Análise Automatizada da Originalidade de Design de Interfaces de Usuário no Contexto Educacional: Um Mapeamento da Literatura

Angélica Siqueira de Souza[1], Nathalia da Cruz Alves[1], Christiane Gresse von Wangenheim[1], Leonardo Kreuch[3]

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC) – Florianópolis – SC – Brasil

angelica.siqueira@posgrad.ufsc.br,
nathalia.alves@posgrad.ufsc.br,c.wangenheim@ufsc.br,
leonardo.k@grad.ufsc.br

Abstract. *Aiming to contribute to the development of important skills in the 21th century, such as creativity by teaching computing in Basic Education, the aim is to teach students to develop original apps with App Inventor. However, one wonders how to assess originality specifically in relation to user interface (UI) design to track learning progress. Thus, this article presents a systematic mapping of existing approaches to automatically assess the UI design originality of Android apps using Artificial Intelligence techniques, which can be used by teachers to assess creativity.*

Resumo. *Com o objetivo de contribuir ao desenvolvimento de habilidades importantes no século XXI, como a criatividade pelo ensino de computação na Educação Básica, visa-se ensinar ao aluno a desenvolver apps originais com App Inventor. Porém, se questiona como avaliar a originalidade especificamente em relação ao design de interface de usuário (UI) para acompanhar o progresso da aprendizagem. Assim, este artigo apresenta um mapeamento sistemático das abordagens existentes para automaticamente avaliar a originalidade de design de UI de apps Android usando técnicas de Inteligência Artificial, que podem ser utilizadas por professores para a avaliação da criatividade.*

1. Introdução

Uma das habilidades importantes do século XXI é a criatividade [Cavallo et al. 2016], especialmente num mundo caracterizado pela automação [WEF 2020]. Assim é essencial promover o desenvolvimento da criatividade dos alunos já a partir da Educação Básica [Cavallo et al. 2016]. Mesmo sendo comumente associada às atividades no ensino da arte, música e literatura, a criatividade pode também ser apoiada pelo ensino da computação [Alves et al. 2020]. A computação pode promover criatividade quando estudantes deixam de ser apenas consumidores de tecnologia e passam a construir artefatos computacionais criativos que podem ter um impacto significativo na sociedade [Yadav e Cooper 2017].

Atualmente, adotando metodologias ativas no ensino de computação na Educação Básica os alunos aprendem, por exemplo, o desenvolvimento de apps usando App Inventor [MIT 2020], um ambiente de programação visual baseado em blocos que permite criar apps para Android, estimulando a geração de ideias e a criatividade [Patton, Tissenbaum e Harunani 2019]. Além da parte funcional o App Inventor suporta

DOI: 10.5753/sbie.2021.217542

também o design de interação e interface [Patton, Tissenbaum e Harunani 2019]. A criação do design de interface permite, à nível de esqueleto, adicionar componentes visuais, como botões, caixas de seleção, escolhas de data, imagens, entre outros [MIT 2020]. Com relação ao *layout*, é possível configurar o posicionamento e alinhamento dos componentes, e no nível de design visual também permite definir o tamanho, cor, fonte, etc. [MIT, 2020]. Levando em consideração a importância do design de interface de usuário (UI - *user interface*) para o sucesso de um app [Ferreira et al. 2020], demonstra-se também a importância do desenvolvimento de uma UI criativa como parte de um app criativo.

O desenvolvimento da criatividade pode ser feito por meio do ensino de desenvolvimento de apps, avaliando os apps dos alunos e dando *feedback* em relação às características criativas do app desenvolvido. Embora diferentes aspectos da criatividade (pessoa, produto/artefato, processo, ambiente) possam ser analisados, a análise do produto/artefato criativo pode permitir *insights* sobre o conceito de criatividade como um todo. Portanto, embora a avaliação da criatividade não seja limitada ao ponto de vista de produtos/artefatos [Ritchie 2001], certamente representa uma parte vital para "estabelecer referências para o conceito 'criatividade' por meio de uma avaliação sistemática das coisas que as pessoas produzem" [Skager, Schultz e Klein 1966]. Considerando que o ensino de computação é tipicamente feito adotando metodologias ativas e tarefas abertas, essa avaliação pode ser baseada no desempenho analisando o artefato computacional (incluindo o design de UI) criado pelo aluno como resultado da aprendizagem [Alves et al. 2020]. Assim, enfocando na criatividade de artefato, que é tipicamente definida pelos fatores de originalidade, utilidade e condensação, visando que um artefato criativo deva ser, respectivamente, original (novo), útil e bem-feito [Runco e Jaeger 2012], prevê-se então também a avaliação da originalidade de um design de UI.

