Rewriting Stack Overflow Questions to Improve Writing Quality

Stanford CS224N Custom Project

Allison Casasola
Department of Mathematics
Stanford University
allyc@stanford.edu

Maximilien Cura
Stanford University
mcura@stanford.edu

Abstract

Stack Overflow (SO) is a popular question-and-answer website for programmers: users seeking help with technical problems submit questions, and other users on the site answer questions according to their interest and technical proficiencies. If a particular question is perceived as high-quality, community members be likely to upvote and/or respond to the question. However, if the question is perceived as low-quality, then community members will be likely to downvote, flag, or edit the question; the absence of positive response leads to lower visibility, and the perception of being low-quality may make members less interested in offering help; thus the question is less likely to be answered. The community perception of question quality may be dependent on many factors, but it is doubtless that the clarity of writing in the question will have an effect on comprehension and perception. Thus, our project wishes to explore how poorly-written SO questions can be rewritten using NLP techniques to improve question quality. We fine-tune multiple instances of the T5 text-to-text model with data from SO questions in several ways, and find that, in addition to certain Grammatical Error Correction (GEC) capabilities, it displays behaviours suitable for the particular domain of SO questions and overall performs better on test data drawn from SO questions than a T5 model fine-tuned on only GEC data.

1 Key Information to include

- Mentor: Heidi Zhang
- External Collaborators (if you have any): None
- Sharing project: No

2 Introduction

Stack Overflow serves as a platform for programmers to ask technical questions as well as respond to others' questions. Since its launch, Stack Overflow has become a ubiquitous feature of programmers' lives, currently hosting over 23.5 million questions[1] that cover an enormous variety of topics.

The questions are often complex in form–many questions include technical jargon recognizable to only a small community of programmers, links to websites or images, snippets of code, etc. Furthermore, the questions span a very broad range of computer science topics. For example, some questions are theoretical/generally applicable while others are much more specific to particular difficulties or problems an individual has run into while programming.

If members of the Stack Overflow community have a positive perception of a question (i.e. they perceive it as high-quality), they may decide to upvote it, having the effect of increasing its visibility

[1] https://stackoverflow.com/questions?tab=Newest

and making it more likely to be answered. If, on the other hand, members of the community perceive a question as low-quality, they may decide to downvote it, edit it, or in severe cases, flag it for review, having the effect of lowering its visibility; additionally, a potential answerer may be less likely to respond to a question they perceive as low-quality, overall having the effect of making the question less likely to be answered. The situation of unanswered questions is fairly common on the platform, with 3.4 million out of [2] of the total 23.5 million questions on the platform having no answers. Perhaps, if some of these questions were revised to improve the writing quality, more of them would have been answered.

Thus motivated, we seek to explore how NLP techniques can be utilized to rewrite poorly-written Stack Overflow questions. Our research questions is thus: "To what extent can natural language models be used to rewrite poorly written Stack Overflow questions and improve their quality of writing?"

3 Related Work

There is a significant body of research on various topics related to Stack Overflow datasets, among them various research related to question quality. For example, Tóth et al. (2019) present results about the usability of NLP methods for classifying question quality; our work, though, is quite different from theirs, since we are rewriting questions to improve quality, not simply classifying questions by quality. Tóth et al. (2019) also notes various other similar research directions, though they make no mention of question-rewriting, the areas they mention being distant from question-rewriting.

Chu et al.Chu et al. (2019) did similar work with 303 other Stack Exchange sites (Stack Overflow is one of the Stack Exchange sites), where they constructed a dataset from large numbers of questions from Stack Exchange sites, and explored various approaches to the question-rewriting problem. Their approach is aimed at rewriting general single-sentence questions from ill-formed to better-formed, whereas, in this project, we attempt to rewrite the entire text of question *posts*. Additionally, their dataset does not use any Stack Overflow data since "it is too specific to programming related questions," whereas our dataset is entirely from Stack Overflow (Chu et al., 2019). In their paper, they mention query and question rewriting, paraphrase generation, and text normalization as related areas, and why each of these areas is different from question rewriting to improve clarity, helpfulness, and utility.

