The Adverse Effects of Code Duplication in Machine Learning Models of Code

Abstract

The field of big code relies on mining large corpora of code to perform some learning task towards creating better tools for software engineers. A significant threat to this approach was recently identified by Lopes et al. [19] who found a large amount of near-duplicate code on GitHub. However, the impact of code duplication has not been noticed by researchers devising machine learning models for source code. In this work, we explore the effects of code duplication on machine learning models showing that reported performance metrics are sometimes inflated by up to 100% when testing on duplicated code corpora compared to the performance on de-duplicated corpora which more accurately represent how machine learning models of code are used by software engineers. We present a duplication index for widely used datasets, list best practices for collecting code corpora and evaluating machine learning models on them. Finally, we release tools to help the community avoid this problem in future research.

大代码领域依赖于挖掘大型代码库来执行一些学习任务，从而为软件工程师创建更好的工具。Lopes等人最近发现了这种方法的一个重大威胁，他们在GitHub上发现了大量近乎重复的代码。然而，为源代码设计机器学习模型的研究人员没有注意到代码复制的影响。在这项工作中,我们探索的影响代码重复机器学习模型显示,性能指标有时报道夸大了100%当测试重复代码语料性能相比减少语料库更准确地代表机器学习模型的代码是如何由软件工程师使用。我们为广泛使用的数据集提供了一个重复索引，列出了收集代码库和评估机器学习模型的最佳实践。最后，我们发布工具来帮助社区在未来的研究中避免这个问题。

1 Introduction

Machine learning models of source code have recently received great attention from the research community. At the intersection of the research fields of software engineering, programming languages, machine learning and natural language processing, multiple communities have been brought together into the field of “Big Code” or “code naturalness” with many fruitful results [1]. Commonly, research in this area relies on large corpora of code which can be used as training and test sets, allowing machine learning methods to learn and probabilistically reason about coding practice at a large scale. The goal is to use the learned models to provide useful tools to software engineers.

源代码的机器学习模型最近受到了研究领域的极大关注。在软件工程、编程语言、机器学习和自然语言处理等研究领域的交叉点上，多个社区融合在“大代码”或“代码自然性”领域，取得了丰硕的成果。通常，这一领域的研究依赖于可作为训练和测试集的大型代码库，这使得机器学习方法能够学习大规模编码实践的概率推理。目标是使用所学的模型为软件工程师提供有用的工具。

However, there is a looming crisis in this newly-founded area, caused by a disproportionately large amount of code duplication. This issue — first observed by Lopes et al. [19] — refers to the fact that multiple file-level (near-)clones appear in large corpora of code, such as those mined from GitHub repositories. This is because software engineers often copy — partially or entirely — files from other projects [11, 19]. Despite the findings of Lopes et al. [19], the research community has not yet investigated how and when code duplication negatively affects its research, the machine learning models it devises, and the practical tools it creates. The core issue arises from the fact that identical or highly similar files appear both in the training and test sets that are used to train and evaluate the machine learning models.

然而，在这个新成立的领域中，存在着一个迫在眉睫的危机，这是由不成比例的大量代码重复造成的。这个问题——Lopes等人首先观察到的[19]——指的是多个文件级(近)克隆出现在大型代码库中，比如那些从GitHub库中挖掘出来的代码。这是因为软件工程师经常从其他项目中部分或全部复制文件[11,19]。尽管Lopes et al.[19]的发现，研究社区还没有调查代码复制如何以及何时负面影响其研究，它设计的机器学习模型，以及它创建的实用工具。核心问题源于这样一个事实，即用于培训和评估机器学习模型的培训和测试集中都出现了相同或高度相似的文件。

In this work, we first describe the impact that code duplication can have on machine learning models. Although not all applications of machine learning models are affected by code duplicates, a large majority of them is. We discuss the biases introduced when evaluating models under duplication and show that duplication can cause the evaluation to overestimate the performance of a model compared to the performance that actual users of the model observe. Then, we replicate the work of Lopes et al. [19] across ten corpora that have been used in “big code” research and we measure the impact of duplication across datasets and machine learning models showing that the performance observed by a user is up to 50% worse compared to reported results. Although this paper does not present any results or ideas that would be unexpected to a statistician or a machine learning expert, we hope that it will help programming language, software engineering and machine learning researchers better understand the issue of code duplication for machine learning on code by clearly illustrating its impact. At the same time, we provide tools and some best practices that can help overcome pitfalls when researching machine learning methods that employ source code data. We hope that this paper contributes the following:

• an application-driven principle for deciding if within the application domain code corpus de-duplication is needed (Section 2);

• the theoretical basis of the effects of code duplication (Section 2) and a demonstration of the effects of code duplication on machine learning models of source code (Section 4);

• an open-source, cross-platform tool that detects near duplicates in C#, Java, Python and JavaScript along with a duplication index for existing datasets, listing existing duplicate files (Section 3);

• a set of suggested best practices to mitigate the code duplication problem for machine learning models of code (Section 5).

