Instituto de Informática - Universidade Federal de Goiás

Análise de estilo de programação com Rede Neural Recorrente para aprovação automática de Pull Request

Lucas Roque, Altino Dantas e Celso G. Camilo-Junior
{lucasroque | altinoneto | celsocamilo}@inf.ufg.br

2018

I4Soft
Intelligence for Software

UFG

INSTITUTO DE INFORMÁTICA
UFG

CAPES

FAPEG
Fundação de Amparo à Pesquisa do Estado de Goiás

Agenda

1. Introdução
2. Proposta
3. Experimentos
4. Resultados
5. Conclusão

Introdução

Propriedades estruturais que afetam a qualidade do software [Dromey 1995]:

Corretude

Estilo

```
namespace MyApplication.MyClass
{
    public class Class1
    {
        public List<int> MyProperty1 { get; set; }

        public void DoSomeWork()
        {
            List<int> items = this.MyProperty1;

            int total = 0;
            string allItems = string.Empty;

            foreach (var item in items)
            {
                total += item;
                allItems += item.ToString() + "|";
            }
        }
    }
}
```

Introdução

- Controle de versão baseado em *Pull Request*:
  - Mercurial, Git, Baazar e etc;
  - *Clone, branch, pull request*.
- Problemas:
  - Dificuldade na definição de regras;
  - Linguagens específicas;
  - Necessidade de controle pela evolução dos sistemas de versionamento.

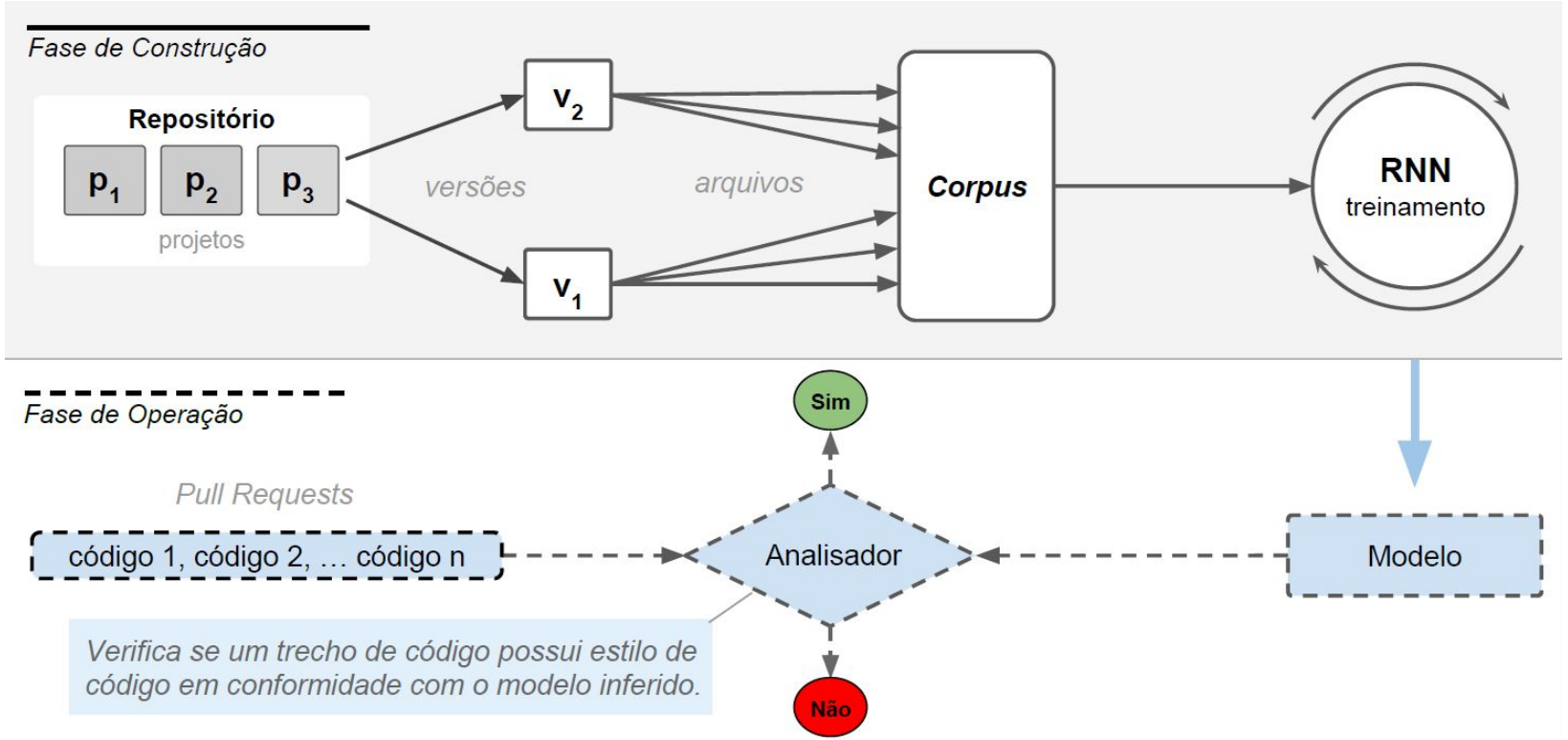
Introdução

- Propõe-se um método que aprende o estilo de codificação de um projeto, e garante que qualquer novo código inserido no projeto seja checado quanto à conformidade em relação ao estilo aprendido.
- Cenários de aplicação:
  - Projetos em desenvolvimento;
  - Projeto ainda em fase inicial.