4 Approach

In this project, we fine-tuned four instances of the T5 model Raffel et al. (2019) with approximately 220 million parameters (the `t5-base` model) and evaluated/compared their success at rewriting Stack Overflow questions.

- **cT5: T5 model fine-tuned on GEC data.** To create a model with standard GEC capability not specific to SO (as a basis for comparison to our models trained on SO data), we fine-tuned a T5-base model on 400k pairs of English learners' texts submitted to lang-8.com, a language learning platform, and the corrected versions of the texts. This serves as our *baseline* model. For the sake of simplicity, we call our baseline model cT5.
- **sT5: T5 model fine-tuned on Stack Overflow data.** We also fine-tuned an off-the-shelf T5 model on 214k Stack Overflow questions that received edits from the community so as to improve the question quality (see Data section for more information on this dataset). For the sake of simplicity, we call this model sT5.
- **csT5: T5 model fine-tuned on GEC and Stack Overflow data.** We took the baseline model cT5 and fine-tuned it again on the same Stack Overflow data used to fine-tune sT5. We call this model csT5.
- **hT5: T5 model fine-tuned on a dataset combining GEC and Stack Overflow data.** We generated a dataset that combined GEC and SO data (see Data section for more information) and fine-tuned an off-the-shelf T5 model on this. We call this model hT5, where the 'h' stands for 'hybrid'.

[2] `https://data.stackexchange.com/stackoverflow/query/1722774/number-of-questions-with-no-answer`

For sT5, csT5, and hT5, we aimed to see an improvement in rewriting SO questions from cT5 since these models are fine-tuned on in-domain SO data.

5 Experiments

5.1 Data

cLang-8. In order to fine-tune a model with GEC capability, we followed the work completed in the paper, "A Simple Recipe for Multilingual Grammatical Error Correction," which demonstrated that fine-tuning a T5 model on the cLang-8 dataset achieves state-of-the-art performance on Grammar Error Correction for English Rothe (2022). The cLang-8 ("cleaned Lang-8") dataset was created by cleaning the popular Lang-8 dataset used for GEC tasks in NLP. The Lang-8 dataset extracted data from Lang-8.com, a language learning platform where native speakers correct others' writing. In the Lang-8 dataset, inputs are English learners' text submitted to lang-8.com, and outputs are versions of the text corrected by other users on lang-8.com. Despite the popularity of Lang-8, it must be noted that "corrected texts frequently contain unnecessary paraphrasing and erroneous or incomplete corrections – phenomena that hurt the performance of a GEC model trained on this data" Rothe (2022). A cleaner version, cLang-8, of this dataset was produced using a state-of-the-art gT5 GEC model, retaining the original outputs and cleaning the outputs.

	cLang-8	Stack Overflow	Hybrid
Training set size	400k	214k	90k
Dev set size	50k	26k	9k
Test set size	-	26k	-

Table 1: Specifications used when fine-tuning with datasets.

Stack Overflow. Stack Exchange (the network that operates Stack Overflow) periodically uploads a data dump[3] which includes all publicly available content, including data about its questions, users, and comments, as well as various associated metadata. All questions that we use are under either CC BY-SA 2.5, CC BY-SA 3.0, or CC BY-SA 4.0 (this was checked programmatically during data processing). We used the `Posts.xml` and `PostHistory.xml` from the Stack Overflow data dump. We first used `Posts.xml` to obtain a list of all question posts in the data, as well as the user IDs of the original authors of the questions. Further, whenever a post is created or edited, a record is created in `PostHistory.xml`; we thus use these edits to create pairs of original and edited questions. Since community edits to questions are likely to improve the quality[4], we treat original versions and their later edited versions as low-quality to higher-quality pairs. We then performed the following processing steps:

1. We, upon inspection, found that edits to many questions involve the original author adding information that they previously had not included. Such edits are clearly not possible for the model to make, since the added information cannot be inferred from the original text. Thus, we filtered the data to only include questions with edits not performed by the original author. To simplify affairs, we kept only questions that had exactly one editor, where said editor was *not* the original author of the post.
2. An additional challenge is that questions often include code. Since code is effectively a separate 'language' (in fact, *languages*, multiple, due to the multiplicity of languages that are asked about in StackOverflow) from English, we decided to remove all questions that had blocks of code or inline code; it must, however, be noted that authors do not always use the formatting markup that indicates particular text to be code. Thus, questions in our dataset may still have varying amounts of short, inline code.

