Classifying Question Quality of StackOverflow.

Francisco Valdez de la Fuente
paco.valdez@berkeley.edu

MIDS W266 Fall 2022

Abstract

StackOverflow (SO) is a popular online platform for developers to ask and answer programming-related questions. This paper proposes a natural language processing (NLP) approach based on popular transformer NLP models to classify the quality of SO questions. We use a dataset of labeled questions based on score and popularity and apply different BERT-based models and a text-to-text transformer model for a multiclass classification task. Our experiments show that the best approach using T5, a text-to-text transformer, achieves a global accuracy of 39% and a maximum precision of 52% for one of the classes.

Introduction

There are multiple online forums where coding questions can be asked. The most interesting problem is to be able to prioritize “good” questions and do the opposite for “bad” questions. The applications to this solution would help these forums provide users feedback while asking a question. Also, “good” questions could be moved to the workflow's next step, an automated classification system, or a manual review. On the other hand, every software developer in the world has dealt with Stack Overflow (SO); the amount of shared knowledge there is incomparable to any other website. Questions in SO are usually annotated and curated by thousands of people, providing metadata about the quality of the question. While recent efforts are aimed at predicting the likelihood of a question being closed to give instant feedback to Stack Overflow users[5], this project aims to provide a broader resolution on the prediction and predict the popularity and score of a question.

Previous research has shown that models based on BERT are good at text classification tasks. Liu et al. (2019)[6] and Howard et al. (2018)[3] have outperformed state-of-the-art classification tasks using fine-tuned BERT. Soumyadeep et al.(2021) have found that even adding a few labeled examples improves the accuracy of text classification [4].

Methodology

Dataset

Stack Overflow publishes all the data about the questions and answers in BigQuery at the dataset `bigquery-public-data.stackoverflow` . Stack Overflow grants “badges”[1] to users that write good questions based on these parameters:

Score-Based	View-Based	Bookmark-Based	Numeric Label	String Label
Great Question (Score >= 100)	Famous Question (Views>=10,000)	Stellar Question (Bookmarks>=100)	0	great
Good Question (Score >= 25)	Notable Question (Views>=2,500)	Favorite Question (Bookmarks>=25)	1	good
Nice Question (Score >= 10)	Popular Question (Views>=1,000)	*Missing Category (Bookmarks>=10)	2	nice
No Badge	No Badge	No Badge	3	bad

Table 1 Definition of label values.

This scoring system provides a 4-value scale intended to provide a higher resolution to the quality of a question. The questions are labeled with a quality value if they fulfill any of each column's criteria. The reasoning behind this labeling is that good questions are usually easier to respond to, have laid out many details about the problem, and are problems that more users are experiencing. There is an inherent class imbalance already noted by Piyush Arora et al. (2016)[2]; a lot more "bad" questions are created than "good" questions (see Figure 1).
The full dataset consists of 23M questions from 2008 to 2022. For the training, classes are balanced, having an equal number of records for each label value by using a random sample for each label value. After balancing the classes, the dataset contains approximately 4M questions, with 1M per value label.[8]

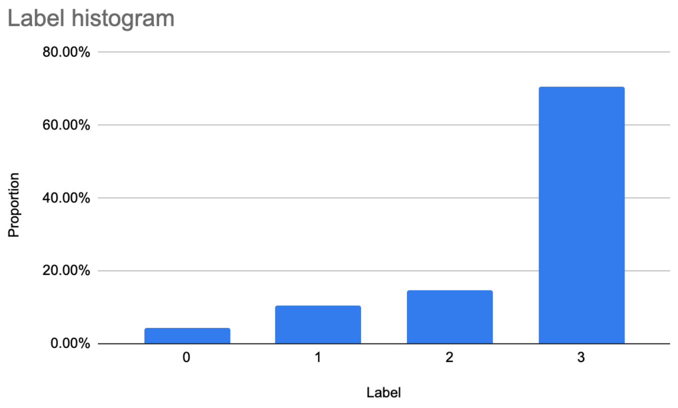


Figure 1 Distribution of the label categories.

