Avaliação da Capacidade de LLMs para Especificar *Workflows**

Paula Woyames[1], Débora Pina[2], Liliane Kunstmann[3], Marta Mattoso[2], Daniel de Oliveira[1]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)

[2]Universidade Federal do Rio de Janeiro (COPPE/UFRJ)

[3]Mendelics Análise Genômica

pnwoyames@id.uff.br,{dbpina,marta}@cos.ufrj.br,

liliane.kunstmann@mendelics.com.br, danielcmo@ic.uff.br

Abstract. *This paper evaluates the use of Large Language Models (LLMs) for specifying workflows from natural language descriptions. Three LLMs (GPT-4o, DeepSeek V3, and Command-A), two prompt versions, and four workflow systems (Nextflow, Parsl, Dask, and Airflow) were compared across three levels of workflow complexity. The results indicate that prompts with examples produce specifications that are syntactically more correct and semantically better aligned with the natural language description, with notable performance from GPT-4o and Dask. Nevertheless, challenges remain in generating complex workflows, particularly those involving parallelism.*

Resumo. *Este artigo avalia o uso de Modelos de Linguagem de Grande Escala (LLMs) na especificação de workflows a partir de descrições em linguagem natural. Foram comparados três LLMs (GPT-4o, DeepSeek V3 e Command-A), duas versões de prompts e quatro sistemas de workflow (Nextflow, Parsl, Dask e Airflow), aplicados a três níveis de complexidade de workflows. Os resultados indicam que prompts com exemplos produzem especificações sintaticamente mais corretas e semanticamente mais alinhadas com a especificação em linguagem natural, com destaque para o desempenho do GPT-4o e do Dask. Ainda assim, desafios persistem na geração de workflows complexos e que envolvem paralelismo.*

1. Introdução

Os *workflows* são abstrações utilizadas para especificar o encadeamento de atividades que compõem um experimento científico baseado em simulação [de Oliveira et al. 2019]. Essas atividades podem envolver o processamento de grandes volumes de dados, o que impõe a necessidade de que a execução dos *workflows* ocorra em ambientes como nuvens, *clusters* e supercomputadores. De forma a gerenciar a execução eficiente dos *workflows*, diversos sistemas de *workflows* foram propostos [de Oliveira et al. 2019]. Esses sistemas oferecem funcionalidades como a coleta de dados de proveniência, o tratamento de falhas, o monitoramento em tempo real, alocação de recursos de forma eficiente, *etc*.

Apesar de os sistemas de *workflow* oferecerem várias facilidades, a especificação de *workflows* ainda é uma tarefa complexa para os usuários. Essa complexidade decorre de uma combinação de fatores. Um dos principais desafios está relacionado à necessidade de aprendizado da linguagem usada no sistema de *workflow*, já que cada sistema possui sua própria linguagem de especificação. Outro fator é a heterogeneidade do ecossistema de *software* que

*Os autores gostariam de agradecer pelo apoio financeiro da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) – Código de Financiamento 001, do CNPq e da FAPERJ.

é usado em um *workflow*. Em muitos casos, a especificação de um *workflow* exige o encadeamento de múltiplas bibliotecas e *frameworks*. Compreender como esses componentes interagem e como configurá-los corretamente não é uma tarefa trivial. Por fim, deve-se considerar o conhecimento sobre a aplicação. A correta especificação de um *workflow* requer entendimento do problema a ser resolvido, incluindo quais atividades devem ser realizadas, em que ordem, *etc*. A combinação desses fatores impõe uma curva de aprendizado grande.

Nos últimos anos, novas tecnologias têm surgido com potencial para facilitar a tarefa de especificação de *workflows*. Um exemplo são os Modelos de Linguagem de Grande Escala (*Large Language Models* ou LLMs), que vêm se consolidando como alternativas para a execução de diversas tarefas computacionais complexas a partir de instruções em linguagem natural, fornecidas por meio de *prompts* textuais [Dong et al. 2024]. Esses modelos já foram aplicados em diferentes domínios, como a detecção de discurso de ódio *online* [Paiva et al. 2025] e a geração de código-fonte [Koziolek et al. 2024]. A aplicação de LLMs na especificação de *workflows* possui potencial, pois pode reduzir a necessidade de conhecimento técnico especializado e o domínio de linguagens específicas de especificação.

