PortuCol: uma pseudolinguagem inspirada em C ANSI para o Ensino de Lógica de Programação e Algoritmos

Lucas Lemos Barbosa, Christian Marlon Souza Couto, Ricardo Terra

Departamento de Ciência da Computação,
Universidade Federal de Lavras (UFLA), Brasil

lbarbosa@computacao.ufla.br,christianmsc@sistemas.ufla.br,terra@dcc.ufla.br

Resumo. *Linguagens de programação são intrinsecamente complexas, devido à variabilidade, generalidade e completude de suas construções. Isso justifica a adoção de pseudolinguagens de alto nível no ensino de Lógica de Programação e Algoritmos. Embora C ANSI e Java sejam as linguagens de programação mais utilizadas, Portugol – uma pseudolinguagem com instruções em português inspirada na linguagem obsoleta Pascal – é ainda largamente adotada por IES. Diante desse cenário, este artigo propõe PortuCol, uma pseudolinguagem de programação com instruções em português assim como Portugol, porém inspirada em C ANSI. Este artigo também descreve PortuCol2C, uma ferramenta que traduz código escrito em PortuCol para C ANSI e conduz estudos empíricos que demonstram estatisticamente que (i) PortuCol é 27% mais similar a C ANSI do que o Portugol e (ii) PortuCol é para C ANSI o que Portugol é para Pascal.*

Abstract. *Programming languages are intrinsically complex due to the variability, generality, and completeness of their constructions. This justifies the adoption of high-level pseudocode languages when teaching Introduction to Algorithms. Although ANSI C and Java are the most used programming languages, Portugol—a pseudocode language with instructions in Portuguese inspired by the outdated Pascal language—is still widely adopted by Brazilian universities. In view of such circumstances, this paper proposes PortuCol, a pseudocode programming language with instructions in Portuguese as Portugol, although inspired by ANSI C. It also describes PortuCol2C, a tool that translates PortuCol to ANSI C, and conducts empirical studies that statistically demonstrate that (i) PortuCol is 27% more similar to ANSI C than Portugol and (ii) PortuCol is for C ANSI what Portugol is for Pascal.*

1. Introdução

No início do ensino superior, é comum que estudantes da área de Ciência da Computação e afins tenham dificuldade no raciocínio lógico de criação de soluções, uma vez que, no ensino médio, esses estudantes estavam acostumados a apenas aplicar fórmulas previamente existentes na solução de problemas. Um estudo atesta essa dificuldade reportando que, em média, mais de 50% dos alunos das disciplinas de introdução à programação são reprovados a cada semestre [2].

Em virtude de que linguagens de programação são intrinsecamente complexas – devido à variabilidade, generalidade e completude de suas construções – docentes de IES

usualmente adotam pseudolinguagens de alto nível nas disciplinas de introdução à Lógica de Programação e Algoritmos. Pseudolinguagens proporcionam ao aluno um maior espaço para o desenvolvimento de seu raciocínio lógico relacionado à resolução de problemas, sem preocupações relativas ao uso correto de uma linguagem de programação.

Portugol é, hoje no Brasil, a pseudolinguagem mais adotada por universidades com esse propósito [9, 11], a qual conta com diversas ferramentas de apoio ao ensino [3, 5, 9, 8, 4]. No entanto, Portugol é o produto de uma simbiose do português e as linguagens de programação ALGOL e Pascal [5], ambas obsoletas [10]. Pascal, por exemplo, sequer está classificada entre as 50 linguagens de programação mais utilizadas no desenvolvimento de sistemas de software [10]. Assim, devido à origem sintática de Portugol, estudantes *não* terão uma transição natural ao aprender linguagens de programação do estado-da-prática como C ANSI e Java.

Como uma alternativa ao Portugol, este artigo propõe PortuCol, uma pseudolinguagem de programação com instruções em português assim como Portugol, porém inspirada em C ANSI. Estudos empíricos atestam que (i) PortuCol é 27% mais similar a C ANSI do que o Portugol, o que proporciona uma transição mais natural para o estudante ao aprender linguagens do estado-da-prática e (ii) PortuCol é para C ANSI o que Portugol é para Pascal, o que retrata apenas uma adaptação da pseudolinguagem às linguagens do estado-da-prática. Além disso, este artigo descreve `PortuCol2C`, uma ferramenta que traduz código PortuCol para C ANSI, o que possibilita aos estudantes se familiarizem com a linguagem C ANSI através da observação do seu código PortuCol traduzido.

Este artigo está organizado como a seguir. A Seção 2 apresenta Portugol, a pseudolinguagem mais adotada em disciplinas de programação do Brasil. A Seção 3 descreve o PortuCol, sua sintaxe e a ferramenta `PortuCol2C`. A Seção 4 reporta os resultados da avaliação de similaridade textual entre algoritmos implementados em PortuCol, C ANSI, Portugol e Pascal. Por fim, a Seção 5 conclui e propõe trabalhos futuros.

