Beyond the Repo: A Case Study on Open Source Integration with GECToR

Sanjna Kashyap **Zhaoyang Xie** **Kenneth Steimel** **Nitin Madnani**
Educational Testing Service
Princeton, NJ, USA
skashyap@ets.org zxie@etscanada.ca ksteimel@ets.org nmadnani@ets.org

Abstract

We present a case study describing our efforts to integrate the open source GECToR code and models into our production NLP pipeline that powers many of Educational Testing Service's products and prototypes. The paper's contributions includes a discussion of the issues we encountered during integration and our solutions, the overarching lessons we learned about integrating open source projects, and, last but not least, the open source contributions we made as part of the journey.

1 Introduction

GECToR (Grammatical Error Correction Tag, not Rewrite)[1] is a set of deep learning models developed by Grammarly for the task of Grammatical Error Correction or GEC (Omelianchuk et al., 2020). GECToR achieves state-of-the-art results for the GEC task and its inference speed is up to 10 times as fast as that of equivalent Transformer-based sequence-to-sequence (seq2seq) GEC systems.

The most commonly proposed systems for the GEC task leverage seq2seq Neural Machine Translation (NMT) models to "translate" from errorful text to corrected text. However, such systems generally suffer from slow inference and require a large amount of data. To deal with these issues, GECToR simplified the task from sequence generation to sequence tagging. To train this tagging system, GECToR utilizes three training stages: pre-training on synthetic data, fine-tuning on parallel datasets that contain both errorful and corrected texts, and further fine-tuning on a combination of high-quality, parallel datasets containing both errorful/corrected as well as error-free texts. However, one of the largest benefits of GECToR to the NLP community is that it was open sourced under a commercially-unrestricted Apache 2.0 license.

2 Requirements

Educational Testing Service (ETS) has developed a pipeline that uses a service-based architecture to combine multiple NLP services into scalable and robust backend applications (Madnani et al., 2018). These applications are used to evaluate the speaking and writing proficiency of students' written essays or spoken responses and provide both automatic scores as well as actionable feedback. Specifically, our pipeline provides descriptive feedback on multiple dimensions such as the student's grammar, mechanics, vocabulary, text complexity, style, organization, among others. Our pipeline is used for various high-stakes assessments, e.g., the Analytical Writing section from GRE (Graduate Record Examinations)[2] and the Independent and Integrated Writing prompts from TOEFL iBT (Test of English as a Foreign Language Internet-Based Test).[3]

Our pipeline has two main requirements: (1) every NLP service should return its results in less than a few seconds to enable near real-time feedback and (2) all models used in services should be optimized for precision over recall to minimize unfair penalization of students. After a careful evaluation of GECToR's GEC performance and inference speed, we felt comfortable in replacing our existing GEC system with GECToR.

GECToR provides three already trained English GEC models based on BERT, RoBERTa, and XLNET. It also has scripts for training and inference. Since our pipeline is English-only for now, the existing GEC models perfectly fit our needs. Therefore, our integration efforts focus entirely on the inference side.

[1] https://github.com/grammarly/gector

[2] https://www.ets.org/gre.html

[3] https://www.ets.org/toefl.html

Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023), pages 78–82
December 6, 2023 ©2023 Association for Computational Linguistics

3 Related Work

Adopting open source software in commercial environments has been a topic of much interest. There are several barriers to open source use: lack of knowledge, inability to incorporate it into existing legacy systems, too many forks created by different groups, technological immaturity, et cetera (Nagy et al., 2010). One formal approach that can be used to assess an open source project before putting it into production is the Open Source Maturity Model (OSMM) which assesses factors like the product, support, training, documentation, product integration and professional services, and gives them a weighted score. However, this model is only a first step in identifying which projects are worth a more in-depth evaluation (Golden, 2005). In this paper, we hope to provide a detailed case study with illustrative, concrete steps for using open source projects.

There are major forks and re-implementations of the original GECToR project available on GitHub. **fast-gector**[4] claims to be a faster and simpler implementation of the original project leveraging AMP[5] and DeepSpeed (Rasley et al., 2020). However, we were unable to reproduce the same results as the original model in our experiments with fast-gector. **gector-large** (Tarnavskyi et al., 2022) focuses on improving GECToR by upgrading the Transformer encoders, and using an ensemble model for span-level edits. According to the paper, while the larger encoders do yield better performance, they do so at the cost of speed with inference being 2.3-2.5x slower.

4 Integration

In this section, we discuss the challenges we faced when integrating GECToR into our pipeline and our solutions.

4.1 Issues

Before we delve into the integration issues, we want to impress upon the reader that the existence of such issues should not detract from or minimize the usefulness of GECToR (or other open source software) to the NLP community. However, we feel that an honest discussion of such real-world issues can provide both the authors and users of such software with useful, actionable information.

