Evaluating Online Question Quality for Technical Subject Domains

Soravit Sophastienphong
UC Berkeley Masters in Data Science

April 10, 2021

1. Abstract

In today's world, the abundance of online resources for technical domains such as computer science and machine learning has made solutions to increasingly complex problems more accessible. However, as a consequence, it can be difficult to find the most insightful among many resources that are related to a particular problem. In this paper, we attempt to accurately classify the quality of questions posted on Stack Overflow, an online question-answering platform focused on computer programming. We first train a baseline Logistic Regression model with TF-IDF word embeddings, upon which we improve our results by building CNN and RoBERTa models. The strong performance of these models indicates potential value in applications such as improving search results or validating online posts, although a larger volume of data would help not only improve our model but also substantiate our findings.

2. Introduction

When encountering a technical challenge, a programmer is likely to seek answers on Stack Overflow under the assumption that someone else is likely to have encountered the same problem before. If they cannot find a relevant post, then they can post the question themselves in hopes that others will answer it. Among the plethora of questions that have been posted on Stack Overflow, a substantial proportion are left unanswered or do not have an answer that is accepted by the inquirer. As of April 3rd, 2021, 6,389,349 out of 21,065,561 questions (30.33%) fall into this category.

There are a variety of reasons why a question might not have an accepted answer, including but not limited to providing insufficient context or detail, being poorly-written, having spelling/grammatical errors, or being uninteresting to other users. Stack Overflow questions are sorted in descending order by number of upvotes, so higher quality questions are more likely to be answered. An automated way of determining question quality could help validate users' questions and provide suggestions before they are asked, thereby increasing the average quality of questions and the proportion of questions with accepted answers.

For this reason, we explore and discuss three text classification models developed for the purpose of accurately predicting question quality. Our CNN and RoBERTa models achieve F1 scores of 0.81 and 0.90 respectively on the test set, not taking into account potential optimizations such as further pretraining and improved data transformation.

3. Background

Text classification is one of the most commonly performed NLP tasks, the typical example being email spam classification. In fact, state-of-the-art language models such as BERT, XLNet, and GPT have achieved remarkable results predicting on well-known datasets such as AG News, DBpedia, and IMDb. As Minaee et al. (2020) suggest, the usage of pretrained language models has led to significant strides across all popular text classification tasks. As for neural networks, Minaee et al. (2020) describe that "RNNs are trained to recognize patterns across time, whereas CNNs learn to recognize patterns across space." Hence, tasks that require capturing long-range dependencies may be more suited for RNNs, whereas CNNs are better at extracting key words and phrases.

In terms of the problem domain, Baltadzhieva et al. (2015) identify various features that significantly contribute to Stack Overflow question quality, measured by number of answers and number of upvotes. For instance, generic (non-stopword) terms have a more positive impact on question quality than technical or domain-specific terms. In their work, other features that significantly impact question quality include grammatical correctness, newness of user, length of question title, length of question body, and the quality of being help-seeking.

4. Methods

4.1 Data

Our models use the Kaggle dataset called "60k Stack Overflow Questions with Quality Rating", which contains 60,000 Stack Overflow questions from 2016-2020 (Moore, 2020). Each question has one of three labels as defined in the description on Kaggle:

1. HQ: High-quality posts with a total of 30+ score and without a single edit.
2. LQ_EDIT: Low-quality posts with a negative score, and multiple community edits. However, they still remain open after those changes.
3. LQ_CLOSE: Low-quality posts that were closed by the community without a single edit.

The *score* here refers to the number of upvotes given to the question, and *edit* refers to the ability of users to suggest or make edits to questions posted by themselves or others. Questions might be edited for a variety of reasons, such as fixing spelling/grammatical errors or providing clarifying details.

To maintain the focus on text processing, only the *Title*, *Body,* and *Tags* columns are extracted from the dataset and used to train our models. Every Stack Overflow post contains a short title, larger body containing details and/or code, and a set of tags representing entities related to the post. To clean our data, we remove all HTML tags from the body, convert each set

of tags into a space-separated string, and make all of our text lowercase. The following is an example of a row from our extracted and cleaned dataset:

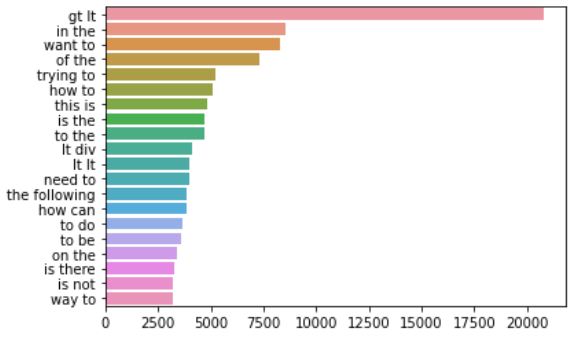
Title	Body	Tags	Label
equivalent of dynamic type "automatically adjusts font" setting for uibutton in interface builder?	a uilabel has a dynamic type: automatically adjusts font check-box in the attributes inspector in interface builder. is there an equivalent in interface builder for automatically adjusting the font size of a uibutton, or does this have to be handled in code? i'm using xcode 9.0 beta 6, targeting ios 11.	ios xcode uibutton interface-builder	HQ

