence (22).

Developing the Model

This work used a DistilBERT model with one additional layer, a Dense output layer, and 40 epochs. The goal of this work was to determine if neural networks can help with predicting quality of Stack Overflow questions. Future work will use hyperparameter tuning to make the model have more feasible results.

For this research, the general-purpose DistilBERT model can be fine-tuned using the training data, where additional layers can be added to the pre-existing model. The DistilBERT models were taken from the Hugging Face 'transformers' library for python, and additional Keras layers were added (23). The DistilBERT Tokenizer model was used to convert the text data into numeric input tensors for feeding into the encoder. The 'prefetch' function from Tensorflow Data was used to build an input pipeline for the TPUs, where the data for the next iteration of the training is fetched in the current iteration itself, to minimize latency and computational inefficiency (24). The 'Autotune' parameter was used to optimize this pipeline.

An input layer for the Neural Net was initialized, and then the vectorised input was processed in the DistilBERT Encoder. An additional Dense Layer with the softmax activation function was added on top of this in our model for classifying the question quality through a probability distribution. The softmax function returns k values in the bounded interval of (0, 1) for a k-vector input, where each value is calculated as shown in Equation 2 (25).

$$S(x_i) = \frac{e^{x_i}}{\sum_{k=1}^{n} e^{x_k}} \quad [2]$$

The loss function used for the model was the sparse categorical cross-entropy loss function (for compatibility with the integer encoding of the labels), which works by comparing the expected value of a particular datapoint with the predicted probability from the classifier (26). The cross-entropy loss function is particularly useful when weights of the Neural Network need to be adjusted during training. This Loss Function (as used in Keras) is as shown in Equation 3 (26).

$$Loss(x_i) = \sum_{i=1}^{n} t_i \log\left(S(x_i)\right) \quad [3]$$

The Optimiser used was the Adam Optimiser, a stochastic gradient descent method, to change the weights of the Neural Network as needed (27).

We used four particular evaluation metrics to test our model. These were accuracy, precision, recall, and F1 score. These metrics evaluate a model based on the frequency of True and False Positives, as well as True and False Negatives (28).

Accuracy depicts the ratio of the number of correct predictions by the model, as compared to the total predictions. In general, it can be calculated as shown in Equation 4.

$$Accuracy = \frac{True\ Positive + True\ Negative}{TP + TN + FP + FN} \quad [4]$$

Precision refers to the ratio of correct positive predictions as compared to the total positive prediction made by the classifier. Thus, it is calculated as shown in Equation 5.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad [5]$$

Recall is the metric that compares the number of correct positive predictions to the total number of positives in the dataset. It is calculated as shown in Equation 6.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad [6]$$

An F1 score refers to the harmonic mean of the precision and recall metrics. This metric is generally used to deal with the Precision-Recall trade-off in real life, where a high precision often comes at the cost of a lower recall, and vice versa (27). To rectify this, an F1 score can be used to judge the performance of the model instead. It is calculated as shown in Equation 7.

$$F1\,Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad [7]$$

Lastly, a callback was also added to the model, which is the 'Early Stopping' callback from the Tensorflow library. This callback is primarily used to prevent overfitting of the model (29). Overfitting, or co-adaptation, refers to the neurons in a model extracting statistical noise from the train set that reduce accuracy, which could lead to a generalization error when tested using unseen data values. This could happen if the model is trained over too many iterations (epochs), which is not ideal. Hence, the 'Early Stopping' callback can be used, which monitors the validation loss of the model in each epoch, and terminates the training once it stops improving or converges on a particular value.

An illustration of the experiment workflow is given in Figure 9.