Porém, realizar a avaliação da originalidade em relação à criatividade do artefato criado pelo aluno não é algo trivial, pois o julgamento humano pode ter influências e preferências variáveis de acordo com sua vivência, tornando uma avaliação subjetiva na qual não há uma "verdade absoluta" [Zen e Vanderdonckt 2016]. Uma alternativa para mitigar essa falta de concordância entre avaliadores humanos e para reduzir esforço e tempo dos professores na Educação Básica é a automatização de parte dessa avaliação [Alves et al. 2020]. Espera-se, com a automatização, uma avaliação mais objetiva, com maior confiabilidade e validade, sem efeito de fatiga do professor e preferências pessoais, além de minimizar esforço, custo e tempo [Beaty e Johnson 2020].

Assim, surge a pergunta de como avaliar a originalidade de UI design neste contexto. Atualmente existem mapeamentos da literatura focando na identificação de abordagens para detecção de plágio de programação no contexto de ensino de computação [Gomes e Matos 2020], para avaliação da criatividade no ensino de computação [Alves et al. 2021] ou outras áreas [Snyder et al. 2019] e para análise automatizada da originalidade de aplicativos Android a partir de features [Alves et al. 2021]. As ferramentas de software automatizadas para avaliação de artefatos computacionais na Educação Básica, em geral, adotam a análise do código de maneiras diferentes, incluindo a automação estática ou dinâmica [Alves et al. 2019]. Mais recentemente surgiu também a utilização de técnicas de *Machine Learning*, como redes

neurais convolucionais para avaliar o grau da estética da UI [Martins 2019]. No entanto, nenhum dos mapeamentos encontrados aborda a questão de como abordagens automatizadas com foco no design de UI poderiam ser aplicadas no contexto educacional buscando identificar o grau de originalidade da UI de um aplicativo criado por um aluno.

Abordando essa questão e seguindo a tendência atual, apresentamos neste artigo um mapeamento sistemático de abordagens existentes para avaliar de forma automatizada a originalidade de design de UI de apps Android usando técnicas de Inteligência Artificial (IA).

2. Definição e execução do estudo de mapeamento sistemático

A fim de levantar o estado da arte foi realizado um mapeamento sistemático da literatura seguindo o procedimento definido por Petersen et al. (2015).

2.1. Definição do Protocolo de Revisão

Pergunta de pesquisa. Quais abordagens existem para automaticamente avaliar a originalidade de design de UI de apps Android usando técnicas de IA?

A pergunta de pesquisa foi refinada nas seguintes perguntas de análise:

PA1. Quais abordagens existem para a avaliação da originalidade do design de UI de apps Android?

PA2. Quais são as características das abordagens de avaliação da originalidade do design de interfaces?

PA3. Quais técnicas de IA são adotadas pelos modelos de avaliação?

PA4. Como foi feita a preparação do conjunto de dados e a rotulação?

PA5. Como é medido e qual é o desempenho das abordagens?

Critérios de inclusão, exclusão e qualidade. Foram incluídos artigos em inglês publicados a partir de 2007 (ano do lançamento do primeiro *smartphone*), acessíveis via Portal CAPES que apresentam alguma abordagem de avaliação da originalidade ou similaridade (conceito inverso da originalidade) em relação ao design de UI de aplicativos de Android. Para ampliar a visão geral, a pesquisa não é limitada ao contexto educacional, porém foram excluídos artigos focados exclusivamente em questões de *malware* e/ou plágio. Em termos de qualidade, foram incluídos apenas artigos contendo informações substanciais indicando como o design de UI é avaliado de acordo com sua originalidade.

Bases de dados. As buscas foram realizadas nas principais bases de dados e bibliotecas digitais da área da computação, incluindo IEEE Xplore, ACM Digital Library, Scopus (que inclui a base Springer), ScienceDirect, e arXiv.org. Foram realizadas também buscas no Google Scholar como uma fonte adicional visando a minimização do risco de omissão [Piasecki et al. 2018].

***String* de Busca.** Com base na pergunta de pesquisa e buscas informais para calibrar a *string* de busca, foi definida a *string* de busca incluindo os termos chaves e sinônimos:

("*mobile application*" OR "*android*" OR "*app inventor*" OR "*app*") *AND* ("*user interface*" OR "*UI design*" OR "*visual languages*") *AND* ("*creativity*" OR "*original*" OR "*similarity*" OR "*novel*") *AND* ("*deep learning*" OR "*artificial intelligence*")

Essa *string* de busca genérica foi adaptada a sintaxe específica de cada base de dados.

