Graduate School of Science and Technology Master's Thesis Abstract

Laboratory name (Supervisor)	Software Design and Analysis (Hajimu Iida (Professor))		
Student ID	2411427	Submission date	2026 / 7 / 22
Name	SUWANACHOTE NABHAN		
Thesis title	Evaluating the Impact of Domain Adaptation for Code Completion Models: A Large Emprical Study		

Abstract

Pre-trained code models have demonstrated strong capabilities in predicting missing code and supporting software development. These models are typically trained on large amounts of code collected from massive software repositories, allowing them to learn common programming patterns across projects. However, such "general" training may not capture the specific APIs, naming conventions, and recurring implementation patterns of an individual project. Although repository-level domain adaptation addresses this limitation by further fine-tuning a code model on project-specific data for code completion, its effectiveness and characteristics across models and projects remain insufficiently investigated.

In this work, we address this gap through a systematic empirical study evaluating three mainstream code models across four representative Java projects. To evaluate and characterize the benefits of repository-level domain adaptation, we compare domain-adapted models with general fine-tuned models. Beyond this, we also study the impact of training data size on adaptation performance under various settings. As the projects' evolution is an inevitable reality during software development, we further investigate whether the benefits of domain adaptation persist as the target projects evolve. Several insightful findings emerge from our study. First, we find that smaller models (e.g., CodeT5) benefit more from repository-level domain adaptation than larger models, albeit with more conspicuous performance fluctuations across different training-data sizes. Secondly, it is observed that project evolution is still a pressing challenge for domain-specific code completion, as evidenced by the ubiquitous performance degradation in all settings. However, degradation appears to depend more on the target project's characteristics than on the model. Lastly, our qualitative analysis reveals that over-generation can have syntactic and behavioral consequences: 55.2% of over-generation cases also involve invalid syntax, while 20.7% involve wrong control flow. This suggests that domain adaptation not only improves completion performance but also helps avoid unnecessary code that is invalid or alters the intended execution. In practice, developers could consider adopting a smaller domain-adapted model; however, they still need to validate its predictions and monitor its performance as the target project evolves.