2. Portugol

O Portugol é a pseudolinguagem mais adotada em classes introdutórias à programação em IES no Brasil [9, 11]. Mais importante, a maioria dos livros utilizados como referência básica nas disciplinas de introdução à programação em universidades brasileiras possui explicações usando alguma notação de Portugol [9, 11]. Em suma, o objetivo de Portugol é facilitar o aprendizado de Lógica de Programação e Algoritmos para alunos não habituados com programação.

Portugol é o produto de uma simbiose do português e as linguagens de programação ALGOL e Pascal [5], ambas obsoletas [10]. Seu propósito – o qual foi atingido – é ser uma pseudolinguagem simples e com instruções em português para favorecer estudantes não familiares com a língua inglesa. Ao utilizar Portugol, o aluno não precisa se preocupar com a sintaxe e comandos em inglês que uma linguagem de programação mais complexa possui, o que foca sua concentração no *como* resolver um problema proposto.

Apesar de trazer benefícios inquestionáveis no aprendizado de Lógica de Programação e Algoritmos, o Portugol não provê, por definição, uma forma de executar o pseudocódigo escrito para que alunos possam testar seus algoritmos, o que se torna indispensável à medida que os algoritmos se tornam mais complexos e passam a requerer códigos

mais elaborados. Esse problema ocorre pois o Portugol tem diversas variações em sintaxe e em comandos dependendo do contexto de utilização. Isso faz com que as ferramentas não sejam padronizadas, i.e., suportando sintaxes e instruções diferentes [3, 5, 9, 8, 4].

Dentre as ferramentas existentes que apoiam o ensino utilizando Portugol podemos destacar: CIFluxProg [3] disponibiliza um ambiente visual com fluxogramas, além de um ambiente textual com uma variação de Portugol; WEB-UNERJOL [5] provê um ambiente *web* que permite o desenvolvimento e execução de algoritmos em Portugol através de um navegador; Portugol Studio [9] é um ambiente de desenvolvimento integrado (IDE) para uma variante do Portugol que utiliza o compilador Portugol Núcleo; Portugol IDE [8] provê um ambiente que possibilita desenvolver algoritmos em Portugol através do uso de fluxogramas como uma linguagem gráfica, além do tradicional modo textual; e, por fim, VisuAlg[4] que adota uma variante de Portugol bem parecida com Pascal, porém sem pontos-e-vírgulas para separar comandos. Os Algoritmos 1 e 2 ilustram tal similaridade apresentando um mesmo algoritmo escrito em Portugol e em Pascal, respectivamente.

Algoritmo 1. Portugol

```
algoritmo "helloWorld"
var
    nome : caractere
inicio
    escreva("Digite seu nome: ")
    leia(nome)
    escreva("Ola ", nome, "!")
fimalgoritmo
```

Algoritmo 2. Pascal

```
Program helloWorld;
var
    nome : string;
Begin
    write('Digite seu nome: ');
    readln(nome);
    write('Ola ', nome, '!');
End.
```

3. PortuCol

Este artigo propõe PortuCol, uma pseudolinguagem de programação com instruções em português assim como Portugol, porém inspirada em C ANSI. O objetivo é adaptar o ensino de Lógica de Programação e Algoritmos através de uma pseudolinguagem cuja sintaxe seja inspirada em uma linguagem de programação do estado-da-prática (i.e., amplamente adotada por fábricas de software).

A Figura 1 ilustra o contexto de utilização de PortuCol. Estudantes desenvolvem o código em PortuCol e, automaticamente, a ferramenta `PortuCol2C` traduz PortuCol para C ANSI e executa o código correspondente. Isso possibilita que estudantes (i) executem seu pseudocódigo; (ii) se familiarizem com a linguagem C ANSI através da observação do seu código PortuCol traduzido; e (iii) tenham uma transição mais natural ao futuramente aprender linguagens do estado-da-prática como C ANSI e Java.

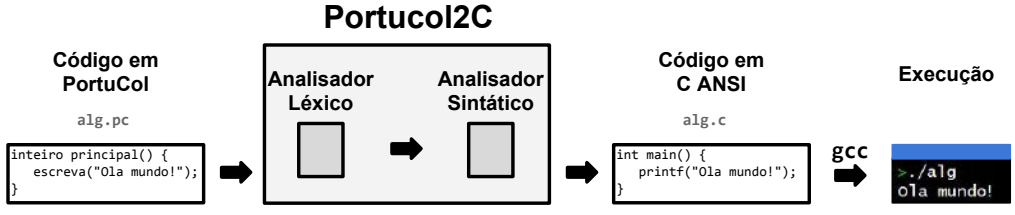

Figura 1. Utilizando PortuCol

3.1. Sintaxe

A definição da sintaxe de PortuCol considerou didática, facilidade de aprendizagem e similaridade com C ANSI de forma que futuramente a transição para essa linguagem de programação seja mais natural para o aluno.