2.2. Execução da busca

A busca foi realizada em março/abril de 2021 por dois autores e revisada pelos coautores. A busca inicial resultou em 19.738 artigos, dos quais foram analisados apenas os 150 primeiros artigos da busca, observando uma queda na relevância dos resultados a partir do 100º artigo. A segunda etapa, de análise dos artigos, também foi realizada por dois autores e revisada pelos coautores durante um mês, dos quais foram selecionados os relevantes de acordo com os critérios definidos (Tabela 1).

Tabela 1. Resultados da busca

Base de Dados	Quantidade de artigos resultantes da busca	Quantidade de artigos analisados	Quantidade de artigos potencialmente relevantes (analisando título e *abstract*)	Quantidade de artigos relevantes (com base no artigo na íntegra)
ACM Digital Library	944	150	9	4
arXiv.org e-print archive	4	4	2	1
Google Scholar	14.200	150	6	3
IEEE Xplore	10	10	1	1
ScienceDirect	766	150	3	0
Scopus	3.814	150	22	11
Total				11 (sem duplicatas)

Alguns artigos encontrados nas buscas foram excluídos por falta de enfoque na originalidade/similaridade do artefato/app, como Jeong, Kim e In (2020) ou por falta de enfoque no design de UI, como Ichinco e Kelleher (2018) e Basu (2019). Vários artigos como Li et al. (2019), Lyu et al. (2016) e Soh et al. (2015), também foram excluídos pelo enfoque na análise de determinados tipos de plágios/clonagem, mas não na originalidade. Por impossibilidade de acesso ao texto na íntegra também foram excluídos os artigos Zhang et al. (2020) e Hu et al. (2020).

2.3. Análise dos dados

2.3.1 Quais artigos existem para a avaliação da originalidade do design de UI de apps Android?

Foram encontradas somente 11 artigos diferentes que contêm alguma forma de avaliação da originalidade do design de interfaces de apps Android (Tabela 2).

Tabela 2. Artigos relevantes

Citação	Referência bibliográfica
(BEHRANG et al., 2018)	Behrang, F.; Reiss, S.P.; Orso, A. GUIfetch. **Proc. of the 5th Int. Conference on Mobile Software Engineering and Systems**, ACM, EUA, 2018.
(CHEN et al., 2020)	Chen, J.; Chen, C.; Xing, Z.; Xia, X.; Zhu, L.; Grundy, J.; Wang, J. Wireframe-based UI Design Search through Image Autoencoder. **ACM Transactions on Software Engineering and Methodology**, 29(3), 2020.
(CHEN et al., 2019)	Chen, X.; Zou, Q.; Fan, B.; Zheng, Z.; Luo, X. Recommending software features for mobile applications based on user interface comparison. **Requirements Engineering**, 24(4), 2018.
(DEKA et al., 2017)	Deka, B.; Huang, Z.; Franzen, C.; Hibschman, J.; Afergan, D.; Li, Y.; Nichols, J.; Kumar, R. Rico. **Proc. of the 30st Annual ACM Symposium on User Interface Software and Technology** , ACM, 2017.

(GE, 2019)	Ge, X. Android GUI Search Using Hand-Drawn Sketches. Proc. of the **41st Int. Conference on Software Engineering**, IEEE, 2019.
(HU et al., 2020)	Hu, Y.; Xu, G.; Zhang, B.; Lai, K.; Xu, G.; Zhang, M. Robust App Clone Detection Based on Similarity of UI Structure. **IEEE Access**, 8, 2020.
(MUSTAFARA J et al., 2017)	Mustafaraj, E.; Turbak, F.; Svanberg, M. Identifying Original Projects in App Inventor. **Proc. of the Florida Artificial Intelligence Research Society Conference**, Marco Island, FL, EUA, 2017.
(TURBAK et al., 2017)	Turbak, F.; Mustafaraj, E.; Svanberg, M.; Dawson, M. Work in Progress: identifying and analyzing original projects in an open-ended blocks programming environment. **Proc. of the 23rd Int. Conference on Distributed Multimedia Systems**, Pittsburgh/EUA, 2017.
(SVANBERG, 2017)	Svanberg, M. Using feature vector representations to identify similar projects in App Inventor. **Proc. of the IEEE Blocks And Beyond Workshop**, IEEE, 2017.
(NGUYEN et al., 2018)	Nguyen, T. T.; Vu, P. M.; Pham, H. V.; Nguyen, T. T. Deep learning UI design patterns of mobile apps. **Proc. of the 40th Int. Conference on Software Engineering**, ACM, 2018.
(XIE et al., 2019)	Xie, Y.; Lin, T.; Xu, H. User Interface Code Retrieval: a novel visual-representation-aware approach. **IEEE Access**, 7, 2019.

Pelo fato de que foram encontrados artigos somente de 2017 até 2020, observa -se que este foco de pesquisa surgiu somente recentemente (Figura 1).

