

Uso de Evidências Baseadas no Comportamento dos Alunos para Identificação de Plágio em Ambientes de Juízes Online

José Lázaro M. de L. Filho¹, Rafael Giusti¹, Leandro Carvalho¹,
Horácio Fernandes¹, David B. F. Oliveira¹

¹ Instituto de Computação – Universidade Federal do Amazonas (UFAM)
Av. General Rodrigo Octávio, 6200 - Coroado I - 69080-900 - Manaus

{jose.marques, rfgiusti, galvao, horacio, david}@icomp.ufam.edu.br

Abstract. *Logs generated by online judges can provide behavioral evidence associated with the transcription of ready-made solutions in programming activities. This work proposes a machine learning-based method to identify coding sessions with patterns compatible with possible copying behavior. The approach uses 19 features extracted from interaction logs, including keystroke latency, editing actions, event proportions, focus loss, and pause periods. To evaluate the method, a dataset of 155 coding sessions was constructed, and nine classification models were trained using nested cross-validation. The best result was achieved by the Gradient Boosting model, with a macro F1-score of 0.89, precision of 0.91, recall of 0.90, and accuracy of 0.89. These results suggest that behavioral features extracted from such logs can support the identification of sessions with indicators of copying behavior.*

Resumo. *Logs gerados por juízes online podem oferecer evidências comportamentais associadas à transcrição de soluções prontas em atividades de programação. Este trabalho propõe um método baseado em aprendizagem de máquina para identificar sessões de codificação com padrões compatíveis com possível comportamento de cópia. A abordagem utiliza 19 características extraídas de logs de interação, incluindo latência entre teclas, ações de edição, proporções de eventos, perda de foco e períodos de pausa. Para avaliar o método, foi construído um conjunto de dados próprio com 155 sessões de codificação, e nove modelos de classificação foram treinados por meio de validação cruzada aninhada. O melhor resultado foi obtido pelo modelo Gradient Boosting, que alcançou 0,89 de F1-score macro, 0,91 de precisão, 0,90 de revocação e 0,89 de acurácia. Esses resultados sugerem que características comportamentais extraídas desses registros podem apoiar a identificação de sessões com indícios de cópia.*

1. Introdução

Juízes online são amplamente utilizados no ensino de programação por permitirem a submissão, execução e correção automática de soluções de código-fonte, além de apoiarem o acompanhamento do desempenho dos estudantes em atividades práticas [Alves et al. 2019, Oliveira et al. 2021]. Esses ambientes também fornecem *feedback* imediato e registram as diferentes interações realizadas durante a resolução das atividades, como edições de código, execuções, submissões e eventos de interação com a

interface. Esses registros podem oferecer informações relevantes sobre o comportamento do estudante durante o processo de codificação, incluindo padrões de digitação que refletem características individuais da interação com o teclado [Khan et al. 2023, Ljubovic and Pajic 2020, Shadman et al. 2025, Dillon and De Marsico 2025].

A maioria das estratégias de detecção de plágio em disciplinas de programação baseia-se na comparação entre pares de códigos-fonte, empregando técnicas de análise de similaridade textual, estrutural ou baseada em fragmentos de código [Prechelt et al. 2002, Schleimer et al. 2003, Gitchell and Tran 1999]. Embora eficazes em diversos contextos, essas abordagens apresentam limitações em cenários nos quais a solução é obtida a partir de fontes externas, como fóruns, repositórios ou Modelos de Linguagem de Grande Escala (LLMs) [Prather et al. 2023, Becker et al. 2023], e, portanto, não está disponível para comparação direta entre códigos-fonte.

Juízes online podem ser configurados para bloquear operações de *copy/paste*, de modo que estudantes que obtenham soluções a partir de fontes externas, sejam obrigados a digitá-las manualmente na IDE do sistema. Embora esse bloqueio dificulte a reprodução direta de soluções prontas pelos alunos, ele não impede que os estudantes realizem a transcrição manual do código obtido externamente. Diante desse cenário, este trabalho investiga o uso de evidências comportamentais extraídas dos logs de interação de um juiz online para identificar sessões de codificação com indícios de transcrição de soluções previamente obtidas. Parte-se da hipótese de que estudantes que desenvolvem suas próprias soluções apresentam padrões de interação com a IDE distintos daqueles observados em estudantes que apenas transcrevem o código. Essas diferenças podem se manifestar em características comportamentais como a latência entre teclas, a ocorrência de rajadas de digitação, a duração e frequência de pausas, a distribuição dos diferentes tipos de eventos registrados e a relação entre o volume de inserções realizadas e o tempo total da sessão de programação.

Para avaliar essa hipótese, foi construída uma base de dados rotulada a partir de sessões de codificação realizadas no juiz on-line *CodeBench*. A partir dos logs coletados, foram extraídas 19 características comportamentais e avaliados nove modelos de classificação. Os resultados indicam que as características extraídas dos logs são capazes de discriminar padrões entre os grupos analisados. O modelo Gradient Boosting obteve o melhor desempenho, alcançando 0,89 de acurácia, 0,91 de precisão, 0,90 de revocação e 0,89 de F1-score macro.

