"Eu posso aprender tudo online": Uma Análise das Notional Machines do TikTok para Aprender Programação

Maria Barbieri[1], **Jamili Amaral**[1], **Rodrigo Duran**[1]

[1]Instituto Federal de Mato Grosso do Sul (IFMS)
Nova Andradina – MS – Brazil

{maria.barbieri, jamili.amaral@estudante.ifms.edu.br;
rodrigo.duran@ifms.edu.br

Abstract. *Unsurprisingly, many instructors and students resort to non-formal educational spaces, especially video-based social networks such as TikTok, when tasked to learn to program, particularly when formal training is scarcely provided. However, it is unclear what kind of educational content such platforms provide to teachers and the quality of such content. In this work, we aim to alleviate this gap by investigating programming videos on TikTok extracting explanations – the Notional Machines – used by presenters. We analyzed 300 videos on these platforms to classify the Notional Machines used to explain topics such as variables, loops, conditionals, and functions. Our results show that, in general, the majority of the videos do not offer an explicit definition of concepts, or use shallow or "textbook" explanations, including some deemed dangerous in the literature. Unfortunately, we could not find new terminology or language that would allow educators to communicate with students from primary and secondary levels successfully. While more work is necessary to analyze a larger corpus of videos, we offer a cautionary tale for using unvetted content as a replacement for well-prepared learning materials.*

Resumo. *Não é surpreendente que muitos instrutores e estudantes recorram a espaços educacionais não-formais, especialmente redes sociais baseadas em vídeo como o TikTok, quando têm a tarefa de aprender a programar, particularmente quando o treinamento formal é raramente fornecido. No entanto, é incerto que tipo de conteúdo educacional tais plataformas fornecem aos professores e a qualidade desse conteúdo. Neste trabalho, visamos diminuir essa lacuna investigando vídeos de programação no TikTok, extraindo explicações – as Notional Machines – usadas pelos apresentadores. Analisamos 300 vídeos dessa plataforma para classificar as Notional Machines usadas para explicar tópicos como variáveis, loops, condicionais e funções. Nossos resultados mostram que, em geral, a maioria dos vídeos não oferece uma definição explícita dos conceitos, ou usam explicações superficiais ou "de livro didático", incluindo algumas consideradas perigosas na literatura. Infelizmente, não conseguimos encontrar uma nova terminologia ou linguagem que permitisse aos educadores se comunicar com sucesso com alunos dos níveis primário e secundário. Embora mais trabalho seja necessário para analisar um corpus maior de vídeos, oferecemos um alerta sobre o uso de conteúdo não verificado como substituto para materiais de aprendizagem bem preparados.*

1. Introdução

Diversos autores apresentam diferentes justificativas para o ensino de computação. Por exemplo, na perspectiva do mercado de trabalho, existe uma grande demanda não atendida por profissionais de computação [Gallindo et al. 2021]; proponentes do Pensamento Computacional (PC) [Wing 2006] argumentam que "a forma de pensar de cientistas da computação, suas heurísticas, e estratégias de solução de problemas são universalmente importantes tanto para aplicar as ideias de computação ao trabalho de outras disciplinas quanto para a vida cotidiana" e desta forma o PC deveria ser ensinado em todos os níveis de educação; a ideia de que pessoas podem "ler e escrever com computação sendo um conjunto de elementos materiais, cognitivos e sociais que geram uma nova forma de conhecer, pensar, aprender e representar o conhecimento" [DiSessa 2001] é poderosa e tem atraído a atenção de educadores e formadores de políticas públicas. Portanto, é imperativo que o acesso à Computação seja feita com equidade para que todos possam ter não apenas acesso aos benefícios da computação, mas também sejam capazes de mensurar seu impacto na sociedade [Blikstein and Moghadam 2019].

A partir de tais argumentos, muitos países expandiram o acesso ao ensino de computação para os níveis elementais de ensino. Por exemplo, em 2022 o Ministério da Educação (MEC) publicou um complemento à Base Nacional Curricular Comum (BNCC) que insere o ensino de Computação no ensino fundamental e médio [da Educação 2022]. O modelo introduzido pelo MEC é multifacetado e introduz o ensino de computação como um tripé composto pelo PC, a cultura digital (computador na sociedade, ética, e letramento digital) e o mundo digital (codificação, distribuição e processamento) [Ribeiro et al. 2023].

Entretanto, os documentos iniciais do complemento e legislação que a acompanha não deixam claro como será construído o currículo deste ensino de Computação e, principalmente, quem serão e como serão capacitados os atores responsáveis pelo ensino de Computação nas escolas. Em um país com aproximadamente 178 mil escolas de nível básico [de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP) 2023], a escala da demanda por profissionais capacitados para o ensino da computação está muito além da oferta dos cursos de licenciatura em Computação (cerca de 10 mil nos últimos 10 anos [1]). Outros formatos de capacitação formais e não-formais serão necessários para suprir esta enorme diferença de demanda.

