**Instituto de Pesquisas Tecnológicas do Estado de São Paulo**

Marcelo de Rezende Martins

**Uso de redes neurais recorrentes na detecção de padrões de erros cometidos por alunos novatos durante a aprendizagem de programação**

São Paulo

2012

Marcelo de Rezende Martins

Uso de redes neurais recorrentes na detecção de padrões de erros cometidos por alunos novatos durante a aprendizagem de programação

Dissertação de Mestrado apresentada ao Instituto de Pesquisas Tecnológicas do Estado de São Paulo - IPT, como parte dos requisitos para a obtenção do título de Mestre em Engenharia da Computação

Data da aprovação \_\_\_\_/\_\_\_\_\_/\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof.Dr. Marco Aurélio Gerosa (Orientador)

USP – Universidade de São Paulo

Membros da Banca Examinadora:

Prof. Dr. Marco Aurélio Gerosa (Orientador)

USP – Universidade de São Paulo

Prof. Dr. YYYY (Membro)

ZZZZ.

Prof. WWWW (Membro)

ZZZZZZ

Marcelo de Rezende Martins

Uso de redes neurais recorrentes na detecção de padrões de erros cometidos por alunos novatos durante a aprendizagem de programação

Pesquisa apresentada ao Instituto de Pesquisas Tecnológicas do Estado de São Paulo - IPT, como requisito para a obtenção do título de Mestre em Engenharia da Computação.

Área de Concentração: Engenharia de Software

Orientador: Prof. Dr. Marco Aurélio Gerosa

São Paulo

Abril/2018

Ficha Catalográfica  
Elaborada pelo Centro de Informação Tecnológica do   
Instituto de Pesquisas Tecnológicas do Estado de São Paulo – IPT

# 

# **A555p**

PEDIR POR EMAIL PARA MARIA DARCI [mdarci@ipt.br](mailto:mdarci@ipt.br) ou telefone 3767 4139.

05-63 CDU 004.41(043)

**Resumo**

Reumo

Palavras chave: .

LISTA DE FIGURAS

**No table of figures entries found.**

LISTA DE QUADROS

No table of figures entries found.

LISTA DE ABREVIATURAS E SIGLAS

MOOC – Curso online, massivo e aberto (Massive Open Online Course)

SUMÁRIO

[1 Introdução 14](#_Toc507865750)

[1.1 Considerações preliminares 19](#_Toc507865752)

[1.2 Objetivo e escopo 19](#_Toc507865753)

[1.3 Contribuições 19](#_Toc507865754)

[1.4 Método de trabalho 19](#_Toc507865755)

[1.5 Estrutura do trabalho 20](#_Toc507865756)

[1.6 Cronograma 21](#_Toc507865757)

[2 Fundamentação teórica 21](#_Toc507865758)

[2.1 Sistemas legados e a evolução do ciclo de vida do software 21](#_Toc507865761)

[2.2 A engenharia reversa aplicada a sistemas legados 22](#_Toc507865762)

[2.3 A engenharia de requisitos e a UML 22](#_Toc507865763)

[2.4 Pesquisas recentes sobre a extração de requisitos (conhecimento) de sistemas legados 22](#_Toc507865764)

[3 Metodologia de pesquisa 22](#_Toc507865765)

[3.1 Sobre a metodologia de pesquisa Pesquisa-ação 22](#_Toc507865767)

[3.2 Ciclos de condução da pesquisa-ação 22](#_Toc507865768)

[3.3 Caracterização da condução da pesquisa-ação 22](#_Toc507865769)

[4 Processo proposto 22](#_Toc507865770)

[5 Execução do 1º ciclo da pesquisa-ação 22](#_Toc507865771)

[5.1 Coleta e validação dos dados 22](#_Toc507865772)

[5.2 Análise dos dados e planejamento das ações 22](#_Toc507865773)

[5.3 Implementação 22](#_Toc507865774)

[5.4 Avaliação 22](#_Toc507865775)

[6 Execução do 2º ciclo da pesquisa-ação 23](#_Toc507865776)

[6.1 Coleta e validação dos dados 23](#_Toc507865777)

[6.2 Análise dos dados e planejamento das ações 23](#_Toc507865778)

[6.3 Implementação 23](#_Toc507865779)

[6.4 Avaliação 23](#_Toc507865780)

[7 Execução do 3º ciclo da pesquisa-ação 24](#_Toc507865781)

[7.1 Coleta e validação dos dados 24](#_Toc507865782)

[7.2 Análise dos dados e planejamento das ações 24](#_Toc507865783)

[7.3 Implementação 24](#_Toc507865784)

[7.4 Avaliação 24](#_Toc507865785)

[8 Análise dos resultados 25](#_Toc507865786)

[9 Conclusão e sugestões 26](#_Toc507865787)

# Introdução

­­­

Em 1936, Alan Turing publicou um artigo chamado *On Computable Numbers, with an application to the Entscheidungsproblem* (TURING, 1936, p. 230—265) no qual descreveu uma máquina teórica capaz de resolver qualquer problema que pudesse ser descrito através de instruções pré-definidas numa fita de papel. Este artigo tornaria-se a base para a computação e invenção dos computadores. Durante a segunda guerra, Turing construiu uma máquina capaz de quebrar a maioria do código Morse secreto usado nas comunicações pela força naval alemã. Segundo alguns especialistas, a guerra teria durado mais tempo se não fosse a máquina inventada por Alan Turing para quebrar a criptografia das mensagens. A humanidade deu um grande salto graças ao avanço dos computadores desde o fim da segunda guerra até os dias de hoje. Este salto tecnológico permitiu o avanço em diversas áreas como saúde, telecomunicações, aviação, engenharia, financeira etc.