A Tabela 1 apresenta a equivalência de instruções PortuCol, C e Portugol. PortuCol possui instruções e palavras reservadas em português similarmente à Portugol e uma sintaxe similar à C ANSI. Em PortuCol, assim como em C ANSI, é necessário delimitar o escopo do código com os símbolos de abre e fecha chaves. Os operadores aritméticos e lógicos são como em C ANSI; exceções (e.g., operadores lógicos `e` e `ou`) ocorreram em virtude do caráter didático que a pseudolinguagem proposta tem como objetivo.

As estruturas básicas de desvio e repetição seguem, em sua maioria, a estrutura da linguagem de programação C ANSI. Contudo, uma das estruturas que foi modificada com o intuito de apresentar um conceito de forma didática foi a estrutura de repetição `para` (`for`) que em PortuCol tem condições de parada fixas (menor ou igual, e maior ou igual). O objetivo dessa modificação foi de habituar o aluno a entender arranjos de tamanho n como estruturas que são acessadas a partir da posição 0 até a posição $n-1$, o que acontece na maioria das linguagens de programação.

Usando o mesmo algoritmo descrito na Seção 2, os Algoritmos 3 e 4 apresentam dois exemplos do mesmo código em PortuCol e C ANSI, respectivamente.

Algoritmo 3. PortuCol

```
1 inteiro principal(){
2     texto nome;
3     escreva("Digite seu nome: ");
4     leia("%texto", nome);
5     escreva("Ola %texto!", nome);
6 }
```

Algoritmo 4. C ANSI

```
1 int main(){
2     char nome[20];
3     printf("Digite seu nome: ");
4     scanf("%s", nome);
5     printf("Ola %s!", nome);
6 }
```

3.2. Ferramenta `PortuCol2C`

Para que os estudantes possam executar seus códigos escritos em PortuCol, foi criado um protótipo de uma ferramenta denominada `PortuCol2C`. Essa ferramenta realiza a tradução dos códigos em PortuCol para a linguagem C ANSI e, por conseguinte, já compila e gera o arquivo executável correspondente. Dessa forma, `PortuCol2C` permite que estudantes (i) "executem" seu código PortuCol e (ii) observem qual seria o código C ANSI correspondente, o que já os familiariza com linguagens de programação reais.

Na atual versão da ferramenta, a execução de `PortuCol2C` é por linha de comando, conforme pode ser observado na Figura 2. Dado um código em PortuCol (`alg.pc`), a execução ferramenta (`portucol2c alg.pc`) gera o código C ANSI correspondente (`alg.c`) e o arquivo executável (`alg`).

O tradutor foi implementado usando *Lex* e *Yacc*,[1] que são ferramentas designadas para o desenvolvimento de compiladores e interpretadores [6]. O *Lex* realiza a análise léxica que consiste na leitura dos caracteres de entrada e produção de uma sequência de *tokens*, enquanto o *Yacc* executa a análise sintática que consiste na obtenção de uma cadeia

[1] http://dinosaur.compilertools.net/

Tabela 1. Equivalência de instruções PortuCol, C e Portugol

Construções	PortuCol	C	Portugol

Estrutura	`inteiro principal(){`
	`  ...`
	`}`
		`int main(){`
		`  ...`
		`}`
			`Algoritmo "nome"`
			`var`
			`  ...`
			`inicio`
			`  ...`
			`fimalgoritmo`

Variável tipo inteiro	`inteiro i;`	`int i;`
		`short int i;`
		`long int i;`
		`long long int i;`
			`i: inteiro`

Variável tipo real	`real d;`	`float d;`
		`double d;`
		`long double d;`
			`d: real`