Existem ainda vários desafios na formação docente na área de Computação [Oliveira and Cambraia 2020, Oliveira et al. 2020, dos Santos et al. 2017], em particular a ausência de formação continuada específica em escala e a falta de material didático específico de qualidade [Guarda and Silveira 2023]. Portanto, não é supresa que muitos instrutores (e alunos) recorram a ambientes não-formais de aprendizado, sendo a abundância de materiais online apontada como uma das principais razões pelo desinteresse no ensino formal por estudantes de cursos de graduação em Computação [Duran et al. 2023]. Desta forma, redes como o YouTube, TikTok, entre outras, acabam por se tornam um importante *locus* de aprendizado não-formal de computação [Haaranen 2017].

[1] https://www.sbc.org.br/documentos-da-sbc?task=download.send&id=1461&catid=133&m=0

Entretanto, ainda não está claro *como* se dá esse processo de ensino não-formal em tais redes. Em particular, é de interesse de instrutores em busca de formação pedagógica as terminologias, analogias, metáforas, em suma, as *explicações* utilizadas para apresentar tópicos de computação, particularmente em programação. Obviamente, tais artefatos pedagógicos que apresentam como conceitos ou pedaços de código são executados por uma máquina, as *Notional Machines* [Duran et al. 2021, Sorva et al. 2012] (NMs), são sensíveis ao contexto e podem ser apresentados em uma progressão, de forma que diferentes explicações atendem a momentos pedagógicos diferentes para pessoas diferentes. Em razão desta contextualização, escolhemos o TikTok como rede social de interesse por diversas razões: ela utiliza vídeos curtos que podem ser caracterizadas como "cápsulas de conteúdo" e potencialmente interessantes para apresentar conceitos de forma rápida utilizando NMs concisas e autocontidas [Bradbury 2016]; é uma rede inegavelmente popular entre estudantes da faixa etária em nível básico [González-Larrea et al. 2021]; e é uma rede com linguagem, formato e conteúdo próprio que as distingue das demais e que pode ser utilizada para o aprendizado de programação [Garcia et al. 2022].

Este trabalho almeja investigar o fenômeno do ensino não-formal de conteúdos de programação em redes sociais de vídeos curtos, tais como o TikTok, e descrever seus temas mais abordados e como esses tipicamente abordados em cursos introdutórios de programação são apresentados nesta rede social. Mais especificadamente, esse trabalho se propõe a responder às seguintes perguntas de pesquisa (PP):

1. **QP1**: Quais os temas mais abordados sobre o ensino de programação no TikTok?
2. **QP2**: Quais Notional Machines foram utilizadas no TikTok para explicar tópicos tipicamente apresentados em disciplinas iniciais de programação?

2. Trabalhos Relacionados

Se no ensino superior existe uma literatura que documenta as extensas e diversas dificuldades encontradas no ensino de computação, sobretudo o eixo codificação (comumente denominado programação) [Qian and Lehman 2017], o mesmo não acontece no ensino fundamental, onde as experiências são mais escassas e frequentemente calculadas apenas no uso de ambientes de programação em blocos [Grover et al. 2014].

A formação profissional para capacitar professores na área de computação ainda é desafiadora e possui muitas perguntas em aberto [Shanley et al. 2023]. Em particular, as licenciaturas em computação foram alvo de diversas investigações sobre as dificuldades da formação docente, que apontam vários fatores, tais como falta de motivação e foco da formação, falta de material didático apropriado, falta de clareza no mercado de atuação do profissional, falta de corpo docente com formação na área de educação de computação, entre outros fatores [dos Santos et al. 2017, Guarda and Silveira 2023, Oliveira and Cambraia 2020, Oliveira et al. 2020].

Outro aspecto na formação docente é a dificuldade em adquirir a experiência necessária para o ensino de programação. Certo é que, de alguma forma, os instrutores devem apresentar um modelo simplificado do computador e como uma linguagem de programação altera o comportamento do computador através de uma Notional Machine (NM) [Du Boulay 1986], "declarações escritas sobre como os programas de computador de um determinado tipo se comportam quando executados" [Duran et al. 2021]. Diferentes contextos, culturas, níveis educacionais e objetivos de aprendizagem podem utilizar

diferentes NM para apresentar o mesmo programa, variando o nível de detalhe, terminologia e outros aspectos. Por exemplo, ao apresentar um trecho de programa escrito em Python que apresente um loop usando a função *range* a estudantes do ensino médio, uma NM possível seria "Repita 10 vezes", enquanto uma NM do mesmo trecho de código para estudantes de graduação salientaria detalhes da função *range*, parâmetros, como loops se comportam e outros aspectos da semântica da linguagem. Ainda foi pouco explorado na literatura quais os aspectos necessários para uma NM e como avaliar empiricamente se uma determinada NM é adequada ou não a um determinado contexto [Duran et al. 2021].