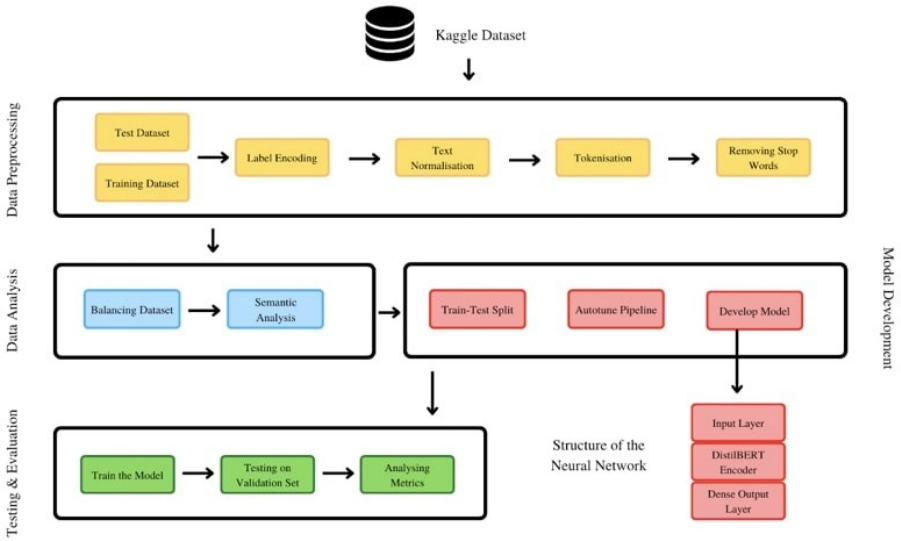

Figure 9. Workflow of the Experiment

Results and Discussion

The final model had one dense layer with 3 neurons, and it used the Adam Optimizer. The training dataset was split into 'training' and 'validation' using the 'train_test_split' stochastic function from the Scikit-Learn library, with validation set size being set to 0.2 (30).

The Google Collab Cloud Tensor Processing Unit (TPU) was utilized for compiling and fitting the model. Tensor Processing Units are custom-designed application-specific integrated circuits (ASICs) developed by Google to accelerate machine learning related tasks with much success (31). The training task was distributed across the TPU Pod using Tensorflow's Strategy API, which offers in-built cluster resolve functions for training the Neural Network on multiple TPUs. The number of devices available was 8. The input data batch size chosen was 32 for each of the 8 TPUs; 256 in total. The number of steps for each epoch was set to 171, by dividing the total values in the training set with the batch size. The patience for the Early Stop callback was set to 5 (which is the minimum number of epochs for without improvement after which the callback is triggered), and the 'restore_best_weights' parameter was set to 'True', in order to retain the best model during the training. The model was set to run for 40 epochs, and the training of the model is shown in Figure 10.

```
Epoch 1/40
INFO:absl:TPU has inputs with dynamic shapes: [<tf.Tensor 'Const:0' shape=() dtype=int32>, <tf.Tensor 'IteratorGetNext:0' shape=(None,
INFO:absl:TPU has inputs with dynamic shapes: [<tf.Tensor 'Const:0' shape=() dtype=int32>, <tf.Tensor 'IteratorGetNext:0' shape=(None,
171/171 [==============================] - ETA: 0s - loss: 0.4410 - accuracy: 0.7908INFO:absl:TPU has inputs with dynamic shapes: [<tf.
171/171 [==============================] - 117s 371ms/step - loss: 0.4410 - accuracy: 0.7908 - val_loss: 0.2309 - val_accuracy: 0.8985
Epoch 2/40
171/171 [==============================] - 51s 297ms/step - loss: 0.2118 - accuracy: 0.9085 - val_loss: 0.2106 - val_accuracy: 0.9115
Epoch 3/40
171/171 [==============================] - 51s 298ms/step - loss: 0.1746 - accuracy: 0.9277 - val_loss: 0.1901 - val_accuracy: 0.9211
Epoch 4/40
171/171 [==============================] - 50s 292ms/step - loss: 0.1512 - accuracy: 0.9388 - val_loss: 0.2060 - val_accuracy: 0.9176
Epoch 5/40
171/171 [==============================] - 50s 292ms/step - loss: 0.1389 - accuracy: 0.9450 - val_loss: 0.1970 - val_accuracy: 0.9230
Epoch 6/40
171/171 [==============================] - 50s 292ms/step - loss: 0.1195 - accuracy: 0.9530 - val_loss: 0.1979 - val_accuracy: 0.9251
Epoch 7/40
171/171 [==============================] - 51s 301ms/step - loss: 0.1059 - accuracy: 0.9583 - val_loss: 0.2000 - val_accuracy: 0.9264
Epoch 8/40
171/171 [==============================] - 53s 312ms/step - loss: 0.0922 - accuracy: 0.9656 - val_loss: 0.2163 - val_accuracy: 0.9256
```

Figure 10. Training the Model

It was observed that the model triggered the Early Stop callback after only 8 epochs, indicating that the model performance had reached a high level rather quickly, with a test loss of only 0.09 during the last epoch. This could be due to the high efficiency of the BERT model in performing text classification. The final obtained accuracy for the test set was 96.6%, and 92.6% for the validation data. These high accuracy scores suggest that model performance is strong and that it can accurately predict the question quality. The loss and accuracy of the model through the epochs was graphed using Plotly and is shown in Figure 11.