Entretanto, estudos indicam que ainda existem restrições no uso de LLMs, especialmente quanto à sensibilidade à formulação dos *prompts*. Pequenas variações na estrutura do *prompt* podem resultar em respostas diferentes tanto em termos da sintaxe quanto da semântica do resultado. Esses aspectos sugerem que, embora os LLMs apresentem potencial como interface entre usuários e sistemas de *workflow*, seu uso na especificação automatizada de *workflows* ainda requer investigações adicionais, com o objetivo de entender e eventualmente contornar suas limitações atuais, assegurando tanto a corretude dos *workflows* gerados quanto a aderência ao que foi originalmente especificado em linguagem natural. Trabalhos anteriores já destacam essas limitações e apresentam resultados preliminares sobre o tema [Yildiz and Peterka 2025, Duque et al. 2023].

Este artigo apresenta um estudo sobre o uso de LLMs para a especificação automatizada de *workflows* a partir de descrições fornecidas em linguagem natural. A metodologia seguida visa investigar a viabilidade e as limitações dessa abordagem, considerando diferentes variáveis. Especificamente, o estudo compara diferentes LLMs, *i.e.*, ChatGPT, DeepSeek e Command, bem como a análise de variações na formulação dos *prompts* e de níveis graduais de complexidade dos *workflows* a serem gerados. Além disso, o estudo considera quatro sistemas de *workflow*, *i.e.*, Parsl, Apache Airflow, Dask e Nextflow, cada um com diferentes características e paradigmas de execução. Essa diversidade permite avaliar a capacidade dos LLMs de adaptar suas respostas a diferentes sintaxes e modelos de execução.

2. Modelos de Linguagem de Grande Escala e Engenharia de *Prompt*

Os LLMs têm se destacado por conta da habilidade de compreender e gerar textos com coerência semântica, mesmo em contextos complexos e com um certo nível de ambiguidade. Os LLMs são construídos a partir do componente decodificador da arquitetura *Transformer* [Vaswani et al. 2017], e se diferenciam não só pela sua escala, *i.e.*, bilhões de parâmetros no modelo, mas também pelo surgimento de capacidades que não se manifestam em versões menores dos modelos como generalização em tarefas não vistas durante o treinamento. O surgimento e a popularização de LLMs como o GPT-3, que serviu de base ou inspiração para diversos LLMs subsequentes, evidenciaram tanto para a comunidade científica quanto para o público geral o potencial desses modelos. Esse potencial deriva, em grande parte, do mecanismo de aprendizado por contexto [Dong et al. 2024].

No aprendizado por contexto, a tarefa a ser desempenhada é apresentada ao LLM por

meio de um *prompt* textual, escrito em linguagem natural, e que pode ser enriquecido com exemplos adicionais. O *prompt* funciona como uma instrução que guia a geração da resposta pelo LLM, sem necessidade de mudanças dos pesos do modelo. Embora o uso de *prompts* represente uma vantagem, essa característica também faz com que o *prompt* influencie diretamente a resposta gerada pelo LLM, uma vez que os mesmos produzem saídas de acordo com a associação estatística entre palavras, frases e contextos previamente observados durante o treinamento. Assim, a qualidade da resposta gerada está condicionada à clareza e precisão semântica do *prompt*, o que torna sua elaboração um fator crítico.