GECToR was open sourced as part of a research publication and aligns well with the needs of the academic community. However, as we attempted to adapt it for use in a commercial NLP pipeline, we felt that some critical requirements were not fully addressed in its original design. This mismatch between the original authors' intended use and users seeking to *productionize* it led to challenges in utilizing GECToR effectively in its original form.

The system was not under active development, which posed challenges for commercial adoption. The dynamic nature of commercial environments needs ongoing development and maintenance to keep pace with rapidly evolving requirements and security issues. A specific challenge in adopting GECToR for our purposes was that it used *significantly* older versions of Python, PyTorch, and AllenNLP (Gardner et al., 2018). Its reliance on these outdated dependencies prevented seamless integration into our existing environments and limited access to the latest features and optimizations.

GECToR was also not packaged for easy installation, further complicating its integration into existing environments. The library could only be used by cloning the repository, downloading the model, and running some scripts created by the authors. Packaging GECToR and streamlining its installation process would be vital to ensure smooth continuous integration and deployment.

GECToR was designed to leverage AllenNLP, a powerful NLP library. However, its usage did not fully exploit the capabilities that AllenNLP offered, namely its high-level abstractions and API, leading to potentially sub-optimal performance. Properly utilizing AllenNLP would significantly enhance the library's overall effectiveness and result in more standardized and maintainable code.

4.2 Solutions

To address the issues we identified with GECToR in the previous section, we undertook several mitigation strategies. These efforts focused on enhancing its functionality, compatibility, and maintainability while still preserving the overall integrity of the original codebase. To implement these strategies, we forked GECToR on GitHub[6] and added the following improvements to our fork.

[4] https://github.com/cofe-ai/fast-gector

[5] https://developer.nvidia.com/automatic-mixed-precision

[6] https://github.com/EducationalTestingService/gector

4.2.1 Regression & unit tests

To ensure that any modifications or updates made by us do not lead to differences in predictions made by the existing models, the *very first* thing we did in our fork was to implement a comprehensive test suite comprising of a total of 95 tests (37 unit tests as well as 58 regression tests). We decided that a good way to get started was to test each AllenNLP component separately, e.g., the tokenizer, the token indexer, the embedder etc. This gave us not only a clear structure to follow but also a reasonable granularity for the test cases.

The test suite not only helps in verifying the correctness of the system but also acts as a safety net to prevent potential differences in the future. We also added a continuous integration plan using Github Actions to automatically run the complete test suite for any changes made to the codebase.

4.2.2 Updated dependencies

One of the critical steps in making GECToR suitable for production was to update its dependencies to modern, supported versions. We carefully updated the library to work seamlessly with the latest versions of Python (3.7 only → 3.8 through 3.10), PyTorch (1.10.0 → 1.12.1), AllenNLP (0.8.4 → 2.10.0), and many other dependencies. By doing so, we ensured compatibility with more modern infrastructure and took advantage of the latest advancements in frameworks and tools. This allows the installation of GECToR in existing CI and deployment environments via package managers such as conda,[7] and minimizes dependency conflicts. Our regression tests ensured that there were no changes to the output as a result of our updating the package versions.

4.2.3 True AllenNLP-ification

We followed AllenNLP's recommended abstractions and guidelines to re-architect our fork of GECToR, enabling a more streamlined integration with AllenNLP. The architecture adopted was largely influenced by the design and modules of AllenNLP. Utilities for training and inference were already provided by AllenNLP so our changes largely reshaped the existing GECToR codebase to function as an extension of AllenNLP. Specifically:

1. To facilitate better configurability and ease of use, we registered GECToR modules with the AllenNLP framework: each file in our fork contains one type of AllenNLP component like tokenizers, token indexers, models, predictors, and dataset readers. The registration allowed model architecture, tokenizer settings and preprocessing options to be referenced and contained in a single `jsonnet` file used for both training and inference.

2. Users can now access GECToR *directly* through AllenNLP-supported configuration files, enhancing its usability.

3. GECToR models can now be bundled into an AllenNLP model archive for easy distribution and inference.

Since we completed this work on GECToR, AllenNLP has been archived and is no longer being maintained. We discuss the implications of this in §5.4.

4.2.4 Packaging & easy installation

We architected our GECToR fork to better support packaging, created a conda recipe, and deployed a publicly-available package on our public conda channel.[8] This makes the installation of GECToR significantly easier and reproducible (Arvan et al., 2022), unlike installing from the Github source repository every time since it is challenging to keep track of any changes made to the code since the last installation from source. We also packaged the GECToR RoBERTa model for easy installation via conda.[9]

By implementing rigorous testing, updating dependencies, aligning with AllenNLP guidelines, and creating easily installable packages, we were able to successfully mitigate the challenges associated with productionizing the original version of GECToR. We argue that these efforts have transformed GECToR into a more robust and adaptable library for grammatical error correction, suitable for use in any production Python environment.

