Tweaking Association Rules to Optimize Software Change Recommendations





Mairieli Wessel, Maurício Aniche, Gustavo Oliva,
Marco Gerosa, Igor Wiese

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Queen's University

TU Delft

NORTHERN ARIZONA UNIVERSITY

A medida que um software evolui, sua estrutura tende a tornar-se mais complexa.

A medida que um software evolui, sua estrutura tende a tornar-se mais complexa.

Recomendação de Mudança

- Regras de Associação
- Identificam acoplamentos de mudança entre artefatos

Se um desenvolvedor alterar o arquivo A, ele provavelmente irá alterar o arquivo B.

Vou usar recomendação de mudanças no meu projeto!

Qual o tamanho do histórico devo usar para gerar as regras?

E os valores de suporte e confiança? Quais devo usar?

Novas mudanças no projeto alteram minhas recomendações?

Objetivo

Investigar como determinar empiricamente os limiares das medidas de interesse e o conjunto de treinamento que geram as recomendações de mudança com maior acurácia.

Função de Regressão Linear

Treinamento = 33974.1

\+ 208.4 x Número de arquivos

\- 3958.9 x Média do tam. dos *commits*

2016 IEEE 16th International Working Conference on Source Code Analysis and Manipulation

Exploring the Effects of History Length and Age on Mining Software Change Impact

Leon Moonen*, Stefano Di Alesio*, Thomas Rolfsnes* and Dave W. Binkley§
*Simula Research Laboratory, Oslo, Norway §Loyola University Maryland, Baltimore, Maryland, USA
leon.moonen@computer.org, {stefano,thomgrol}@simula.no, binkley@cs.loyola.edu

Abstract—The goal of Software Change Impact Analysis is to identify artifacts (typically source-code files) potentially affected by a change. Recently, there is an increased interest in *mining* software change impact based on evolutionary coupling. A particularly promising approach uses association rule mining to uncover potentially affected artifacts from patterns in the system's change history. Two main considerations when using this approach are the *history length*, the number of transactions from the change history used to identify the impact of a change, and *history age*, the number of transactions that have occurred since these patterns were mined. The resulting rules directly affect the quality of any change impact analysis based on mined evolutionary coupling. However, while reviewing the literature, we found that the effects of history length and age on mined association rules have not been systematically studied. We address this shortcoming.

Index Terms—change impact analysis, evolutionary coupling, association rule mining, parameter tuning.

I. Introduction

When software systems evolve, the interactions in the source code grow in number and complexity. As a result, it becomes increasingly challenging for developers to predict the overall effect of making a change to the system. Change Impact Analysis [1] has been proposed as a solution to this problem, aimed at identifying software artifacts (e.g., files, methods, classes) affected by a given change. Traditionally, techniques for change impact analysis are based on static or dynamic analysis, which identify dependencies, for example, methods calling or called by a changed method [2, 3, 4]. However, static and dynamic analysis are generally language-specific, making them hard to apply to modern heterogeneous software systems [5]. In addition, dynamic analysis can involve considerable overhead (e.g., from code instrumentation), while static analysis tends to over-approximate the impact of changes [6].

To address these challenges, alternative techniques have been proposed that identify dependencies through *evolutionary coupling* [7, 8, 9, 10]. In essence, evolutionary coupling exploits a developers inherent knowledge of the dependencies in the system, which manifest themselves through commit comments, bug reports, context switches in IDEs, and so on [8]. These couplings differ from those found through static and dynamic analysis, because they are based on how the software system has evolved over time, rather than how system components are interconnected.

This paper considers *historical co-change* between artifacts as the basis for uncovering evolutionary coupling. Known techniques [10, 11, 12, 13] for mining evolutionary couplings from artifact co-changes build on *association rule mining* (or *association rule learning*) [14], and differ in the way that the association rules are generated from the history. Nevertheless, key to all such techniques is the *history* used to learn from. There are two main factors related to the history that impact the mined rules: (1) the *history length*, the number of transactions considered while mining co-change *age*, the number of transactions that have occurred since these patterns were mined. The resulting rules directly affect the quality of any change impact analysis based on mined evolutionary coupling. However, while reviewing the literature, we found that the effects of history length and age on mined association rules have not been systematically studied. We address this shortcoming.