Existem pouquíssimas NM documentadas na literatura [Duran et al. 2021] e a maioria é voltada ao público de graduação de cursos de tecnologia. Fincher et al. [Fincher et al. 2020] apresentam alguns exemplos e taxonomias para NMs; Berry et al. [Berry and Kölling 2013] apresentam uma notação visual para uma NM para o ensino de Java, enquanto Lewis [Lewis 2021] concretiza essa notação utilizando elementos físicos para representação de memória em Java. Em suma, a literatura relacionada a NMs é recente, mas em franca expansão devido ao interesse em como representar um modelo de execução de programas para diferentes audiências [Dickson et al. 2020].

Em geral, NMs podem utilizar termos técnicos e detalhes que estão além dos objetivos educacionais salientados pelo modelo proposto pelo MEC e seriam pouco adequados a públicos do ensino fundamental e médio. Entretanto, existe uma área correlata que explorou como diferentes tipos de analogias e metáforas são aplicadas no ensino de programação. Neste trabalho distinguimos metáforas de NMs pois NMs são atreladas a um contexto e geralmente apresentam uma semântica específica de uma linguagem de programação (ou pelo menos de um grupo de linguagens de programação similares). Contudo, essa distinção é semântica e muitas vezes uma analogia ou metáfora pode ser um exemplo de NM e vice-versa, o que frequentemente observaremos em nossos resultados. Embora os termos não sejam exatamente intercambiáveis, consideramos que o que exploramos neste trabalho são NMs apresentados em vídeo. Cao et al. [Cao et al. 2016] observaram que analogias possuem valor didático, principalmente em curto prazo, e que tais analogias são dependentes de contexto. Portanto, Bettin e Ott [Bettin and Ott 2023] propõem um modelo teórico de design de analogias isomórficas em diferentes contextos que podem ser úteis a educadores que trabalham em diferentes níveis educacionais. Harper et al. [Harper et al. 2024] exploraram um framework semelhante para investigar o valor de metáforas propostas pelos próprios estudantes para o aprendizado de programação, com alguns resultados promissores. Todavia, vários autores já exploraram como algumas metáforas são potencialmente nocivas e podem induzir a *misconceptions* [Chiodini et al. 2021], modelos mentais incorretos e persistentes sobre o funcionamento de programas. Um exemplo é a metáfora de que "uma variável é uma caixa", que induz à misconception de que uma variável pode armazenar vários valores simultaneamente [Du Boulay 1986].

Este trabalho visa explorar um viés semelhante ao de Harper et al., mas extraindo NMs de vídeos em redes sociais, onde potencialmente é empregada uma linguagem mais acessível e próxima dos estudantes do ensino básico. As redes sociais também possuem a vantagem de oferecerem conteúdos curtos, mais próximos da capacidade de atenção humana [Bradbury 2016] e altamente engajantes não apenas para entretenimento, mas também para outros tópicos, tais como ativismo social [Milton et al. 2023].

Uma das redes sociais mais utilizadas é o TikTok, uma rede social originalmente chinesa que se expandiu rapidamente pelo mundo justamente por apresentar vídeos muito curtos, de em média 30 segundos, sobre os mais variados tópicos e sendo atualmente uma das redes sociais mais acessadas por jovens e possui um dos maiores crescimentos de público entre outras redes sociais [González-Larrea et al. 2021]. Como apresenta Haaranen, [Haaranen 2017], o ato de programar em vídeo pode ser engajador e atrair um público diferente, menos habituado ao aprendizado formal, para o universo da computação e Garcia et al. [Garcia et al. 2022] se valem do mesmo fenômeno para investigar o potencial do TikTok como plataforma de aprendizado de programação online utilizando vídeos curtos.

3. Metodologia

Para responder à **QP1**, foi realizada uma análise qualitativa em duas etapas: a primeira procurou atribuir *labels* a cada um dos vídeos; a segunda agrupou os labels em temas similares. Para a primeira etapa, foram analisados os primeiros 50 vídeos mais relevantes (de acordo com o motor de busca do TikTok) utilizando a query "programação". Como critério de inclusão, os vídeos deveriam tratar do tema para evitar bots que adicionam tags automaticamente a vídeos e serem em Português. Para cada vídeo selecionado foram coletados seus links, tags, título e sua descrição. Os vídeos do TikTok foram armazenados manualmente pois os links gerados pela plataforma não são permanentes. Cada vídeo foi posteriormente assistido e sumarizado.

Em seguida, um lote de 10 vídeos foi alocado para duas avaliadoras (autoras do trabalho) para atribuição de labels iniciais a cada vídeo, tarefa esta realizada de forma independente por cada avaliadora. Posteriormente, as mesmas reuniram-se para avaliar a concordância entre os labels. Labels similares, com utilização apenas de termos distintos, mas com significado similar, foram unificados. Labels cujo significado fosse dissimilar não foram alterados nesta etapa. Posteriormente, foi calculado o κ de Cohen como critério de concordância [McHugh 2012]. Caso o κ fosse inferior a 0.7, uma rodada de discussão seria realizada para harmonizar os labels e o processo se repetiria enquanto não houvesse concordância. Como o valor de κ foi atingido logo após a primeira rodada de discussão, as avaliadoras prosseguiram aplicando os labels ao restante dos vídeos. Posteriormente, cada avaliadora ficou responsável por aplicar de forma independente os labels produzidos anteriormente no corpus restante de vídeos (45 vídeos para cada avaliadora). Caso necessário, novos labels eram acrescentados e o processo de refinamento repetido até que todo o corpus tivesse sido analizado atingindo um grau de concordância satisfatório.