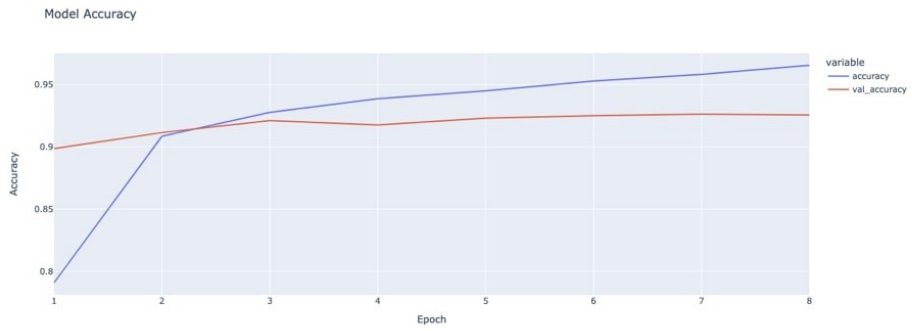

Figure 11a. Model Accuracy for Training and Test Set during Training

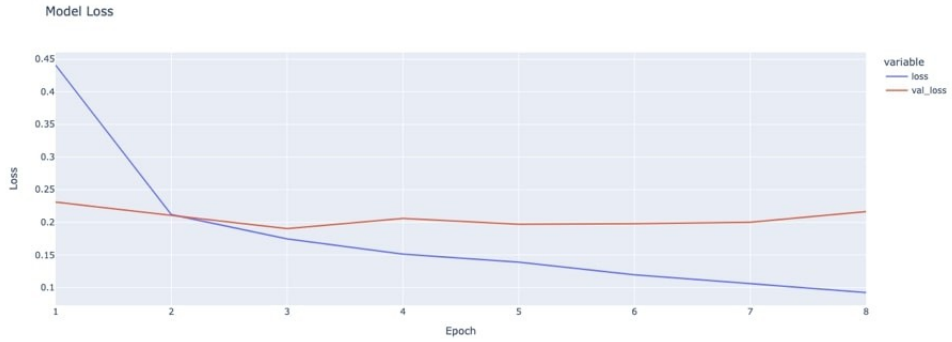

Figure 11b. Model Loss for Training and Test Set during Training

The model was then used to make predictions on the unseen data in the test dataset from Kaggle. Given a particular Stack Overflow question, the model returned the probability of the question falling in each category. We then consider the largest probability (using the Numpy 'argmax' function) to determine the classification for the question.

A confusion matrix was plotted for the results from the predictions for the validation dataset, as shown in Figure 12. The confusion matrix clearly indicates that the model correctly predicted the quality of Stack Overflow questions most of the time. A classification report was generated for the Model Prediction using the Scikit-Learn library and the following was the output (30). This is shown in Table 2.

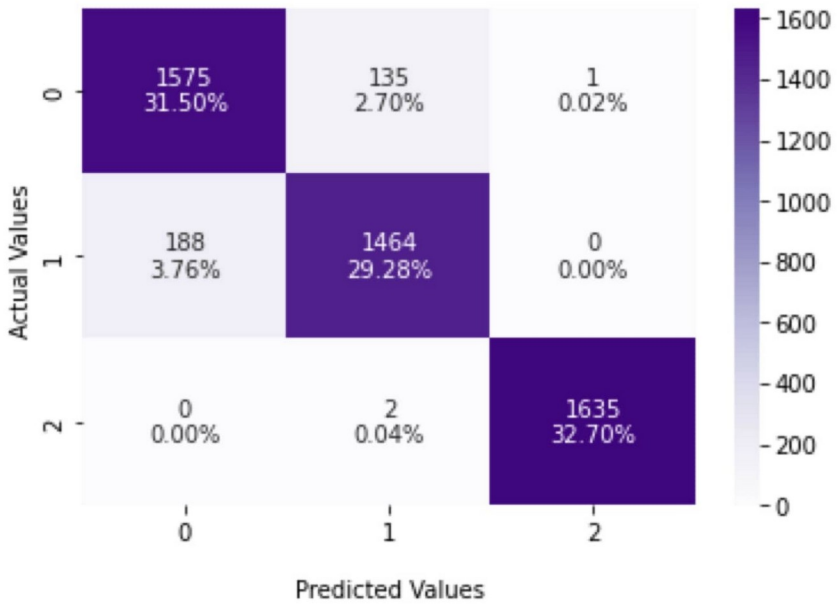

Figure 12. Confusion Matrix

Table 2: Classification Report (Metric Scores)

	Precision	Recall	F1-Score
HQ	0.89	0.92	0.91
LQ-Edit	0.91	0.89	0.90
LQ-Close	1	1	1

Table 2 shows that the model yields a high F1 score value for each classification category, 91%, 90% and 100% respectively. The accuracy score was found to be 93.5%. This implies that the model is highly precise and performs its task well. The similarity in the accuracy during training and testing suggests that the model did not face the problem of overfitting either, hence implying that it can be used practically for making predictions about the quality of Stack Overflow questions accurately.