Contributions: This paper presents a series of systematic experiments using the change histories of two large industrial systems and 17 large open source systems. The study makes two key contributions: (1) we investigate the extent to which history length and age affect the quality (formalized in Section IV-F) of the change impact sets derived via association rule mining, and (2) we derive practical guidelines for selecting an appropriate system-specific value for history length and for determining at what age a model has sufficiently deteriorated to benefit from rebuilding. The guidelines enable a team of engineers to best exploit association rule mining for change impact analysis in the context of their project.

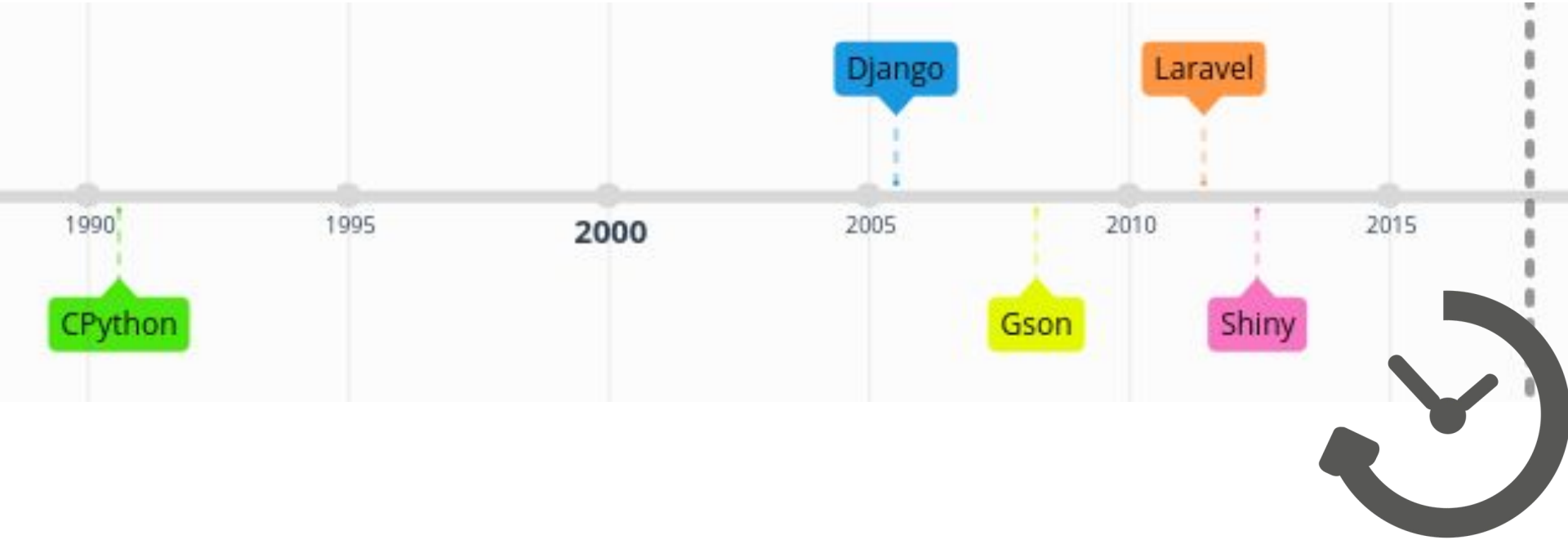
Overview: The remainder of this paper is organized as follows: Section II provides background on mining evolutionary coupling. Section III presents our research questions. Section IV describes the setup of our empirical investigation, whose results are presented in Section V. Finally, Section VI presents related work, and then Section VII provides some concluding remarks.

