

Recomendação Sequencial com Filtragem Pedagógica baseada em Knowledge Tracing para Plataformas de Ensino de Algoritmos

Djefferson Maranhão¹, Carlos de Salles Soares Neto¹

¹Departamento de Informática – Universidade Federal do Maranhão (UFMA)
São Luis – MA – Brazil

djefferson.maranhao@gmail.com, carlos.salles@ufma.br

Abstract. *This paper proposes and evaluates a four-stage pipeline for programming exercise recommendation that integrates sequential recommendation with knowledge tracing. The method combines GRU4Rec, selected among five sequential recommendation architectures, with Deep Knowledge Tracing (DKT), chosen among five knowledge tracing models for offering the best balance between accuracy and computational cost. The DKT acts as a pedagogical filter over the ranking produced by GRU4Rec, suppressing exercises whose estimated probability of success falls below an adaptive per-exercise threshold. Experiments on seven datasets from programming platforms (Cosmo, BePKT, CodeNet, CodeNet-len, CodeNet-time, POJ, and CodeForces), spanning varied profiles of scale and sparsity, show that the pedagogical filtering yields gains across all ranking metrics evaluated, with the most expressive impact on the positioning of the relevant exercise at the top of the ranking.*

Resumo. *Este trabalho propõe e avalia um pipeline de quatro estágios para recomendação de exercícios de programação que integra recomendação sequencial e rastreamento do conhecimento. O método combina o GRU4Rec, selecionado entre cinco arquiteturas de recomendação sequencial, com o Deep Knowledge Tracing (DKT), escolhido entre cinco modelos de Knowledge Tracing pelo melhor equilíbrio entre acurácia e custo computacional. O DKT atua como filtro pedagógico sobre o ranking do GRU4Rec, suprimindo exercícios cuja probabilidade de acerto estimada esteja abaixo de um limiar adaptativo por exercício. Experimentos conduzidos em sete datasets de plataformas de programação (Cosmo, BePKT, CodeNet, CodeNet-len, CodeNet-time, POJ e CodeForces), com perfis variados de escala e esparsidade, mostram que a filtragem pedagógica produz melhorias em todas as métricas de ranqueamento avaliadas, com impacto mais expressivo no posicionamento do exercício relevante no topo do ranking.*

1. Introdução

O ensino de programação é reconhecido como uma área desafiadora, especialmente pela dificuldade dos estudantes em compreender conceitos abstratos e formular soluções algorítmicas [Barbosa et al. 2024]. Nesse contexto, plataformas de juiz *online*, que avaliam automaticamente submissões de código, têm sido usadas como recurso complementar ao ensino, oferecendo *feedback* imediato e acesso a exercícios práticos

[Watanobe et al. 2022]. Entretanto, a grande quantidade de exercícios disponíveis pode dificultar a escolha adequada pelo estudante, tornando relevantes mecanismos de recomendação personalizados.

Nesse cenário, os Sistemas de Recomendação Sequencial (RS) constituem uma abordagem promissora para modelar a natureza progressiva e temporal das interações dos usuários [Wang et al. 2020, Quadrana et al. 2018]. Esses modelos capturam dependências temporais em sequências de comportamentos passados e produzem *rankings* de itens candidatos com base no histórico de interações [Hidasi et al. 2016]. Quando aplicados a plataformas de aprendizagem de programação, podem utilizar a sequência de exercícios realizados como representação da trajetória do estudante, sugerindo os próximos itens de forma adaptativa.

Embora apresentem desempenho competitivo em tarefas de predição sequencial, os modelos clássicos de RS possuem uma limitação importante quando aplicados diretamente ao domínio educacional: não representam explicitamente o estado cognitivo do estudante. Em geral, essas abordagens tratam o histórico de interações como uma sequência de identificadores de itens e exploram principalmente a ordem e as relações entre os exercícios acessados. Dessa forma, não estimam o domínio do estudante sobre os conceitos envolvidos, o que pode levar à recomendação de exercícios incompatíveis com sua proficiência.

Para contornar essa limitação, este trabalho propõe uma abordagem integrada que combina recomendação sequencial e rastreamento do conhecimento (*Knowledge Tracing* — KT). A proposta utiliza um modelo de KT [Piech et al. 2015] para estimar a probabilidade de acerto do estudante em cada exercício e emprega essas estimativas como um indicador de adequação pedagógica aplicado ao *ranking* produzido pelo modelo de recomendação sequencial. Com isso, exercícios considerados incompatíveis com o nível estimado de proficiência do estudante são filtrados, permitindo ajustar as recomendações ao seu estado cognitivo atual.

Este trabalho contribui com a proposição de uma arquitetura modular que integra a predição sequencial ao estado pedagógico estimado por rastreamento do conhecimento, enfrentando a limitação cognitiva dos modelos sequenciais clássicos. A abordagem proposta é avaliada experimentalmente em sete conjuntos de dados reais de programação, com perfis distintos de escala, esparsidade e padrões de submissão. Os resultados demonstram que a filtragem pedagógica produz melhorias nas métricas de ranqueamento em todos os *datasets* avaliados.

O restante do artigo está organizado da seguinte forma: a Seção 2 revisa os trabalhos relacionados; a Seção 3 descreve a abordagem proposta; a Seção 4 apresenta a metodologia; a Seção 5 reporta os resultados; a Seção 6 discute os achados; e a Seção 7 apresenta as conclusões do trabalho.

2. Trabalhos Relacionados

Esta seção revisa os três tópicos que fundamentam a abordagem proposta: modelos de recomendação sequencial (Seção 2.1), sua aplicação em ambientes de aprendizagem de programação (Seção 2.2) e técnicas de rastreamento do conhecimento (Seção 2.3).

2.1. Recomendação Sequencial

Modelos de recomendação sequencial buscam prever as próximas interações de um usuário com base na ordem temporal de seu histórico [Wang et al. 2020, Quadrana et al. 2018]. O GRU4Rec [Hidasi et al. 2016], baseado em unidades recorrentes do tipo GRU, modela sessões de interação com baixo custo computacional. Arquiteturas baseadas em autoatenção, como o SASRec [Kang and McAuley 2018] e o BERT4Rec [Sun et al. 2019], ampliaram a capacidade de capturar dependências de longo alcance. Abordagens mais recentes, como o CORE [Hou et al. 2022] e o SINE [Tan et al. 2021], propuseram, respectivamente, representações consistentes no espaço de *embeddings* e modelagem de interesses esparsos.