Na segunda estapa de análise da QP1, ambas avaliadoras analisaram em conjunto os labels produzidos na primeira etapa e os agrupavam em temas similares em um processo interativo onde temas cada vez mais complexos eram produzidos até que fosse atingida uma saturação [Lewis 2012] e novos temas não puderam ser acrescidas.

Para responder às **QP2**, metodologia similar foi empregada, com pequenas alterações: a query "programação" foi estendida com os tópicos de interesse ("programação variáveis", "programação condicionais", "programação loops" e "programação funções"); para cada query foram selecionados os 50 primeiros vídeos sugeridos pelo motor de busca da plataforma; os labels foram construídos a partir da *definição* (Notional Machine) apresentada em cada vídeo, ao invés de seu sumário; os labels foram posteriormente agrupados em temas.

Para todos os temas foi calculado a frequência deste em relação aos demais. Vídeos que não apresentavam uma definição visual ou oral foram classificados como "Indefinidos". O dataset do trabalho esta disponível em `https://osf.io/u72sq/?view_only=f015720aa9c44e0b96e13f9d08569551`.

4. Resultados & Discussão

4.1. QP1: Quais os temas mais abordados sobre o ensino de programação no TikTok

A análise dos vídeos de programação no TikTok evidencia que o foco primordial dos criadores de conteúdo está no entretenimento, não em aspectos educacionais. Cerca de 38% desses vídeos foram categorizados como "Vida de Programador", nos quais os autores produzem conteúdos voltados para suas experiências e rotinas envolvendo programação, expectativa/realidade da profissão e aspectos positivos e negativos da área. Adicionalmente, 23% foram categorizados como "Humor", apresentando abordagens humorísticas sobre o tema com o intuito de divertir aqueles que já possuem familiaridade com o tema.

Alguns vídeos apresentavam caráter educativo. Cerca de 28% foram classificados como "Dicas para Iniciantes", oferecendo orientações para aqueles que estão começando ou pretendem ingressar na área, enquanto 2% dos vídeos foram nomeados como "Cursos", nos quais são recomendados cursos para aprendizes. Cerca de 9% dos vídeos foi categorizado como "Tecnologia", abordando inovações e aspectos tecnológicos, como inteligência artificial, e seu uso no dia a dia.

4.2. Quais Notional Machines foram utilizadas no TikTok para explicar tópicos tipicamente apresentados em disciplinas iniciais de programação?

4.3. Variáveis

Cerca de 42% dos vídeos analisados no TikTok foram categorizados como "Indefinido", ou seja, não apresentam uma definição ou explicação clara sobre o conceito. Aproximadamente 28% dos vídeos apresentam variáveis como "caixas, gavetas, ou blocos", 22% definiram variáveis como um "espaço na memória para armazenamento", e 8% como um "nome definido para armazenamento". Apresentamos agora alguns exemplos de NMs em cada um dos temas.

Caixas, gavetas ou blocos: Alguns vídeos apresentam este tema com NMs como *"variável é um espaço que a gente aloca na memória, como uma gaveta que se coloca uma etiqueta"*, *"variáveis são como caixas que amazenam valores"*, ou como *"variáveis são caixas com memória de armazenamento"*. Estas NMs, embora frequentemente utilizadas, podem causar concepções incorretas nos estudantes [Du Boulay 1986, Sorva et al. 2012, Chiodini et al. 2021, Qian and Lehman 2017], uma vez que o estudante pode criar o modelo mental de que variáveis podem armazenar vários valores simultaneamente, ou que variáveis possuem histórico de valores.

Espaço na memória para armazenamento: Este tema inclue NMs que definem variáveis como *"um espaço na memória do computador usado para armazenar dados que podem ser acessados e manipulados ao longo do programa"*, *"variáveis são espaços criados dentro da memória RAM que permitem o armazenamento temporário de dados"*. Estas NMs utilizam-se de conceitos que devem ser apresentados de forma concomitante

aos estudantes. A ideia de que existem tipos de memórias no computador (e.g., RAM vs armazenamento), de que esta memória é limitada e portanto deve ser alocada e desalocada, e a própria ideia de mutabilidade das variáveis são desejáveis para alguns públicos (e.g., superior) mas de difícil compreensão para outros (e.g., fundamental). Consideramos este um caso típico onde tal NM deve ser acompanhada de um critério para uso [Duran et al. 2021].