Trabalhos Relacionados

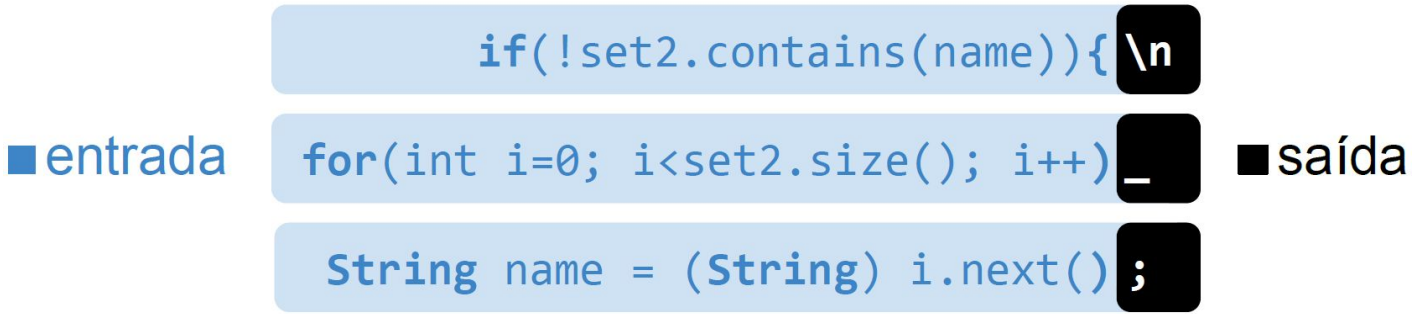
- Análise de autoría [Man and Cook 1989];
- Análise de estilo em Pascal [Lake and Cook 1990];
- Auxílio para estudantes C++ [Ala-Mutka *et al*. 1995];
- Taxinomia do estilo de programação [Dromey 1995];

Proposta | visão geral



Proposta | Exemplo

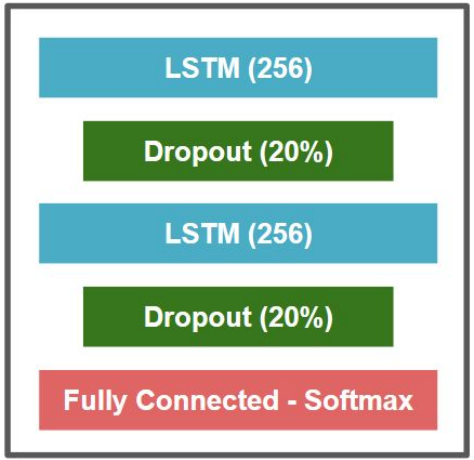
Exemplo de operação do modelo recorrente treinado, com entrada e saída.



Experimentos

- Projeto em C# com mais de 10 mil linhas;
- Caracteres alvo: “{”, “}”, “(”, “)” e “;”;
- Análise de frequência;
- Análise de estilo de programação;
- Arquitetura rede neural recorrente:

Arquitetura recorrente



Resultados | Análise de frequência

Estrutura	{	}	(	)	;
if	\n, “ ” , \t	\n, “ ”,)	alpha,), (	\n, “ ”, ;	
else	\n, “ ”	\n, “ ”, ;			
for	\n, “ ”	\n,), “ ”	alpha,), “ ”	;,], \n	
while	\n	\n	alpha,), (	\n,), ;	
try	\n, “ ”	\n, “ ”,)			
catch	\n	\n, “ ”	alpha, “ ”,)	\n, ;, “ ”	
função			alpha,), \n	\n,], “ ”	\n, “ ”, \t
atribuição			alpha,), “ ”	;,),]	\n, “ ”, \t

Resultados | Análise de estilo de programação

Estrutura	{	}	(	)	;
if	100%	100%	91%	86%	
else	100%	100%			
for	100%	100%	100%	84%	
while	100%	100%	100%	100%	
try	100%	100%			
catch	100%	100%	100%	100%	
função			80%	80%	92%
atribuição			94%	93%	100%

Conclusões

- Propôs uma nova abordagem para análise de estilo de programação, que mostrou-se eficiente para o aprendizado do estilo de programação presente no projeto alvo;

- Trabalhos Futuros:
  - Melhorar análise da precisão do modelo;
  - Estender para outras linguagens de programação, e aplicar o estilo aprendido a projetos distintos.

Perguntas?

Obrigado!