Este trabalho está dividido em oito seções. A Seção 2 apresenta os trabalhos relacionados à detecção de plágio em exercícios de programação, com ênfase nos casos em que o código é obtido de fontes externas. A Seção 3 descreve a metodologia proposta, a extração das características e o protocolo de treinamento e avaliação dos modelos. A Seção 4 apresenta a estratégia adotada para construção do dataset experimental. A Seção 5 apresenta os resultados obtidos, com a comparação entre os modelos avaliados, a análise do impacto do filtro temporal e a matriz de confusão do modelo de melhor desempenho. A Seção 6 discute as limitações do trabalho. A Seção 7 apresenta as conclusões e os trabalhos futuros. Por fim, a Seção 8 apresenta a declaração sobre uso de Inteligência Artificial.

2. Trabalhos Relacionados

A detecção de plágio em atividades de programação tem sido amplamente investigada por meio de métodos baseados na comparação entre códigos-fonte submetidos. Ferramentas como JPlag [Prechelt et al. 2002], MOSS [Schleimer et al. 2003] e Sim [Gitchell and Tran 1999] analisam similaridades textuais, estruturais ou baseadas em fragmentos de código para identificar possíveis cópias entre soluções. Embora eficazes quando os pares de código estão disponíveis para comparação, essas abordagens apresentam limitações em cenários nos quais o estudante obtém o código a partir de fontes externas, como fóruns, repositórios públicos ou LLMs.

Diante dessas limitações, alguns trabalhos passaram a explorar informações relacionadas ao processo de desenvolvimento do código. Schneider et al. (2018) e Oliveira et al. (2021) analisaram registros de edição produzidos durante a construção das soluções, demonstrando que informações do processo de desenvolvimento podem complementar técnicas tradicionais de detecção de plágio. De forma semelhante, Ljubovic e Pajic (2020) extraíram características comportamentais de repositórios de granularidade ultrafina, mostrando que padrões de uso da IDE podem auxiliar na identificação de comportamentos suspeitos. Li et al. (2023) extraíram features comportamentais do processo de codificação para identificar plágio em atividades de programação, obtendo resultados promissores. No entanto, essas estratégias dependem de eventos de copy/paste como uma importante fonte de evidências para a detecção de plágio, tornando-se menos eficazes em juízes on-line que bloqueiam esse tipo de interação durante o desenvolvimento das soluções.

Fora do contexto de programação, Trezise et al. (2019) relatam um padrão comportamental qualitativamente compatível com o observado neste trabalho: tarefas de transcrição apresentaram rajadas de digitação mais longas e regulares e menor frequência de pausas em comparação a tarefas de escrita autêntica. Os resultados, porém, não são diretamente comparáveis, uma vez que os autores empregaram clusterização não supervisionada em um domínio distinto (produção textual), sem métricas de classificação equivalentes às reportadas aqui. Khan et al. (2023) também extraíram características a partir de eventos de teclado em contexto de programação, porém com o objetivo de prever desempenho acadêmico dos estudantes, e não de identificar padrões de cópia ou transcrição.

A crescente capacidade de LLMs em gerar soluções funcionais para exercícios de programação representa um desafio adicional para os mecanismos tradicionais de detecção de plágio [Prather et al. 2023].

Diferentemente de abordagens centradas apenas na similaridade entre códigos, este trabalho investiga evidências comportamentais registradas em logs de um juiz on-line. A proposta busca identificar sessões com padrões compatíveis com a transcrição de soluções previamente obtidas, considerando características como latência entre teclas, eventos de edição, execuções, submissões, perda de foco e interações com a interface. Em contraste com trabalhos anteriores baseados em comportamento, o método proposto não utiliza eventos de *copy/paste* como fonte de evidências, permitindo sua aplicação mesmo em ambientes nos quais esse tipo de interação é bloqueado ou não é registrado pela IDE.

3. Método Proposto

Nesta seção é apresentado o método proposto para identificar sessões de codificação com padrões comportamentais compatíveis com a transcrição de soluções previamente obtidas. A abordagem pressupõe a existência de uma base de dados rotulada, composta por sessões de codificação nas quais se conhece previamente se o estudante desenvolveu a solução diretamente na plataforma ou se realizou a transcrição manual de uma solução obtida externamente. A Seção 4 descreve o protocolo utilizado para a construção da base empregada neste trabalho. A partir dessa base, são extraídas características comportamentais dos logs de interação registrados pelo juiz online, com o objetivo de representar os padrões de interação associados ao desenvolvimento próprio e à transcrição de soluções, conforme descrito na seção a seguir.

3.1. Extração e Engenharia de Características

O método baseia-se na hipótese de que estudantes que desenvolvem suas próprias soluções apresentam padrões de interação com a IDE distintos daqueles observados em estudantes que transcrevem manualmente soluções obtidas externamente. Essas diferenças podem se manifestar em aspectos como o ritmo de digitação, a frequência e a duração das pausas, o padrão de edição do código e a forma como o estudante interage com os recursos da plataforma.