Nome definido para armazenamento: Uma pequena porcentagem dos vídeos apresentaram variáveis como um *"nome definido para armazenar dados de forma simples"*, ou que *"permitem guardar um valor específico dentro de um nome previamente definido"*. Tais definições apresentam variáveis como um *alias* para um dado, mas sem fazer referência explícita a memória. Podem ser utilizados em casos onde o instrutor pode fazer referência ao conceito de imutabilidade, e até mesmo trabalhar a ideia de que existe uma ordem de ações para a definição de uma variável onde também seria possível inferir que uma variável sempre armazena um valor (e.g., não seria possível criar variáveis não inicializadas).

4.4. Condicionais

Cerca de 60% dos vídeos com o tema condicionais foram classificados como "Indefinidos". A análise mostra que 17,5% dos vídeos apresentam condicionais como uma "condição for satisfeita/favorável/cumprida", e 7,5% dos vídeos definem condicionais como uma "frase onde se passa uma lógica condicional" e 15% dos vídeos apresentam condicionais como "If e else são estruturas alteráveis".

Condição for satisfeita/favorável/cumprida: Os vídeos nesta categoria apresentam condicionais como uma *"condição que está estabelecida no código foi verificada e comprovada como verdadeira"*, ou que *"condicionais são perguntas: se determinada condição for propícia irá executar um bloco e senão outro bloco"*, ou ainda que a *"estrutura condicional só funciona se a condição for satisfeita"* ou *"o código executa apenas quando uma situação acontecer"*. Novamente, pelo menos duas NMs apresentadas estão intimamente ligadas a uma *misconception* de que o código de um condicional só executa quando o resultado da expressão é verdadeiro (e portanto nada pode acontecer quando o resultado é falso) [Qian and Lehman 2017, Sorva et al. 2012, Chiodini et al. 2021]. Obviamente, o instrutor pode fazer uso de uma NM *incompleta* com objetivos pedagógicos e não introduzir um bloco else, mas tal nuance deveria ser explicitada: estudantes cometem erros ao escrever blocos else [Sorva et al. 2012], portanto faria sentido introduzir NMs em partes [Duran et al. 2021, DiSessa 2018] e posteriormente apresentar o bloco else.

Frase onde se passa uma lógica condicional: Alguns vídeos apresentaram condicionais como *"uma frase que permite que o código execute diferentes ações com base em um valor ou condição"*, ou que *"condicionais têm como principal função o desvio condicional do algoritmo"*, similar à NM apresentada por outro vídeo: *"expressão condicional é uma frase onde se passa uma lógica condicional dentro dela"*. Nessas NMs temos explicitamente introduzidos os conceitos de expressão, lógica condicional e desvio de código, mesmo que sem explicitamente mencionar blocos. Tais NMs poderiam necessitar da introdução de tópicos como expressões Booleanas e outros conceitos relacionados de forma concomitante.

If e else são estruturas alteráveis: Nestes vídeos as estruturas condicionais foram apre-

sentadas como *estruturas mutáveis*, ou seja, estruturas que podem ser usadas para alterar o fluxo de execução de um programa. Por exemplo, um apresentador denotou a NM que *"if é o se sim, else é o se não"* e que *"se a condição for verdadeira executa o corpo if, senão, o else"*. Outros exemplos de NMs desta categoria são que *"testes lógicos, usados para mostrar um resultado ou mudar a sequência lógica da aplicação, quando uma condição for falsa ou verdadeira"*, ou que *"if-else é uma estrutura de controle condicional que permite que o programa execute diferentes blocos de códigos com base em uma condição booleana"*. Aqui temos explicitamente a introdução tanto da sintaxe (if e else) quanto do conceito dos Booleanos (verdadeiro e falso).

4.5. Loops

Na análise dos vídeos de Loops, 78% foram classificados como "Indefinido", 13% dos vídeos explicaram que loops eram "Estruturas de repetição", para se utilizar quando quer repetir código, por exemplo, e outros 9% que eles "Funcionam com uma condição específica sobre uma variável".

Estruturas de repetição: Uma pequena parcela dos vídeos definem loops como *"são estruturas de repetição com um recurso para desenvolver tarefas repetidas em um loop contínuo"* ou *"estruturas de repetição que são utilizadas quando se quer repetir código"*. Nota-se que tais NMs são vagas e pouco informativas. Não há menção sobre o que se repete ou quantas vezes uma repetição deveria ser realizada, critérios para se definir ou criar uma repetição, ou mesmo exemplos concretos de uso. O segundo exemplo, inclusive, dá a noção que a repetição é usada quase como um recurso "copia e cola" de código, o que embora possa ser um dos aspectos de uso de uma repetição, pode levar a modelos mentais incorretos ou incompletos [DiSessa 2018, Qian and Lehman 2017, Chiodini et al. 2021]. Não é absurdo conjecturar que alguns estudantes, expostos a tal NM, possam desenvolver o modelo mental de que o código dentro do laço de repetição é sempre igual, o que leva ao mesmo *comportamento* do programa repetidas vezes (o que só é verdade em alguns casos específicos: e.g., desenhar figuras geométricas utilizando símbolos, tais como uma linha). Obviamente, se tais NMs levam a tais modelos ainda é uma pergunta de pesquisa aberta (veja a Seção 5 para mais detalhes).