2.2. Recomendação em Ambientes de Aprendizagem de Programação

A aplicação de sistemas de recomendação a plataformas de aprendizagem de programação tem recebido atenção crescente, com abordagens que vão desde a organização de exercícios por árvores de estrutura de conhecimento [Zheng et al. 2022] até o uso de aprendizado por reforço para integrar estilos de aprendizagem [Xu et al. 2022] e a conversão de submissões em avaliações implícitas para filtragem colaborativa [Muepu and Watanobe 2024]. [Zhu et al. 2022] propuseram um modelo de rastreamento do conhecimento voltado a juízes online de programação, acompanhado do *dataset* BePKT. Embora essas abordagens incorporem informações pedagógicas, nenhuma utiliza modelos de recomendação sequencial como base, deixando inexplorada a integração entre modelagem temporal e estado cognitivo do estudante.

2.3. Rastreamento do Conhecimento

Técnicas de rastreamento do conhecimento (*Knowledge Tracing*) modelam a evolução do estado cognitivo do estudante ao longo do tempo, estimando a probabilidade de acerto em futuras interações [Shen et al. 2021]. O DKT [Piech et al. 2015] introduziu o uso de RNNs para essa tarefa. Aprimoramentos subsequentes incluem o DKVMN [Zhang et al. 2017], com memória externa para representar conceitos, o AKT [Ghosh et al. 2020], com atenção contextual, e modelos como o LPKT e o UKT, que incorporam sinais de persistência do aprendiz e aprendizado contrastivo. No domínio da programação, [Zhu et al. 2022] disponibilizaram o *dataset* BePKT, voltado ao rastreamento do conhecimento em juízes online.

Diferentemente dos trabalhos anteriores, que abordam recomendação e rastreamento do conhecimento de forma isolada, este trabalho investiga a integração dessas duas abordagens em um *pipeline* modular. A proposta combina a capacidade de modelagem temporal de um modelo de recomendação sequencial com as estimativas de proficiência produzidas por um modelo de KT, utilizando-as como camada de filtragem pedagógica sobre o *ranking* gerado. Essa integração permite que cada componente seja treinado e atualizado de forma independente, diferindo de abordagens que buscam modelos unificados.

3. Abordagem Proposta

Esta seção descreve a abordagem proposta para integrar recomendação sequencial e rastreamento do conhecimento em plataformas de aprendizagem de programação. A Seção 3.1 apresenta a visão geral do *pipeline* e a Seção 3.2 formaliza o mecanismo de filtragem pedagógica.

3.1. Visão geral do *pipeline*

A abordagem proposta é estruturada em quatro estágios complementares, conforme ilustrado na Figura 1. Seja U o conjunto de estudantes e I o conjunto de exercícios disponíveis. Para cada estudante $u \in U$, considera-se seu histórico de interações ordenado cronologicamente, representado por $s_u = (x_1, x_2, \dots, x_t)$, em que cada interação x_k inclui o exercício acessado $i_k \in I$ e a resposta observada do estudante.

No primeiro estágio, as sequências de interação dos estudantes são utilizadas para produzir, para cada estudante, um *ranking* de itens candidatos $R_u = (r_1, r_2, \dots, r_K)$, ordenados segundo uma pontuação de relevância estimada. Esse estágio tem como objetivo capturar padrões temporais de comportamento e identificar exercícios potencialmente relevantes com base no histórico de interações do estudante.

No segundo estágio, é incorporada uma camada pedagógica responsável por estimar a adequação dos exercícios candidatos ao estado de conhecimento do estudante. Para cada par estudante–exercício, essa camada produz uma estimativa $P_{\text{ped}}(u, i) \in [0, 1]$, interpretada como a probabilidade de o estudante responder corretamente ao item. Essa estimativa fornece uma indicação do nível de domínio do estudante em relação aos exercícios recomendados, permitindo avaliar se determinado item é compatível com seu conhecimento atual.

No terceiro estágio, o *ranking* produzido no primeiro estágio é refinado com base nas estimativas pedagógicas. Exercícios cuja probabilidade de acerto estimada esteja abaixo de um limiar adaptativo por exercício têm sua pontuação rebaixada, de modo que o ranqueamento final reflita não apenas a relevância sequencial, mas também a adequação pedagógica ao nível de conhecimento do estudante. O mecanismo de filtragem é formalizado na Subseção 3.2.

Por fim, no quarto estágio, o *ranking* refinado é avaliado por meio de métricas de ranqueamento. Para garantir uma comparação justa, o mesmo critério de filtragem é aplicado ao exercício de referência (*ground truth*): quando esse exercício fica abaixo do limiar pedagógico, o caso é excluído da avaliação. Dessa forma, o desempenho do *ranking* original e o do *ranking* filtrado são medidos exatamente sobre os mesmos estudantes, isolando o efeito da reordenação pedagógica. O protocolo completo é descrito na Seção 4.3.