Com o aumento do poder de processamento computacional dobrando a cada 18 meses, Lei de Moore, novos avanços em tecnologia surgem como a Internet da Coisas (IoT), cidades inteligentes (Smart Cities), Indústria 4.0. Todas tem um ponto em comum: computação úbiqua, uso de inteligência artificial e big data. E um outro ponto é a automação. Governos em todo o mundo estão preocupados com o advento da inteligência artificial e automação, pois muitos especialistas dizem que haverá um desemprego em massa. A maioria dos empregos que tem uma tarefa repetitiva, que não exige criatividade e interação social, serão automatizados. Em sua maioria, são empregos exercidos por pessoas com baixa qualificação (SMIT, 2016, p. 26). Segundo um relatório da OCDE, este desemprego em massa não ocorrerá tão rapidamente, pois o computador consegue fazer somente tarefas bem definidas e a maioria dos trabalhos não tem uma definição clara das tarefas a serem feitas (ARNTZ. Melanie; GREGORY. Terry; ZIERAHN. Ulrich., 2016, p. 22).

Em um item todos os governos concordam: há uma necessidade em preparar a nova geração para esta nova era. Hoje há um defícit de programadores em todo o mundo. E com o advento do IoT, Indústria 4.0, a demanda por desenvolvedores e programadores irá aumentar ainda mais. Governos de todo o mundo estão empenhados em um projeto de ensino de programação no currículo básico das escolas para atender a esta demanda. Alan Perlis disse em 1960 que programação deveria ser ensinada a todos. O modo de pensar "algoritmicamente" é útil para diversos problemas em nosso cotidiano. E com a popularização e a presença em massa dos computadores em nosso cotidiano, a importância de saber programar aumenta. Governos e empresas multinacionais juntaram forças para ensinar programação à populacao através de iniciativas como o code.org, udacity, scratch etc.

Mas aprender a programar não é uma tarefa fácil. Diversos estudos que apontam para a alta taxa de evasão dos alunos dos cursos de Ciência da Computação (WATSON, Christopher; LI, Frederick W.B., 2014, p. 39—44). E outros estudos apontam a alta taxa de reprovação por parte dos alunos nos cursos de introdução à programação (WATSON, Christopher; LI, Frederick W.B., 2014, p. 39—44; BOSSE, Yorah; GEROSA, Marco Aurélio, 2015, p. 1389--1397). Segundo um estudo preliminar feito por Bosse (2015), a taxa de reprovação nos cursos introdutórios de programação ultrapassam a taxa de 30% somente na Universidade de São Paulo. A reprovação no curso de introdução à programação é um dos fatores que contribuem para a evasão do curso. Segundo os dados do censo do Ensino Superior divulgado pelo MEC em 2016, a taxa de evasão média para cursos de bacheralado é em torno de 22%, enquanto para os cursos de bacharelado em Computação, a taxa de evasão supera 28%.

Segundo Kalelioglu (2016), o pensamento computacional é um processo para resolução de problemas. Inicia-se com a identificação do problema através da abstração e decomposição. Para entender melhor e solucionar o problema, é necessário coletar, representar e analisar os dados. No processo de coleta e análise, reconhecimento de padrões, conceptualização e representação dos dados devem ser levados em consideração. Para obter soluções mais precisas, processos cognitivos como lógica matemática, algortimos podem ser empregados. Para implementar a solução, modelagem, automação e simulação devem ser utilizadas para verificar a eficácia da solução. Por último, aplicar a solução obtida para diferentes tipos de problemas deve ser realizado para generalizar a solução. Segundo Kalelioglu (2016) o pensamento computacional envolve diversos modelos mentais como abstração, resolução de problemas, reconhecimento de padrões, decomposição, generalização etc.

Além da necessidade de desenvolver o pensamento computacional, é importante saber quais são as principais dificuldades encontradas pelos alunos. Segundo Jenkins (2002), há diversos fatores que contribuem para a dificuldade do aprendizado. Programar exige múltiplas habilidades, é uma tarefa de vários processos, é novidade para maioria dos alunos ingressantes, depende da motivação do aluno, interese, ritmo da aula etc.