Condição específica sobre uma variável: Os vídeos deste tema fazem referência explícita ou implícita ao que alguns autores chamam de *variável de controle* na repetição. Por exemplo, *"repetição de código que especifica a quantidade de números nele contido"*, ou que a repetição *"funciona quando você obtém de uma variável e uma condição que afirme essa variável"*. A primeira parece fazer referência a um loop com número definido de repetições, ao estilo de um *for*. Essa NM foi utilizada em diversas linguagens de programação, tais como Python e Scratch, onde tal explicação poderia ser apropriada, desde o objetivo fosse apresentar a ideia, simplificada, de que toda repetição acontece por um número determinado de vezes [Duran et al. 2021]. Obviamente, nem sempre este é o objetivo instrucional e tal explicação seria limitada. O segundo exemplo atrela a ideia de repetição a uma variável, sem apresentar entretanto conceitos como testes Booleanos e outros conceitos relacionados a expressões que tornariam a explicação mais completa (veja [Duran et al. 2021] para mais critérios de avaliação de NMs).

4.6. Funções

Os resultados da análise dos vídeos sobre funções mostraram um aumento na parcela de vídeos categorizados como "Indefinido", correspondendo a aproximadamente de 75% do total. Cerca de 17.5% dos vídeos foram categorizados como "Definições específicas do sistema ou linguagem", enquanto 5% dos vídeos foram categorizados como "função é usada para não ter que repetir código". O tema "nomenclaturas que podem receber parâmetros e retornar um valor" representou 2.5% dos vídeos analisados.

Definições específicas do sistema ou linguagem: Uma parte dos vídeos analisados apresenta a NM de que funções representam funções específicas de determinada linguagem, sistema ou biblioteca. Por exemplo, ao apresentar funções, um apresentador utilizando Python explica que a *"função Range é como período, sendo usada em estruturas de laço"*. Outros vídeos que apresentam outras funções, tais como a *concat*, que *"retorna um novo array contendo todos os arrays ou valores passados como parâmetro"*, ou a função *reverse* a qual *"é usada para inverter a ordem dos elementos de um array"*. Vídeos que focam em apenas uma função preexistente em uma ou mais linguagens de programação têm seu valor pedagógico, particularmente para iniciantes buscando funcionalidades úteis para resolvr um problema. Outros vídeos apresentam uma definição mais abstrata de funções, tal como uma *"função é uma funcionalidade criada no sistema ou que vem pronta"*. A ideia de "consumir antes de produzir" funções não é nova e trabalhos anteriores [Fisler et al. 2016, Duran et al. 2018] e designs de curso [2] exploram uma trajetória pedagógica onde os estudantes são apresentados primeiramente ao funcionamento de funções prontas (bibliotecas), seu uso para posteriormente definir suas próprias funções.

Função é usada para não ter que repetir código: Alguns apresentadores exemplificam que *"função é usada para não ter que repetir código, para usá-lo para uma mesma finalidade"*, o que uma *"função na programação é uma parte guardada para ser usada em determinada parte, ou evento específico"*. Estas NMs reforçam os conceitos de reuso de código, frequentemente associadas ao uso de funções como sub-rotinas, ou até mesmo parte de uma biblioteca do usuário. Embora tais NMs sejam úteis como parte de uma série de argumentos em favor do uso de funções, estudantes, principalmente iniciantes, observam pouco valor neste argumento, principalmente quando as atividades não forçam o reuso de código ou os programas são suficientemente pequenos para que o estudante não sinta que seja necessário "organizar" o código em sub-elementos.

Nomenclaturas que podem receber parâmetros e retornar um valor: A NM que apresenta funções como *"blocos de código nomeados que podem receber parâmetros e retornar um valor"* foi usada por alguns apresentadores. Essa NM pode introduzir um viés mais funcional, evocando conceitos da matemática como parâmetros e retorno obrigatório de um valor. Diversos autores são ardentes defensores deste tipo de abordagem [Felleisen et al. 2018], argumentando que a mesma fornece uma abstração mais completa e livre de efeitos, induzindo menos *misconceptions* [Sorva et al. 2012].

5. Conclusão

A análise dos vídeos de programação no TikTok se mostrou ligeiramente decepcionante. Não em relação à PQ1, afinal o TikTok não é uma plataforma educacional e portanto

[2] `https://plus.cs.aalto.fi/o1/2023/`

não é surpreendente que a maioria dos vídeos mostre a "vida de programador" como entretenimento. De certa forma, os vídeos foram educativos ao mostrar muitas situações profissionais de forma descontraída desmistificando elementos da profissão.