3.2. Mecanismo de filtragem pedagógica

A incorporação do sinal pedagógico é realizada por meio de um mecanismo de filtragem aplicado sobre o *ranking* R_u produzido no primeiro estágio. Para cada par estudante–exercício (u, i) , considera-se a estimativa pedagógica $P_{\text{ped}}(u, i)$, interpretada como a probabilidade de o estudante u responder corretamente ao exercício i , conforme estimado pela camada de rastreamento do conhecimento. Assim, o ranqueamento ajustado é obtido a partir de uma pontuação modificada:

$$\tilde{s}(u, i) = \begin{cases} s_{\text{rec}}(u, i), & \text{se } P_{\text{ped}}(u, i) \geq \tau_i, \\ -\infty, & \text{se } P_{\text{ped}}(u, i) < \tau_i, \end{cases} \quad (1)$$

em que $s_{\text{rec}}(u, i)$ representa a pontuação originalmente atribuída pelo recomendador ao exercício i para o estudante u . O ranqueamento final $\hat{R}_u$ é obtido pela ordenação dos

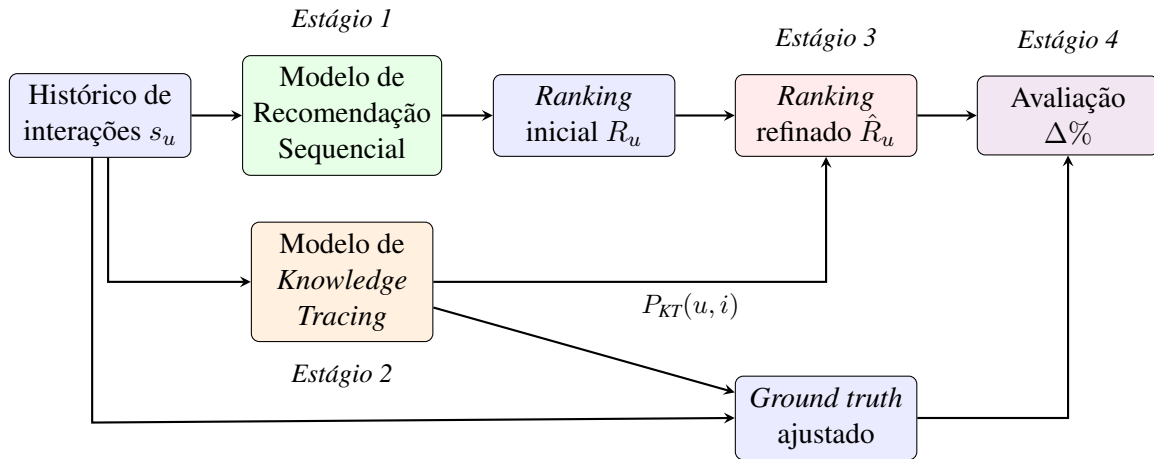


Figura 1. Visão geral do *pipeline* proposto.

exercícios candidatos segundo $\tilde{s}(u, i)$. Assim, candidatos abaixo do limiar pedagógico não são removidos da estrutura de avaliação, mas recebem a menor prioridade possível no ranqueamento.

O limiar τ_i é associado ao exercício i e calculado a partir das probabilidades estimadas para esse exercício no conjunto de treinamento:

$$\tau_i = \frac{1}{|U|} \sum_{u \in U} P_{\text{ped}}(u, i). \quad (2)$$

Essa definição adapta o limiar à dificuldade de cada exercício: exercícios mais difíceis apresentam valores menores de τ_i , exigindo menos do estudante para que o item seja mantido; exercícios mais fáceis apresentam τ_i mais elevado, tornando o critério mais exigente. Dessa forma, o filtro calibra automaticamente as exigências de acordo com o perfil de dificuldade de cada exercício, sem a necessidade de um limiar fixo.

4. Metodologia

Esta seção descreve a metodologia utilizada nos experimentos. A Seção 4.1 apresenta os conjuntos de dados. A Seção 4.2 detalha os procedimentos de treinamento e avaliação dos modelos. A Seção 4.3 descreve o protocolo de avaliação da integração.

4.1. Conjuntos de Dados

Os experimentos utilizam sete conjuntos de dados provenientes de cinco plataformas distintas de programação. O *Cosmo* é extraído de um juiz online utilizado em disciplinas introdutórias de programação na Universidade Federal do Maranhão (UFMA).¹ O *BePKT* [Zhu et al. 2022] é um *dataset* voltado ao rastreamento do conhecimento em ambientes de ensino de programação. O *CodeNet* [Puri et al. 2021] é uma base de larga escala disponibilizada pela IBM, contendo milhões de submissões de programação competitiva. A

¹Os dados foram gerados no contexto de uso regular da plataforma em disciplinas curriculares e tratados de forma anonimizada. Nos experimentos, os estudantes são representados apenas por identificadores, sem informações pessoais diretamente identificáveis.

partir dessa base, [Liu et al. 2023] derivaram dois recortes: o *CodeNet-len*, que seleciona estudantes com sequências de interação mais longas, e o *CodeNet-time*, que aplica um recorte temporal sobre as submissões. O *POJ* [Mou et al. 2016] é oriundo do *Peking University Online Judge*. O *CodeForces* [Volostnov 2024] é extraído de uma plataforma de programação competitiva, com alto volume de interações e exercícios.

A Tabela 1 apresenta as estatísticas descritivas dos sete conjuntos de dados, cujos perfis contrastantes favorecem uma avaliação mais abrangente. Em termos de escala, os conjuntos variam de 313 estudantes, no Cosmo, a 154.180, no CodeNet. A esparsidade oscila entre 78,65%, no Cosmo, e 99,78%, no POJ, evidenciando cenários em que cada estudante interage com apenas uma pequena parcela dos exercícios disponíveis. Quanto ao desempenho, o BePKT apresenta a menor taxa de acertos, de 32,03%, enquanto o POJ registra a maior média de tentativas por exercício, de 7,29.

Tabela 1. Estatísticas descritivas dos conjuntos de dados.

Dataset	Estudantes	Interações	Submissões	Exercícios	Esparsidade ²	Taxa de Acertos ³	Tentativas/Exercício ⁴
Cosmo	313	4.545	13.916	68	78,65%	37,04%	3,06
BePKT	997	23.920	75.993	553	84,85%	32,03%	3,18
CodeNet-len	1.121	328.726	614.661	3.348	91,24%	56,75%	1,87
CodeNet-time	26.269	428.970	811.465	2.464	99,34%	53,42%	1,89
CodeForces	14.957	7.780.135	17.607.999	30.614	98,30%	44,80%	2,26
POJ	22.916	136.608	996.240	2.750	99,78%	35,65%	7,29
CodeNet	154.180	6.768.445	13.920.865	7.930	99,45%	53,59%	2,06

4.2. Treinamento e Avaliação dos Modelos

Os modelos de recomendação sequencial (BERT4Rec, CORE, GRU4Rec, SASRec e SINE) foram treinados por meio da biblioteca RecBole [Zhao et al. 2021]. As interações de cada estudante foram organizadas em ordem temporal e divididas na proporção 80%/10%/10% (treino/validação/teste), de modo que as últimas interações de cada sequência constituem os conjuntos de validação e teste. Essa divisão temporal busca reproduzir um cenário mais próximo de uma situação real de uso, evitando que informações posteriores ao momento da recomendação sejam utilizadas durante o treinamento. A avaliação foi realizada no modo *full-sort*, no qual todos os exercícios disponíveis são considerados na geração do *ranking*, tornando a avaliação mais exigente.