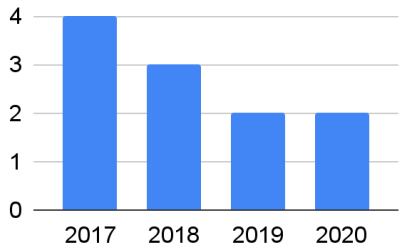

Figura 1. Distribuição de publicações relevantes pelo ano de publicação

2.3.2. Quais são as características das abordagens de avaliação da originalidade do design de interfaces?

As abordagens variam entre analisar a originalidade [Mustafaraj et al. 2017; Svanberg 2017] e a similaridade [Behrang et al. 2018; Nguyen et al. 2018], considerando o conceito da similaridade como o inverso da originalidade. Abordagens que analisam a originalidade, simplesmente avaliam se o design da UI é original ou não original. Por outro lado, abordagens que analisam a similaridade, utilizam escalas de medição de similaridade referente à semelhança entre design de UIs [Chen et al. 2019] ou criam rankings de apps com base no seu grau de originalidade [Ge 2019; Behrang et al. 2018].

Em relação aos critérios utilizados, a maior parte das abordagens (8 de 11) avalia critérios relacionados à estrutura da UI, podendo analisar *screenshots* diretamente (ex.: Chen et al. (2020) ou *sketches* (ex.: Behrang et al. (2018)). Xie et al. (2019) transforma um *sketch* de UI como entrada em uma representação de árvore em que cada nó da árvore representa um *widget* ou *layout* e Ge (2019) em esqueletos da UI (UI *skeleton*). Por outro lado, Deka et al. (2017), Chen et al. (2020) e Hu et al. (2020), analisam posições e tamanhos dos elementos de cada UI a partir de um *screenshot*, diferenciando elementos textuais e não textuais e informações estruturais sobre o *layout*. Além disso, Nguyen et al. (2018) e Behrang et al. (2018), adotam a estratégia de analisar não somente a estrutura da UI a partir de uma tela, mas também outros elementos. Nguyen et al. (2018), por exemplo, analisa elementos avulsos, como uma parte da tela ou um elemento de IU e suas descrições e Behrang et al. (2018) utiliza o *sketch* e um conjunto

de palavras-chave. Apenas Mustafaraj et al. (2017), Svanberg (2017) e Turbak et al. (2017) não analisam a estrutura da UI, considerando somente elementos avulsos, como componentes e blocos de programação a partir do código do aplicativo.

No contexto da avaliação da originalidade compara-se um artefato "novo" com um universo de referência, de forma a identificar se e com quantos artefatos desse universo de referência o artefato "novo" é similar. Neste contexto, Mustafaraj et al. (2017), Svanberg (2017) e Turbak et al. (2017), consideram projetos criados por alunos a partir de tutoriais globais ou exemplos específicos de cursos para aprender a desenvolver apps com App Inventor. Já a maioria das abordagens utilizam como universo de referência, apps Android disponibilizados em repositórios públicos, como Google Play Store [Nguyen et al. 2018; Hu et al. 2020; Deka et al. 2017; Ge 2019], Apple App Store [Xie et al. 2019] e Github [Behrang et al. 2018].

Observou-se também que a maioria das abordagens encontradas são direcionadas ao uso profissional de design de UI [Deka et al. 2017; Nguyen et al. 2018]. Somente as abordagens relatadas por Mustafaraj et al. (2017), Svanberg (2017) e Turbak et al. (2017) são voltadas especialmente ao contexto educacional.

2.3.3. Quais técnicas de IA são adotadas pelos modelos de avaliação?

Várias abordagens utilizam medidas de similaridade/distância, que permitem medir o quanto dois aplicativos são parecidos. Behrang et al. (2018), por exemplo, faz uma busca de similaridade, buscando *screenshots* similares a partir de um *sketch*. Algumas abordagens usam essas medidas de similaridade/distância em conjunto com métodos de *clustering*. Mustafaraj et al. (2017) e Svanberg (2017), extraindo vetores de frequência de comandos do código de apps, utilizam medidas de similaridade/distância, como Euclidiana, Cosseno e Jaccard em conjunto com *clustering* e métodos de agrupamento hierárquico para classificar automaticamente os projetos como não originais ou originais.

Seguindo uma estratégia diferente, algumas abordagens utilizam *deep learning*, com modelos *Convolutional Neural Networks* (CNNs), comumente aplicados para analisar imagens [Valueva et al. 2020]. Xie et al. (2019), por exemplo, convertem a UI em uma representação de árvore rotulada como entrada para treinar o modelo CNN (VGGNet). Chen et al. (2020) treinam um CNN (CNN-*based image autoencoder architecture*) usando *wireframes* de design de IU não rotulados. Ge (2019) faz a comparação entre árvores e o esboço dos esqueletos da UI correspondente, utilizando o algoritmo *Largest Weighted Common Subtree Embeddings* (LWCSE).