Com base nessa hipótese, são extraídas características comportamentais dos logs de interação registrados pelo juiz online. As características consideradas neste trabalho podem ser organizadas em cinco grupos: (i) latência entre inserções consecutivas de caracteres, (ii) rajadas de digitação, (iii) períodos de pausa, (iv) proporções de eventos e (v) relação entre volume de interações e tempo total da sessão. Em conjunto, esses grupos buscam capturar tanto aspectos temporais da digitação quanto padrões mais gerais de interação com o ambiente.

As características são extraídas das sessões de codificação, cada uma correspondente às interações realizadas por um estudante durante a resolução de um exercício. Os eventos de inserção de caracteres são filtrados e, com base nos *timestamps* associados a esses eventos, calculam-se as diferenças temporais entre inserções consecutivas, formando uma sequência de latências por sessão. Esses eventos são utilizados porque representam diretamente o processo de digitação do estudante, aspecto central para distinguir sessões em que uma solução é construída daquelas em que uma solução obtida externamente é apenas transcrita para o ambiente.

A sequência de latências é segmentada em dois grupos com interpretações distintas: intervalos curtos, associados à digitação ativa, e intervalos longos, associados a pausas na atividade. Os limiares temporais dessa segmentação são definidos a partir da análise da distribuição das latências e da avaliação do desempenho dos classificadores, conforme descrito na Seção 5. Dessa forma, em novos conjuntos de dados, o mesmo procedimento pode ser replicado para ajustar os limiares ao comportamento temporal observado.

Nas seções seguintes, cada grupo de características é descrito em detalhe, começando pela latência entre inserções consecutivas de caracteres.

3.1.1. Latência entre Inserções Consecutivas de Caracteres

Espera-se que a transcrição manual produza um ritmo de digitação distinto do desenvolvimento próprio, capturado pelas latências entre inserções consecutivas de caracteres.

As latências são filtradas dentro de um intervalo mínimo e máximo correspondente à faixa em que se concentra a maior parte dos períodos de digitação ativa.

A partir do vetor de latências filtrado de cada sessão estudante-exercício, são extraídas duas características: a média e o desvio padrão dos intervalos entre inserções consecutivas de caracteres. Essas medidas representam, respectivamente, a tendência central e a variabilidade do ritmo de digitação da sessão.

3.1.2. Rajadas de Digitação

Uma rajada de digitação, ou *burst*, corresponde a uma sequência contínua de inserções de caracteres separadas por intervalos curtos, sem pausas prolongadas [Khan et al. 2023]. Neste trabalho, considera-se que uma rajada termina quando o intervalo entre duas inserções consecutivas ultrapassa o limiar estabelecido para caracterizar uma pausa na atividade.

Espera-se que sessões de cópia apresentem rajadas mais longas e regulares, e sessões de desenvolvimento, rajadas mais curtas e heterogêneas, intercaladas por pausas de reflexão.

Para cada rajada, são calculados seu tamanho, definido como a quantidade de teclas pressionadas, e sua duração, medida pela soma dos intervalos entre pressionamentos consecutivos na sequência. Os valores obtidos para todas as rajadas de uma sessão são agregados por meio da média e do desvio padrão, resultando em quatro características: média e desvio padrão do tamanho das rajadas, e média e desvio padrão da duração das rajadas.

3.1.3. Períodos de Pausa

Além das latências associadas à digitação ativa, são extraídas características relacionadas aos períodos de pausa observados durante a sessão de codificação. Essas pausas podem refletir momentos de leitura do enunciado, de reflexão sobre a solução, de revisão da lógica implementada ou de interrupção temporária da atividade.

Espera-se que sessões de desenvolvimento apresentem pausas mais frequentes e distribuídas, enquanto sessões de cópia concentrem a atividade em digitação contínua, com menos interrupções.

Para representar diferentes níveis de interrupção, os intervalos de pausa são agrupados em cinco faixas temporais a partir do limiar de 5 segundos: de 5 a 10 segundos, de 10 a 15 segundos, de 15 a 30 segundos, de 30 a 60 segundos e acima de 60 segundos. Essa segmentação busca capturar padrões de pausas curtas, intermediárias e longas durante a sessão de codificação. Pausas mais longas podem indicar afastamento temporário da atividade ou retomada posterior da resolução, em contraste com as pausas curtas associadas à reflexão imediata.

Cada faixa temporal origina uma característica correspondente à quantidade de pausas observadas naquele intervalo, totalizando cinco características.

3.1.4. Proporções de Eventos

Este conjunto de características está relacionado à proporção entre determinados tipos de eventos e o total de ações registradas na sessão. Essa normalização é adotada porque a contagem bruta de eventos pode ser influenciada pelo tamanho da solução, pela complexidade do exercício ou pela duração da sessão. Assim, as razões entre eventos e total de ações permitem representar o comportamento relativo do estudante independentemente do tamanho do *log*.

São consideradas proporções associadas aos principais eventos observados durante a interação com o ambiente: inserção, remoção, seleção, submissão, execução, perda de foco do editor e desfazer. Essas características buscam capturar diferenças nos padrões de edição, navegação e interação dos estudantes durante a resolução das atividades.

Além dessas proporções, é extraída uma característica adicional correspondente à razão entre o número de eventos de remoção e o número de eventos de inserção. Espera-se que sessões de desenvolvimento apresentem maior proporção de remoções, refletindo ajustes e reformulações, enquanto sessões de cópia demandem menos revisões estruturais.