Os modelos de rastreamento do conhecimento (DKT, AKT, DKVMN, UKT e LPKT) foram treinados por meio da biblioteca PyEdMine, seguindo o protocolo *pykt_setting*. Nesse protocolo, 20% dos estudantes são separados para teste por meio de amostragem estratificada (por comprimento de sequência e taxa de acertos), e os 80% restantes são divididos em cinco partições (*5-fold*), das quais uma é utilizada como validação e as demais como treino. Esses modelos estimam a probabilidade de acerto do estudante em cada exercício, sendo essa estimativa utilizada como sinal pedagógico para ajustar as recomendações ao nível de conhecimento estimado do estudante.

Os principais hiperparâmetros utilizados nos modelos de recomendação sequencial e nos modelos de rastreamento do conhecimento são apresentados, respectivamente,

²Calculada por $1 - \frac{\#Interações}{(\#Estudantes \times \#Exercícios)}$.

³Calculada por $\frac{\#Submissões\ Corretas}{\#Submissões}$.

⁴Calculada por $\frac{\#Submissões}{\#Interações}$.

nas Tabelas 2 e 3. Esses parâmetros foram definidos de modo a manter uma configuração consistente entre os diferentes *datasets* avaliados.

Tabela 2. Hiperparâmetros dos modelos de recomendação sequencial.

Parâmetro	BERT4Rec	CORE	GRU4Rec	SASRec	SINE
Dim. embedding	64	64	64	32	64
Dim. oculta	64	64	32	32	64
Num. camadas/blocos	2	2	1 (GRU)	2	2
Num. cabeças atenção	2	2	–	2	2
Dim. <i>inner</i>	256	256	–	256	–
Dropout (atenção)	0,2	0,5	–	0,5	–
Dropout (oculta/geral)	0,2	0,5	0,25	0,5	0,3
Comprimento máx. seq.	10	10	10	10	10
Função de perda	CE	CE	CE	CE	CE
<i>gru.layers</i>	–	–	5	–	–
Interesses esparsos	–	–	–	–	3
<i>reg.weight</i>	–	10^{-4}	–	10^{-4}	–

Tabela 3. Hiperparâmetros dos modelos de Knowledge Tracing.

Parâmetro	Grupo	Knowledge Tracing				
		DKT	AKT	DKVMN	UKT	LPKT
Otimizador	Treino	Adam	Adam	Adam	Adam	Adam
Taxa de aprend.	Treino	0,001	0,0001	0,001	0,0001	0,001
Weight decay	Treino	0,0001	0	0	10^{-5}	10^{-5}
Tamanho de lote	Treino	64	24	64	24	64
Máx. épocas	Treino	100	100	100	100	100
Paciência (ES)	Treino	10	10	10	10	10
Métrica (ES)	Treino	AUC	AUC	AUC	AUC	AUC
Dim. latente	Modelo	256	256	64	256	64
Dropout	Modelo	0,30	0,10	0,20	0,10	0,30
Num. blocos/camadas	Modelo	1 (GRU)	2	–	2	–
Num. cabeças	Modelo	–	8	–	8	–
Comp. sequência	Modelo	–	200	–	200	–
Scheduler LR	Treino	Não	Não	Não	Não	StepLR
Grad. clip	Treino	Não	10,0	Não	Não	Não

A avaliação dos modelos de recomendação sequencial foi realizada com métricas de ranqueamento no corte $K = 10$, incluindo HR@10, NDCG@10, MRR@10 e Recall@10. Essas métricas permitem verificar se o item relevante aparece entre as primeiras posições do *ranking* e, também, se ele é posicionado próximo ao topo da lista de recomendações. Para os modelos de rastreamento do conhecimento, foram utilizadas as métricas AUC, ACC, MAE e RMSE. Essas métricas avaliam a capacidade dos modelos de prever corretamente o desempenho dos estudantes e de estimar probabilidades de acerto próximas às respostas observadas.

4.3. Protocolo de Avaliação da Integração

A avaliação da integração corresponde ao quarto estágio do *pipeline* descrito na Seção 3.1. Inicialmente, o modelo de recomendação sequencial atribui uma pontuação $s_{\text{rec}}(u, i)$ a todos os exercícios do catálogo para cada estudante do conjunto de teste (*full-sort*). Em seguida, o modelo de rastreamento do conhecimento estima, para cada par estudante–exercício, a probabilidade de acerto $P_{\text{ped}}(u, i)$. O limiar adaptativo τ_i é calculado conforme a Equação 2. A pontuação modificada $\tilde{s}(u, i)$ é obtida conforme a Equação 1, atribuindo $-\infty$ aos exercícios abaixo do limiar pedagógico e preservando a pontuação original dos demais. O ranqueamento final é obtido pela ordenação segundo $\tilde{s}(u, i)$.

Para garantir uma comparação justa, o mesmo critério de filtragem é aplicado ao exercício de referência (*ground truth*) de cada estudante. Se o exercício que o estudante de fato resolveu no conjunto de teste apresenta $P_{\text{ped}}(u, i) < \tau_i$, esse exercício é removido do conjunto de referência. Quando nenhum exercício de referência permanece após a filtragem, o caso é desconsiderado na avaliação. Dessa forma, tanto a condição sem filtragem (antes) quanto a condição com filtragem (depois) são avaliadas sobre o mesmo subconjunto, isolando o efeito da reordenação pedagógica.