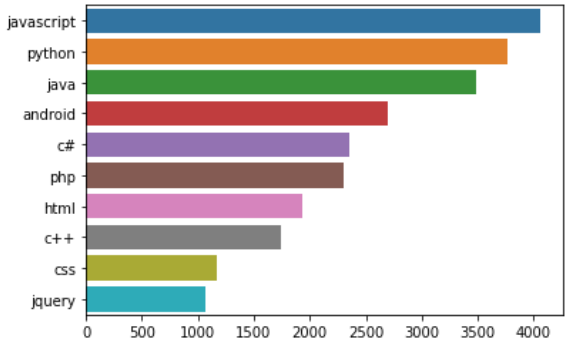
As shown in this example, *Body* values are generally much longer than *Title* values, and *Tags* often include specific technologies, programming languages, or frameworks. In addition to these, we also add features for each of these features word lengths, since they are potential indicators of question quality.

For training and testing our models, we use a 50:25:25 split of our dataset, where 30,000 questions are used for training, and the remaining 30,000 are split between validation and testing. (15,000 each)

4.2 Baseline

For our baseline, we train a logistic regression model because it is easy to interpret and supports multiclass classification.





The top 20 bigrams of the *Body* are general phrases containing stopwords such as “how to” that are used to formulate questions. However, the top non-stopwords are all programming languages and frameworks. In order to give more weight to non-stopwords, we generated

TF-IDF embeddings for all of our examples. These embeddings are matrices of TF-IDF features that would capture the relevance and importance of specific tokens to the documents in which they reside.

Given their ability to capture semantic similarity between words, we also experimented with generating GloVe and Doc2Vec embeddings from our text features and training logistic regression models with these as inputs. However, neither of these models' F1 scores surpassed that achieved with TF-IDF embeddings, which also takes the least time to generate. We hypothesize that our training set size was too small for these more complex approaches to develop effective embeddings, keeping in mind that these should be considered if more data was available.

4.3 Convolutional Neural Network

Since our data contains many unique terms and phrases, particularly source code, we are more concerned with feature detection than capturing long-range dependencies. Hence, in order to improve on our baseline, we decided to build a CNN that can help detect local patterns in our text that are common across examples.

To do this, we tokenize our text features, one-hot-encode our labels, and train a CNN with embedding, convolutional, max pooling, and dropout (20%) layers to minimize cross-entropy loss. In this case, the embedding layer handles our document embeddings instead of the TF-IDF vectorizer that was used for our baseline model.

4.4 RoBERTa

The RoBERTa model not only allows us to leverage the massive text corpora on which it was pretrained, but also optimizes on top of the original BERT model via improved pretraining and masking, as detailed by Liu et al. (2019). Like with our CNN, we feed the tokenized text into the pretrained RoBERTa model, followed by a softmax output layer.

5. Results

Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression (Baseline)	0.73	0.73	0.73	0.73
CNN	0.81	0.81	0.81	0.81
RoBERTa	0.91	0.90	0.90	0.90

In terms of all four performance metrics, most notably the F1 score, the RoBERTa model outperformed both baseline and CNN models on the test set by a noticeable margin.

7. Error Analysis

Label	Precision	Recall	F1 Score
HQ	0.94	0.81	0.87
LQ_CLOSE	0.78	0.91	0.84
LQ_EDIT	0.98	0.99	0.99

The label for which the RoBERTa model had the highest F1 score was *LQ_EDIT*. low-quality examples with multiple community edits. For the other two labels, F1 scores were similar, but high-quality questions achieved higher precision and lower recall, whereas low-quality questions (without edits) achieved higher recall and lower precision.

Examining the examples from the validation set for which the model predicted the wrong label, the majority were *HQ* examples predicted as *LQ_CLOSE* and vice-versa, reaffirming that the model has developed the strongest interpretation of the *LQ_EDIT* class compared to the other two classes. The set of wrongly predicted examples, the *wrongly predicted set*, does not differ significantly from the set of correctly predicted examples, the *correctly predicted set*, in terms of their most frequent n-grams and non-stopwords as well as their word length distributions. However, the proportion of unique tokens, calculated by dividing unique word count by total word count, is higher for the *wrongly predicted set* (19.08%) than than the *correctly predicted set* (13.29%). This suggests that there are unique tokens for which the model does not have enough examples to develop a meaningful understanding of its relationship with question quality.