978-1-5090-3848-0/16 $31.00 © 2016 IEEE
DOI 10.1109/SCAM.2016.9

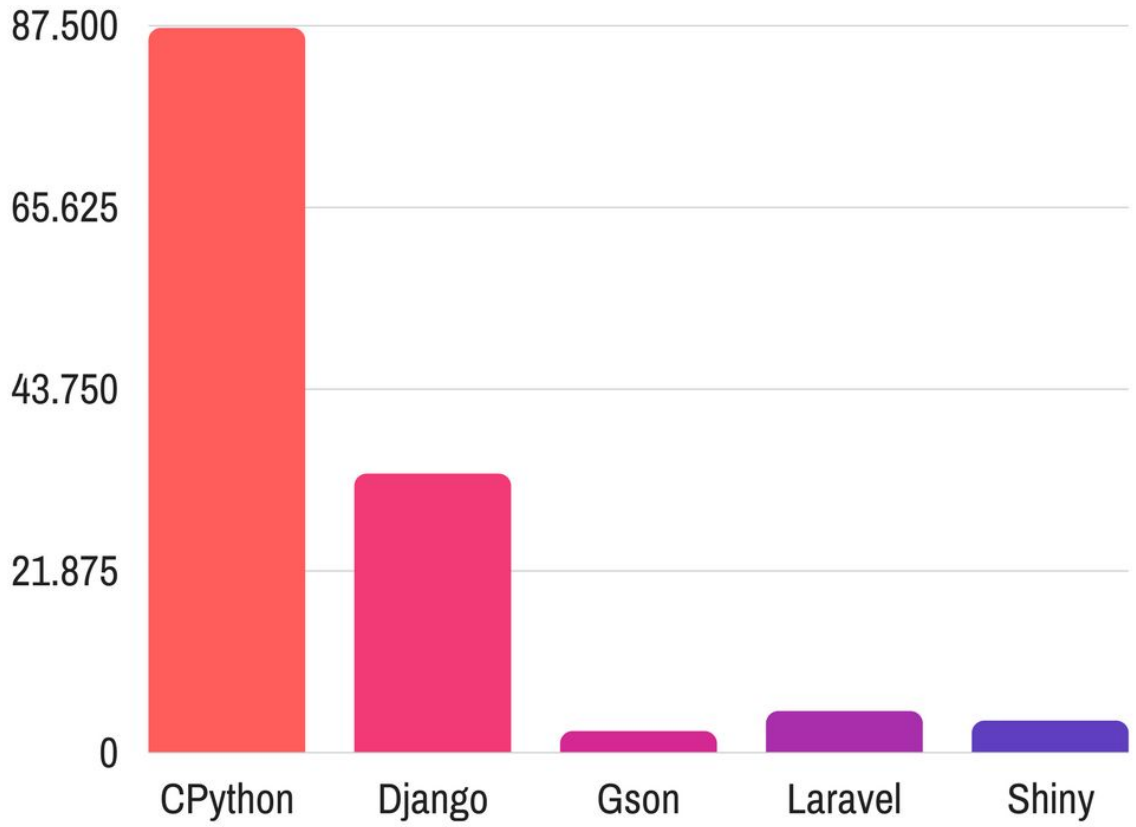
5

diferentes projetos analisados

Tempo de Histórico



Quantidade de transações



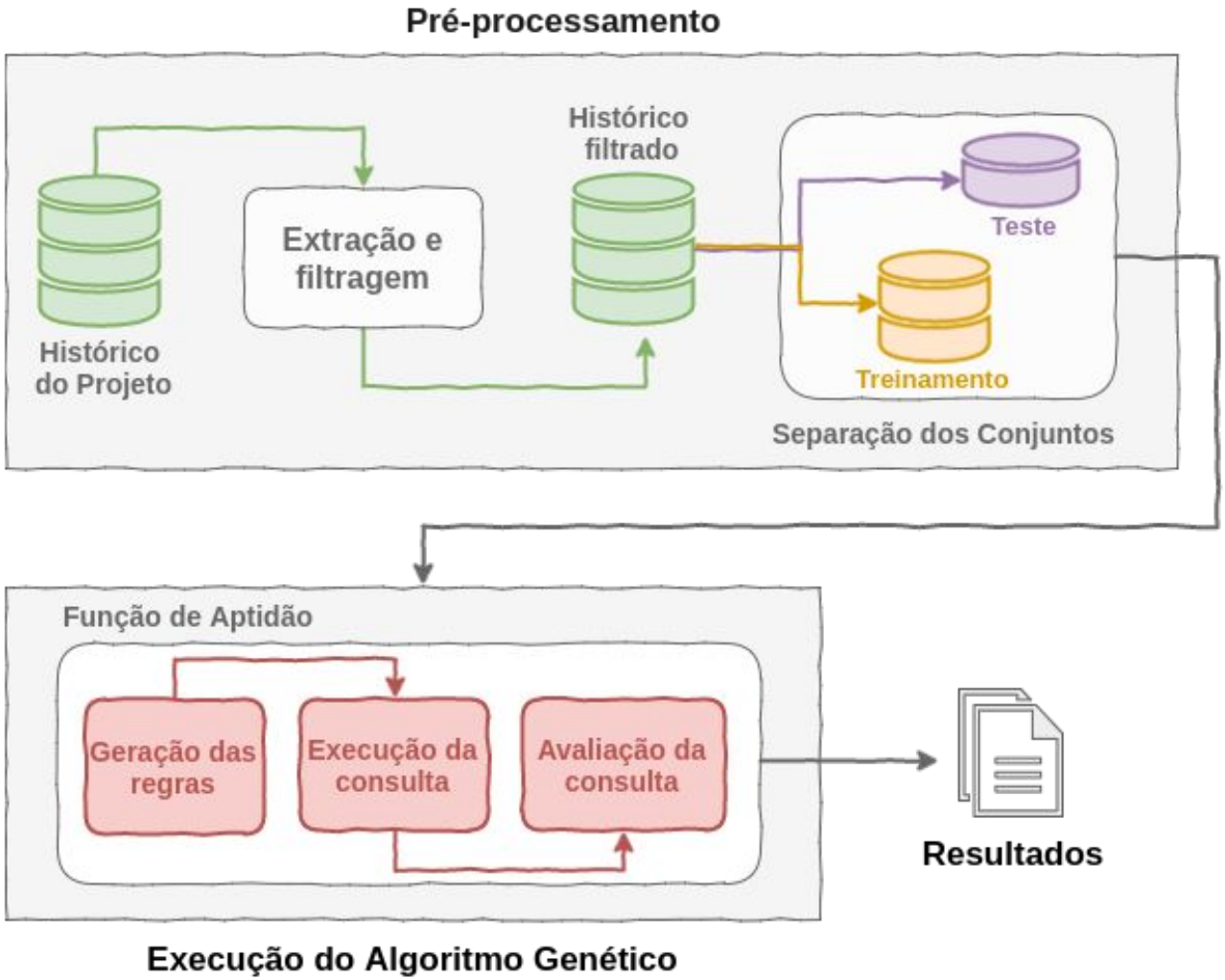
QP1

Como a acurácia do modelo de recomendação de mudanças baseado em Algoritmo Genético se compara àquela do modelo proposto pela Regressão?

QP2

Como a acurácia dos modelos de recomendação de mudança se comporta quando o conjunto de teste aumenta?