Por fim, o efeito da integração é quantificado pelo ganho relativo $\Delta\%$, calculado para cada filtro pedagógico e *dataset*:

$$\Delta\% = \frac{M_{\text{Depois}} - M_{\text{Antes}}}{M_{\text{Antes}}} \times 100 \quad (3)$$

em que M_{Antes} é o valor da métrica obtido pelo modelo de recomendação sequencial no subconjunto de estudantes cobertos pelo filtro, e M_{Depois} é o valor obtido após a filtragem pedagógica (reordenação do *ranking*).

5. Resultados

Esta seção reúne os resultados das três etapas experimentais conduzidas neste estudo. Inicialmente, são avaliados os modelos de recomendação sequencial (Seção 5.1) e de rastreamento do conhecimento (Seção 5.2). Em seguida, apresentam-se os resultados da abordagem integrada baseada nos modelos GRU4Rec e DKT (Seção 5.3).

5.1. Comparação dos modelos de recomendação sequencial

A seleção do modelo de ranqueamento do Estágio 1 considerou dois critérios: acurácia e custo computacional. A Tabela 4 compara cinco arquiteturas nos sete *datasets*. Em termos de acurácia, GRU4Rec e SASRec obtiveram os melhores resultados nas métricas NDCG@10 e HR@10, com o SASRec apresentando vantagem em *datasets* menores e mais densos, enquanto o GRU4Rec alcançou resultados competitivos em praticamente todos os cenários. Em determinados *datasets*, o BERT4Rec também supera o GRU4Rec em métricas específicas; contudo, essa vantagem pontual não se sustenta de forma consistente em todos os cenários, o que reforça a escolha pelo GRU4Rec como modelo base.

Em termos de custo computacional (Tabela 5), as diferenças são substanciais. O BERT4Rec exige tempos de treinamento elevados (como 277 min no CodeForces) e o SINE apresenta alta sensibilidade à escala dos dados (como 1270 min no CodeNet-time). O GRU4Rec, por sua vez, apresenta consistentemente o menor tempo de treinamento (como 0,9 min no Cosmo e 14,7 min no CodeNet) com consumo de memória estável. Considerando o melhor equilíbrio entre acurácia e escalabilidade, o GRU4Rec foi selecionado como modelo de ranqueamento inicial.

Tabela 4. Resultados dos modelos de recomendação sequencial.

Dataset	Métrica	Algoritmos				
		BERT4Rec	CORE	GRU4Rec	SASRec	SINE
Cosmo	NDCG@10	0,7403	0,7552	0,7611	0,7661	0,4600
Cosmo	HR@10	0,9067	0,9176	0,9158	0,9176	0,6929
Cosmo	Recall@10	0,9067	0,9176	0,9158	0,9176	0,6929
Cosmo	MRR@10	0,6879	0,7045	0,7114	0,7179	0,3885

Continuação da Tabela 4

Dataset	Métrica	Algoritmos				
		BERT4Rec	CORE	GRU4Rec	SASRec	SINE
BePKT	NDCG@10	0,7686	0,7697	0,7636	0,7718	0,4003
BePKT	HR@10	0,8993	0,9089	0,9029	0,9087	0,6524
BePKT	Recall@10	0,8993	0,9089	0,9029	0,9087	0,6524
BePKT	MRR@10	0,7264	0,7254	0,7183	0,7277	0,3225
CodeNet	NDCG@10	0,6665	0,5953	0,6638	0,6384	0,3655
CodeNet	HR@10	0,7911	0,7775	0,7898	0,7781	0,5141
CodeNet	Recall@10	0,7911	0,7775	0,7898	0,7781	0,5141
CodeNet	MRR@10	0,6257	0,5365	0,6227	0,5929	0,3182
CodeNet-len	NDCG@10	0,6368	0,5682	0,6243	0,5727	0,1531
CodeNet-len	HR@10	0,7612	0,7475	0,7507	0,7477	0,3169
CodeNet-len	Recall@10	0,7612	0,7475	0,7507	0,7477	0,3169
CodeNet-len	MRR@10	0,5964	0,5109	0,5833	0,6154	0,1918
CodeNet-time	NDCG@10	0,6847	0,5835	0,6878	0,6625	0,3788
CodeNet-time	HR@10	0,8191	0,8063	0,8187	0,8125	0,5440
CodeNet-time	Recall@10	0,8191	0,8063	0,8187	0,8125	0,5440
CodeNet-time	MRR@10	0,6407	0,5119	0,6449	0,6136	0,3260
POJ	NDCG@10	0,3942	0,4152	0,3967	0,3989	0,2385
POJ	HR@10	0,4628	0,4942	0,4673	0,4666	0,3433
POJ	Recall@10	0,4628	0,4942	0,4673	0,4666	0,3433
POJ	MRR@10	0,3731	0,3908	0,3749	0,3780	0,2064
CodeForces	NDCG@10	0,7079	0,6549	0,7056	0,6863	0,2641
CodeForces	HR@10	0,8241	0,8177	0,8218	0,8123	0,4172
CodeForces	Recall@10	0,8241	0,8177	0,8218	0,8123	0,4172
CodeForces	MRR@10	0,6697	0,6020	0,6674	0,6451	0,2162

Tabela 5. Tempo de treinamento e consumo de memória dos modelos de recomendação sequencial.

Dataset	Métrica	Algoritmos				
		BERT4Rec	CORE	GRU4Rec	SASRec	SINE
Cosmo	Tempo (min)	23.0	13.3	0.9	6.5	13.8
Cosmo	Memória (GB)	1.3	1.3	1.2	1.2	1.2
BePKT	Tempo (min)	37.8	12.9	4.3	67.9	25.5
BePKT	Memória (GB)	1.3	1.3	1.2	1.3	1.3
CodeNet	Tempo (min)	146.4	57.8	14.7	51.3	26.4
CodeNet	Memória (GB)	18.5	18.6	18.4	18.5	18.5
CodeNet-len	Tempo (min)	105.8	11.7	9.4	51.3	27.0
CodeNet-len	Memória (GB)	2.0	2.0	1.9	1.9	1.9
CodeNet-time	Tempo (min)	105.8	15.7	9.9	60.0	1270.9
CodeNet-time	Memória (GB)	1.7	2.2	2.1	0.8	1.3
POJ	Tempo (min)	15.3	38.3	13.0	52.4	53.9
POJ	Memória (GB)	3.0	1.2	2.3	0.8	1.0
CodeForces	Tempo (min)	277.5	81.1	23.7	69.5	36.2
CodeForces	Memória (GB)	23.1	23.2	23.0	23.1	23.1

5.2. Comparação dos modelos de rastreamento do conhecimento

A seleção do modelo do Estágio 2 seguiu a mesma lógica: acurácia e custo. A Tabela 6 compara o desempenho preditivo dos cinco modelos de rastreamento do conhecimento. O AKT alcança os melhores resultados em AUC na maioria dos *datasets*, beneficiando-se de sua modelagem baseada em atenção. O DKT apresenta desempenho consistentemente próximo (como AUC de 0,766 no Cosmo, contra 0,772 do AKT).