Dada essa característica, surge a área de engenharia de *prompts*, cujo objetivo é investigar estratégias de formulação textual que maximizem o desempenho de LLMs em tarefas específicas. Entre as estratégias destacam-se a inclusão de exemplos diretamente no corpo do *prompt*, o que caracteriza a abordagem de *few-shot prompting* [Dong et al. 2024]; a incorporação de personas que influenciam o comportamento e o estilo discursivo do modelo [Choi and Li 2024]; e a decomposição do raciocínio em etapas sucessivas, conforme a técnica de *chain-of-thought prompting* (CoT) [Wei et al. 2022]. Tais estratégias têm apresentado resultados promissores em uma variedade de domínios e tarefas. Entretanto, a escolha do *prompt* adequado para cada tarefa ainda é um desafio em aberto.

3. Trabalhos Relacionados

[Xu et al. 2024] propõem uma ferramenta de geração automatizada de *workflow* chamada LLM4Workflow. A LLM4Workflow utiliza a capacidade de aprendizado contextual dos LLMs para gerar especificações de *workflows* corretos e executáveis. Apesar de representar um avanço, a LLM4Workflow não utiliza somente *prompts*, mas necessita de uma base de conhecimento de APIs, bibliotecas, *etc* para que a especificação seja gerada. [Yildiz and Peterka 2025] exploram quatro LLMs (ChatGPT 4.0, o1-preview, Claude 3.5 Sonnet e LLaMA-3.8B) com cinco sistemas de *workflow* (ADIOS, Henson, PyCOMPSs e Wilkins) para avaliar a capacidade de LLMs de gerar especificações de *workflows* baseados em descrições em linguagem natural. O LLM foi avaliado de acordo com a corretude do *script* executável gerado para cada sistema de *workflow*. Entretanto, os autores não focam no uso de prompts progressivos, o que pode fazer com que o LLM não gere as melhores especificações possíveis.

[Sänger et al. 2024] apresentam uma avaliação da eficiência do ChatGPT-3.5 para compreender e explicar, sugerir e aplicar mudanças, adaptar e estender *workflows*. Os autores apontam que a escolha das palavras no *prompt* é relevante para os resultados. Foram conduzidos três estudos que mostram que o ChatGPT tem boas capacidades de compreensão, adaptação e extensão de *workflows*, porém, encontra dificuldades em tarefas mais complexas e exploratórias, ou na identificação de termos ambíguos. [Zhang et al. 2024] propõem um *dataset* textual chamado MASSW, que apresenta dados de sumarização de múltiplos aspectos de *workflows* científicos. Este conjunto inclui mais de 150K publicações de 17K conferências. O *dataset* permite que pesquisadores desenvolvam e avaliem métodos de IA que permitam a geração de *workflows* eficazes, acelerando as descobertas e inovações na área. O trabalho utiliza três LLMs (ChatGPT-3.5 e 4, e Mixtral 8x7B) para extrair cinco aspectos das publicações armazenadas, que são: contexto, ideia central, metodologia, resultados e impacto futuro. A extração é validada semântica e sintaticamente. A qualidade da extração é feita comparando a extração da LLM com anotações feitas por humanos. Na comparação semântica foi observada uma pequena diferença nas anotações e na sumarização da LLM, sendo o Mixtral o modelo mais similar às especificações feitas por humanos.

4. Metodologia

A metodologia proposta neste artigo para investigar o impacto de diferentes *prompts* submetidos aos LLMs na tarefa de especificação de *workflows* a partir de linguagem natural foi estruturada em quatro etapas interdependentes: (i) a definição de um conjunto de *workflows*-alvo, *i.e.*, aqueles que se deseja especificar por meio de linguagem natural, juntamente com seus respectivos "gabaritos", (ii) a elaboração de duas versões de *prompts*, variando em complexidade e grau de informação de suporte; (iii) a aplicação dos LLMs selecionados; e (iv) a avaliação dos resultados obtidos.