Overall, the high accuracy and precision obtained during this experiment suggests that the BERT transformer can effectively be used to design a Neural Network for accurately predicting Stack Overflow question quality. It can also be inferred that the described Deep-learning based Natural Language Processing technique is highly effective for performing this task, and can hence be used in helping Stack Overflow authors judge the quality of their questions before posting.

The results obtained in this paper are comparable to those in the literature (4, 32). Table 3 presents the performance of some of the literature discussed before.

Table 3: Comparison of Best Validation Results from the literature

	F1 Score	Accuracy	Recall
Tóth L et al (4)	75%	74%	74%
Chitre Y (32)	93%	85%	97%
DistilBERT	100%	93.5%	100%

As can be seen, this model performs at par with those from previous papers in terms of model accuracy, F1 score, and recall, where it even outperforms the previous models.

Limitations

One potential limitation of this research is that the question quality in the Stack Overflow dataset utilized in this model was predicted and labeled through another machine learning model, as is proposed in a previous paper (10). Although the research resulted in high accuracy levels of 95.6% as a multi-class task, claiming to outperform all previously-published models, machine learning can still be biased. Hence, the labels on the questions in the dataset could potentially face some minor inaccuracies which could adversely impact the results of this research. In addition to this, the dataset also lacks information about the percent of questions in each category that are answered, unanswered and deleted, which restricts evaluation on whether the dataset is balanced on these parameters. In the case that it is not, it could lead to a bias in the trained model and limit the practical utility in aiding Stack Overflow authors.

Future Work

Some potential ideas for future work are proposed to improve upon this research. One addition to this research could be to use the model on new data, using other datasets from multiple sources to validate the model and verify its accuracy. A dataset with additional labels like ‘answered’, ‘unanswered’ and ‘deleted’ questions on Stack Overflow must be sampled, which can offer insight into the parameters in each category which increase the probability of a question being answered, and how those change over time. Subsequently, the model could be tuned to predict whether a question is likely to be answered or remain unanswered, which would permit the Stack Overflow moderation committee to put up better question templates on their website to increase the chances of users obtaining meaningful answers to their questions.

A data crawl can be performed on Stack Overflow, to extract new authentic data and test the model performance further. Moreover, other Q/A platforms beside Stack Overflow can also be considered, for instance other Stack Exchange platforms, and Quora could be potential candidates for future research (33). The model can be used to

assess the quality of questions on these platforms as well, and fitted accordingly.

When considering other Q/A Platforms, Hyperparameter tuning could be used with the Grid Search CV Method to fit the model (34). Hyperparameters refer to the parameters that are decided before the training of a Machine Learning model and affect its learning rate. For Neural Networks like ours, these could include the learning rate, the dropout rate, etc, which could be tuned appropriately for optimizing the model performance. Grid Search explores and evaluates different hyperparameter combinations to select the one that results in the most optimum outcome. The 'GridSearchCV' function from the Scikit-Learn library in Python can be used for this purpose (30). Identifying the best hyperparameter set can hence help in compiling a less biased and more efficient model for predicting the quality of Stack Overflow questions.

For datasets with limited data points like ours, it would also be ideal to consider data resampling procedures such as k-fold Cross Validation to improve the accuracy of the training (35). Cross Validation is a method used popularly to test the accuracy of machine learning models on test data and to reduce bias while training. In K-Fold Cross Validation, the dataset is split into K groups, and the model is trained and tested K times. In each iteration, one of the dataset groups is chosen for testing and the rest are used for training, and the model is evaluated on controlled metrics. The performance of the model is then characterized as its overall average accuracy over each iteration. Using this approach for our research could allow better fine-tuning for parameters, more accurate performance, and less bias in the model. Cross Validation is illustrated in Figure 13.

Figure 13. K-Fold Cross Validation. Figure taken from (35)

Another important addition to this research could be to consider sentiment analysis to assess the correlation between the expression of emotions in a Stack Overflow question, and its ranked quality (36). Inspiration can be taken from past research; the salient features of the question body can be extracted and the classification model can be fine-tuned to assess the correlation between emotions and question quality.