A diferença entre os modelos está no custo computacional. Conforme a Tabela 7, o DKT requer tempos de treinamento significativamente menores (como 0,3 min no Cosmo e 37,7 min no CodeNet-time, comparado a 4,6 min e 126,9 min do LPKT nos mesmos *datasets*). Como a camada pedagógica precisa operar com baixo custo para viabilizar a integração em tempo real, o DKT foi selecionado pelo melhor equilíbrio entre capacidade preditiva e eficiência.

Tabela 6. Resultados dos modelos de Knowledge Tracing nos datasets avaliados.

Dataset	Métrica	Algoritmos				
		AKT	DKT	UKT	DKVMN	LPKT
Cosmo	AUC	0,77234	0,76644	0,74914	0,75048	0,73390
Cosmo	ACC	0,73750	0,73462	0,34415	0,72468	0,71447
Cosmo	MAE	0,34994	0,36066	0,48450	0,36533	0,39790
Cosmo	RMSE	0,42216	0,42464	0,49224	0,43133	0,43883
BePKT	AUC	0,74104	0,71861	0,69729	0,70651	0,70999
BePKT	ACC	0,72245	0,70907	0,32518	0,70349	0,71705
BePKT	MAE	0,36255	0,37630	0,49325	0,37376	0,40239
BePKT	RMSE	0,43001	0,43731	0,49739	0,44332	0,44144
CodeNet	AUC	0,80623	0,79232	0,79084	0,79685	0,76607
CodeNet	ACC	0,73019	0,71831	0,53618	0,72282	0,70237
CodeNet	MAE	0,34676	0,37542	0,44454	0,36660	0,39792
CodeNet	RMSE	0,42390	0,43044	0,46212	0,42807	0,44302
CodeNet-len	AUC	0,79150	0,77374	0,76982	0,77395	0,76611
CodeNet-len	ACC	0,72206	0,70726	0,56796	0,71063	0,70277
CodeNet-len	MAE	0,35661	0,38237	0,43703	0,37235	0,39135
CodeNet-len	RMSE	0,42949	0,43753	0,46057	0,43754	0,44097
CodeNet-time	AUC	0,80010	0,79389	0,78300	0,79036	0,77805
CodeNet-time	ACC	0,72321	0,71594	0,53258	0,71684	0,70709
CodeNet-time	MAE	0,35291	0,38032	0,44558	0,36790	0,38740
CodeNet-time	RMSE	0,42708	0,43162	0,46411	0,43161	0,43761
POJ	AUC	0,65440	0,64835	0,61394	0,63760	0,64881
POJ	ACC	0,66712	0,66328	0,35575	0,65896	0,66203
POJ	MAE	0,42632	0,42667	0,49665	0,43000	0,43001
POJ	RMSE	0,46228	0,46305	0,49866	0,46604	0,46338
CodeForces	AUC	0,76455	0,74879	0,74814	0,75345	0,73369
CodeForces	ACC	0,70138	0,68787	0,44791	0,69250	0,68226
CodeForces	MAE	0,38620	0,40803	0,46913	0,39529	0,42097
CodeForces	RMSE	0,44118	0,44940	0,48146	0,44584	0,45464

Tabela 7. Tempo de treinamento dos modelos de Knowledge Tracing.

Dataset	Métrica	Algoritmos				
		AKT	DKT	UKT	DKVMN	LPKT
Cosmo	Tempo (min)	0.9	0.3	0.8	1.3	4.6
BePKT	Tempo (min)	1.9	0.2	1.4	1.6	13.6
CodeNet	Tempo (min)	51.3	14.7	53.8	35.1	1858.1
CodeNet-len	Tempo (min)	42.2	13.2	5.6	12.8	30.7
CodeNet-time	Tempo (min)	152.8	37.7	121.6	51.3	126.9
POJ	Tempo (min)	39.8	6.1	26.2	10.9	61.5
CodeForces	Tempo (min)	34.1	110.8	693.5	100.8	378.8

5.3. Resultados da integração dos modelos

A Tabela 8 apresenta os resultados da aplicação dos Estágios 3 e 4 do *pipeline* proposto (Subseção 3.1): a filtragem pedagógica do *ranking* pelo GRU4Rec, seguida da avaliação com *ground truth* ajustado. A coluna *GRU4Rec* corresponde ao desempenho do modelo de recomendação sequencial isolado, a coluna *Antes* indica o desempenho no subconjunto coberto pelo DKT antes da filtragem, a coluna *Depois* indica o valor após a filtragem, e a coluna $\Delta\%$ indica o ganho relativo, calculado conforme a Equação 3.

Tabela 8. Resultados da integração GRU4Rec + DKT.