Variável tipo lógico	`logico b;`	`bool b; (C99)`	`b: logico`
Variável tipo caractere	`texto s;`	`char s;`	`s: caractere`
Variável tipo *string*	`texto s;`	`char s[];`	`s: caractere`
Variável tipo vetor	`tipo arranjo[linhas];`	`tipo arranjo[linhas];`	`v: arranjo[linha1..linhaN] de tipo`
Variável tipo matriz	`tipo arranjo[linhas][colunas];`	`tipo arranjo[linhas][colunas];`	`v: arranjo[linha1..linhaN, coluna1..colunaN] de tipo`
Ler valor	`leia("%tipo", i);`	`scanf("%tipo", i);`	`leia(i)`
Escrever	`escreva("...");`	`printf("...");`	`escreva("...")`
Escrever com quebra de linha	`escreva("...\n");`	`printf("...\n");`	`escreval("...")`
Escrever com parâmetros	`escreva("... %tipo ...` `%tipo ...", a, b);`	`printf("... %tipo ...` `%tipo ...", a, b);`	`escreval("...", a, "...",` `b, "...")`
Operação de atribuição	`a = b;`	`a = b;`	`a <- b` `a := b`
Operação de adição	`a + b;`	`a + b;`	`a + b`
Operação de subtração	`a - b;`	`a - b;`	`a - b`
Operação de multiplicação	`a * b;`	`a * b;`	`a * b`
Operação de divisão	`a / b;`	`a / b;`	`a / b`
Operação de divisão inteira	`a / b;`	`a / b;`	`a div b`
Operação de resto da divisão	`a % b;`	`a % b;`	`a % b` `a mod b`
Operação de potenciação	`potencia(a, b);`	`pow(a, b)  <math.h>`	`a ^ b`
Operação de raiz quadrada	`raizquadrada(a);`	`sqrt(a);  <math.h>`	`RaizQ(a)`

Operação de concatenação	`concatena("texto","texto");`
	`concatena("texto", a);`
	`concatena(a, "texto");`
		`strcat("texto", "texto");`
		`strcat("texto", a);`
		`strcat(a, "texto");`
			`"texto" + "texto"`
			`"texto" + a (string)`
			`a (string) + "texto"`

Operação de tamanho do texto	`tamanhotexto(a);`	`strlen(a);  <string.h>`	`compr(a)`
Operador menor	`a < b;`	`a < b;`	`a < b`
Operador menor ou igual	`a <= b;`	`a <= b;`	`a <= b`
Operador maior	`a > b;`	`a > b;`	`a > b`
Operador maior ou igual	`a >= b;`	`a >= b;`	`a >= b`
Operador igual	`a == b;`	`a == b;`	`a = b`
Operador diferente	`a != b;`	`a != b;`	`a <> b`
Operador e	`a e b`	`a && b`	`a e b`
Operador ou	`a ou b`	`a || b`	`a ou b`
Operador de negação	`!a`	`!a`	`nao a`

Condicional se	`se (condição) {`
	`  ...`
	`} senao {`
	`  ...`
	`}`
		`if (condição) {`
		`  ...`
		`} else {`
		`  ...`
		`}`
			`se condição entao`
			`  ...`
			`senao`
			`  ...`
			`fimse`

Condicional aninhado	`se (condição){`
	`  ...`
	`} senao se (condição){`
	`  ...`
	`} senao {`
	`  ...`
	`}`
		`if (condição){`
		`  ...`
		`} else if (condição){`
		`  ...`
		`} else {`
		`  ...`
		`}`
			`se condição entao`
			`  ...`
			`senao`
			`  se condicao entao`
			`  ...`
			`  senao`
			`  ...`
			`  fimse`
			`fimse`

Condicional caso	`escolha (variável) {`
	`  caso valor1:`
	`  ...`
	`  caso valor2, valor3:`
	`  ...`
	`  padrao:`
	`  ...`
	`}`
		`switch (variável) {`
		`  case valor1:`
		`  ...`
		`  break;`
		`  case valor2:`
		`  case valor3:`
		`  ...`
		`  break;`
		`  default:`
		`  ...`
		`  break;`
		`}`
			`escolha variável`
			`  caso valor1`
			`  ...`
			`  caso valor2, valor3`
			`  ...`
			`  outrocaso`
			`  ...`
			`fimescolha`

Condicional enquanto	`enquanto (condição) {`
	`  ...`
	`}`
		`while (condição) {`
		`  ...`
		`}`
			`enquanto condição faca`
			`  ...`
			`Fimenquanto`

Condicional para	`para (variável = valor1 ;`
	`valorN ; P) {`
	`  ...`
	`}`
		`for (variável = valor1;`
		`variável <= valorN;`
		`variável += P) {`
		`  ...`
		`}`
			`para variável de valor1`
			`ate valorN passo P faca`
			`  ...`
			`fimpara`

Comando de interrupção	`interrompa;`	`break;`	`interrompa`

Criar procedimento	`semretorno nome(tipo`
	`parâmetro, ...) {`
	`  ...`
	`}`
		`void nome(tipo parâmetro,`
		`...) {`
		`  ...`
		`}`
			`procedimento nome(`
			`parâmetro: tipo; ...)`
			`var`
			`  ...`
			`inicio`
			`  ...`
			`fimprocedimento`