Abordagem Proposta

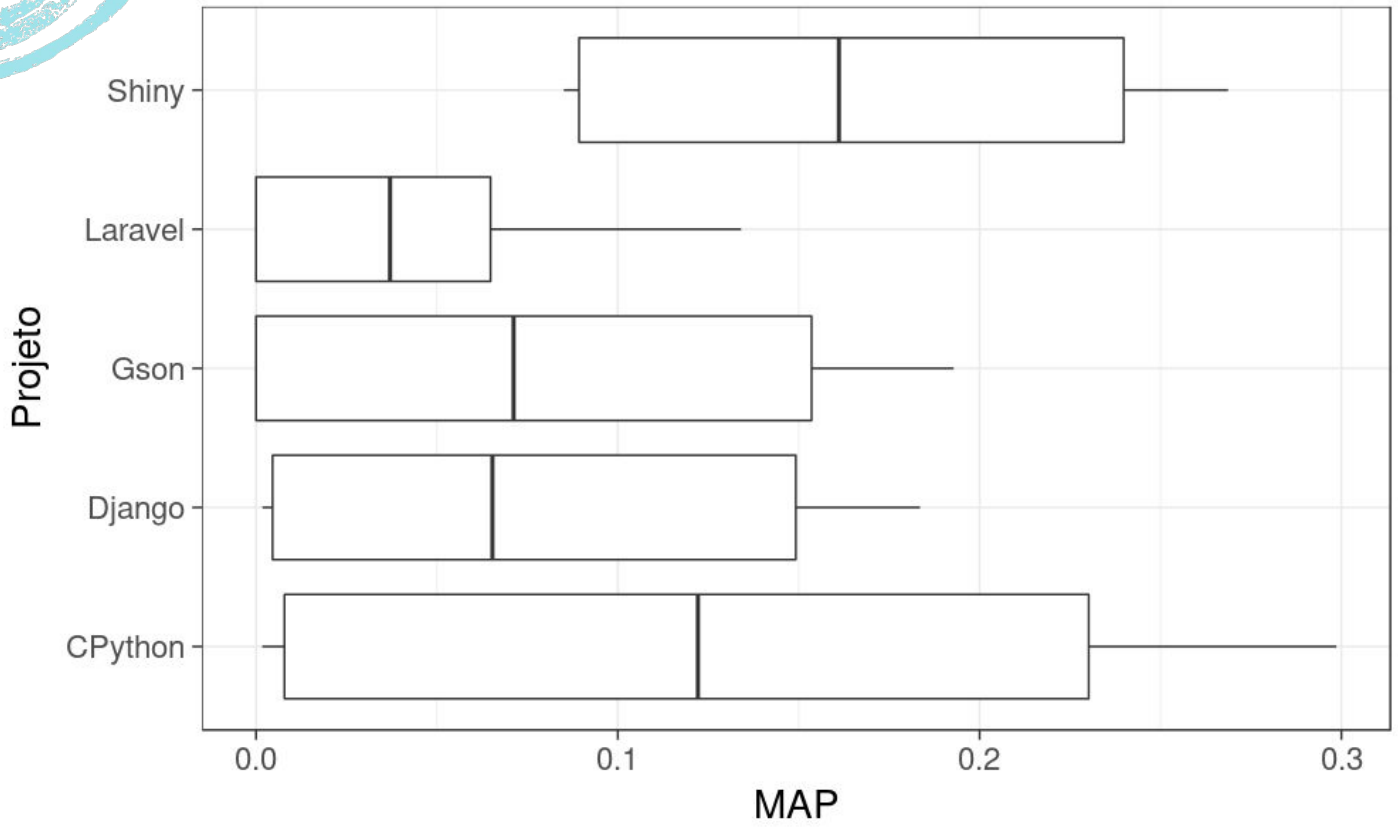


Resultados

QP1

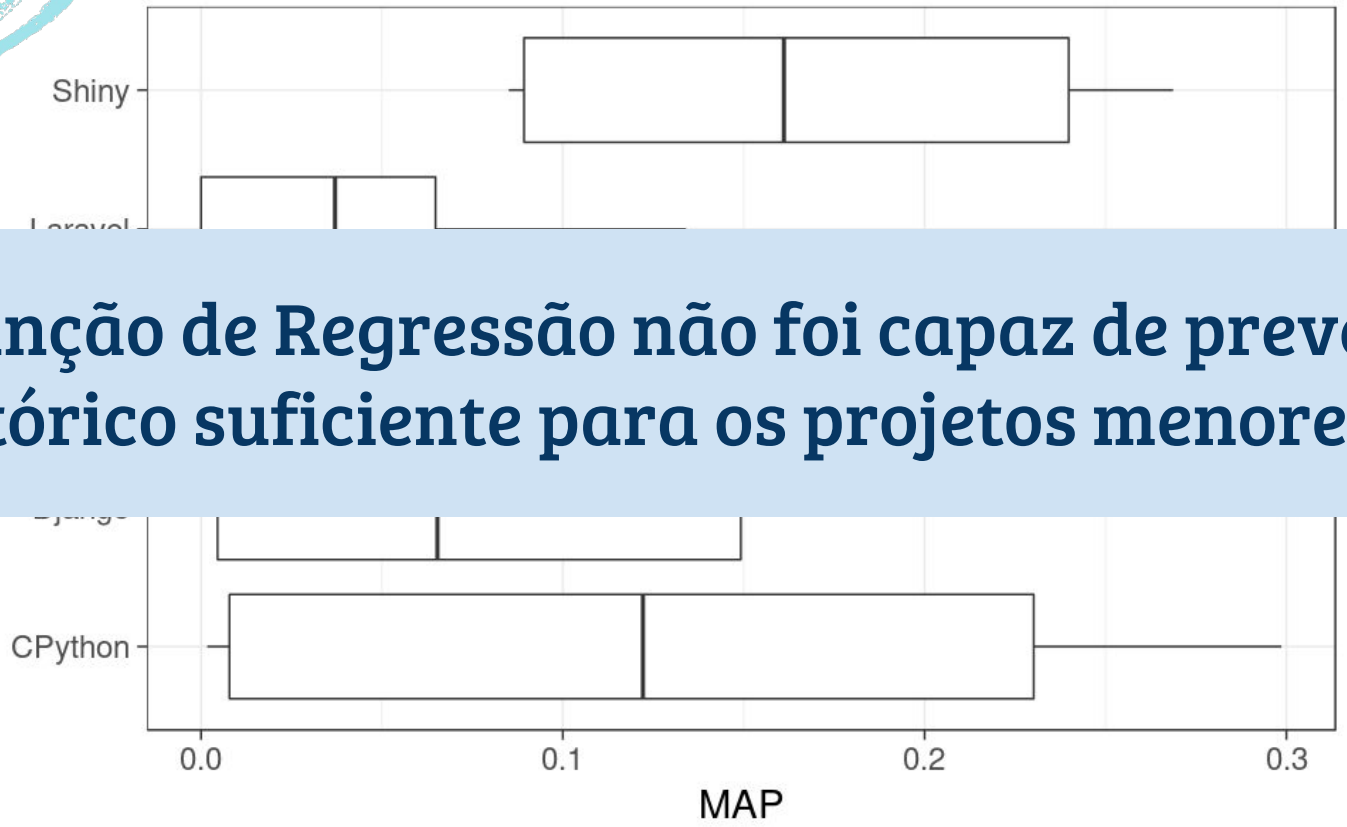
5% das modificações mais recentes de cada projeto para testar os modelos.

QP1



Distribuição do modelo estático para cada projeto.