Nguyen et al. (2018) utilizam redes neurais recorrentes e redes adversárias geradoras para identificar a similaridade de *screenshots* de apps representados em forma descrição em linguagem natural, descrevendo o design da UI. Para isso, utilizam *Deep-Visual*, uma *Recurrent Neural Network* (RNN), que é treinada para aprender os padrões de design de UIs de apps.

Somente Chen et al (2019) utiliza algoritmos genéticos em conjunto com medidas de similaridade/distância a partir do texto extraído da UI para medir a similaridade dos componentes da IU usando o modelo *Latent Dirichlet Allocation*

(LDA). O algoritmo genético é utilizado para otimizar a eficiência do tempo de determinar as correspondências quase ótimas de componentes da UI.

Várias abordagens utilizam técnicas de *deep learning* para buscar em um conjunto de dados as UIs similares e calcular uma pontuação de similaridade entre o *sketch* e todas as IUs encontradas, apresentando um modelo que foi treinado para automaticamente comparar a pontuação de similaridade entre o *sketch* e a UI correspondente [Behrang et al. 2018; Xie et al. 2019; Chen et al. 2019].

2.3.4. Como foi feita a preparação do conjunto de dados e a rotulação?

Várias abordagens utilizam como tipos de entradas *sketches* e/ou *screenshots*. Behrang et al. (2018) e Xie et al. (2019), por exemplo, partem de um *sketch* feito à mão. Algumas abordagens utilizam uma tela, uma parte da tela ou um elemento de UI em conjunto com suas descrições [Nguyen et al. 2018]. Outras abordagens usam componentes e blocos de programação, como as abordagens de Mustafaraj et al. (2017), Turbak et al. (2017) e Svanberg (2017). Somente uma das abordagens utiliza a similaridade de texto na interface para medir a similaridade dos componentes da IU [Chen et al. (2019)].

A maioria das abordagens realiza algum tipo de pré-processamento dos dados de entrada. Algumas abordagens utilizam técnicas de *deep learning* para criar *wireframe* e *encode* em forma de vetor de característica (*autoencoder*) [Chen et al. 2020]. Analisando a similaridade de texto, é utilizado o modelo *latent dirichlet allocation* (LDA) para modelar um documento e misturar as palavras, após misturar as palavras são utilizadas duas ferramentas para calcular a similaridade de texto [Chen et al. (2019)]. Por outro lado [Mustafaraj et al. 2017; Turbak et al. 2017; Svanberg 2017] extraem vetores de frequência de uso de comandos/blocos a partir do código de apps criados com App Inventor e rotulam manualmente os apps em original ou não original. Com relação aos conjuntos de dados utilizados para o treinamento de modelos de ML, observa-se que, em geral, a quantidade de instâncias varia entre menos de 1000 apps a quantidades que chegam próximo a meio milhão de apps (Figura 3).

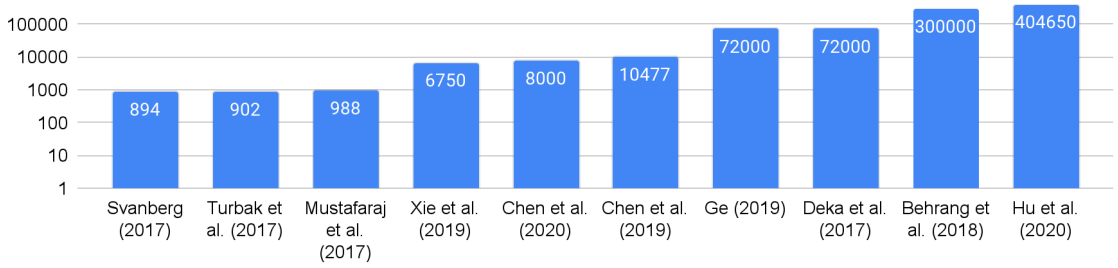

Figura 3. Quantidade de instâncias
(Nguyen et al. (2018) não relata a quantidade de instâncias utilizada)

2.3.5. Como é medido e qual é o desempenho das abordagens?

A maioria dos trabalhos apresenta avaliações do desempenho das abordagens. Porém, observa-se que são utilizados diferentes tipos de medidas, dependendo do tipo de avaliação adotado, para apresentar o desempenho das abordagens (Tabela 3).