Definição dos *Workflows*-Alvo. A primeira etapa da metodologia consiste na definição dos *workflows*-alvo a serem especificados por meio de linguagem natural. Para isso, selecionamos quatro sistemas de *workflow* utilizados em diferentes contextos de aplicação: Nextflow [Di Tommaso et al. 2017], Parsl [Babuji et al. 2019], Dask [Matthew Rocklin 2015] e Apache Airflow. Cada um desses sistemas possui características distintas, refletindo diferentes paradigmas de modelagem e execução de *workflows*. O Nextflow é um sistema de *workflow* com foco em aplicações na área de Bioinformática. A especificação dos *workflows* nesse sistema é realizada por meio de uma linguagem de *script* baseada em *Groovy*, que permite uma descrição declarativa das atividades. O Nextflow oferece suporte à contêineres e à ambientes de computação em nuvem. O Parsl, por sua vez, é um sistema de *workflow* baseado em *scripts* Python, cuja estratégia para a execução paralela de atividades se dá por meio da adição de *decorators* às funções definidas no *script*. Além disso, o Parsl é compatível com diversos gerenciadores de fila, *e.g.*, SLURM, o que o torna apropriado para ambientes de computação de alto desempenho. O Dask é uma biblioteca Python de computação paralela e distribuída e que oferece integração com soluções como o Pandas e o Scikit-learn. Os *workflows* no Dask, assim como no Parsl, são especificados por meio de *scripts* Python, mas ao invés de usar *decorators*, o Dask usa estruturas de dados nativamente paralelas. Por fim, o Airflow é um sistema de *workflow* comumente utilizado em *pipelines* ETL (extração, transformação e carga). O Airflow possui integração com *frameworks* como o Hadoop, Spark, e bancos de dados relacionais, além de oferecer suporte à execução em contêineres e em nuvem.

Para cada sistema de *workflow* anteriormente citado, foram especificados três *workflows* distintos, cada um representando um nível diferente de complexidade. Esses *workflows* foram usados como "gabaritos" para avaliar o resultado da tarefa de especificação automática por meio de linguagem natural, empregando LLMs. É importante ressaltar que decidimos utilizar nessa primeira avaliação *workflows* mais simples e que não fossem específicos de um domínio (*e.g.*, Bioinformática). Ainda que simples, os *workflows*-alvo definidos já permitem avaliar aspectos fundamentais, como o encadeamento das atividades, um fator crítico em *workflows* mais complexos, e a possibilidade de paralelismo. O primeiro *workflow* (Figura 1), classificado como nível 1 ($N1$), recebe uma *string* como entrada, inverte a ordem de seus caracteres e retorna a *string* resultante como saída. Esse *workflow* foi projetado para representar operações simples de transformação de dados, sem necessidade de paralelismo.

Figura 1. Especificação do *Workflow* nível $N1$.

O segundo *workflow* (Figura 2), classificado como nível 2 ($N2$), realiza o processamento paralelo de dados numéricos provenientes de um arquivo de entrada. Esse arquivo contém

um conjunto de valores associados a um atributo numérico. O *workflow* realiza, de forma concorrente, a separação do arquivo em dois *chunks*, e calcula a soma dos números pares em um arquivo e a soma dos números ímpares no outro. Ao final da execução, é gerado um arquivo de saída contendo as duas somas. Esse *workflow* exemplifica a capacidade do sistema de lidar com múltiplas fontes de dados e operações básicas de agregação de forma paralela.

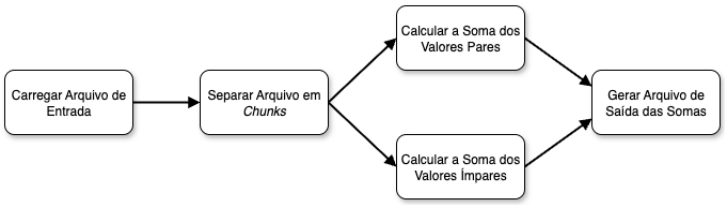

Figura 2. Especificação do *Workflow* nível $N2$.

O terceiro *workflow* (Figura 3), classificado como nível 3 ($N3$), recebe como entrada um *dataset* no formato CSV contendo diversos atributos numéricos. A tarefa consiste em dividir o *dataset* em múltiplos blocos (*chunks*) de tamanho aproximado, processando cada um desses blocos de forma paralela. Para cada *chunk*, o *workflow* aplica uma função de agregação definida em cada atributo numérico presente. O resultado final do *workflow* é o resultado da função de agregação global de cada atributo, obtida a partir dos resultados parciais. Este *workflow* visa avaliar a habilidade dos LLMs em lidar com tarefas mais robustas de processamento distribuído e agregação estatística.