[3] `https://archive.org/download/stackexchange`

[4] `https://stackoverflow.com/help/editing`

3. Because T5 performs poorly with longer text, and due to the limits of our own computational resources, we performed a processing step eliminating question pairs that, when tokenized by T5, were likely to exceed 512 tokens.

This generated a set of approximately 268k source-target pairs. We then split the data into a training set of 214k pairs, a dev set of 26k pairs, and a test set of 26k pairs.

Hybrid. We also created a hybrid data set with the following composition where the training set

	cLang-8 examples	Stack Overflow examples	Total
Training set size	50k	40k	90k
Dev set size	5k	4k	9k

Table 2: Composition of the hybrid dataset

.

consisted of 50k examples from training set of the cLang-8 dataset, and 40k examples from the training set of the Stack Overflow data, and the dev set consisted of 5k and 4k examples from the two datasets' respective dev sets.

5.2 Evaluation method

To evaluate the success of cT5, sT5, csT5, and hT5 we used SacreBLEUPost (2018), a metric that computes BLEU scores. We performed these evaluations using a dataset of approximately 26k source-target pairs from the Stack Overflow dataset.

5.3 Experimental details

The parameters listed in Table 3 were used when fine-tuning cT5, sT5, csT5, and hT5, and each model achieved the dev set loss when fine-tuned on their respective datasets indicated in Table 5.

Max epochs	10
Optimizer	AdamW
Learning rate	3×10^{-4}
Max length	512 tokens
Training batch size	8
Dev set evaluation interval	400 steps

Table 3: Specifications used during fine-tuning.

We additionally performed early stopping based on the loss values from the dev set.

The code used for training the models can be found at `https://github.com/max-cura/cs224n-final-project`; it uses Hugging Face's Transformers library Wolf et al. (2020) and the SacreBLEU metric. To write the code, we used tutorials found in the Hugging Face platform[5] as well as one found here.

5.4 Results

The following SacreBLEU scores were computed when evaluating the three models on the 26k SO source-target pairs in the SO test set.

[5] `https://huggingface.co/`

	cT5	sT5	csT5	hT5
Dev set loss	0.151	0.2469	0.2499	0.2217

Table 4: Dev set loss after fine-tuning each model on their respective datasets. Note that the loss indicated for csT5 is the loss from its second round of fine-tuning, since csT5 is the product of fine-tuning cT5 *again* on SO data.

	cT5	sT5	csT5	hT5
SacreBLEU Score	65.98	84.59	84.72	82.69

Table 5: SacreBLEU scores computed on SO data.

Clearly, sT5, csT5, and hT5 perform better on the task of rewriting Stack Overflow questions than cT5, with little to no difference in score between sT5 and csT5, and hT5 performing slightly worse (possibly due to the smaller size of the dataset used in training hT5). The data demonstrates a difference in score of 19.74 between csT5 and cT5, with csT5 having a score 29.9% greater than that of cT5. Thus we find that our models learned various features of well-written questions between simple grammaticality.

6 Analysis

We find that sT5, csT5, and hT5 perform the task of accurately rewriting SO questions better than cT5 (with csT5 performing the best). While all four models share similar capacities in correcting spelling mistakes and capitalization of common words, there are several reasons why sT5, csT5, and hT5 outperform cT5. In this section, we motivate these reasons with examples.