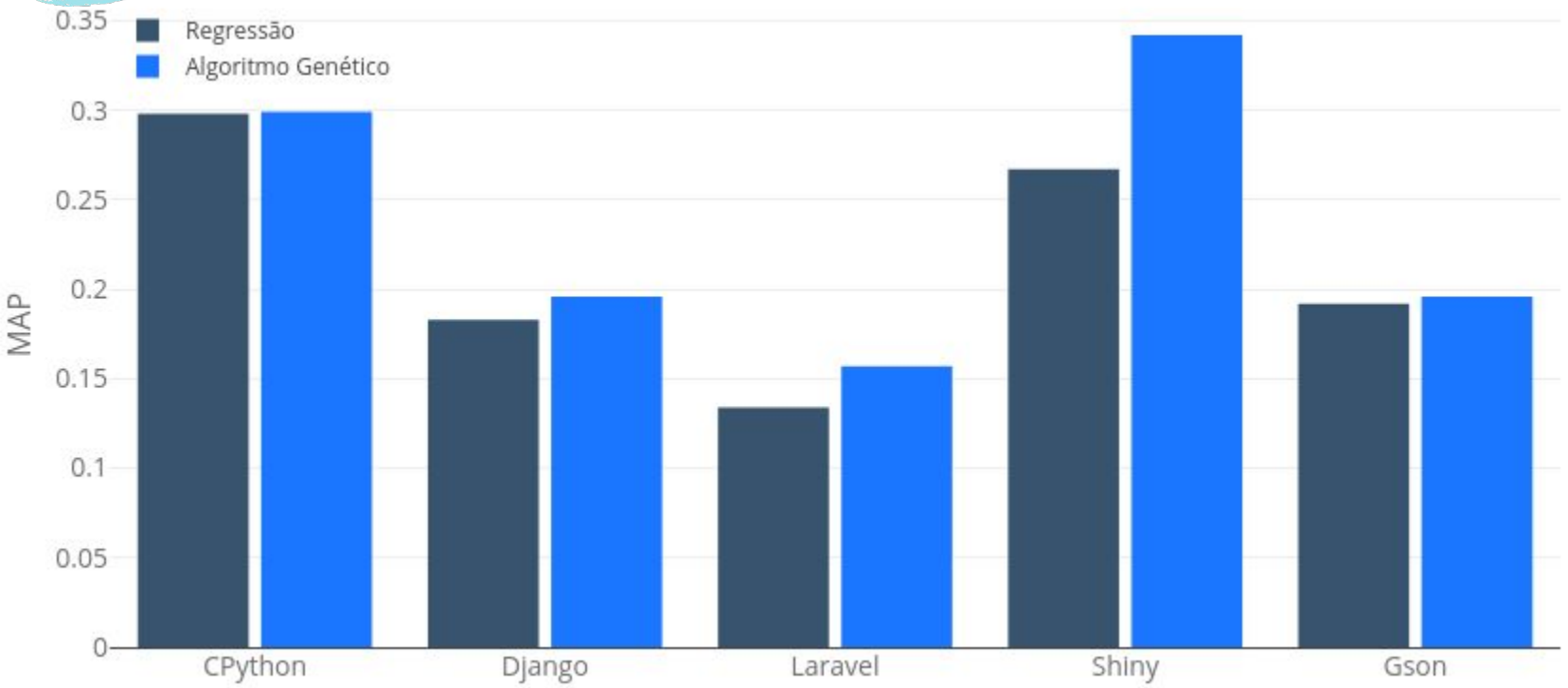
QP1



A função de Regressão não foi capaz de prever histórico suficiente para os projetos menores.

Distribuição do modelo estático para cada projeto.

QP1



Resultado da execução dos modelos com 5% de teste.

QP1

O Algoritmo Genético se adapta ao tamanho do projeto.

- 11000 transações no treinamento

django

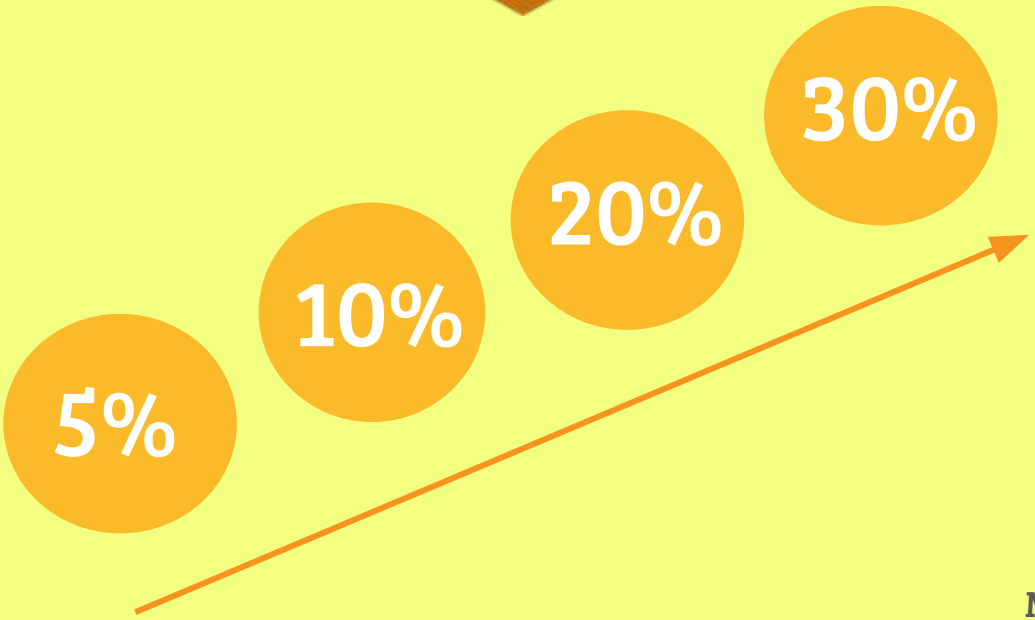
Resultado da execução dos modelos com 5% de teste.