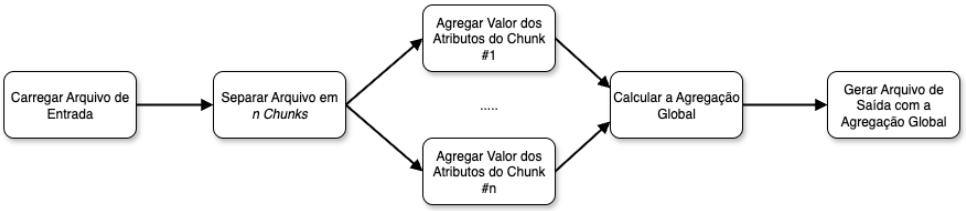

Figura 3. Especificação do *Workflow* nível $N3$.

Elaboração dos *Prompts*. Foram elaboradas duas versões distintas e progressivamente mais informativas de *prompts*. A primeira versão do *prompt* (denominada $P1$) é a mais básica dentre as três e serve como *baseline* (*zero-shot*). Ela se limita a fornecer uma descrição sucinta da tarefa, definindo o objetivo do *workflow* e qual sistema de *workflow* alvo. Nessa versão, o LLM é instruído da seguinte maneira para cada um dos *workflows* ($N1$, $N2$ e $N3$):

Prompt $P1$ - Workflow $N1$

Escreva um código para um *workflow* utilizando a ferramenta **[nome do sistema de *workflow*]** que receba uma sequência de caracteres (*string*) como entrada em um arquivo e forneça um arquivo de saída com o reverso da sequência de entrada.

Prompt $P1$ - Workflow $N2$

Escreva um código para um *workflow* utilizando a ferramenta **[nome do sistema de *workflow*]** que processe um arquivo de entrada com uma série de números e depois separe essa série em dois arquivos distintos. Em seguida, calcule a soma de números pares de um arquivo e ímpares do outro. Por fim, gere um arquivo de saída que contenha os valores agregados.

Prompt $P1$ - Workflow $N3$

Escreva um código para um *workflow* utilizando a ferramenta **[nome do sistema de *workflow*]** que receba como entrada um *dataset* e divida-o em partes menores, onde cada parte será feita uma **[operação de agregação]** dos atributos em paralelo. No fim, faça a **[operação de agregação]** dos resultados de cada arquivo em um mesmo arquivo de saída como um relatório final.

A segunda versão do *prompt* ($P2$) mantêm a estrutura do $P1$, mas introduz exemplos para cada um dos *workflows*, com o intuito de oferecer ao LLM instruções mais concretas sobre o tipo de *workflow* que deve ser gerado (*few-shot*). Nessa versão, o exemplo é fornecido de acordo com o texto "**Exemplo:** o seguinte código acessa um arquivo com um número aleatório de números, faz a divisão em dois arquivos de acordo com uma regra previamente estabelecida, realiza operações e salva os resultados nos mesmos arquivos de entrada. No fim, apenas une as informações em um novo arquivo. **Código:** [exemplo na linguagem do sistema de *workflow*]".

Aplicação dos Modelos. Foram utilizados três LLMs: (i) o Command-A, (ii) o GPT-4o, e (iii) o DeepSeek V3. Devido aos custos de acesso às APIs dos modelos GPT-4o e DeepSeek V3, os *prompts* foram submetidos por meio das interfaces públicas de *chat* das respectivas plataformas. Todos os LLMs foram submetidos às duas versões progressivas de *prompts* ($P1$ e $P2$) para cada sistema de *workflow* e para cada nível de complexidade ($N1$, $N2$ e $N3$), permitindo uma análise comparativa do impacto das variações no desempenho dos *workflows* gerados. Essa abordagem possibilita avaliar como o conteúdo das instruções fornecidas nos *prompts* influencia a sensibilidade e a precisão dos LLMs na especificação automática de *workflows*.