Parte da decepção veio da análise dos vídeos relacionados à PQ2. A primeira observação é de que, quanto mais avançado o conceito, menos definições eram apresentadas. Muitos vídeos eram apenas sessões de codificação sem qualquer explicação (similares aos vídeos analisados por [Haaranen 2017]). Esses vídeos parecem presumir que seu público alvo já possui um bom conhecimento prévio de programação, incompatível com os professores ou estudantes do nível fundamental. Não fomos capazes de encontrar a linguagem ou formato inovadores que hipotetizamos inicialmente. Afinal de contas, se vários estudantes apontam que "eu consigo aprender tudo online" [Duran et al. 2023], era de se imaginar que estes apresentadores pudessem apresentar uma linguagem, terminologia, analogias ou explicações que melhor se conectassem com esse público mais jovem presente no TikTok. De forma geral, as NMs apresentadas são muito similares às apresentadas em materiais didáticos em nível de graduação, incluindo algumas controversas ("Variáveis são caixas" [Du Boulay 1986]), ou outras muito superficiais.

Outro aspecto a ser destacado é a ampla variedade de linguagens apresentadas na pesquisa, entre as quais se destacam Python, JavaScript, Java e C + +. No entanto, é interessante observar que a maioria desses criadores, apesar de muitos trabalharem ou estarem estudando na área, não alegam possuir uma formação acadêmica em tecnologia, sendo assim, muito da sua bagagem sendo adquirida por meio de plataformas de exibição de vídeos, como o TikTok, ou mesmo por cursos e jogos na internet. Nenhum utilizava ambientes de programação mais voltadas ao ensino fundamental (e.g., Scratch).

É possível apenas conjecturar se tais NMs são "boas" ou "ruins", especialmente sem um contexto de aplicação específico e a falta de pesquisas empíricas de seu uso. Como trabalhos futuros, sugerimos colocar tais NMs a prova em diferentes contextos de aprendizado.

5.1. Limitações

O método de seleção dos vídeos do TikTok impõem importantes limitações e viéses a este trabalho. Além do fato do motor de busca apresentar os vídeos mais relevantes de acordo com algum critério que não é amplamente conhecido, foi observado que os resultados da busca mudam de forma substancial com o tempo, o que dificulta de sobremaneira a replicabilidade dos resultados deste trabalho. Desta forma, nossas reivindicações sobre generalização são tênues.

Outra limitação é que não necessariamente os vídeos analisados foram produzidos com intento educacional, e portanto dificultam a sua classificação como Notional Machines. Não surpreendentemente, os resultados da QP1 mostraram que a maioria dos vídeos eram entretenimento, e não educacionais. Apesar de estarmos mais confidentes de que os vídeos selecionados para análise da QP2 eram mais voltados ao ensino do tópico em questão, não há forma de garantir que sempre era esta a intenção. A classificação dos vídeos também pode ter tido sua validade diminuída pela dificuldade em classificar alguns desses vídeos, precisamente pelo seu teor, com poucas explicações e não voltado especificadamente ao ensino. As autoras buscaram mitigar estes aspectos seguindo um percurso metodológico utilizado anteriormente na literatura.

Referências

Berry, M. and Kölling, M. (2013). The design and implementation of a notional machine for teaching introductory programming. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education*, WiPSE '13, page 25–28, New York, NY, USA. Association for Computing Machinery.

Bettin, B. and Ott, L. (2023). Pedagogical prisms: Toward domain isomorphic analogy design for relevance and engagement in computing education. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*, ITiCSE 2023, page 410–416, New York, NY, USA. Association for Computing Machinery.

Blikstein, P. and Moghadam, S. H. (2019). Computing education. *The Cambridge handbook of computing education research*, pages 56–78.

Bradbury, N. A. (2016). Attention span during lectures: 8 seconds, 10 minutes, or more?

Cao, Y., Porter, L., and Zingaro, D. (2016). Examining the value of analogies in introductory computing. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*, ICER '16, page 231–239, New York, NY, USA. Association for Computing Machinery.

Chiodini, L., Moreno Santos, I., Gallidabino, A., Tafliovich, A., Santos, A. L., and Hauswirth, M. (2021). A curated inventory of programming language misconceptions. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, ITiCSE '21, page 380–386, New York, NY, USA. Association for Computing Machinery.

da Educação, M. (2022). Base nacional comum curricular: Computação, complemento à bncc.

de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP), B. I. N. (2023). Censo escolar da educação básica 2022: Resumo técnico.

Dickson, P. E., Brown, N. C. C., and Becker, B. A. (2020). Engage against the machine: Rise of the notional machines as effective pedagogical devices. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '20, page 159–165, New York, NY, USA. Association for Computing Machinery.

DiSessa, A. A. (2001). *Changing minds: Computers, learning, and literacy*. Mit Press.

DiSessa, A. A. (2018). A friendly introduction to "knowledge in pieces": Modeling types of knowledge and their roles in learning. In *Invited lectures from the 13th international congress on mathematical education*, pages 65–84. Springer.

dos Santos, W. O., Silva, C., and Hinterholz, L. (2017). Licenciatura em computação: Desafios e oportunidades na perspectiva do estudante. In *Anais do Workshop de Informática na Escola*, volume 23, pages 885–894.