Há estudos que tentam identificar quais são os erros mais comuns cometidos pelos alunos na aprendizagem de programação (HRISTOVA, 2006, p.153 – 156; CACEFFO, Ricardo et al., 2016, p. 364--369). A partir dos erros comumente cometidos pelos alunos, é possível elaborar materiais e alterar o ritmo da aula para um melhor aprendizado do estudante. Porém, a maioria dos estudos adotam a abordagem de entrevista com alunos e professores, catalogação dos erros verificados e posterior análise para validá-los.

Com o aumento do poder computacional, crescente uso de ferramentas de aprendizagem de máquina e a grande oferta de repositórios de código-fontes como Github, por exemplo, nasce o termo Big Code (BHATIA, Sahil; SINGH, Rishabh, 2016). Big Code refere-se ao conhecimento adquirido a partir da análise dos repositórios de código fonte através de ferramentas de aprendizagem de máquina. Construir ferramentas que sejam capazes de aprender a partir dos repositórios de código fonte é um novo desafio. Com o conhecimento que podemos adquirir a partir da análise dos repositórios, podemos construir ferramentas para analisar a arquitetura de um projeto, verificar se o código fonte ou o nome do método está de acordo com a convenção, detectar erros de execução de forma antecipada.

Com as ferramentas de aprendizagem de máquina, é possível também identificar quais são os erros mais comuns cometidos pelos alunos. Esta técnica é utilizada para dar uma resposta automática para o aluno, permitindo que ele receba uma avaliação mais rápida que a dada pelo professor ou monitor. Segundo um estudo feito por Bhatia (2016), o uso de aprendizagem de máquina permitiu criar uma ferramenta para detectar erros de sintaxe e corrigi-los. Um texto publicado no site da Microsoft, mostrou como o uso do algoritmo de *Deep Learning* foi eficaz para identificar as correções necessárias no código fonte submetido pelo aluno.

Esta dissertação tentará responder a seguinte pergunta:

1. Como as redes neurais recorrentes podem ser utilizadas para identificar padrões de erros cometidos por alunos previamente analisados por monitores e instrutores?

## Considerações preliminares

O uso de algoritmo de aprendizagem para correção automática de exercícios não é novidade. Porém, boa parte dos estudos foram feitos para plataformas MOOC, no qual a maioria dos exercícios são avaliados por meio de uma base de testes e a nota representa a porcentagem de testes que o exercício passou. Neste estudo, identificaremos os padrões de erros por meio do uso do algoritmo de redes neurais aplicado num conjunto de exercícios previamente avaliados por monitores e professores. A avaliação feita por monitores e instrutores leva em consideração se o aluno entendeu conceitos como instruções de repetição, instruções de decisão, chamada de métodos, recursão, uso de vetores, ponteiros, passagem de parâmetros etc. Este tipo de avaliação é diferente das avaliações baseadas em porcentagem de testes.

## Objetivo e escopo

O objetivo da pesquisa é verificar o uso das redes neurais recorrentes como técnica para detecção de padrões de erros cometidos por alunos previamente analisados por monitores e instrutores. O escopo deste estudo será limitado inicialmente a uma classe de problema. Normalmente, os exercícios feitos pelos alunos visam reforçar um conceito visto em sala de aula. Por exemplo, o propósito de um programa para verificar se um ano é bissexto ou não é reforçar os conceitos de instrução de decisão, operadores booleanos. Já um programa para exibir um calendário reforça os conceitos de instrução de decisão, instrução de repetição, operadores booleanos, formatação da saída, porém não exige o conhecimento de matrizes, por exemplo.

## Contribuições

Conforme mencionado anteriormente, o contexto do uso de redes neurais recorrentes na detecção de padrões de erros será a principal contribuição deste trabalho. Contribuições secundárias serão o levantamento de padrões de erros cometidos por alunos novatos durante a aprendizagem de programação, definição de classe de problemas e exercícios, definição dos atributos que serão levados em consideração para o treinamento e análise e verificação do uso de redes neurais recorrentes para detecção de padrões de erros no contexto deste trabalho.

## Método de trabalho

## Estrutura do trabalho

## Cronograma

# Fundamentação teórica



## Sistemas legados e a evolução do ciclo de vida do software

## A engenharia reversa aplicada a sistemas legados

## A engenharia de requisitos e a UML

## Pesquisas recentes sobre a extração de requisitos (conhecimento) de sistemas legados

# Metodologia de pesquisa



## Sobre a metodologia de pesquisa Pesquisa-ação

## Ciclos de condução da pesquisa-ação

## Caracterização da condução da pesquisa-ação

# Processo proposto

# Execução do 1º ciclo da pesquisa-ação

## Coleta e validação dos dados

## Análise dos dados e planejamento das ações

## Implementação

## Avaliação

# Execução do 2º ciclo da pesquisa-ação

## Coleta e validação dos dados

## Análise dos dados e planejamento das ações

## Implementação

## Avaliação

# Execução do 3º ciclo da pesquisa-ação

## Coleta e validação dos dados

## Análise dos dados e planejamento das ações

## Implementação

## Avaliação

# Análise dos resultados

# Conclusão e sugestões