Um Modelo Operacional Automático para Detecção de Padrões Lógico-Linguisticos na Programação

Isabel. H.G. P. Barros, Carlo E. T. de Oliveira, Claudia L. R. da Motta
Universidade Federal do Rio de Janeiro, UFRJ.

CONTEXTO

A baixa qualidade dos programas na maioria das vezes advém da dificuldade de entender a abstração dos problemas, a qual depende da condição cognitiva do desenvolvedor. Portanto, tal dificuldade é um desafio no aprendizado e essencial no desenvolvimento e na qualidade do software.

PROPOSTA

Esta pesquisa apresenta um modelo computacional que fornece recursos para a detecção de padrões de códigos em variados níveis de programação, avalia a qualidade dos códigos e identifica as competências lógico-linguísticas usadas na resolução de problemas.

METODOLOGIA

Método da investigação: É um experimento com variáveis controladas. A análise da pesquisa é quantitativa e qualitativa com a participação ativa do publico alvo.

Público alvo:
Aprendizes de programação, programadores e educadores.

Instrumento: Para detectar padrões e competências lógico-linguísticos de aprendizes de programação, desenvolveu-se um engenho, uma API, desenhado na linguagem Python.

PROCEDIMENTO

Para validar o modelo foi desenvolvida uma ferramenta, Eidoloom, que classifica e sugere aperfeiçoamentos no código de forma a conduzir o aprendiz a otimizar seu programa. A abordagem é baseada na transcrição da abstração simbólica da metodologia Montessori.

Experimento 1: Para verificar a qualidade do código e corroborar com os pressupostos de Casalnuovo et al(2019) expõe-se que as limitações humanas na programação não são apenas no âmbito da sintaxe, mas apontam para a estrutura contextual do código que é adversa a linguagem natural humana. Neste contexto busca-se a relação gramatical e sintática no código afim de evitar a repetitividade, fundamental na busca de qualidade no software. Foram identificadas morfologicamente as 34 palavras reservadas da linguagem Python, associadas aos símbolos da metodologia Gramatical Montessori e adaptadas aos símbolos unicodes. (Tabela 1).

Tabela 1: Identificação morfológica das palavras reservadas.

CLASSES GRAMATICIAS	SÍMBOLOS	PALAVRAS RESERVADAS
Substantivo (nomeia)	▲	**Básico:** true, false **Intermediário:** none
Adjetivo (Qualifica)	◭	**Básico:** def **Intermediário:** nonlocal, global **Avançado:** class
Verbo (indica ação, estado)	●	**Básico:** break, pass, import, return **Intermediário:** continue, del **Avançado:** try, assert, raise, lambda, yeld, exec
Advérbio (modifica)	◎	**Básico:** in **Intermediário:** is **Avançado:** except, finally, not
Preposição (liga)	◠	**Básico:** for, from **Intermediário:** as, with
Conjunção (liga)	━	**Básico:** while, if, elif, else, or, and

Experimento 2: Segundo o método Montessori é possível descrever as classes gramaticais em duas grandes famílias, que são as bases para iniciar o entendimento da estrutura sintática, ou seja, a relação das palavras (tokens) utilizados no código. Desta forma, amplia-se a investigação da base de dados que passa a receber a lista bruta de tokens. Para a leitura e manipulação de dados foi criado um Data Frame. Os módulos Abstract Syntax Tree (AST) responsáveis pela análise semântica foram usados para possibilitar a investigação dos códigos no processamento da gramatica de sintaxe abstrata. Com o intuito de ensinar a máquina foram coletados códigos fonte de games escritos na linguagem Python. A árvore PLL aponta para a instância lógica do desenvolvedor (Figura 1).