在这项工作中，我们首先描述代码复制对机器学习模型的影响。虽然并不是所有的机器学习模型应用程序都受到代码重复的影响，但其中大部分都受到影响。我们讨论了在重复情况下评估模型时引入的偏差，并表明，与模型的实际用户所观察到的性能相比，重复会导致评估高估模型的性能。然后我们复制的工作洛佩斯等。[19]在十全集,用于“大代码”研究和我们测量的影响复制数据集和机器学习模型显示,观察到的性能由用户50%相比更糟糕的结果。虽然本文不存在任何结果或想法,会出现意想不到的统计学家或机器学习专家,我们希望这将有助于编程语言,软件工程和机器学习研究人员更好地理解代码重复的问题对机器学习代码通过清晰地说明它的影响。同时，我们提供了一些工具和一些最佳实践，可以帮助您在研究使用源代码数据的机器学习方法时克服一些缺陷。我们希望本文能对以下方面做出贡献:

•应用程序驱动原则，用于决定是否需要在应用程序域代码库中进行重复数据删除(第2节);

•代码复制效果的理论基础(第2节)和代码复制对源代码机器学习模型影响的演示(第4节);

•一个开源的跨平台工具，可以检测c#、Java、Python和JavaScript中的几乎重复项，并为现有数据集提供重复索引，列出现有的重复文件(第3节);

•一组建议的最佳实践，以减轻代码机器学习模型的代码复制问题(第5节)。

2 Code Duplication & Machine Learning

Code duplication refers to the idea that a large snippet of code appears multiple times with no or small differences within a corpus of code. Duplicates are a relatively small subset of code clones [25] — a well-studied field of software engineering. The existence of duplicates was noticed much earlier [27] but their negative effect became significantly more noticeable due to recent advancements that allowed the collection of large code corpora [19]. In this paper, we are specifically interested in illustrating the effects of code duplication on machine learning models of code1 . This endeavor sets different parameters for searching, understanding and classifying code duplication. To understand the effects of duplicates, we first need to discuss the practical applications of machine learning models for code.

代码复制是指一个大的代码片段出现多次，而在一个代码库中没有或只有很小的差异。重复是代码克隆[25]的一个相对较小的子集，[25]是软件工程的一个研究领域。复制的存在早在[27]出现之前就被注意到了，但是由于最近的改进允许收集大型代码库[19]，它们的负面影响变得更加明显。在本文中，我们特别感兴趣的是说明代码复制对code1机器学习模型的影响。这项工作为搜索、理解和分类代码复制设置了不同的参数。为了理解重复的效果，我们首先需要讨论代码的机器学习模型的实际应用。

Why do we want to train machine learning models on source code? At a high-level, the goal is to train models on existing code, such that the learned models capture the statistical properties of some particular aspect of coding practice, which can then be useful within a tool used by a software engineer. Some examples of recently researched models include:

• code completion models [14, 15, 20, 24] aiming to assist code construction in an editor when a developer is writing new code. Such models are widely used in practice today.

[14] Vincent J Hellendoorn and Premkumar Devanbu. 2017. Are deep neural networks the best choice for modeling source code?. In Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering.

ACM, 763–773.

[15] Abram Hindle, Earl T Barr, Zhendong Su, Mark Gabel, and Premkumar Devanbu. 2012. On the naturalness of software. In Software Engineering (ICSE), 2012 34th International Conference on. IEEE, 837–847.

[20] Chris Maddison and Daniel Tarlow. 2014. Structured generative models of natural source code. In Proceedings of the International Conference on Machine Learning (ICML). 649–657.

[24] Veselin Raychev, Martin Vechev, and Eran Yahav. 2014. Code completion with statistical language models. In Proceedings of the Symposium on Programming Language Design and Implementation (PLDI), Vol. 49.

ACM, 419–428.

• Type prediction models [13, 23] where the goal is to infer (or provide probabilistic hints for) the types of new, previously untyped, programs (e.g. in JavaScript) ;

[13] Vincent J Hellendoorn, Christian Bird, Earl T Barr, and Miltiadis Allamanis.

2018. Deep learning type inference. In Proceedings of the

2018 26th ACM Joint Meeting on European Software Engineering Conference

and Symposium on the Foundations of Software Engineering. ACM,

152–162.

[23] Veselin Raychev, Martin Vechev, and Andreas Krause. 2015. Predicting

program properties from Big Code. In Proceedings of the Symposium on

Principles of Programming Languages (POPL), Vol. 50. ACM, 111–124.

• code summarization [3, 5, 7, 16] where the goal is to summarize some code into a short natural language utterance.

[3] Miltiadis Allamanis, Hao Peng, and Charles Sutton. 2016. A convolutional

attention network for extreme summarization of source code.

In Proceedings of the International Conference on Machine Learning

(ICML). 2091–2100.

[5] Uri Alon, Omer Levy, and Eran Yahav. 2010. code2seq: Generating

Sequences from Structured Representations of Code. In Proceedings of

the International Conference on Learning Representations (ICLR).

[7] Antonio Valerio Miceli Barone and Rico Sennrich. 2017. A Parallel Corpus

of Python Functions and Documentation Strings for Automated

Code Documentation and Code Generation. In Proceedings of the Eighth

International Joint Conference on Natural Language Processing (Volume

2: Short Papers), Vol. 2. 314–319.

[16] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer.

2016. Summarizing source code using a neural attention model. In Proceedings

of the 54th Annual Meeting of the Association for Computational

Linguistics (Volume 1: Long Papers), Vol. 1. 2073–2083.

为什么我们要在源代码上训练机器学习模型?在高层次上，目标是在现有代码上训练模型，这样所学习的模型就可以捕获编码实践的某些特定方面的统计特性，这些特性可以在软件工程师使用的工具中使用。最近研究的模型包括:

•代码完成模型[14,15,20,24]，目的是在开发人员编写新代码时帮助编辑器进行代码构建。这种模型在今天的实践中得到了广泛的应用。

•类型预测模型[13,23]，其目标是推断(或提供概率提示)以前未类型化的新程序的类型(如JavaScript);

代码摘要[3,5,7,16]，目的是将一些代码总结成简短的自然语言。