Tabela 3. Desempenho das abordagens

Referência	Medida/método de análise	Resultado
Xie et al. (2019)	*Intraclass Correlation Coefficient* (ICC)	ICC = 0.725 a 0.889, média de 0.822 ($p < 0.0001$)
	Coeficiente de Correlação de Pearson (PPC)	PPC = 0.883
	Mean Squared Error (MSE)	MSE = 0.027
Ge (2019)	Comparação de um escore de similaridade com ranking	A abordagem encontra as 6 GUIs mais similares a uma GUI de consulta
Behrang et al. (2018)	Tau de Kendall	TAU = Valores variando de 0.6 a 1 (p variando de 0.015 a 0.142)
	Rho de Spearman	RHO = Valores variando de 0.7 a 1 (p variando de 0.037 a 0.188)
Mustafaraj et al. (2017)	Acurácia	Acurácia = 0.89 (considerando uma distância de Jaccard a um threshold de 0.4)
Svanberg (2017)	Comparação de medidas por meio de revocação (*recall*)	Revocação > 0.8 (Cosseno, Jaccard, Euclidiana e Cityblock sem grandes diferenças)
	Comparação de seleção de *features* por meio de revocação (*recall*)	Revocação > 0.9 (apenas blocos ou blocos com componentes) Revocação > 0.5 (apenas componentes)
	Comparação de normalização de *features* por meio de revocação (*recall*)	Revocação > 0.7 (TF-IDF e Count) Revocação > 0.5 (Binário)
Chen et al. (2019)	*Average Precision* (AP) para 30 apps	AP = 0.42
	Mean Average Precision (MAP) para 30 apps	MAP (30) = 0.418
Chen et al. (2020)	Precision@k (Pre@k) (k=1)	Precision@1= 70% (component scaling ratio 20%) Precision@1= 84.6% (component-removal ratio 20%) Precision@1= 70.6% (fully connected neural network baseline ratio 20%) Precision@1= 47.6% (fully connected neural network baseline ratio 30%) Precision@1= 0.5 (scaling-10% and removal-20%)
	Mean Reciprocal Rank (MRR)	MRR = 0.73 (component scaling ratio 20%) MRR = 0.88 (component-removal ratio 20%) MRR = 0.77 (fully connected neural network baseline ratio 20%) MRR = 0.57 (fully connected neural network baseline ratio 30%) MRR = 0.62 (scaling-10% and removal-20%)
	Relevância usando Precision@k (Pre@k) (k=1, 5 e 10) e *Mean Reciprocal Rank* (MRR)	Relevância precision@1 = 0.44 Relevância precision@5 = 0.40 Relevância precision@10 = 0.38 MRR = 0.59
Hu et al. (2020)	Eficácia (razão falso-positivos (FPR) e razão falso-negativos (FNR))	FPR = 0.02% FNR = 0.42%
	Eficiência (Tempo em segundos (T))	T = 0.053s para comparar um par de apps
Deka et al. (2017)	Não relatado	Não relatado
Nguyen et al. (2018)	Não relatado	Não relatado
Turbak et al.	Não relatado	Não relatado

(2017)		

Abordagens que propõem um ranking automatizado de originalidade de vários apps utilizam medidas de estatística para avaliar o desempenho da correlação entre rankings, como Rho de Spearman, Tau de Kendall, Coeficiente de Correlação de Pearson, etc. [Ge 2019; Xie et al. 2019; Behrang et al. 2018]. Por outro lado, abordagens que propõem a avaliação do grau de originalidade de um app, utilizam medidas como acurácia [Mustafaraj et al. 2017], revocação [Svanberg 2017], eficácia [Hu et al. 2020], etc. Cabe ressaltar que apenas uma abordagem também realiza uma avaliação da eficiência, buscando apresentar como o uso de um algoritmo genético auxilia na rapidez do processamento da análise de originalidade [Hu et al. 2020].

De forma geral os resultados apresentam bom desempenho, tanto em medidas de correlação, como de acurácia, revocação, etc. No entanto, algumas abordagens apresentam uma comparação cujos resultados denotam que algumas configurações podem ser melhores/piores, como os resultados de comparação apresentados por Svanberg (2017) em que apenas blocos ou blocos com componentes apresentam resultados superiores a utilizar somente componentes do App Inventor.

2.3.6 Ameaças à validade

Como em qualquer mapeamento sistemático, existem possíveis ameaças à validade dos resultados. Um dos principais riscos é o de omitir estudos relevantes. Para mitigá-lo, a *string* de busca foi construída buscando utilizar todos os termos relevantes e incluindo sinônimos. Além disso, a busca foi realizada em diversas bases de dados científicas, reduzindo as chances de omitir estudos relevantes. Para mitigar riscos na seleção dos estudos e extração de dados, foi utilizada uma definição detalhada dos critérios de inclusão e exclusão. Foi definido e documentado um protocolo rígido para a seleção do estudo, que foi realizada e revisada cuidadosamente em conjunto entre os autores até que um consenso foi atingido.