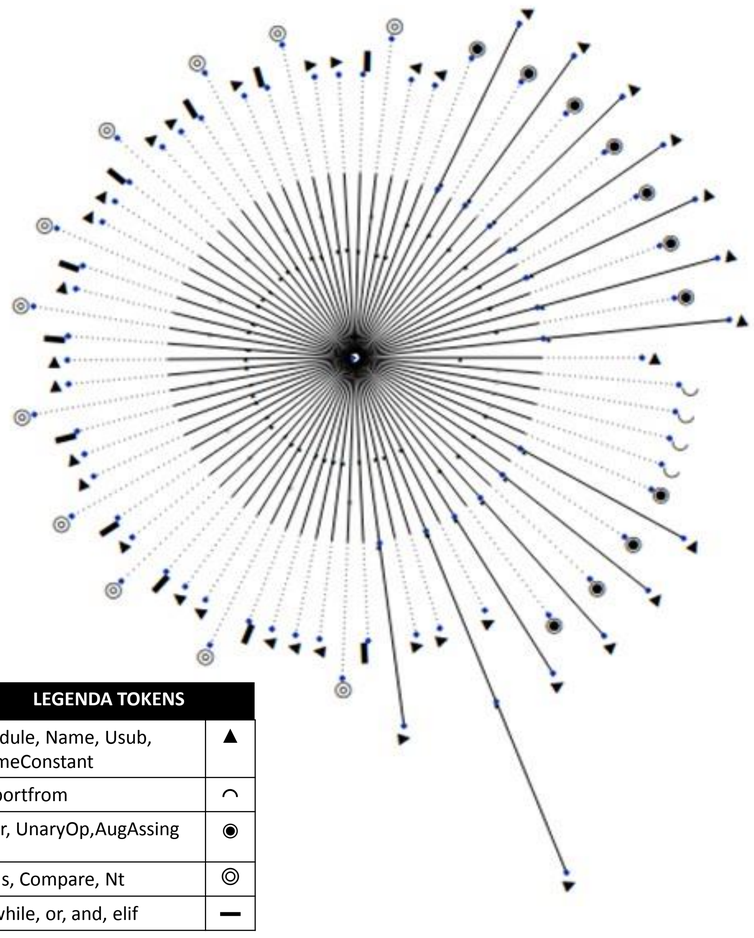

Fig. 1 Árvore PLL Instância lógica do desenvolvedor do Game Jokenpo

RESULTADOS

O Gráfico 1 exibe os valores gerados no conjunto de dados contidos no Data frame. É possível verificar a relação, a intensidade da relação e a incidência dos tokens nos códigos fonte a partir dos Games investigados.

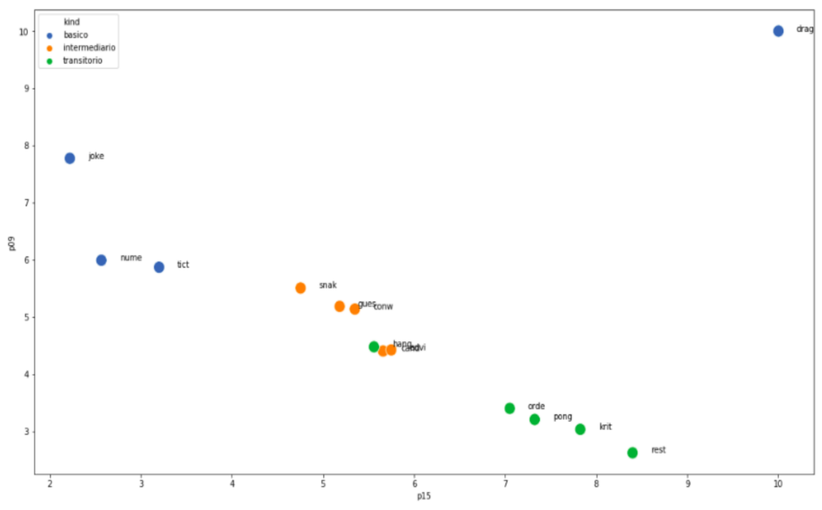

Gráfico 1: Parâmetros p15 x p09

PROXIMOS PASSOS

- Complementar Análise quantitativa
- Análise qualitativa
- Intervenção: método de elaboração dirigida
- Resultados e aprimoramento do código

Referências

Casalnuovo, C., Sagae, K., Devanbu, P. (2019) “Studying the Difference Between Natural and Programming Language Corpora”. Empirical Software Engineering. https://doi.org/10.1007/s10664-018-9669-7.ISSN1382-3256.
Hindle A., Barret E T., Gabel M., Devambu P. (2012), “On the naturalness of software”. In: Proceedings of the 34th International Conference on Software Engineering, IEEE Press,Piscataway, NJ, USA, ICSE ’12, pp 837–847. ISBN: 978-1-4673-1067-3.
Knuth, D. E. (1992), Literate Programming. California: Stanford University Center for the Study of Language and Information.
Montessori, M., Tornar, C. Fresco, G. H. (2017), Psicogrammatica. Franco Angeli.
Neto, A C; Ribeiro, E; Conceição J; Santos, K., Jesus, G S .(2018) “Avaliação de uma abordagem para auxiliar a correção de erros de aprendizes de programação”. CBIE 2018, Anais do XXIX Simpósio Brasileiro de Informática na Educação (SBIE).