Another phenomenon in the dataset is that the old questions tend to have accumulated better scores than newer ones; this may be due to how StackOverflow works. When a new question is asked, it will be closed if a similar question already exists, even if the new question is worded in a way that is more comprehensible or if it has more details or code examples. We conducted a few experiments using data from all times but then focused on using an arbitrary time frame to try to control for the inherent temporality in the dataset. As new technologies arise, people start to ask questions about them, so there is still plenty of room for new questions, and many old questions lose relevance. It still takes some time for questions to reach their full potential. With this reasoning, we picked 2016 data as the subset of the data to further analyze different NLP models.[9]

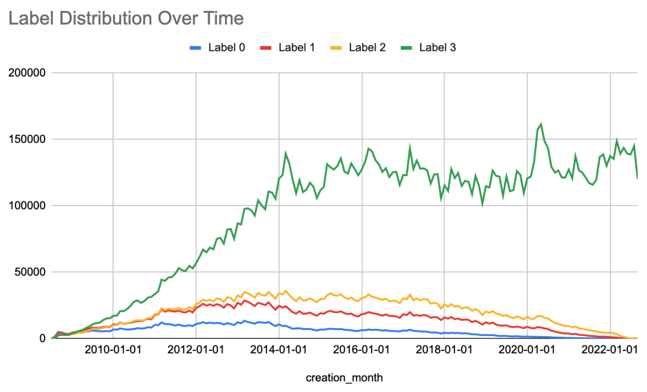


Figure 2 Distribution of Labels from 2008 to 2022

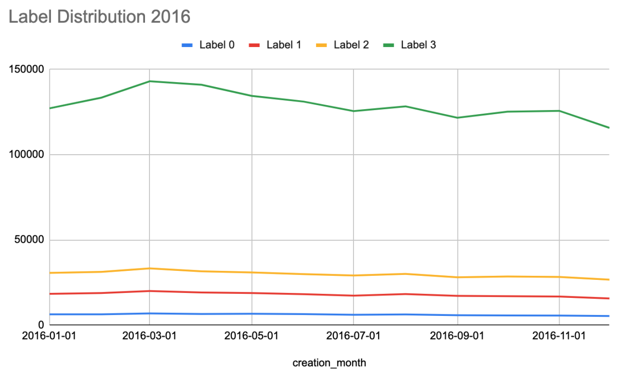


Figure 3 Distribution of Labels for 2016

Model

The baseline model is a fine-tuned Uncased BERT model[7] using StackOverflow questions from 2008 to 2022. BERT is trained in the BooksCorpus and English Wikipedia, and SO questions are written in more colloquial but very technical English. We expect that BERT will pick up the unwritten conventions of how questions are made in SO. The models tested were Uncased-BERT[7], BERTweet[10], CodeBERT[11] and T5[12]. All the BERT-based models were trained using the same architecture, and the only outlier is the model based on T5, where we are just using a fined tuned t5 text-to-text transformer. The BERT-based trained models consist of the input layer followed by a BERT layer, then an Activation Layer using softmax as the activation function, and finally, an Adam optimizer with a sparse categorical cross-entropy loss function.

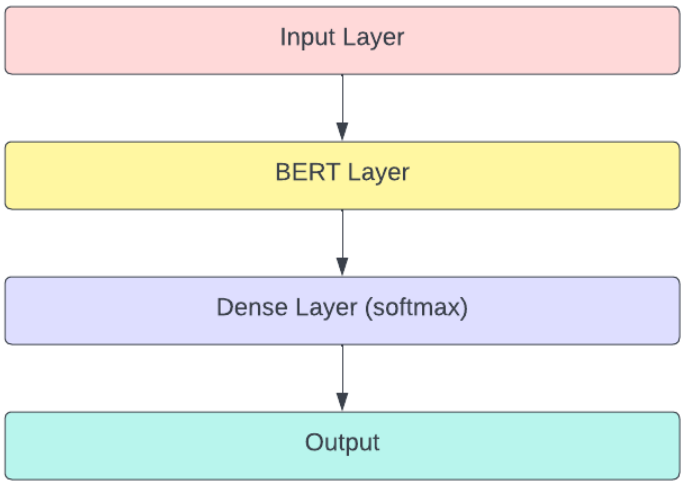
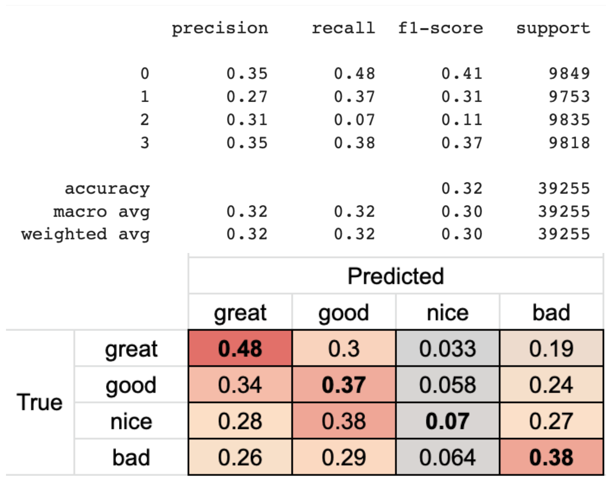