3. Conclusão

O presente artigo expõe um mapeamento de abordagens existentes para avaliar automaticamente a originalidade de design de UI de apps Android usando técnicas de Inteligência Artificial (IA). Foram identificadas 11 abordagens diferentes que contêm alguma forma a avaliação da originalidade do design de interfaces de apps Android. Os resultados demonstram que o tema de avaliação de originalidade enfocando no design de UI de apps Android é recente e ainda pouco abordado. A maioria das abordagens avalia a originalidade utilizando como base a estrutura do *layout* do aplicativo, seja por meio de *screenshots* ou *sketches*. Foram encontradas somente três abordagens voltadas ao contexto educacional referentes a apps criados com App Inventor avaliando critérios enfocando características do App Inventor, como blocos e componentes. As abordagens encontradas variam em termos de conjuntos de dados utilizando ou dados próprios ou de repositórios públicos. Os resultados também demonstram grande variação nos modelos de IA adotados, apontando que há diversos possíveis caminhos para o problema em questão. Em termos de resultados de desempenho, a maioria das abordagens alcançou bons resultados. Assim, visando a automatização da avaliação da originalidade de apps desenvolvidos no contexto do ensino de computação observa-se

ainda uma grande lacuna apontando a necessidade de mais pesquisas nesta área não somente melhorando o estado da arte mas também explorando alternativas. Além de uma integração mais integral dentro de um modelo de avaliação mais completa e sistematicamente validado.

Porém, a maioria é utilizada no contexto profissional, sendo que poucas estão relacionadas ao contexto educacional. Várias abordagens focam no processo de desenvolvimento da solução e não em detalhes do modelo resultante, resultando na falta de informações sobre a avaliação do desempenho, como fatores avaliados, métodos de coleta de dados, tamanho da amostra utilizada e suas respectivas descobertas. De forma geral, as abordagens não focam tanto na avaliação e divulgação dos resultados, dificultando a comparação entre os modelos encontrados.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001 e do Conselho Nacional de Desenvolvimento Científico e Tecnológico – Brasil (CNPq).

Referências

Alves, N. da C., Alberto, M., Gresse von Wangenheim, C. (2021) Análise Automatizada da Originalidade de Aplicativos Android no Contexto Educacional: Um Mapeamento da Literatura. Anais do 29º WEI – Workshop sobre Educação em Computação, online.

Alves, N. da C., Gresse von Wangenheim, C., Alberto, M.; Martins-Pacheco, L. H. (2020) Uma Proposta de Avaliação da Originalidade do Produto no Ensino de Algoritmos e Programação na Educação Básica. Anais do Xxxi Simpósio Brasileiro de Informática na Educação. SBC. http://dx.doi.org/10.5753/cbie.sbie.2020.41.

Alves, N. da C., Gresse von Wangenheim, C., Hauck, J. C.R. (2019) Approaches to Assess Computational Thinking Competences Based on Code Analysis in K-12 Education: a systematic mapping study. Informatics In Education, 18(1), p. 17-39. http://dx.doi.org/10.15388/infedu.2019.02.

Basu, S. (2019) Using Rubrics Integrating Design and Coding to Assess Middle School Students' Open-ended Block-based Programming Projects. Proceedings Of The 50Th Acm Technical Symposium On Computer Science Education, p. 1211-1217. ACM. http://dx.doi.org/10.1145/3287324.3287412.

Beaty, R., Johnson, D. R. (2020) Automating Creativity Assessment with SemDis: an open platform for computing semantic distance. Psyarxiv, 53, p. 757-780. http://dx.doi.org/10.31234/osf.io/nwvps.

Cavallo, D., Singer, H., Gomes, A., Bittencourt,I., Silveira, I. (2016). Inovação e Criatividade na Educação Básica: Dos conceitos ao ecossistema. Revista Brasileira de Informática na Educação, 24(2).

Ferreira, M. N. F., Gresse von Wangenheim, C., Goncalves, B. S., Hauck, J.C.R., Medeiros, G. (2020) Ensino de Design Visual de Aplicativos Móveis no Ensino

Fundamental. In: Anais da XI Conferência Computer on the Beach, Balneário Camboriú/SC, p. 199-205.

Gomes, K. P. e Matos, S. N. (2020). Detection of Programming Plagiarism in Computing Education: A Systematic Mapping Study. In: Anais do Simpósio ui; Brasileiro de Informática na Educação, Natal, Brasil.