Criar função	`tipo nome(tipo parâmetro,`
	`...) {`
	`  ...`
	`  retorne valor;`
	`}`
		`tipo nome(tipo parâmetro,`
		`...) {`
		`  ...`
		`  return valor;`
		`}`
			`funcao nome(parâmetro:`
			`tipo; ...): tipo`
			`var`
			`  ...`
			`inicio`
			`  ...`
			`retorne valor`
			`fimfuncao`

```
toy_example — -bash — 52×17
[[toy_example@terra] $ cat alg.pc
inteiro principal(){
        escreva("Ola mundo!");
}

[[toy_example@terra] $ ./portucol2c alg.pc
Codigo C ANSI alg.c gerado e ja compilado com GCC.

[[toy_example@terra] $ cat alg.c
#include <stdio.h>

int  main () {
    printf("Ola mundo!");
}

[[toy_example@terra] $ ./alg
Ola mundo!
```

Figura 2. Execução do `PortuCol2C`

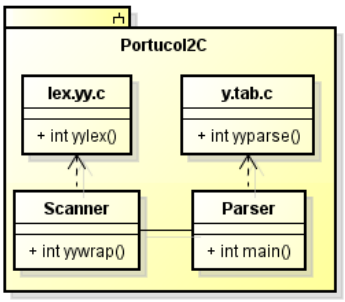

Figura 3. Arquitetura do `PortuCol2C`

de *tokens* e verificação de quaisquer erros de sintaxe de uma forma inteligível [1]. Conforme ilustrado na Figura 3, a ferramenta segue então uma arquitetura com os seguintes dois módulos principais:

1. *Scanner*: Responsável por realizar a análise léxica do código-fonte escrito em PortuCol. Sua implementação consiste em definições de expressões regulares a serem convertidas em *tokens* tais como números, variáveis e palavras reservadas. Posteriormente, são definidas as regras para cada *token*, sendo que no caso do *Scanner* as regras estabelecidas consistem em retornar cada *token* para o *Parser*. A compilação do *Scanner* pelo *Lex* gera o arquivo `lex.yy.c`, o qual é utilizado para viabilizar a detecção dos *tokens* no código-fonte do PortuCol.
2. *Parser*: Responsável por realizar a análise sintática do código-fonte escrito em PortuCol. O *Parser* é constituído por regras formadas através dos *tokens* recebidos pelo *Scanner*, sendo que cada regra possui subrotinas associadas. No nosso caso, as subrotinas têm como finalidade imprimir, no arquivo de saída, a tradução para código C ANSI da sequência de *tokens* relacionado à cada regra. A compilação do *Parser* pelo *Yacc* gera o arquivo `y.tab.c`, o qual é utilizado para possibilitar a análise sintática dos *tokens*.

4. Avaliação

4.1. Questões de Pesquisa

Um estudo foi projetado para responder às seguintes questões de pesquisa:

QP #1 – PortuCol se aproxima à C ANSI mais do que Portugol?

QP #2 – PortuCol se aproxima à C ANSI mais do que Portugol se aproxima à Pascal?

4.2. *Dataset*[2]

Para efetivamente comparar a similaridade entre algoritmos nas linguagens PortuCol, C, Portugol e Pascal, foram especificados 30 algoritmos, os quais foram implementados em cada uma das linguagens. A Tabela 3 descreve sucintamente os algoritmos implementados e aponta as principais estruturas utilizadas.

[2] *Dataset* publicamente disponível em: `http://www.dcc.ufla.br/~terra/papers/2016_wei_portucol`

Tabela 3. Descrição dos algoritmos

Algoritmo	Descrição	Principais estruturas utilizadas
Alg_1	Cadastro e acesso de produtos	Procedimentos, enquanto, para, caso, se
Alg_2	Equação quadrática	Tipo de dado real, se senao se
Alg_3	Idade por ano de nascimento	Tipo de dado arranjo, para
Alg_4	Bubble sort	Tipo de dado arranjo, para, se
Alg_5	Busca binária	Função, enquanto, se senao se
Alg_6	Busca em arranjo	Faca enquanto, se, senao
Alg_7	Calculadora (+, -, *)	Caso, tipo de dado texto
Alg_8	Média de notas	Leitura de entrada, se, senao
Alg_9	Soma e subtração de reais	Para, se, senao
Alg_10	Combinação	Escrita e leitura de dados, Para
Alg_11	Decomposição de inteiro	Para, faca enquanto, se
Alg_12	Aprovação de aluno por nota	Caso
Alg_13	Fatorial recursivo	Função recursiva, se senao se
Alg_14	Fibonacci iterativo	Função, para
Alg_15	Selection sort	Para, tipo de dado arranjo, se
Alg_16	Mostrar coluna de matriz	Matriz 4x4, para
Alg_17	Média de reais	Função, tipo de dados real
Alg_18	Inverter texto	Tipo de dados texto, para
Alg_19	Números palíndromos	Enquanto, se, senao
Alg_20	Primos até certo número	Para, enquanto, se
Alg_21	Aproximação de Pi	Para, operação de raiz quadrada
Alg_22	Potenciação de inteiros (xˆy)	Para
Alg_23	Comb sort	Enquanto, se, para
Alg_24	Soma de matrizes	Matrizes 2x2, para
Alg_25	Soma de algarismos	Enquanto, operação de resto de divisão
Alg_26	Classificação de texto por caso	Caso, comparação de texto
Alg_27	Validação de CPF	Para, se, conversão de texto para inteiro
Alg_28	Divisão de arranjos entre par e ímpar	Repita enquanto, para, se, senao
Alg_29	Aproximação de raiz quadrada	Se, enquanto, senao
Alg_30	Concatenação e comparação de texto	Se, senao, operações de concatenação e comparação de texto