Figure 4 Architecture diagram for BERT-based models.

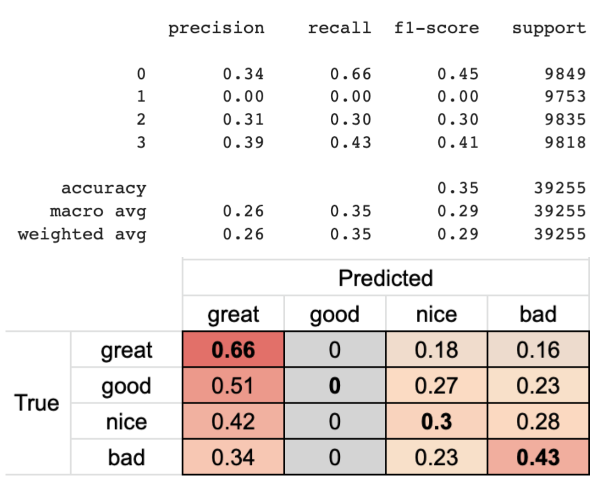
Results

The initial experiments we did use all history of SO questions to capture the constant change of the writing style and the introduction of new technologies. Given the size, time, and resources required to train the model with the full dataset, it was required to take a 20% sample of the dataset and validate its accuracy. The first iteration of the experiments was to compare the baseline BERT with another model trained with a different corpus. BERTweet[10] is pre-trained with an 850M English tweets corpus. We expected BERTweet to perform better, given that SO rewards concise questions and that questions are written in a more colloquial language than Wikipedia or other books used to pre-train the baseline BERT. The baseline BERT achieved 32% global accuracy and a maximum of 35% precision on the *great* and *bad* classes (see table 1 for class definition). BERTweet achieved 35% global accuracy and a maximum of 39% precision in the *bad* class.



```
              precision    recall  f1-score   support

           0       0.35      0.48      0.41      9849
           1       0.27      0.37      0.31      9753
           2       0.31      0.07      0.11      9835
           3       0.35      0.38      0.37      9818

    accuracy                           0.32     39255
   macro avg       0.32      0.32      0.30     39255
weighted avg       0.32      0.32      0.30     39255
```

		Predicted			
		great	good	nice	bad
True	great	**0.48**	0.3	0.033	0.19
	good	0.34	**0.37**	0.058	0.24
	nice	0.28	0.38	**0.07**	0.27
	bad	0.26	0.29	0.064	**0.38**



```
              precision    recall  f1-score   support

           0       0.34      0.66      0.45      9849
           1       0.00      0.00      0.00      9753
           2       0.31      0.30      0.30      9835
           3       0.39      0.43      0.41      9818

    accuracy                           0.35     39255
   macro avg       0.26      0.35      0.29     39255
weighted avg       0.26      0.35      0.29     39255
```

		Predicted			
		great	good	nice	bad
True	great	**0.66**	0	0.18	0.16
	good	0.51	**0**	0.27	0.23
	nice	0.42	0	**0.3**	0.28
	bad	0.34	0	0.23	**0.43**

Table 2 Accuracy Results and Confusion Matrix from baseline BERT Full History (Left) and BERTweet Full History (Right).

Then in the next iteration, we added a Special Token to replace code blocks from SO questions. SO questions are formatted using HTML, and users usually add code snippets or error logs to add more details to the question(See Table 3 for an example of the text formatting and processing). The BERT models being tested are not trained on code. Therefore, it is vital to translate the code block into a Special Token that indicates a code snippet or an error log.