3.1.5. Relação com o Tempo Total no Exercício

O último conjunto de características considera a relação entre a quantidade de eventos registrados e o tempo total da sessão. O tempo total é calculado pela diferença entre os *timestamps* do último e do primeiro evento registrados na sessão estudante-exercício.

Espera-se que sessões de desenvolvimento distribuam interações ao longo do tempo necessário à resolução, enquanto sessões de cópia concentrem eventos em intervalos menores, refletindo reprodução do código.

A Tabela 1 apresenta um resumo das características propostas, organizadas por grupo, descrição e quantidade.

Tabela 1. Consolidação das características extraídas para os modelos de classificação.

Grupo	Descrição	Quantidade
Latência entre Teclas	Média e desvio padrão dos intervalos entre teclas associados à digitação ativa.	2
Rajadas de Digitação	Média e desvio padrão do tamanho e da duração das rajadas de digitação.	4
Períodos de Pausa	Contagem de pausas distribuídas em cinco faixas temporais.	5
Proporções de Eventos	Razão de tipos de eventos pelo total de ações e razão entre remoções e inserções.	7
Relação com o Tempo Total	Razão entre o total de eventos e o tempo total da sessão.	1
Total		19

3.2. Protocolo Experimental

Definidas as características comportamentais, passa-se à definição da *pipeline* experimental do método proposto. Essa *pipeline* abrange desde a criação da tabela de características até a avaliação final dos modelos.

3.2.1. Construção do Conjunto de Características

O conjunto de características é organizado em uma tabela na qual cada linha corresponde a um par estudante-exercício e cada coluna representa uma das 19 características descritas anteriormente. Cada linha sintetiza o comportamento de um estudante durante a resolução de um exercício específico.

Para compor o conjunto, são incluídas sessões que atendam a pelo menos um dos seguintes critérios: o estudante realizou ao menos uma submissão correta; ou acumulou no mínimo 100 eventos de inserção de caracteres. O segundo critério é adotado porque sessões sem submissão correta podem ainda conter evidências de resolução efetiva, uma vez que eventos de inserção refletem diretamente a atividade de digitação do estudante. A combinação dos critérios permite ampliar o conjunto sem comprometer a representatividade das amostras.

3.2.2. Treinamento e Avaliação dos Modelos

Para validar o método, nove modelos de classificação são treinados e avaliados: *Logistic Regression*, *Support Vector Machine* (SVM), *Decision Tree*, *Naive Bayes*, *K-Nearest Neighbors* (KNN), *Random Forest*, *AdaBoost*, *Gradient Boosting* e *XGBoost*. Esses algoritmos foram selecionados por sua ampla utilização em problemas de classificação supervisionada [Sarker 2021], bem como pela frequente competitividade de modelos baseados em árvores e *ensembles* em dados tabulares [Grinsztajn et al. 2022, Chen and Guestrin 2016].

As características são normalizadas por meio do *StandardScaler*, e o treinamento adota uma estratégia de validação cruzada aninhada (*Nested Cross-Validation*), composta por dois *loops* de *StratifiedGroup 5-Fold*: um *loop* externo, responsável pela avaliação final do modelo, e um *loop* interno, responsável pelo ajuste de hiperparâmetros, evitando que o desempenho reportado seja otimista. Além disso, essa estratégia assegura a separação de estudantes entre os conjuntos de treino e teste, evitando que o modelo aprenda padrões individuais de comportamento em vez de padrões associados à transcrição. No *loop* externo, o conjunto de teste é reservado para avaliação final; no *loop* interno, aplicado apenas sobre os dados de treino de cada partição externa, é executada a otimização de hiperparâmetros por meio de busca aleatória (*Random Search*), avaliando 30 combinações por modelo. Para garantir replicabilidade e estabilidade estatística, os dados são embaralhados previamente com semente aleatória fixada em 42 para todas as divisões.

A métrica monitorada durante o ajuste fino é o F1-score macro, escolhido para que os modelos deem igual atenção a ambas as classes. O melhor classificador de cada iteração tem suas métricas extraídas a partir da matriz de confusão: acurácia, revocação (*recall*), precisão e F1-score macro. Ao fim do processo, calcula-se a média dessas métricas para comparação do desempenho geral, e as matrizes de confusão de cada

iteração são acumuladas para fornecer uma visão global dos erros e acertos dos modelos.

3.3. Cuidados Éticos

A coleta de dados foi realizada mediante a assinatura do Termo de Consentimento Livre e Esclarecido (TCLE) por todos os participantes, previamente informados sobre os objetivos, procedimentos e destinação das informações coletadas, em atendimento aos princípios éticos da Resolução CNS nº 510/2016. A participação foi espontânea em todas as coletas, inclusive na terceira, realizada durante uma atividade avaliativa da disciplina: os estudantes que optaram por não autorizar o uso de seus dados na pesquisa realizaram a atividade normalmente, sem qualquer influência sobre sua avaliação. Foi assegurado a todos os participantes o direito de interromper a participação a qualquer momento, e o sigilo das informações foi preservado por meio da anonimização dos registros e do acesso restrito à equipe de pesquisa, de forma que nenhum participante é passível de identificação individual.