Dataset	Métrica	GRU4Rec	Knowledge Tracing (DKT)		
			Antes	Depois	$\Delta\%$
Cosmo	NDCG@10	0,715	0,723	0,730	+1,01%
	HR@10	0,877	0,881	0,890	+1,03%
	Recall@10	0,877	0,881	0,890	+1,03%
	MRR@10	0,664	0,672	0,679	+1,00%

Continuação da Tabela 8

Dataset	Métrica	GRU4Rec	Knowledge Tracing (DKT)		
			Antes	Depois	Δ %
BePKT	NDCG@10	0,694	0,686	0,694	+1,11%
	HR@10	0,859	0,850	0,854	+0,47%
	Recall@10	0,859	0,850	0,854	+0,47%
	MRR@10	0,640	0,632	0,641	+1,38%
CodeNet	NDCG@10	0,680	0,672	0,685	+1,93%
	HR@10	0,813	0,812	0,818	+0,67%
	Recall@10	0,813	0,812	0,818	+0,67%
	MRR@10	0,637	0,626	0,642	+2,45%
CodeNet-len	NDCG@10	0,605	0,611	0,617	+0,99%
	HR@10	0,727	0,739	0,742	+0,50%
	Recall@10	0,727	0,739	0,742	+0,50%
	MRR@10	0,565	0,570	0,576	+1,19%
CodeNet-time	NDCG@10	0,701	0,684	0,692	+1,15%
	HR@10	0,822	0,802	0,806	+0,51%
	Recall@10	0,822	0,802	0,806	+0,51%
	MRR@10	0,661	0,645	0,654	+1,42%
POJ	NDCG@10	0,381	0,294	0,295	+0,37%
	HR@10	0,451	0,368	0,369	+0,46%
	Recall@10	0,451	0,368	0,369	+0,46%
	MRR@10	0,359	0,272	0,273	+0,33%
CodeForces	NDCG@10	0,700	0,692	0,699	+1,06%
	HR@10	0,825	0,822	0,825	+0,35%
	Recall@10	0,825	0,822	0,825	+0,35%
	MRR@10	0,659	0,649	0,658	+1,36%

A filtragem pedagógica produz ganhos positivos em todos os *datasets* e métricas avaliados. Os maiores incrementos ocorrem no CodeNet (+2,45% no MRR@10) e no CodeNet-time (+1,42% no MRR@10). O MRR@10 apresenta os ganhos mais expressivos em todos os cenários, indicando que a integração melhora sobretudo a posição do exercício relevante no topo do *ranking*. O POJ, por sua vez, registra os menores ganhos (+0,33% a +0,46%). Esses padrões são analisados em detalhe na Seção 6.

6. Discussão

A integração do modelo de *Knowledge Tracing* ao *pipeline* de recomendação sequencial endereça uma limitação fundamental dos modelos clássicos, a ausência de informações sobre o estado cognitivo do estudante na geração do *ranking*. Os ganhos observados em todos os cenários avaliados confirmam que o sinal pedagógico complementa a modelagem temporal, acrescentando uma dimensão de adequação ao nível do aprendiz que o modelo sequencial não captura.

Os maiores ganhos no MRR@10 em relação ao HR@10 indicam que a filtragem atua principalmente no reposicionamento do exercício relevante para posições mais elevadas, e não apenas em sua presença entre os primeiros itens. Esse efeito é pedagogicamente relevante, pois o estudante tende a interagir com as primeiras recomendações. Em *datasets* esparsos, como o CodeNet-time, o sinal pedagógico compensa a escassez de informações do modelo sequencial. Em cenários mais densos, como Cosmo e BePKT, os ganhos permanecem estáveis, indicando consistência entre diferentes perfis de dados. O POJ constitui uma exceção, com os menores ganhos, possivelmente devido à maior complexidade de seus exercícios, com média de 7,29 tentativas, o que pode dificultar a estimação da proficiência pelo DKT e reduzir a precisão da filtragem.

Em alguns *datasets*, os valores da coluna *Depois* são inferiores aos da coluna *GRU4Rec*. A diferença decorre do uso de conjuntos distintos: *GRU4Rec* considera todo o

conjunto de teste, enquanto *Antes* e *Depois* se referem ao subconjunto cujo exercício de referência passou pelo filtro pedagógico. Em bases com muitas ressubmissões, o GRU4Rec captura tentativas repetidas, pois modela a sequência sem distinguir acertos de erros. O filtro exclui sobretudo ressubmissões previsíveis com baixa probabilidade estimada de acerto. Assim, o efeito da filtragem deve ser medido pelo $\Delta\%$, que compara *Antes* e *Depois* no mesmo subconjunto.

Os ganhos referem-se a métricas de ranqueamento *offline*; a confirmação de benefícios pedagógicos requer estudos com usuários reais. O estudo também apresenta ameaças à validade. Internamente, a comparação antes/depois opera sobre o subconjunto de estudantes cujo exercício de referência foi aprovado pelo filtro pedagógico, de modo que estudantes cujo próximo exercício ficou abaixo do limiar são excluídos da avaliação, e os hiperparâmetros dos modelos não foram otimizados por *dataset*. Externamente, os resultados se restringem a plataformas de programação e podem não se generalizar para outros domínios educacionais. Quanto ao constructo, a probabilidade de acerto é uma aproximação da adequação pedagógica e não representa toda a complexidade da relação entre dificuldade e aprendizagem.

7. Conclusão

Este trabalho investigou a limitação dos modelos de recomendação sequencial quando aplicados a plataformas de aprendizagem de programação: a ausência de sensibilidade ao estado cognitivo do estudante, que pode resultar em recomendações baseadas unicamente na ordem dos exercícios previamente acessados. Para endereçar essa limitação, foi proposta uma abordagem integrada em quatro estágios que combina o GRU4Rec, responsável pela geração do *ranking* sequencial, com o DKT, que estima a probabilidade de acerto do estudante em cada exercício e atua como filtro pedagógico sobre o *ranking*.

Os experimentos conduzidos em sete conjuntos de dados de programação, com diferentes níveis de escala e esparsidade, demonstraram que a filtragem pedagógica gera ganhos em todas as métricas e cenários avaliados. O principal benefício está no reposicionamento do exercício relevante para o topo do *ranking* (MRR@10), resultado desejável do ponto de vista pedagógico. Os resultados também indicam que o sinal pedagógico é especialmente útil em cenários de alta esparsidade, nos quais o modelo sequencial dispõe de poucas informações sobre cada estudante. Isso sugere que a integração entre recomendação sequencial e rastreamento do conhecimento é uma estratégia viável para produzir sugestões mais alinhadas ao nível de conhecimento do aprendiz.