4.3. Similaridade Textual

Para obter a similaridade entre os algoritmos implementados em PortuCol, C , Portugol e Pascal foi utilizada a similaridade de cosseno (*cosine similarity*) – técnica mais adotada em recuperação de informação e mineração de texto para comparação de documentos [7] – a qual mensura a similaridade de dois documentos com base no ângulo entre seus vetores representativos.[3]

Portanto, a similaridade de cosseno entre dois documentos d_1 e d_2 é computada a partir dos seus vetores representativos $\overrightarrow{V}(d_1)$ e $\overrightarrow{V}(d_2)$ conforme Equação 1.

$$sim(d_1, d_2) = \frac{\overrightarrow{V}(d_1) \cdot \overrightarrow{V}(d_2)}{|\overrightarrow{V}(d_1)||\overrightarrow{V}(d_1)|} \tag{1}$$

onde $\overrightarrow{V}(d_i)$ denota o vetor derivado de um documento d_i, com um componente no vetor para cada termo do dicionário. Assim, cada termo do documento é armazenado uma única vez no vetor e é associado a um peso. O peso dos termos normalmente leva em consideração o $tf{-}idf_{t,d}$, onde tf é a frequência dos termos (*Term Frequency*), idf é a frequência inversa do documento (*Inverse Document Frequency*), t representa os termos e d o documento.[4] Enfim, a similaridade entre dois documentos consiste na razão entre (i) o produto escalar e (ii) o produto das normas euclidianas dos dois vetores representativos.

[3] A similaridade varia de 0 (nada similares) a 1 (totalmente similares).

[4] Particularmente neste artigo, foi considerada apenas a frequência dos termos, uma vez que diminuir o peso de termos frequentes não é desejável.

4.4. Metodologia

Foram conduzidas as seguintes atividades para se responder às questões de pesquisa:

1. Definir uma particular versão de Portugol, devido à existência de variações do Portugol encontradas na literatura, onde diferentes ferramentas [3, 5, 9, 8, 4] possuem sua próprias variações de Portugol. Assim, foi escolhida a variante adotada pelo VisuAlg – um interpretador de Portugol desenvolvido por Souza [4] – a qual é completa e amplamente aceita.
2. Construir um *dataset* contendo uma vasta gama de algoritmos implementados nas linguagens PortuCol, C, Portugol e Pascal (veja Seção 4.2).
3. Comparar a similaridade textual – utilizando *cosine similarity* descrito na Seção 4.3 – dos algoritmos implementados na linguagem PortuCol com C, Portugol com C e Portugol com Pascal. É importante mencionar que textos referentes à inclusão de bibliotecas (e.g. `#include <stdio.h>` e `uses crt`) foram retirados dos códigos C e Pascal.
4. Determinar estatisticamente a relação de similaridade de PortuCol com C e de Portugol com C, a fim de responder a QP #1.
5. Determinar estatisticamente a relação de similaridade de PortuCol com C e de Portugol com Pascal, a fim de responder a QP #2.

4.5. Resultados

A Tabela 4 reporta os resultados da similaridade textual dos algoritmos implementados na linguagem Portugol com C (coluna 1), PortuCol com C (coluna 2) e Portugol com Pascal (coluna 3). A Figura 4 representa visualmente o intervalo de confiança para a média das similaridades textuais de cada par de linguagens.