4. Construção do Dataset Experimental

A construção do conjunto de dados foi realizada a partir de logs gerados no ambiente de juiz online *CodeBench*, no qual operações de copiar e colar entre fontes externas e o editor de código são bloqueadas, exigindo que soluções obtidas externamente sejam digitadas manualmente na plataforma.

Os participantes foram estudantes voluntários, informados de que a pesquisa buscava identificar padrões de cópia ou transcrição de código.

O protocolo de coleta foi conduzido em três sessões presenciais de resolução de listas de exercícios, cada uma com duração de duas horas. Durante essas sessões, os estudantes interagiram com o editor do juiz online enquanto suas ações eram registradas automaticamente, incluindo eventos de inserção e remoção de caracteres, ações de desfazer, seleção de trechos de código, submissões, execuções e alternância de foco na interface. Cada sessão de codificação foi posteriormente associada a uma das classes utilizadas na tarefa de classificação.

Para a rotulação dos dados, foram definidos dois cenários de resolução. No primeiro, os estudantes deveriam desenvolver as soluções diretamente na plataforma, sem acesso a soluções previamente elaboradas, compondo o Grupo de Desenvolvimento. No segundo, os estudantes receberam soluções prontas em um papel impresso para os exercícios da lista e foram instruídos a digitá-las manualmente no sistema, compondo o Grupo de Cópia.

As soluções fornecidas ao Grupo de Cópia foram geradas por LLM nas duas primeiras sessões e elaboradas pelo professor da disciplina na terceira.

Com o objetivo de favorecer o balanceamento entre as classes e ampliar a diversidade dos registros coletados, o protocolo também permitiu uma dinâmica alternada, na qual um mesmo estudante desenvolvia parte dos exercícios e transcrevia soluções prontas na outra parte da lista. Assim, a unidade de análise adotada no conjunto de dados não foi a sessão completa do estudante, mas o par estudante-exercício, permitindo que exercícios diferentes resolvidos por um mesmo estudante fossem rotulados conforme o cenário efetivamente aplicado a cada um.

As atividades utilizadas na coleta contemplaram tópicos introdutórios e intermediários de programação, incluindo estruturas condicionais, estruturas de repetição, busca binária, recursão e algoritmos de ordenação. A escolha desses conteúdos buscou abranger exercícios com diferentes níveis de complexidade e diferentes demandas cognitivas, de modo a produzir sessões de codificação variadas para a análise comportamental. As soluções trabalhadas nesses exercícios apresentavam, em média, entre 15 e 20 linhas de código.

Ao final da coleta, foram obtidos logs de 205 pares estudante-exercício. Após a aplicação dos critérios de inclusão descritos na Seção 3, o conjunto final ficou composto por 155 sessões de codificação, sendo 85 pertencentes ao Grupo de Cópia e 70 ao Grupo de Desenvolvimento.

5. Experimentos e Resultados

Nesta seção são apresentados os resultados obtidos ao longo de duas etapas. Primeiro, analisa-se a distribuição das latências entre pressionamentos de teclas, que orienta a definição dos limiares temporais utilizados na extração de características. Em seguida, avalia-se o desempenho dos modelos de classificação, considerando o impacto do filtro temporal, bem como o comportamento detalhado do modelo de melhor desempenho.

5.1. Definição dos Limiares Temporais de Latência

Os limiares temporais utilizados na extração de características são definidos a partir da análise da distribuição das latências entre inserções consecutivas de caracteres.

5.1.1. Distribuição dos Intervalos entre Pressionamentos de Teclas

Os intervalos entre pressionamentos de teclas variam em uma escala ampla, de menos de 0,01 segundos a mais de 2.600 segundos, e por isso são representados na escala logarítmica de base 10. Essa base foi adotada por facilitar a interpretação direta dos valores em segundos: $\log_{10}(1) = 0$ corresponde a 1 segundo, $\log_{10}(10) = 1$ corresponde a 10 segundos, e assim sucessivamente. A partir dos dados transformados, são gerados gráficos de densidade (KDE) por grupo, conforme apresentado na Figura 1.

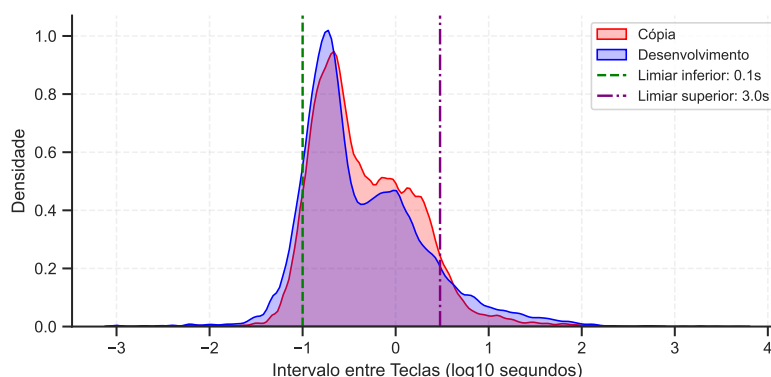


Figura 1. Distribuição dos intervalos entre pressionamentos de teclas em escala logarítmica.