O estudo apresenta limitações que devem ser consideradas. Embora o limiar de filtragem seja adaptativo por exercício, a escolha da média como estatística de corte pode ser aprimorada, pois outras estratégias ou limiares por perfil de estudante podem ampliar os ganhos observados. Além disso, a probabilidade de acerto, embora forneça um sinal objetivo e escalável, não captura plenamente a relação entre dificuldade e aprendizagem, podendo remover exercícios desafiadores potencialmente úteis. A avaliação foi conduzida exclusivamente em modo *offline*, o que requer estudos com usuários reais para verificar se os ganhos se traduzem em benefícios pedagógicos concretos. Como trabalhos futuros, pretende-se incorporar modelos de diagnóstico cognitivo (*Cognitive Diagnosis*), capazes de estimar simultaneamente a proficiência do estudante em múltiplas habilidades.

Disponibilização de Artefatos

Os artefatos gerados neste estudo estão disponíveis em: https://github.com/dssmaranhao/sbie_2026. Em virtude da limitação de espaço do repositório, não foi possível disponibilizar a totalidade dos datasets e modelos treinados.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) – Brasil – Código de Financiamento 001. Também conta com financiamento no âmbito do projeto “Wanda: Card Games para Ensino de Lógica de Programação”, processo 88887.180797/2025-00 contemplado pelo Edital nº 3/2025 CAPES “InovaEducação”.

Declaração sobre uso de Inteligência Artificial

Na elaboração deste artigo, ferramentas de IA generativa foram utilizadas como apoio à revisão gramatical e ortográfica e à organização estrutural do texto. A concepção da pesquisa, a análise e a interpretação dos dados, os resultados, as conclusões e as contribuições científicas do trabalho foram desenvolvidos pelos autores, que assumem integral responsabilidade pelo conteúdo apresentado.

Referências

- Barbosa, P. L. S., Carmo, R. A. F. d., Gomes, J. P., and Viana, W. (2024). Adaptive learning in computer science education: A scoping review. *Education and Information Technologies*, 29(8):9139–9188.
- Ghosh, A., Heffernan, N., and Lan, A. S. (2020). Context-aware attentive knowledge tracing. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2330–2339.
- Hidasi, B., Karatzoglou, A., Baltrunas, L., and Tikk, D. (2016). Session-based recommendations with recurrent neural networks. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*.
- Hou, Y., Mu, S., Zhao, W. X., Li, Y., Ding, B., and Wen, J.-R. (2022). CORE: Simple and effective session-based recommendation within consistent representation space. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1796–1801.
- Kang, W.-C. and McAuley, J. (2018). Self-attentive sequential recommendation. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 197–206.
- Liu, Y., Zhu, R., and Gao, M. (2023). Personalized programming guidance based on deep programming learning style capturing. In *International Conference on Computer Science and Education*, pages 214–231. Springer.
- Mou, L., Li, G., Zhang, L., Wang, T., and Jin, Z. (2016). Convolutional neural networks over tree structures for programming language processing. In *Proceedings of the AAAI conference on artificial intelligence*.

- Muepu, D. M. and Watanobe, Y. (2024). From code to ratings: Converting programming data to enhance collaborative filtering in educational online judge systems. *IEEE Access*, 12:20239–20251.
- Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L. J., and Sohl-Dickstein, J. (2015). Deep knowledge tracing. *Advances in neural information processing systems*, 28.
- Puri, R., Kung, D. S., Janssen, G., Zhang, W., Domeniconi, G., Zolotov, V., Dolby, J., Chen, J., Choudhury, M., Decker, L., et al. (2021). Codenet: A large-scale ai for code dataset for learning a diversity of coding tasks. *arXiv preprint arXiv:2105.12655*.
- Quadrana, M., Cremonesi, P., and Jannach, D. (2018). Sequence-aware recommender systems. *ACM Computing Surveys*, 51(4):1–36.
- Shen, S., Liu, Q., Huang, Z., Zheng, Y., Yin, M., Wang, M., and Chen, E. (2021). A survey of knowledge tracing: Models, variants, and applications. *IEEE Transactions on Learning Technologies*.
- Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., and Jiang, P. (2019). BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformers. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1441–1450.
- Tan, Y., Liu, H., Gao, J., Lin, K., and Shi, Y. (2021). SINE: Sparse interest network for sequential recommendation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 202–210.
- Volostnov, D. (2024). Codeforces Users Submissions Dataset (End of 2024). <https://huggingface.co/datasets/denkCF/UsersCodeforcesSubmissionsEnd2024>. Acesso em: 2024.
- Wang, S., Hu, L., Wang, Y., Cao, L., Sheng, Q. Z., and Orgun, M. (2020). Sequential recommender systems: Challenges, progress and prospects. *arXiv preprint arXiv:2001.04830*.
- Watanobe, Y., Rahman, M. M., Matsumoto, T., Rage, U. K., and Ravikumar, P. (2022). Online judge system: Requirements, architecture, and experiences. *International Journal of Software Engineering and Knowledge Engineering*, 32(06):917–946.
- Xu, Y., Ni, Q., Liu, S., Mi, Y., Yu, Y., and Hao, Y. (2022). Learning style integrated deep reinforcement learning framework for programming problem recommendation in online judge system. *International Journal of Computational Intelligence Systems*, 15(114).
- Zhang, J., Shi, X., King, I., and Yeung, D.-Y. (2017). Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*, pages 765–774.
- Zhao, W. X., Mu, S., Hou, Y., Lin, W., Chen, Y., Pan, X., Li, K., Lu, Y., Wang, H., Tian, C., et al. (2021). RecBole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM)*, pages 4653–4664.

Zheng, W., Du, Q., Fan, Y., Tan, L., Xia, C., and Yang, F. (2022). A personalized programming exercise recommendation algorithm based on knowledge structure tree. *Journal of Intelligent & Fuzzy Systems*, 42:2169–2180.

Zhu, R., Zhang, D., Han, C., Gaol, M., Lu, X., Qian, W., and Zhou, A. (2022). Programming knowledge tracing: A comprehensive dataset and a new model. In *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 298–307. IEEE.