Lastly, another extension to this work can be identifying the features that distinguish "low quality" and "high quality" questions in Stack Overflow, perhaps using Logistic Regression Models and considering to find correlations between features like parts-of-speech, question structure, lexical fields, punctuation, etc, and the perceived quality of the question. This will help the Stack Overflow users to optimize the quality of their posts and subsequently increase the chances of receiving a helpful answer. Inspiration can be taken from the reviewed literature to pursue this in the future (4, 5).

Conclusion

The task of predicting the quality of questions posted on online Question and Answer platforms such as Stack Overflow is of utmost importance, especially considering the increase in people's dependence on it (1). Prediction of question quality before posting can allow authors to improve their questions which will in turn allow for three primary benefits: (a) users can increase the chances of their question getting answered, (b) the moderation process on Stack Overflow can be expedited (c) other programmers can save time and experience better efficiency when seeking technical support due to the reduction in the amount of clutter.

The quality prediction problem is technically a particularly challenging one in the field of Natural Language Processing, since human language can be rather ambiguous and abstract for machines to comprehend and quantify (8). As a potential solution, this paper proposes a Deep Learning-based approach to this classification problem, which uses the DistilBERT model to train a Neural Network which predicts question quality in Stack Overflow.

The experiment results were very promising. The model displayed an accuracy of 96.6% with the training dataset, 92.6% with the unseen test dataset, and an accuracy of 93.5% with the validation set. These results suggest that this proposed model can effectively solve the problem of predicting question quality in Stack Overflow, and hence has the potential to be implemented practically. Pursuing further research and the proposed future work can help explore better solutions to the aforementioned question quality problem and meaningfully expand the boundaries of the field of Natural Language Processing and Text Classification.

References

1. Mondal S. Investigating the Quality Aspects of Crowd-Sourced Developer Forum: A Case Study of Stack Overflow. 2020.
https://clones.usask.ca/pubfiles/articles/SaikatMondalMScThesis2020.pdf .

2. Yazdaninia M, Lo D, Sami A. Characterization and Prediction of Questions without Accepted Answers on Stack Overflow. 29th IEEE International Conference on Program Comprehension, 2021. https://doi.org/10.48550/arXiv.2103.11386 .

3. Code of Conduct. Stack Overflow, 2019. https://stackoverflow.com/conduct .

4. Tóth L, Nagy B, Janthó D, Vidács L, Gyimóthy T. Towards an Accurate Prediction of the Question Quality on Stack Overflow using a Deep-Learning-Based NLP Approach. 14th International Conference on Software Technologies, 2019. https://doi.org/10.5220/0007971306310639 .

5. Baltadzhieva A, Chrupała G. Predicting the quality of questions on Stackoverflow. Proceedings of the International Conference Recent Advances in Natural Language Processing, pages 32–40, 2015. https://research.tilburguniversity.edu/en/publications/predicting-the-quality-of-questions-on-stackoverflow .

6. Lin JW, Lin TC, Schaedler P. Predicting the Best Answers for Questions on Stack Overflow. 2016. https://www.semanticscholar.org/paper/Predicting-the-Best-Answers-for-Questions-on-Stack-Lin/9aec97f5381769c8cdbf5fde75ca221c8ad1113e .

7. Zhao LX, Zhang L, Jiang J. Hot question prediction in Stack Overflow. IET Software, 15(1), 90–106, 2021. https://doi.org/10.1049/sfw2.12013 .

8. Torfi A, Shirvani R, & Keneshloo Y, Tavvaf N, Fox E. Natural Language Processing Advancements By Deep Learning: A Survey. 2020. https://arxiv.org/pdf/2003.01200.pdf .

9. Kaur J, Saini J. A Study of Text Classification Natural Language Processing Algorithms for Indian Languages. VNSGU Journal of Science and Technology. 4. 162-167, 2015. https://www.vnsgu.ac.in/oldwebsite/dept/publication/vnsgujst41july2015/20.pdf .

10. Annamoradnejad I, Habibi J, Fazli M. Multi-view approach to suggest moderation actions in community question answering sites. Information Sciences, 600, 144-154, 2022. http://dx.doi.org/10.1016/j.ins.2022.03.085 .

11. Sanh V, Debut L, Chaumond J, Wolf T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. 2019. https://doi.org/10.48550/arXiv.1910.01108 .