Tabela 4. Similaridade Textual

Algoritmo	Portugol/C	PortuCol/C	Portugol/Pascal
Alg_1	0.4004	0.6031	0.4365
Alg_2	0.7009	0.6824	0.7607
Alg_3	0.6348	0.7314	0.7291
Alg_4	0.7824	0.8713	0.7850
Alg_5	0.6803	0.7712	0.6591
Alg_6	0.5866	0.7072	0.6256
Alg_7	0.6674	0.6292	0.7227
Alg_8	0.6190	0.5930	0.6347
Alg_9	0.3036	0.7593	0.6543
Alg_10	0.7386	0.6616	0.8338
Alg_11	0.7262	0.8586	0.7523
Alg_12	0.2761	0.3639	0.5355
Alg_13	0.6385	0.6468	0.6156
Alg_14	0.6230	0.6863	0.5823
Alg_15	0.7190	0.8915	0.7118
Alg_16	0.5420	0.6057	0.5809
Alg_17	0.4449	0.4355	0.5536
Alg_18	0.6502	0.7662	0.7963
Alg_19	0.8511	0.8685	0.8348
Alg_20	0.3073	0.7611	0.7110
Alg_21	0.3267	0.7708	0.7780
Alg_22	0.6261	0.6199	0.6581
Alg_23	0.8580	0.9192	0.8247
Alg_24	0.0407	0.8187	0.9045
Alg_25	0.1079	0.8729	0.7621
Alg_26	0.6490	0.7286	0.7338
Alg_27	0.6202	0.8472	0.5964
Alg_28	0.5844	0.6647	0.6306
Alg_29	0.8454	0.8956	0.8604
Alg_30	0.4611	0.5484	0.4665
Média ($\bar{x}$)	**0.5671**	**0.7193**	**0,6910**
Dev. Pad. (s)	**0.2089**	**0.1361**	**0,1161**

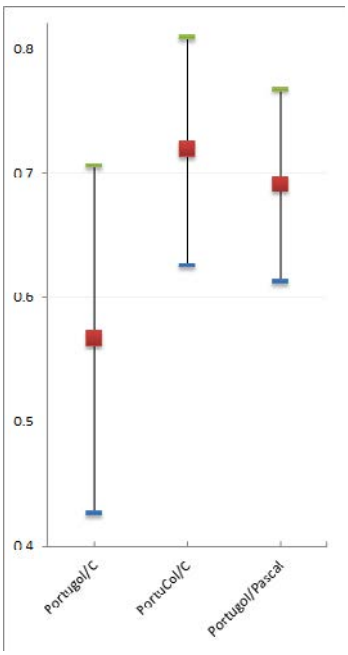

Figura 4. Intervalo de confiança

QP #1 – *PortuCol se aproxima à C ANSI mais do que Portugol?*

Foram comparadas as similaridades textuais dos algoritmos implementados (i) em Portugol e em C e (ii) em PortuCol e em C, cujos resultados podem ser observados nas primeiras duas colunas da Tabela 4 e nas duas primeiras linhas da Figura 4.

O *dataset* disponível para avaliação apresenta os mesmos algoritmos implementados em diferentes linguagens (veja Seção 4.2). Isso permite que seja conduzida uma observação pareada dos resultados de similaridade textual. Portanto, considerando uma distribuição normal dos resultados, a distribuição probabilística *t-Student* foi utilizada no cálculo do intervalo de confiança para responder à QP #1, conforme Equação 2.[5]

$$IC = \overline{x} \pm t_{[1-\frac{\alpha}{2};n-1]}(\frac{s}{\sqrt{n}}) = 0,1523 \pm 3,6590(\frac{0,2158}{\sqrt{30}}) = [0{,}0081\ ,\ 0{,}2964] \quad (2)$$

O *dataset* utilizado possui trinta algoritmos ($n = 30$). A diferença da similaridade textual entre cada algoritmo Portugol/C e PortuCol/C (colunas 1 e 2 da Tabela 4) foi calculada. Assim, obteve-se a média aritmética ($\overline{x} = 0,1523$) e o desvio padrão ($s = 0,2158$) das diferenças. A confiança adotada para as amostras da distribuição de *t-Student* foi 99,9% com grau de liberdade de 29, onde $t_{[0.9995;29]}$ é igual a 3,659. Assim, o teste t resultou no intervalo de $[0{,}0081\ ,\ 0{,}2964]$, o qual *não* inclui zero. Isso indica que, com 99,9% de confiança, PortuCol se aproxima 26,85% mais à C ANSI do que Portugol.

QP #2 – *PortuCol se aproxima à C ANSI mais do que Portugol se aproxima à Pascal?*

Similarmente à QP #1, foram comparadas as similaridades textuais dos algoritmos implementados (i) em PortuCol e em C e (ii) em Portugol e em Pascal, cujos resultados podem ser observados nas últimas duas colunas da Tabela 4 e nas duas últimas linhas da Figura 4. É importante mencionar que os resultados comparativos entre PortuCol e C já tinham sido previamente calculados na resposta à QP #1.