Os intervalos de ambos os grupos concentram-se principalmente entre 0,1 e 3 segundos, faixa adotada como critério de filtragem para a extração das características de

digitação ativa. Valores abaixo de 0,1 segundo podem estar associados a eventos muito rápidos ou ruídos, enquanto valores acima de 3 segundos podem representar pausas ou interrupções na atividade.

O Grupo de Cópia apresenta um segundo pico de densidade entre 1 e 3 segundos, enquanto o Grupo de Desenvolvimento concentra-se em intervalos superiores a 3 segundos, confirmando o comportamento esperado e motivando os limiares adotados.

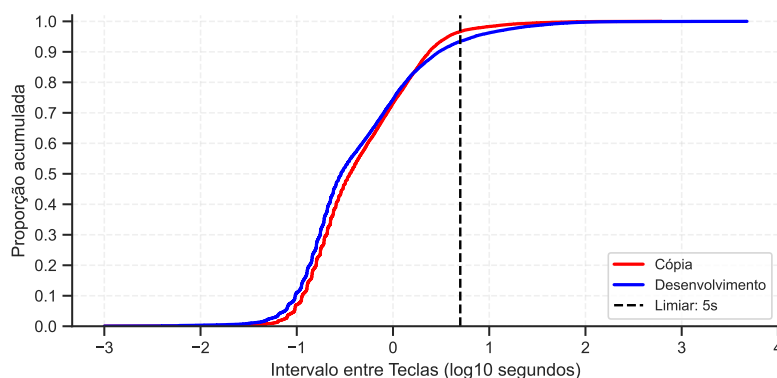


Figura 2. Distribuição acumulada dos intervalos entre pressionamentos de teclas em escala logarítmica.

Para complementar a análise, gerou-se também o gráfico da função de distribuição acumulada empírica (ECDF), apresentado na Figura 2, que representa a proporção acumulada das latências em função do intervalo de tempo. Conforme pode ser observado no gráfico, aproximadamente 95% das latências de ambos os grupos situam-se abaixo de 5 segundos. Esse valor é consistente com a separação natural entre intervalos de digitação ativa e pausas na atividade, e foi adotado como limiar para a definição de rajadas e como valor inicial para a segmentação das características de pausa descritas na Seção 3.

5.2. Resultados da Classificação

Os resultados obtidos pelos modelos avaliados estão consolidados na Tabela 2. O modelo que apresentou o melhor desempenho geral foi o *Gradient Boosting*, com F1-score macro de 0,89 e precisão de 0,91. O segundo melhor desempenho foi obtido pelo *KNN*, com F1-score macro de 0,88 e precisão de 0,89. O menor desempenho foi observado na *Decision Tree*, com F1-score macro de 0,72 e precisão de 0,75.

De modo geral, os resultados indicam que o conjunto de características comportamentais extraídas dos logs foi capaz de discriminar padrões associados às classes de cópia e desenvolvimento direto da solução na plataforma. O desempenho obtido pelo *Gradient Boosting* sugere que as características propostas capturam informações relevantes sobre o comportamento de codificação dos estudantes. O bom desempenho do *KNN* como segundo melhor modelo reforça essa interpretação, indicando que sessões de cópia e desenvolvimento formam grupos consistentemente separáveis no espaço de características.

O *Random Forest* apresentou a maior estabilidade entre os *folds* (desvio de 0,02), enquanto a *Decision Tree* foi a menos estável (0,10), o *Gradient Boosting*, apesar do melhor desempenho médio, também apresentou desvio considerável (0,08), o que deve ser ponderado dado o tamanho reduzido do conjunto de dados.

Tabela 2. Desempenho dos modelos de classificação (média $\pm$ desvio padrão dos 5 folds).

Modelo	Acurácia	Precisão	Revocação	F1 Macro
<i>Gradient Boosting</i>	0,89 $\pm$ 0,07	0,91 $\pm$ 0,09	0,90 $\pm$ 0,05	0,89 $\pm$ 0,08
<i>KNN</i>	0,88 $\pm$ 0,07	0,89 $\pm$ 0,07	0,89 $\pm$ 0,08	0,88 $\pm$ 0,07
<i>Logistic Regression</i>	0,86 $\pm$ 0,04	0,91 $\pm$ 0,06	0,84 $\pm$ 0,02	0,86 $\pm$ 0,04
<i>Support Vector Machine</i>	0,86 $\pm$ 0,06	0,91 $\pm$ 0,06	0,84 $\pm$ 0,07	0,86 $\pm$ 0,06
<i>AdaBoost</i>	0,86 $\pm$ 0,05	0,89 $\pm$ 0,07	0,86 $\pm$ 0,05	0,86 $\pm$ 0,05
<i>Random Forest</i>	0,85 $\pm$ 0,02	0,87 $\pm$ 0,06	0,86 $\pm$ 0,06	0,85 $\pm$ 0,02
<i>Naive Bayes</i>	0,83 $\pm$ 0,04	0,85 $\pm$ 0,09	0,84 $\pm$ 0,06	0,82 $\pm$ 0,04
<i>XGBoost</i>	0,80 $\pm$ 0,07	0,80 $\pm$ 0,07	0,85 $\pm$ 0,07	0,80 $\pm$ 0,07
<i>Decision Tree</i>	0,74 $\pm$ 0,08	0,75 $\pm$ 0,12	0,81 $\pm$ 0,06	0,72 $\pm$ 0,10

Após a comparação geral entre os modelos, avaliou-se também o impacto do filtro temporal aplicado às latências entre teclas. Para isso, comparou-se o desempenho dos dois modelos de maior destaque considerando duas configurações: uma utilizando todas as latências e outra utilizando apenas os intervalos entre 0,1 e 3 segundos. A Tabela 3 apresenta essa comparação.