QP1

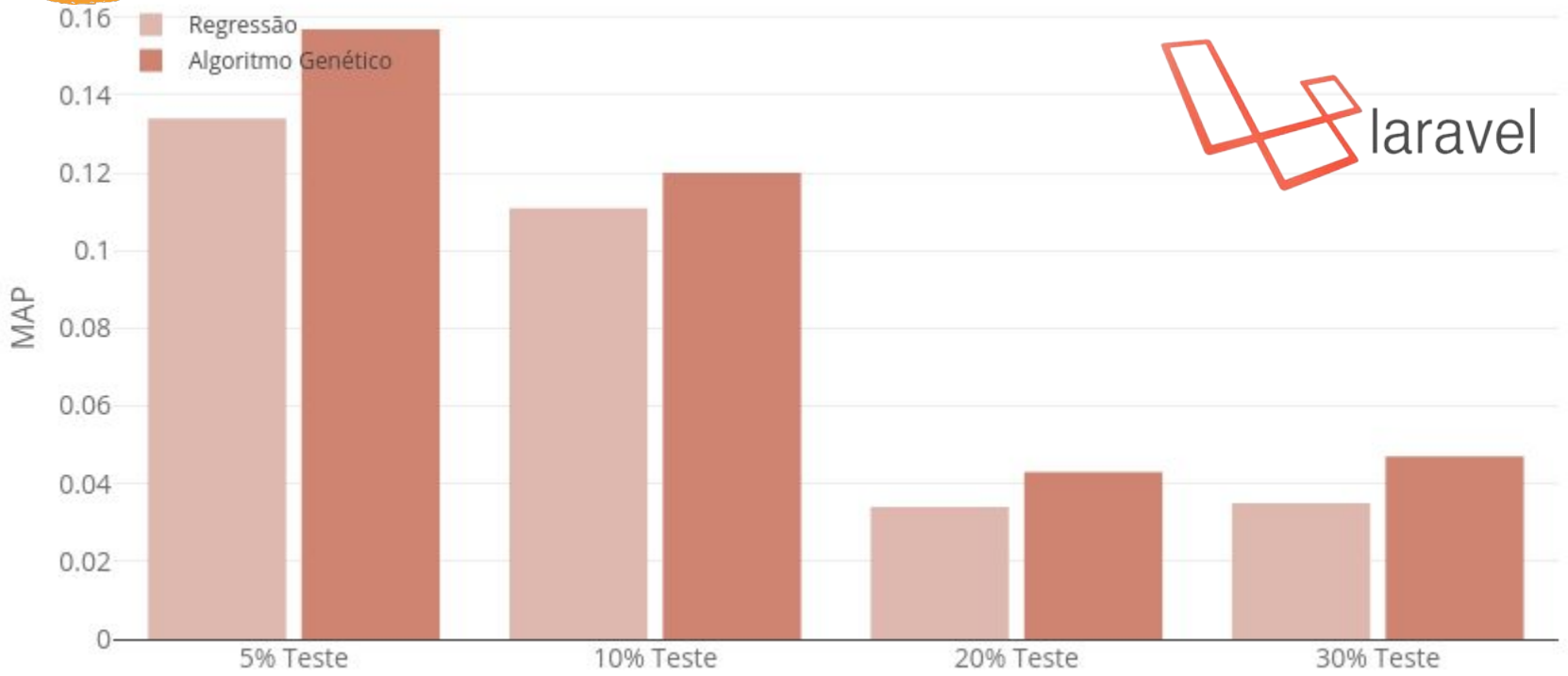
A melhoria na acurácia das recomendações diminui o esforço do desenvolvedor em encontrar arquivos para realizar uma mudança.

QP2

4 tamanhos de teste



QP2



Resultado da execução dos modelos para o projeto Laravel.

QP2

Inserir mudanças no sistema deteriora a estabilidade do modelo de recomendações de mudanças.

Considerações Finais

- O modelo proposto se adapta a diferentes tamanhos de históricos de projetos.
- Dispensa preocupação com as medidas de interesse.

Tweaking Association Rules to Optimize Software Change Recommendations





Mairieli Wessel, Maurício Aniche, Gustavo Oliva, Marco Gerosa, Igor Wiese

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Queen's University

TU Delft

NORTHERN ARIZONA UNIVERSITY