Removing unnecessary text. We discover that csT5, sT5, and hT5 cut out text that is considered irrelevant to the actual question and hence is often edited out by community members. Consider the following example:[6]

Source Input[7]: `Hello developers out there, I wanted to know how I can get the Fragment which is onloaded on my Acticvity. The background behind this is, that I want to change the onBackPressed method that it's switching to the right fragments. At the moment when I press "Back" the app closes, because I work alot with fragments. Thanks for the help.`

Target Output: `I wanted to know how I can get the Fragment which is onloaded on my Acticvity. The background behind this is, that I want to change the onBackPressed method that it's switching to the right fragments. At the moment when I press &quot;Back&quot; the app closes, because I work alot with fragments.`

cT5 Prediction: `Hello developers out there, I wanted to know how I can get the fragment which is onloaded on my computer. The background behind this is that I want to change the onBackPressed method so that it's switching to the right fragments. At the moment when I press "Back" the app closes, because I work a lot with fragments. Thanks for the help.`

csT5 Prediction: `I wanted to know how I can get the Fragment which is onloaded on my Activity. The background behind this is, that I want to change the onBackPressed method that it's switching to the right fragments. At the moment when I press "Back" the app closes, because I work alot with fragments.`

[6]To reduce redundancy, we did not include sT5 and hT5 predictions in our examples. These predictions are close enough to our csT5 model's that including them would be redundant.

[7]`https://stackoverflow.com/questions/47982647/how-do-i-get-the-onloaded-fragment`. Asked by *Pumpanickel*. Edited by *Phantômaxx*.

In this and many other examples, csT5, sT5, and hT5 remove unnecessary greetings at the beginning of the question and unnecessary "thank you"s at the end of the question while cT5 does not. This makes sense as such edits are often made in Stack Overflow for the sake of brevity.

Recognizing technical jargon. csT5, sT5, and hT5 also perform well at recognizing and preserving technical terms while cT5 often does not. In the cases where cT5 does not recognize a technical term, it either changes the word to something more familiar, changes the sentence, or removes the sentence with the word entirely. In the following example, cT5 auto-corrects "oauth2" (for open authorization) as "orath 2" while csT5, sT5, and hT5 preserve "oauth2" as "oauth2."

Source Input [8]: `I need some help regarding google ouath2. I want to implement google login in my hybrid application but facing some problems . If anyone has demo solution please help,that would be great help for me.; Please Thanks`

Target Output: `I need some help regarding Google OAuth2. I want to implement Google login in my hybrid application but am facing some problems. If anyone has a demo application/solution please help,that would be great help for me. Please Thanks`

cT5 Prediction: `I need some help regarding google orath 2. I want to implement google login in my hybrid application but am facing some problems. If anyone has a demo solution please help. That would be a great help for me. Please Thanks`

csT5 Prediction: `I need some help regarding google ouath2. I want to implement google login in my hybrid application but facing some problems. If anyone has demo solution please help,that would be great help for me.`

Preserving links and special text. While csT5 and sT5 preserve links and images, cT5 often splits up the text for links/images into parts. Consider the following example:

Source Input[9]: `I am unable to select birthday date in calender picker in appium IOS, can anyone please suggest me on this issue. In appium inspector elements are not recognized. https://i.stack.imgur.com/Swql0.png`

Target Output: `I am unable to select birthday date in calender picker in appium IOS. In appium inspector elements are not recognized. https://i.stack.imgur.com/Swql0.png`

cT5 Prediction: `I am unable to select a birthday date in the calendar in iOS. Can anyone please suggest this issue. In the iOS inspector elements are not recognized. https://i.stack.imgur.com / Swql0. png`

csT5 Prediction: `I am unable to select birthday date in calender picker in appium IOS, can anyone please suggest me on this issue. In appium inspector elements are not recognized. https://i.stack.imgur.com/Swql0.png`

It is clear to see here that the text for the link to the image is split up in the cT5 prediction while the csT5 prediction preserves the original link.

Preserving conversational text. While cT5 does a good job of grammatical error correction, it is found that such corrections are not always necessary to rewrite a "better" Stack Overflow question. Many well-written SO questions take on short form or do not adhere *perfectly* to the rules of grammar. This may be because perfect grammar is not necessary for clarity. Consider the following example:

Source Input[10]: `I am tried to load a page with django formset, I clearly see in the logs that the backend logic is executed well within 1 second, I have a logger printing right before return render() and it is very clear that the backend process is almost done. The point here is,it is taking way too much time to render the template and nginx throwing bad gateway at me.`

[8] `https://stackoverflow.com/questions/47989078/how-to-implement-google-login-in-cordova-phonegap`. Asked by *Beginner*. Edited by *MozenRath*.