12. Hardesty L. Explained: Neural networks. MIT News | Massachusetts Institute of Technology, 2017. https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414.

13. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I. Attention Is All You Need. 2017. https://doi.org/10.48550/arXiv.1706.03762 .

14. Reddy KM. Model Selection Using Cross Validation and GridSearchCV. Medium, 2018. https://medium.com/@kesarimohan87/model-selection-using-cross-validation-and-gridsearchcv-8756aac1e9d7 .

15. Bird S, Klein E, Loper E. Natural Language Processing with Python. O'Reilly Media Inc, 2009. https://www.nltk.org/ .

16. Devlin J, Chang MW, Kenton L, Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2019. https://doi.org/10.48550/arXiv.1810.04805 .

17. Hinton G, Vinyals O, Dean J. Distilling the Knowledge in a Neural Network. 2015. https://doi.org/10.48550/arXiv.1503.02531 .

18. Gou J, Yu B, Maybank SJ, Tao D. Knowledge Distillation: A Survey. International Journal of Computer Vision, 129(6), 1789–1819, 2021. https://doi.org/10.1007/s11263-021-01453-z .

19. Khalid S. BERT Explained: A Complete Guide with Theory and Tutorial. Medium, 2021. https://medium.com/@samia.khalid/bert-explained-a-complete-guide-with-theory-and-tutorial-3ac9ebc8fa7c .

20. Castellucci G, Bellomaria V, Favalli A, Romagnoli R. Multi-lingual Intent Detection and Slot Filling in a Joint BERT-based Model. 2019. https://doi.org/10.48550/arXiv.1907.02884 .

21. Phi M. Illustrated Guide to Transformers- Step by Step Explanation. Medium, 2021. https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc02 .

22. Patchigolla D. Understanding BERT architecture - Analytics Vidhya. Medium, 2021. https://medium.com/analytics-vidhya/understanding-bert-architecture-3f35a264b187.

23. Hugging Face. DistilBERT. 2020. https://huggingface.co/docs/transformers/model_doc/distilbert .

24. Module: tf.keras | TensorFlow Core v2.9.1. TensorFlow, 2022. https://www.tensorflow.org/api_docs/python/tf/keras .

25. Nwankpa CE, Ijomah W, Gachagan A, Marshall S. Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. 2018. https://arxiv.org/pdf/1811.03378.pdf .

26. Koech KE. Cross-Entropy Loss Function - Towards Data Science. Medium, 2022. https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e .

27. Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. 2017. https://doi.org/10.48550/arXiv.1412.6980 .

28. Shung KP. Accuracy, Precision, Recall or F1? - Towards Data Science. Medium, 2020. https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9 .

29. Vijay U. Early Stopping to avoid overfitting in neural network- Keras. Medium, 2021. https://medium.com/zero-equals-false/early-stopping-to-avoid-overfitting-in-neural-network-keras-b68c96ed05d9 .

30. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: Machine Learning in Python. JMLR, 12: 2825-2830, 2011. https://scikit-learn.org/stable/ .

31. Jouppi N, Borchers A, Boyle R, Cantin P, Chao C, Clark C, Coriell J, Daley M, Dau M, Dean J, Gelb B, Young C, Ghaemmaghami T, Gottipati R, Gulland W, Hagmann R, Ho C, Hogberg D, Hu J, Boden N. In-Datacenter Performance Analysis of a Tensor Processing Unit, 2017. https://doi.org/10.1145/3079856.3080246 .

32. Chitre Y. Stack Overflow Quality Prediction. Github. 2020. https://github.com/yashchitre03/Stack-Overflow-Quality-Prediction .

33. Ansari N, Sharma R. Identifying Semantically Duplicate Questions Using Data Science Approach: A Quora Case Study. ACM Conference. ACM, New York, NY, USA., 2019. https://doi.org/0000001.0000001 .

34. Liao L, Li H, Shang W, Ma L. An Empirical Study of the Impact of Hyperparameter Tuning and Model Optimization on the Performance Properties of Deep Neural Networks. ACM Transactions on Software Engineering and Methodology. 31, 3, Article 5, 2022. https://doi.org/10.1145/3506695 .

35. Cross-Validation. MATLAB & Simulink, 2022. https://www.mathworks.com/discovery/cross-validation.html .

36. Mondal A, Rahman MM, Roy C. Embedded Emotion-based Classification of Stack Overflow Questions Towards the Question Quality Prediction. 2016. https://clones.usask.ca/pubfiles/articles/RahmanSEKE2016Sentiment.pdf .