For a deeper analysis, we found all of the tokens that appear only one time in the *wrongly predicted set* and take a random sample:

```
['foreach(student', 'beginbackgroundtaskwithname(taskname:', 'bmp',
'"c:/wamp/bin/php/php5.6.16/ext/"', 'for(i=0;', 'xmppframework', '"gtx',
'excessive.', '$school', 'auth:']
```

The majority of tokens from this sample appear to be extracted from code, such as “foreach(student”. Any examples for which the question body largely consists of code or domain-specific tokens like the ones above could be difficult for the model to predict, given that the model has little or no information about them. These types of examples are likely one of the primary sources of error in our RoBERTa model.

6. Conclusion

While the performance of our final model is solid, there are certain enhancements that could potentially improve our results:

1. We could pretrain the RoBERTa model further on a dataset that contains tokens specific to computer science, such as common syntax from popular programming languages. The RoBERTa model was trained on a massive corpus, but it was not particularly tuned for text that is specific to Stack Overflow's subject domain.
2. We can augment our text and length features with more numerical features that impact question quality. For example, the question creation date or overall sentiment could be good candidates.
3. Our dataset is very small relative to those used to train most state-of-the-art models. If we could obtain a large volume of labeled data or leverage unsupervised methods then our model could absorb a much greater amount of information and make better predictions.
4. We could improve our data cleaning and transformation to better handle numbers, code, special characters, and rare terms, such that our model would not be as influenced by tokens that only appear a handful of times. If, for example, we could map any *function calls* in code to a token, then that could potentially add valuable information to our document embeddings.

Based on these suggestions, our final model certainly has room for improvement. However, we were ultimately able to build a model that achieves a testing accuracy over 90%. If Stack Overflow were to use a scoring model to validate users' questions before they are posted, then they could potentially reduce the disparity between number of questions asked and number of questions answered. Similarly, the Stack Overflow search engine could assign higher weight to questions predicted to be high-quality, such that upvotes are not the sole factor determining how questions get ranked. Although these examples are specific to Stack Overflow, our model could be applied to various other platforms where evaluating text quality adds value. For example, Reddit subreddits could evaluate the quality of new comments to determine where they should show up in a user's feed, and live-stream Q&As could determine which viewers' questions should be answered.

References

Papers

Baltadzhieva, A., & Chrupała, G. (2015). Predicting the quality of questions on Stackoverflow. RANLP.

Le, Q.V., & Mikolov, T. (2014). Distributed Representations of Sentences and Documents. ArXiv, abs/1405.4053.

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., & Gao, J. (2020). Deep Learning Based Text Classification: A Comprehensive Review. ArXiv, abs/2004.03705.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. ArXiv, abs/1907.11692.

Pennington, J., Socher, R., & Manning, C.D. (2014). Glove: Global Vectors for Word Representation. EMNLP.

Sun, C., Qiu, X., Xu, Y., & Huang, X. (2019). How to Fine-Tune BERT for Text Classification? ArXiv, abs/1905.05583.

Websites

Freischlag, C. (2020, July 05). Combining numerical and text features in deep neural networks. Retrieved April 09, 2021, from https://towardsdatascience.com/combining-numerical-and-text-features-in-deep-neural-networks-e91f0237eea4

Guide : Tensorflow core. (n.d.). Retrieved April 09, 2021, from https://www.tensorflow.org/guide

Moore. (2020, October 12). 60K stack Overflow questions with quality rating. Retrieved April 09, 2021, from https://www.kaggle.com/imoore/60k-stack-overflow-questions-with-quality-rate

RoBERTa. (n.d.). Retrieved April 09, 2021, from https://huggingface.co/transformers/model_doc/roberta.html

Shahul ES Freelance Data Scientist | Kaggle Master Data science professional with a strong end to end data science/machine learning and deep learning (NLP) skills. Experienced working in a Data Science/ML Engineer role in multiple startups. K, ES, S., Freelance Data Scientist | Kaggle Master Data science professional with a strong end to end data science/machine learning and deep learning (NLP) skills. Experienced working in a Data Science/ML Engineer role in multiple

startups. Kaggle Kernels Master ra, & On, F. (2021, April 09). Exploratory data analysis for natural language processing: A complete guide to python tools. Retrieved April 09, 2021, from https://neptune.ai/blog/exploratory-data-analysis-natural-language-processing-tools

Team, K. (n.d.). Keras documentation: Keras API reference. Retrieved April 09, 2021, from https://keras.io/api/