5 Lessons learned

While bolstering GECToR, we learned a few general lessons about open source development and integration that we would like to share.

5.1 Projects should explicitly state a purpose.

In our opinion, open source authors should explicitly state the purpose of their projects. Document-

[7] https://docs.conda.io/en/latest/

[8] https://anaconda.org/ets/gector

[9] https://anaconda.org/ets/gector-roberta

ing whether the codebase is intended solely for research purposes or whether it is ready for production can help potential users easily estimate the level of effort required for integration. We hope our message is clear. Open source research projects like GECToR are invaluable for the community. However, the level of involvement required to productionize research codebases can vary significantly.

5.2 Estimation of effort is hard but necessary.

It is crucial to perform a careful analysis of the effort involved in integrating an open source project into a production codebase. Most popular open source projects are under active development and are used by a large number of users and organizations. Such projects are usually created or maintained by commercial organizations with dedicated teams working on them. Any issues or feature requests have a higher chance of being addressed and implemented respectively.

However, sometimes a smaller project or one open sourced as part of a research publication might be more suitable for your needs. In such cases, you must do your best to examine the codebase and develop a reasonable estimate for the integration effort. It is essential to carefully test the project and develop a plan to gauge whether the level of work as indicated by the resulting estimate can be offset by the value provided to your own product or project.

5.3 Test, test, test!

Many open source projects do not implement any form of testing since it's not perceived to be necessary for research projects. However, in our opinion, a good testing setup is critical *irrespective* of the eventual use case since it assures users that the code actually behaves as expected. This may seem like a large cost to take on upfront, but can significantly reduce technical debt in the future. We strongly recommend that everyone who contributes to open source consider adding unit/regression testing along with a CI plan which can help identify bugs and failures as and when they happen, regardless of whether their project is meant only for experimental purposes.

5.4 Always have a contingency plan.

One of the primary concerns with open source projects is the risk of abandonment by their original authors. While many projects have active maintainers and thriving communities, there have been instances where developers discontinue support and maintenance. The case of AllenNLP[10] serves as an example of how a once prominent open source library can become stagnant or archived due to shifting priorities or organizational changes.

This can have serious consequences for those who have integrated such projects into their production environment since bugs and security vulnerabilities will likely go unaddressed in the future. Therefore, it is important to be prepared for such scenarios and have a well-defined strategy in place for either taking on the entire maintenance of the library or transitioning to an alternate solution.

6 Future Work

In the future, we plan to take our own advice from §5.4 and transition our fork of GECToR away from AllenNLP. We plan to replicate the functionality we need by using actively maintained open source projects from Huggingface such as Transformers (Wolf et al., 2020), Datasets (Lhoest et al., 2021), and Accelerate[11] directly.

7 Conclusion

We presented a case study on integrating an open source project into a commercial, production NLP pipeline. While we consider the primary contributions of this paper to be a clear and concise description of the issues we faced (and solved) as well as the larger lessons we learned, we also hope that the NLP community will benefit from our fork of GECToR that is actively maintained, more modern, more robustly tested, and easier to use for inference.

References

Mohammad Arvan, Luís Pina, and Natalie Parde. 2022. Reproducibility in computational linguistics: Is source code enough? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2350–2361, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018.

[10] https://github.com/allenai/allennlp

[11] https://huggingface.co/docs/accelerate/index

AllenNLP: A Deep Semantic Natural Language Processing Platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

B. Golden. 2005. *Succeeding with Open Source*. Addison-Wesley information technology series. Addison-Wesley.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Sasko, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clement Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander M. Rush, and Thomas Wolf. 2021. Datasets: A Community Library for Natural Language Processing. *CoRR*, abs/2109.02846.

Nitin Madnani, Aoife Cahill, Daniel Blanchard, Slava Andreyev, Diane Napolitano, Binod Gyawali, Michael Heilman, Chong Min Lee, Chee Wee Leong, Matthew Mulholland, and Brian Riordan. 2018. A Robust Microservice Architecture for Scaling Automated Scoring Applications. *ETS Research Report Series*, 2018(1).

Del Nagy, Areej M. Yassin, and Anol Bhattacherjee. 2010. Organizational adoption of open source software: Barriers and remedies. *Commun. ACM*, 53(3):148–151.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr" Skurzhanskyi. 2020. GECToR – Grammatical Error Correction: Tag, Not Rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170. Association for Computational Linguistics.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Maksym Tarnavskyi, Artem Chernodub, and Kostiantyn Omelianchuk. 2022. Ensembling and knowledge distilling of large sequence taggers for grammatical error correction. In *Accepted for publication at 60th Annual Meeting of the Association for Computational Linguistics (ACL 2022)*, Dublin, Ireland.

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. HuggingFace's Transformers: State-of-the-art Natural Language Processing.