Original Text	**Post Processed Text with Special Token**
`Vuex getter not updating <p>I have the below getter:</p> <pre class="lang-js prettyprint-override"> <code>withEarmarks: state =&gt; { var count = 0; for (let l of state.laptops) if (l.earmarks.length &gt; 0) count++; return count; } </code></pre> <p>And in a component, this computed property derived from that getter:</p> <pre class="lang-js prettyprint-override"> <code> withEarmarks() { return this.$store.getters.withEarmarks; }, </code></pre> <p>The value returned is correct, until I change an element within the laptops array, and then the getter doesn't update.</p>`	`vuex getter not updating i have the below getter` **`[CODE]`** `and in a component this computed property derived from that getter` **`[CODE]`** `the value returned is correct until i change an element within the laptops array and then the getter doesnt update`

Table 3 Example of the original text and post-processed text with Special Tokens (highlighted).

BERTweet without Special Token improved considerably more than the baseline, and we expected that the baseline would improve with Special Token, but it did not. BERT with Special Token achieved a global accuracy of 31% (vs. 32% baseline), although it surpassed the baseline on maximum precision of any class with 39% at the *bad* class (vs. 35% baseline). BERTweet with Special Token had the same global precision as without Special Token (35%) but improved in the maximum precision with 42% (vs. 39% w/o Special Token).

```
              precision    recall  f1-score   support

           0       0.34      0.53      0.41      9849
           1       0.26      0.44      0.32      9753
           2       0.31      0.03      0.06      9835
           3       0.39      0.25      0.30      9818

    accuracy                           0.31     39255
   macro avg       0.32      0.31      0.27     39255
weighted avg       0.32      0.31      0.27     39255
```

```
              precision    recall  f1-score   support

           0       0.35      0.61      0.45      9849
           1       0.30      0.13      0.18      9753
           2       0.31      0.30      0.30      9835
           3       0.42      0.37      0.39      9818

    accuracy                           0.35     39255
   macro avg       0.34      0.35      0.33     39255
weighted avg       0.34      0.35      0.33     39255
```

		Predicted			
		great	good	nice	bad
True	great	**0.53**	0.35	0.015	0.11
	good	0.4	**0.44**	0.029	0.13
	nice	0.34	0.48	**0.034**	0.15
	bad	0.3	0.42	0.032	**0.25**

		Predicted			
		great	good	nice	bad
True	great	**0.61**	0.1	0.16	0.13
	good	0.45	**0.13**	0.24	0.17
	nice	0.37	0.12	**0.3**	0.21
	bad	0.29	0.083	0.26	**0.37**

Table 4 Accuracy Results and Confusion Matrix from BERT Full History with Special Token (Left) and BERTweet Full History with Special Token (Right).

In the next iteration, we used only 2016 data to try to compensate for the temporality of the data. The baseline BERT with 2016 data and Special Tokens improved the global accuracy to 33% and a maximum precision of 37%. BERTweet with 2016 data and Special Tokens achieved 37% global accuracy and a maximum accuracy of 47%.

```
              precision    recall  f1-score   support

           0       0.34      0.66      0.44     14988
           1       0.35      0.00      0.01     14850
           2       0.29      0.26      0.28     14808
           3       0.37      0.41      0.39     14684

    accuracy                           0.33     59330
   macro avg       0.34      0.33      0.28     59330
weighted avg       0.34      0.33      0.28     59330
```

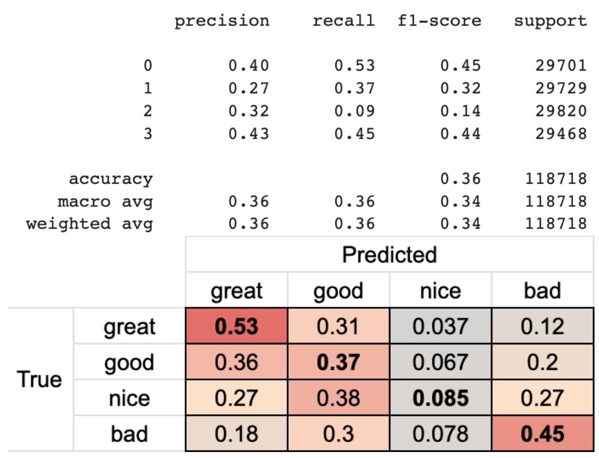
```
              precision    recall  f1-score   support

           0       0.47      0.37      0.41     14988
           1       0.29      0.28      0.28     14850
           2       0.31      0.30      0.30     14808
           3       0.41      0.53      0.46     14684

    accuracy                           0.37     59330
   macro avg       0.37      0.37      0.36     59330
weighted avg       0.37      0.37      0.36     59330
```