Tabela 3. Impacto do filtro temporal no desempenho dos modelos de melhor desempenho.

Modelo	Filtro	Acurácia	Precisão	Revocação	F1 Macro
<i>Gradient Boosting</i>	Sem filtro	0,87 $\pm$ 0,04	0,88 $\pm$ 0,06	0,90 $\pm$ 0,07	0,87 $\pm$ 0,04
<i>Gradient Boosting</i>	0,1–3s	0,89 $\pm$ 0,07	0,91 $\pm$ 0,09	0,90 $\pm$ 0,05	0,89 $\pm$ 0,08
<i>KNN</i>	Sem filtro	0,85 $\pm$ 0,06	0,85 $\pm$ 0,07	0,89 $\pm$ 0,07	0,85 $\pm$ 0,06
<i>KNN</i>	0,1–3s	0,88 $\pm$ 0,07	0,89 $\pm$ 0,07	0,89 $\pm$ 0,08	0,88 $\pm$ 0,07

Observa-se que a aplicação do filtro temporal contribuiu para o aumento do desempenho em ambos os modelos. No *Gradient Boosting*, o F1-score macro passou de 0,87 para 0,89, enquanto no *KNN* passou de 0,85 para 0,88. Esse resultado sugere que a faixa entre 0,1 e 3 segundos concentra intervalos mais representativos da digitação ativa, reduzindo a influência de eventos muito rápidos ou de pausas longas nas características temporais.

Após essa análise, investigou-se com maior detalhe o comportamento do *Gradient Boosting*, modelo de melhor desempenho geral. A Tabela 4 apresenta sua matriz de confusão acumulada ao longo dos cinco folds da validação cruzada externa.

Tabela 4. Matriz de confusão acumulada para o modelo *Gradient Boosting*.

	Desenv. (0)	Cópia (1)
Desenv. (0)	62	8
Cópia (1)	9	76

Observa-se que o modelo classificou corretamente 62 das 70 sessões de desenvolvimento e 76 das 85 sessões de cópia, totalizando 138 classificações corretas entre 155 amostras. Os erros distribuíram-se de forma relativamente equilibrada entre as classes,

com 8 sessões de desenvolvimento classificadas como cópia e 9 sessões de cópia classificadas como desenvolvimento, sem concentração majoritária de erros em nenhum dos grupos.

6. Limitações do Trabalho

Os resultados demonstram que o método proposto apresenta viabilidade para identificar se um estudante desenvolveu a solução diretamente no ambiente de juiz online. No entanto, o trabalho apresenta limitações que devem ser consideradas.

A primeira diz respeito ao conjunto de dados construído, que não contempla situações em que o próprio estudante obteve a solução por meio de uma LLM ou a desenvolveu genuinamente em outro ambiente antes de transcrevê-la, cenários em que o comportamento de digitação pode diferir do observado e comprometer a capacidade discriminativa do método.

A segunda refere-se à definição empírica dos limiares temporais, sem garantia de valores ótimos, além disso, os limiares foram definidos a partir de todos os dados disponíveis, incluindo as amostras de teste, o que introduz um pequeno vazamento de dados na avaliação.

A terceira refere-se à ciência dos participantes sobre o objetivo da pesquisa, o que pode ter gerado viés de reatividade não controlado neste trabalho.

7. Conclusão e Trabalhos Futuros

Este trabalho propôs um método para identificar indícios comportamentais associados à transcrição de soluções previamente obtidas em atividades de programação realizadas em ambientes de juízes *online*. A abordagem utilizou evidências extraídas de *logs* de interação gerados durante o processo de codificação, considerando 19 características comportamentais relacionadas a padrões de pressionamento de teclas, pausas, proporções de eventos e ações realizadas na interface.

Para avaliar a abordagem proposta, foi construído um conjunto de dados próprio composto por sessões de codificação rotuladas. A partir desse conjunto, nove modelos de classificação foram treinados e avaliados: *Logistic Regression*, *Support Vector Machine* (SVM), *Decision Tree*, *Naive Bayes*, *K-Nearest Neighbors* (KNN), *Random Forest*, *AdaBoost*, *Gradient Boosting* e *XGBoost*. O melhor desempenho foi obtido pelo *Gradient Boosting*, que alcançou F1-score macro de 0,89 e precisão de 0,91. Esses resultados sugerem que características comportamentais extraídas de *logs* podem contribuir para a identificação de sessões com padrões compatíveis com cópia ou transcrição de soluções.