[9] `https://stackoverflow.com/questions/48006211/unable-to-select-date-in-calender-picker-in-appium-ios`. Asked by *Sudarshan K*. Edited by *James Z*.

[10] `https://stackoverflow.com/users/6613831/kireeti-k`. Asked by *Kireeti K*. Edited by *Mahi Parmar*.

What should I do?

Target Output: I am trying to load a page with django formset, I can clearly see in the logs that the backend logic is executed well within 1 second, I have a logger printing right before return render() and it is very clear that the backend process is almost done. The point is that,it is taking way too much time to render the template and nginx throwing bad gateway at me. What should I do?

cT5 Prediction: I am trying to load a page with a django formset. I clearly see in the logs that the backend logic is executed well within 1 second. I have a logger printing right before returning rendering() and it is very clear that the backend process is almost done. The point here is, it is taking way too much time to render the template and nginx throwing bad gateway at me. What should I do?

csT5 Prediction: I am tried to load a page with django formset, I clearly see in the logs that the backend logic is executed well within 1 second, I have a logger printing right before return render() and it is very clear that the backend process is almost done. The point here is, it is taking way too much time to render the template and nginx throwing bad gateway at me. What should I do?

In this example, cT5 detects that the first sentence is a run-on sentence and splits it into three different sentences with periods. However, the target output preserves the run-on sentence and its commas. This may be due to the fact that formal language is not always necessary for comprehensibility.

Despite the success of csT5, sT5, and hT5, it must be noted that these generally fail to suggest entire sentence format revisions that require reordering words of a sentence. This is most likely due to the fact that few target outputs in the Stack Overflow dataset actually reformat entire sentences. Most target outputs make minimal changes, such as inserting words, removing words, and fixing grammar, perhaps in the interest of preserving the original meaning of the question. And for the target outputs that do reformat entire sentences, it must be noted that there are multiple ways to reformat a sentence, which makes it difficult for the model to predict the "correct" way.

7 Conclusion

Summary. Conclusively, we find that csT5, sT5, and hT5 outperform cT5 in rewriting Stack Overflow questions to increase the overall quality, with the best model, csT5, outperforming cT5 by 29.9%. Fine-tuning a model so that it performs general Grammatical Error Correction tasks enables it to make some meaningful edits(such as capitalization, punctuation, and spelling corrections) as demonstrated by cT5; however, when applied to poorly written SO questions, it fails to recognize and remove unnecessary text, recognize technical jargon, preserve links and special phrases, and preserve conversational language. In contrast, fine-tuning a model on Stack Overflow questions enables the model to perform these kinds of tasks that are not captured by GEC models.

Limitations. Our work does not account for Stack Overflow questions that contain code. While this was an intentional choice, we recognize that this limits the applicability of our model on SO questions. Additionally, due to time and computational limitations, we had to make restrictions on the sizes of our models and the amount of data we trained them on.

Future work. Building off of this project, future work may attempt to do the following:

- **Fine-tuning on a larger dataset.**By fine-tuning on a larger SO dataset, a model might better learn good question-asking practices, at least within the SO community's notion of good questions, as well as become able to recognize more technical terms and other SO-specific features of the questions.
- **Exploring with larger models.** Fine-tuning larger models and comparing results can give more insight as to what the trade-off is between computational costs and accuracy.
- **Incorporating code.** Devising methods to work with posts with code would open up the ability to train on larger sections of the Stack Overflow dataset.

References

Zewei Chu, Mingda Chen, Jing Chen, Miaosen Wang, Kevin Gimpel, Manaal Faruqui, and Xiance Si. 2019. How to ask better questions? A large-scale multi-domain dataset for rewriting ill-formed questions. *CoRR*, abs/1911.09247.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Et al. Rothe, Sascha. 2022. A simple recipe for multilingual grammatical error correction.

László Tóth, Balázs Nagy, Dávid Janthó, Laszlo Vidacs, and Tibor Gyimóthy. 2019. Towards an accurate prediction of the question quality on stack overflow using a deep-learning-based nlp approach.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.