		Predicted			
		great	good	nice	bad
True	great	**0.66**	0.0021	0.17	0.17
	good	0.52	**0.0042**	0.24	0.24
	nice	0.44	0.0036	**0.26**	0.29
	bad	0.35	0.002	0.23	**0.41**

		Predicted			
		great	good	nice	bad
True	great	**0.37**	0.28	0.17	0.18
	good	0.21	**0.28**	0.26	0.26
	nice	0.14	0.24	**0.3**	0.33
	bad	0.077	0.16	0.23	**0.53**

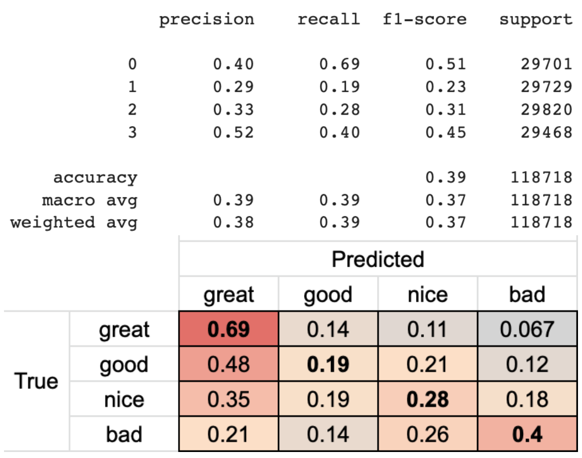
Table 5 Accuracy Results and Confusion Matrix from BERT with 2016 data and Special Token (Left) and BERTweet with 2016 data and Special Token (Right).

For the final experiment, we tried to use different approaches, first try with a BERT-based model pre-trained with programming source code. We selected CodeBERT[11] base since it is trained with multiple programming languages, as with documentation and notes inside the source code. CodeBERT with 2016 data and almost no text pre-processing, achieved a 36% global accuracy and a maximum precision of 43%. The last experiment was based on T5[12], which required using a text label to use the text-to-text transformation capabilities. T5 achieved a global accuracy of 39% and a maximum precision of 52% at the *bad* class. This result made T5 the best performer of all the models and training data we tested.



```
              precision    recall  f1-score   support

           0       0.40      0.53      0.45     29701
           1       0.27      0.37      0.32     29729
           2       0.32      0.09      0.14     29820
           3       0.43      0.45      0.44     29468

    accuracy                           0.36    118718
   macro avg       0.36      0.36      0.34    118718
weighted avg       0.36      0.36      0.34    118718
```

		Predicted			
		great	good	nice	bad
True	great	**0.53**	0.31	0.037	0.12
	good	0.36	**0.37**	0.067	0.2
	nice	0.27	0.38	**0.085**	0.27
	bad	0.18	0.3	0.078	**0.45**



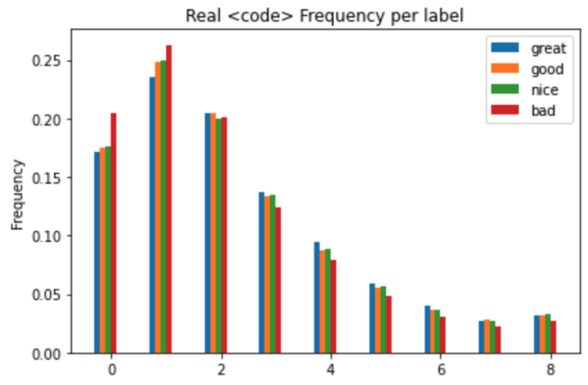
```
              precision    recall  f1-score   support

           0       0.40      0.69      0.51     29701
           1       0.29      0.19      0.23     29729
           2       0.33      0.28      0.31     29820
           3       0.52      0.40      0.45     29468

    accuracy                           0.39    118718
   macro avg       0.39      0.39      0.37    118718
weighted avg       0.38      0.39      0.37    118718
```

		Predicted			
		great	good	nice	bad
True	great	**0.69**	0.14	0.11	0.067
	good	0.48	**0.19**	0.21	0.12
	nice	0.35	0.19	**0.28**	0.18
	bad	0.21	0.14	0.26	**0.4**

Table 6 Accuracy Results and Confusion Matrix from CodeBERT with 2016 data (Left) and T5 with 2016 data (Right).