Como trabalhos futuros, pretende-se expandir o conjunto de dados por meio da coleta de novas sessões de codificação, contemplando diferentes turmas, disciplinas, tipos de exercícios e soluções de maior extensão e complexidade, além de protocolos que simulem cenários não abordados neste trabalho, como situações em que os próprios alunos obtêm soluções por meio de LLMs. Um conjunto de dados mais amplo poderá viabilizar uma avaliação mais robusta da abordagem, bem como a exploração de novas características comportamentais. Pretende-se também investigar métodos mais rigorosos para a definição dos limiares temporais utilizados na extração de características, visando reduzir a dependência de escolhas empíricas e o vazamento de dados identificado como limitação deste trabalho.

8. Declaração sobre uso de Inteligência Artificial

Declara-se que ferramentas de Inteligência Artificial foram empregadas apenas no aprimoramento textual de partes pontuais do artigo. A formulação da pesquisa, a elaboração do método, o tratamento e a interpretação dos dados, assim como a escrita do conteúdo científico e a construção das conclusões, couberam exclusivamente aos autores. Tais ferramentas não participaram da produção de resultados, das escolhas metodológicas nem da formulação das conclusões aqui expostas.

9. Agradecimentos

O presente trabalho foi realizado com o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (AUXPE-CAPES-PROEX) - Código de Financiamento 001. Adicionalmente, este trabalho foi parcialmente financiado pela Fundação de Amparo à Pesquisa do Estado do Amazonas - FAPEAM - por meio dos projetos PDPG-CAPES e POSGRAD 2025-2026 e POSGRAD 2026-2027.

Referências

- Alves, A., Carvalho, L. S. G. d., Oliveira, E., and Fernandes, D. (2019). Análise comportamental em juízes online para predição do desempenho final de alunos em disciplinas de computação. In *Anais do XXX Simpósio Brasileiro de Informática na Educação*, pages 1906–1915, Porto Alegre, RS, Brasil. SBC.
- Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., and Santos, E. A. (2023). Programming is hard — or at least it used to be: Educational opportunities and challenges of AI code generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education*, volume 1 of *SIGCSE 2023*, pages 500–506, New York, NY, USA. ACM.
- Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM.
- Dillon, R. and De Marsico, M. (2025). Behavioral biometrics for remote exam integrity: Continuous authenticity assessment via keystroke dynamics. *Procedia Computer Science*, 274:402–411.
- Gitchell, D. and Tran, N. (1999). Sim: A utility for detecting similarity in computer programs. In *Proceedings of the Thirtieth SIGCSE Technical Symposium on Computer Science Education*, pages 266–270. ACM.
- Grinsztajn, L., Oyallon, E., and Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on typical tabular data? In *Advances in Neural Information Processing Systems*, volume 35.
- Khan, M. F. A., Edwards, J., Bodily, P., and Karimi, H. (2023). Deciphering student coding behavior: Interpretable keystroke features and ensemble strategies for grade prediction. In *2023 IEEE International Conference on Big Data (BigData)*, pages 5799–5808. IEEE.
- Li, Z. et al. (2023). Applying coding behavior features to student plagiarism detection on programming assignments. *Journal of Circuits, Systems and Computers*, 32(17):2350286.

- Ljubovic, V. and Pajic, E. (2020). Plagiarism detection in computer programming using feature extraction from ultra-fine-grained repositories. *IEEE Access*, 8:96505–96514.
- Oliveira, D. B. F. d., Lavareda Filho, R. M., Oliveira, E. H. T. d., Carvalho, L. S. G. d., Pereira, F. D., Colonna, J. G., and Menezes, A. (2021). Um método de detecção de plágio para sistemas juiz on-line baseado no comportamento dos alunos. In *Anais do XXXII Simpósio Brasileiro de Informática na Educação*, pages 836–848, Porto Alegre, RS, Brasil. SBC.
- Prather, J., Denny, P., Leinonen, J., Becker, B. A., Albluwi, I., Craig, M., Keuning, H., Kiesler, N., Kohn, T., Luxton-Reilly, A., MacNeil, S., Petersen, A., Pettit, R., Reeves, B. N., and Savelka, J. (2023). The robots are here: Navigating the generative ai revolution in computing education. In *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '23, page 108–159, New York, NY, USA. Association for Computing Machinery.
- Prechelt, L., Malpohl, G., and Philippsen, M. (2002). Finding plagiarisms among a set of programs with jplag. *Journal of Universal Computer Science*, 8(11):1016–1038.
- Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, 2(3):160.
- Schleimer, S., Wilkerson, D. S., and Aiken, A. (2003). Winnowing: Local algorithms for document fingerprinting. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 76–85. ACM.
- Schneider, J., Bernstein, A., vom Brocke, J., Damevski, K., and Shepherd, D. C. (2018). Detecting plagiarism based on the creation process. *IEEE Transactions on Learning Technologies*, 11(3):348–361.
- Shadman, R., Wahab, A. A., Manno, M., Lukaszewski, M., Hou, D., and Hussain, F. (2025). Keystroke dynamics: Concepts, techniques, and applications. *ACM Computing Surveys*, 57(11):1–35.
- Trezise, K., Ryan, T., de Barba, P. G., and Kennedy, G. E. (2019). Detecting contract cheating using learning analytics. *Journal of Learning Analytics*, 6(3):137–150.