Du Boulay, B. (1986). Some difficulties of learning to program. *Journal of Educational Computing Research*, 2(1):57–73.

Duran, R., Bim, S. A., Gimenes, I., Ribeiro, L., and Correia, R. C. M. (2023). Potential factors for retention and intent to drop-out in brazilian computing programs. *ACM Trans. Comput. Educ.*, 23(3).

Duran, R., Sorva, J., and Leite, S. (2018). Towards an analysis of program complexity from a cognitive perspective. In *Proceedings of the 2018 ACM Conference on International Computing Education Research*, ICER '18, page 21–30, New York, NY, USA. Association for Computing Machinery.

Duran, R., Sorva, J., and Seppälä, O. (2021). Rules of program behavior. *ACM Transactions on Computing Education (TOCE)*, 21(4):1–37.

Felleisen, M., Findler, R. B., Flatt, M., and Krishnamurthi, S. (2018). *How to design programs: an introduction to programming and computing*.

Fincher, S., Jeuring, J., Miller, C. S., Donaldson, P., du Boulay, B., Hauswirth, M., Hellas, A., Hermans, F., Lewis, C., Mühling, A., Pearce, J. L., and Petersen, A. (2020). Notional machines in computing education: The education of attention. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '20, page 21–50, New York, NY, USA. Association for Computing Machinery.

Fisler, K., Krishnamurthi, S., and Siegmund, J. (2016). Modernizing plan-composition studies. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, page 211–216, New York, NY, USA. Association for Computing Machinery.

Gallindo, S., Oliveira, M., and Persona, H. (2021). Demanda de talentos em tic e estratégia stcem. *Relatório técnico Brascomm*, 1(1):1–32.

Garcia, M. B., Juanatas, I. C., and Juanatas, R. A. (2022). Tiktok as a knowledge source for programming learners: a new form of nanolearning? In *2022 10th International Conference on Information and Education Technology (ICIET)*, pages 219–223. IEEE.

González-Larrea, B., Hernández-Serrano, M. J., and Renés-Arellano, P. (2021). Analysis of the prevalence type in the adolescents' social networks use by age and gender. In *Ninth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'21)*, pages 423–427.

Grover, S., Cooper, S., and Pea, R. (2014). Assessing computational learning in k-12. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*, pages 57–62.

Guarda, G. F. and Silveira, I. F. (2023). Desafios e caminhos para a implementação da bncc computação no ensino médio. In *Anais do XXIX Workshop de Informática na Escola*, pages 798–809. SBC.

Haaranen, L. (2017). Programming as a performance: Live-streaming and its implications for computer science education. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, pages 353–358.

Harper, C., Tran, K., and Cooper, S. (2024). Conceptual metaphor theory in action: Insights into student understanding of computing concepts. In *Proceedings of the 55th*

ACM Technical Symposium on Computer Science Education V. 1, SIGCSE 2024, page 463–469, New York, NY, USA. Association for Computing Machinery.

Lewis, C. M. (2012). The importance of students' attention to program state: a case study of debugging behavior. In *Proceedings of the ninth annual international conference on International computing education research*, pages 127–134.

Lewis, C. M. (2021). Physical java memory models: A notional machine. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, SIGCSE '21, page 383–389, New York, NY, USA. Association for Computing Machinery.

McHugh, M. L. (2012). Interrater reliability: the kappa statistic. *Biochemia medica*, 22(3):276–282.

Milton, A., Ajmani, L., DeVito, M. A., and Chancellor, S. (2023). "i see me here": Mental health content, community, and algorithmic curation on tiktok. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–17.

Oliveira, W. and Cambraia, A. C. (2020). Desafios na formação de professores de computação: Reflexões e ações em construção. In *Anais do XXVI Workshop de Informática na Escola*, pages 319–328. SBC.

Oliveira, W., França, R., Lemos, A., da Cruz, M. K., Scaico, P., Amaral, H., and Teixeira, L. P. (2020). Os desafios enfrentados pela licenciatura em computação que a comunidade de educação em computação precisa conhecer. In *Anais do XXVIII Workshop sobre Educação em Computação*, pages 191–195. SBC.

Qian, Y. and Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Trans. Comput. Educ.*, 18(1).

Ribeiro, L., Foss, L., Cavalheiro, S. A. D. C., Kniphoff da Cruz, M. E. J., and Soares de França, R. (2023). The brazilian school computing standard. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, pages 53–58.

Shanley, N., Pérez-Quiñones, M. A., Martin, F., Pugalee, D., Ahlgrim-Delzell, L., and Hart, E. (2023). K-12 teacher experiences from online professional development for teaching apcsa. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, pages 1001–1006.

Sorva, J. et al. (2012). *Visual program simulation in introductory programming education*. Aalto University.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3):33–35.