Considerando uma distribuição normal dos resultados, a distribuição probabilística *t-Student* novamente foi utilizada no cálculo do intervalo de confiança para responder à QP #2, conforme Equação 3.

$$IC = \overline{x} \pm t_{[1-\frac{\alpha}{2};n-1]}(\frac{s}{\sqrt{n}}) = 0,0283 \pm 3.659(\frac{0.1006}{\sqrt{30}}) = [-0{,}0389\ ,\ 0{,}0955] \quad (3)$$

Similarmente à *QP #1*, a confiança adotada para as amostras da distribuição de *t-Student* foi 99,9% com grau de liberdade de 29. Assim, o teste t resultou no intervalo de $[-0{,}0389\ ,\ 0{,}0955]$. Como o intervalo de confiança inclui zero, podemos considerar que o PortuCol se aproxima à C ANSI da mesma maneira que Portugol se aproxima à Pascal.

5. Considerações Finais

O uso de pseudolinguagens no ensino de Lógica de Programação e Algoritmos é uma estratégia muito utilizada em IES. Embora C ANSI e Java sejam as linguagens de programação mais utilizadas no desenvolvimento de sistemas de software [10], a pseudolinguagem mais utilizada no Brasil é o Portugol [9, 11], que é inspirada no obsoleto Pascal.

[5]O teste t resulta em um intervalo de confiança. Caso o intervalo inclua zero, as amostras *não* são estatisticamente diferentes. Caso contrário, pode-se afirmar que as amostras *são* estatisticamente diferentes, o que, neste caso, indica que a similaridade em um par de linguagens é maior que na outra.

Diante desse cenário, este artigo propôs PortuCol, uma pseudolinguagem de programação com instruções em português assim como Portugol, porém inspirada em C ANSI. Como contribuição acadêmica, uma análise estatística da similaridade textual demostrou que: (i) a pseudolinguagem PortuCol é estatisticamente 26,85% mais similar à C ANSI do que Portugol, o que proporciona uma transição mais natural ao estudante ao aprender linguagens do estado-da-prática como C ANSI e Java; e (ii) PortuCol é para C ANSI o que Portugol é para Pascal, o que retrata apenas uma adaptação da pseudolinguagem às linguagens do estado-da-prática. Como contribuição prática, este artigo também apresenta `PortuCol2C`, uma ferramenta que executa código PortuCol através da prévia tradução para código C ANSI, o que possibilita aos estudantes se familiarizem com a linguagem C ANSI através da observação do seu código PortuCol traduzido.

O *dataset* dos algoritmos utilizados na avaliação e a ferramenta `PortuCol2C` (binário e código fonte) estão publicamente disponíveis em: `http://www.dcc.ufla.br/~terra/papers/2016_wei_portucol`

Como trabalhos futuros, pretende-se (i) avaliar o uso da pseudolinguagem PortuCol no ensino de Lógica de Programação e Algoritmos e (ii) melhorar a ferramenta `PortuCol2C` com um ambiente de desenvolvimento integrado (IDE) e um depurador.

Agradecimentos

Este trabalho foi apoiado pela FAPEMIG, CAPES e CNPq.

Referências

[1] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compiladores: Princıpios, técnicas e ferramentas*. LTC, 2 edition, 2008.

[2] Ricardo Luiz B. L. Campos. Lógica de programação: Há como melhorar o aprendizado fugindo dos padrões estabelecidos nos livros didáticos e adotados pela maioria dos docentes? In *XVII Congresso Iberoamericano de Educación Superior em Computación (CIESC)*, pages 22–25, 2009.

[3] Rafael de Santiago and Rudimar Luís Scaranto Dazzi. Ferramenta de apoio ao ensino de algoritmos. In *XIII Seminário de Computação (SEMINCO)*, pages 1–8, 2004.

[4] Cláudio Morgado de Souza. VisuAlg - ferramenta de apoio ao ensino de programação. *TECCEN*, 2(2):1–9, 2009.

[5] Mauri Ferrandin and Simone Lilian Stephani. Ferramenta para o ensino de programação via internet. In *I Congresso Sul Catarinense de Computação (SULCOMP)*, pages 1–8, 2005.

[6] John R. Levine, Tony Mason, and Doug Brown. *Lex & Yacc*. O'Reilly, 2 edition, 1992.

[7] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[8] António Manso, Luís Oliveira, and Célio Gonçalo Marques. Ambiente de aprendizagem de algoritmos – Portugol IDE. In *VI Conferência Internacional de TIC na Educação*, pages 969–983, 2009.

[9] Luiz F. Noschang, Fillipi Pelz, Elieser A. de Jesus, and André L. A. Raabe. Portugol Studio: Uma IDE para iniciantes em programação. In *XXII Workshop sobre Educação em Computação (WEI)*, pages 1287–1296, 2014.

[10] TIOBE. Programming languages index. `http://www.tiobe.com/tiobe_index`, March 2016.

[11] Adriana Salvador Zanini and André Luís Alice Raabe. Análise dos enunciados utilizados nos problemas de programação introdutória em cursos de ciência da computação no Brasil. In *XX Workshop sobre Educação em Computação (WEI)*, 2012.