Hu, R. M., Cai, L., Chen, W. (2020) Detection and Segmentation of Graphical Elements on GUIs for Mobile Apps Based on Deep Learning. Lecture Notes Of The Institute For Computer Sciences, Social Informatics And Telecommunications Engineering, p. 187-197. Springer International Publishing. http://dx.doi.org/10.1007/978-3-030-64214-3_13.

Ichinco, M. Kelleher, C. (2018) The Example Guru: Suggesting Examples to Novice Blocks Programmers in an Artifact-Based Context Increases Use of New Blocks. In: Blocks + 2018, Boston, p. 1-2.

Jeong, J., Kim, N., In, H. P. (2020) Detecting usability problems in mobile applications on the basis of dissimilarity in user behavior. International Journal Of Human-Computer Studies, v. 139, p. 102364. Elsevier BV. http://dx.doi.org/10.1016/j.ijhcs.2019.10.001.

Li, L., Bissyandé, T. F., Wang, H., Klein, J. (2019) On Identifying and Explaining Similarities in Android Apps. Journal Of Computer Science And Technology, [S.L.], v. 34, n. 2, p. 437-455. Springer Science and Business Media LLC. http://dx.doi.org/10.1007/s11390-019-1918-8.

Lyu, F., Lin, Y., Yang, J., Zhou, J. (2016) SUIDroid: an efficient hardening-resilient approach to android app clone detection. 2016 Ieee Trustcom/Bigdatase/Ispa, [S.L.], p. 511-518. IEEE. http://dx.doi.org/10.1109/trustcom.2016.0104.

Martins, O. P. H. R. (2019) Desenvolvimento de um Modelo para Avaliação da Estética Visual de Interfaces de Usuários de Aplicativos Usando Deep Learning. 2019. 116 f. TCC (Graduação) - Curso de Ciência da Computação, Universidade Federal de Santa Catarina. Florianópolis.

MIT. (2020) MIT App Inventor, http://appinventor.mit.edu/, Outubro, 2020.

Patton, E. W., Tissenbaum, M., Harunani, F. (2019) MIT App Inventor: objectives, design, and development. Computational Thinking Education, 3, p. 31-49. http://dx.doi.org/10.1007/978-981-13-6528-7_3.

Petersen, K., Vakkalanka, S., Kuzniarz, L. (2015) Guidelines for conducting systematic mapping studies in software engineering: an update. Information and Software Technology, 64, p. 1-18. http://dx.doi.org/10.1016/j.infsof.2015.03.007.

Piasecki, J., Waligora, M. & Dranseika, V. (2018) Google Search as an Additional Source in Systematic Reviews. Sci Eng Ethics 24, p. 809–810.

Ritchie, G. (2001) Assessing Creativity. Proceedings of the AISB symposium on AI and creativity in arts and science. York: The Society for the Study of Artificial Intelligence and Simulation of Behaviour. p. 3–11.

Runco, M. A., Jaeger, G. J. (2012) The Standard Definition of Creativity. Creativity Research Journal, 24(1), p. 92-96. http://dx.doi.org/10.1080/10400419.2012.650092.

Skager, R. W.; Schultz, C. B.; Klein, S. P. (1966) Points of view about preference as tools in the analysis of creative products. Perceptual and Motor Skills, v. 22, p. 83-94.

Soh, C., Tan, H. B. K., Arnatovich, Y. L., Wang, L. (2015) Detecting Clones in Android Applications through Analyzing User Interfaces. Ieee 23Rd International Conference On Program Comprehension, p. 163-173. IEEE. http://dx.doi.org/10.1109/icpc.2015.25.

Snyder, H. T., Hammond, J. A., Grohman, M. G., Katz-Buonincontro, J. (2019) Creativity measurement in undergraduate students from 1984–2013: A systematic review. Psychology of Aesthetics, Creativity, and the Arts, 13(2), 133–143.

WEF. (2020) The Future of Jobs Report 2020. 2020. Disponível em: https://www.weforum.org/reports/the-future-of-jobs-report-2020, Junho, 2021.

Yadav, Aman; Cooper, Steve. (2017) Fostering creativity through computing. Communications of the ACM, 60(2), p. 31-33. http://dx.doi.org/10.1145/3029595.

Zen, M., Vanderdonckt, J. (2016) Assessing User Interface Aesthetics based on the Inter-subjectivity of Judgment. International BCS Human Computer Interaction Conference, BCS Learning & Development, p. 1-12. http://dx.doi.org/10.14236/ewic/hci2016.25.

Zhang, T., Liu, Y., Gao, J., Gao, L. P., Cheng, J. (2020) Deep Learning-Based Mobile Application Isomorphic GUI Identification for Automated Robotic Testing. Ieee Software, v. 37, n. 4, p. 67-74. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/ms.2020.2987044.