Conclusion

StackOverflow question data is an excellent example of data that can be used for NLP. There are multiple possible tasks, and predicting quality is an interesting problem that is difficult to separate from predicting future performance, a known limitation of Machine Learning. BERT-based models struggled with understanding the code blocks, and even replacing the code blocks with Special Tokens did not improve the accuracy drastically. The existence of code blocks is not itself a predictor of performance (see figure 7). Possible future improvements could be to build a more complex model on top of a BERT-based model, and in the case of using CodeBERT only use a subset of the questions that are asked in the programming languages used to pre-train it. Temporality is still a problem that could be addressed by adjusting the score based on when the question earned its votes or views. It is not a big surprise that T5 performed better based on its performance with traditional benchmarks.



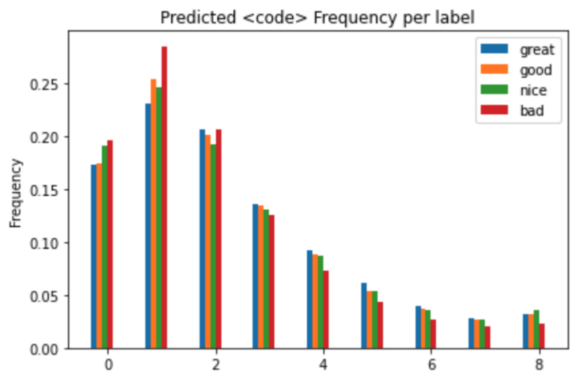


Figure 7 Frequency of <code> blocks in Actual data (Left) and Predicted data (T5) (Right)

References

1. Stack Overflow - Badges https://stackoverflow.com/help/badges Retrieved September 2022
2. Arora, P., Ganguly, D., & Jones, G., (2016). Nearest Neighbour based Transformation Functions for Text Classification: A Case Study with StackOverflow. In Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval (ICTIR '16). Association for Computing Machinery, New York, NY, USA, 299–302. https://doi.org/10.1145/2970398.2970426
3. Howard, J., & Ruder, S., 2018. Universal Language Model Finetuning for Text Classification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 328–339. https://doi.org/10.18653/v1/P18-1031
4. Roy, S., Chakraborty, S., Manda, A., Balde, G., Sharma, P., Natarajan, A., Khosla, M., Sural, S., & Ganguly, N., (2021) Knowledge-Aware Neural Networks for Medical Forum Question Classification. In Proceedings of the 30th ACM International Conference on Information &amp; Knowledge Management (CIKM '21). Association for Computing Machinery, New York, NY, USA, 3398–3402. https://doi.org/10.1145/3459637.3482128
5. Tóth, L., Nagy, B., Gyimóthy, T., & Vidács, L., (2020). Why will my question be closed? NLP-based pre-submission predictions of question closing reasons on stack overflow. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER '20). Association for Computing Machinery, New York, NY, USA, 45–48. https://doi.org/10.1145/3377816.3381733
6. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). ROBERTA: A robustly optimized BERT pretraining approach. arXiv:1907.11692. https://arxiv.org/pdf/1907.11692.pdf
7. Devlin, J., Chang, M., Lee, K., & Toutanova, K., (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics. https://arxiv.org/pdf/1810.04805.pdf
8. Francisco Valdez, Stackoverflow Dataset. Retrieved from https://huggingface.co/datasets/pacovaldez/stackoverflow-questions
9. Francisco Valdez, Stackoverflow Dataset from 2016. Retrieved from https://huggingface.co/datasets/pacovaldez/stackoverflow-questions-2016
10. Nguyen, D., Vu, T., & Nguyen, A., (2020). BERTweet: A pre-trained language model for English Tweets. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 9–14, Online. Association for Computational Linguistics. https://aclanthology.org/2020.emnlp-demos.2
11. Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., & Zhou, M. (2020). CodeBERT: A Pre-Trained Model for Programming and Natural Languages. arXiv. https://doi.org/10.48550/arXiv.2002.08155
12. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res., 21(140), 1-67. http://jmlr.org/papers/v21/20-074.html