5. Discussão dos Resultados

A avaliação foi conduzida com base na *proximidade* entre o *workflow* gerado e o gabarito de referência elaborado para cada nível de complexidade. Os *workflows* produzidos foram classificados em três categorias: (i) *workflows* que executaram corretamente e cumpriram integralmente a tarefa proposta (✓); (ii) *workflows* que não correspondiam à especificação fornecida em linguagem natural (✗); e (iii) *workflows* que apresentaram erros de execução ou falharam parcialmente na tarefa, mas puderam ser corrigidos com ajustes mínimos no código (*i.e.*, menos de cinco linhas modificadas) (±). A Tabela 1 apresenta os resultados consolidados obtidos por cada um dos LLMs avaliados, nas duas versões progressivas de *prompt* ($P1$ e $P2$), para os três níveis de *workflow* ($N1$, $N2$ e $N3$) e para os diferentes sistemas de *workflow* testados. De maneira geral, os resultados indicam que o *prompt* $P2$, que inclui exemplos, proporcionou melhor desempenho para todos os LLMs, níveis de *workflow* e sistemas avaliados. Esse padrão evidencia que a adição de informações de suporte nos *prompts* contribui de forma significativa para a melhoria da capacidade dos LLMs em gerar *workflows* mais aderentes às especificações.

Podemos também perceber que o LLM que apresentou os melhores resultados foi o GPT-4o, seguido pelo DeepSeek V3 e, por último, pelo Command-A. O GPT-4o apresentou a maior taxa de geração de *workflows* sintaticamente corretos e semanticamente coerentes com as especificações em linguagem natural. Do ponto de vista dos sistemas de *workflow*, o Nextflow foi, consistentemente, o sistema que apresentou maior dificuldade de adaptação por parte dos LLMs, principalmente nos níveis de complexidade $N2$ e $N3$. Essa dificuldade pode ser atribuída à sintaxe da linguagem Groovy (que não é tão comum quanto Python no *corpus* de treinamento dos LLMs) e à necessidade de definições de processos e canais, características que não foram bem interpretadas pelos LLMs. Por outro lado, o Dask foi o que obteve os melhores resultados, com *workflows* bem estruturados e com menor incidência de erros, o que provavelmente se deve à sua especificação em Python (fortemente presente no *corpus*).

Nos testes realizados com *workflows* $N1$ e utilizando o *prompt* $P1$, observou-se que

Tabela 1. Resultados de avaliação para os LLMs com variação dos *prompts*.

Sistema de *Workflow*	LLM	Nível de Dificuldade					
		$N1$		$N2$		$N3$	
		P1	**P2**	**P1**	**P2**	**P1**	**P2**
Nextflow	GPT-4o	✗	✓	✗	±	±	✓
	DeepSeek V3	✓	✓	✗	✓	✗	✗
	Command-A	✗	✓	✗	✗	✗	✗
Airflow	GPT-4o	✓	✓	✗	✓	✓	✓
	DeepSeek V3	±	✓	±	✓	±	±
	Command-A	±	✓	±	✓	✗	✓
Dask	GPT-4o	✓	✓	±	✓	±	✓
	DeepSeek V3	✓	✓	±	✓	✗	±
	Command-A	✓	✓	±	✗	✗	✗
Parsl	GPT-4o	✓	✓	±	✓	±	±
	DeepSeek V3	±	✓	✗	✓	±	✓
	Command-A	±	✓	±	✗	±	✓

os LLMs falharam por pequenos detalhes na especificação, *e.g.*, importações de bibliotecas e declaração de variáveis. No entanto, esses pontos foram corrigidos com alterações inferiores a cinco linhas de código e os *workflows* executaram corretamente. Quando a versão do *prompt* foi a $P2$, todos os LLMs conseguiram gerar *workflows* corretos para esse nível de complexidade, sem a necessidade de ajustes adicionais. Para os *workflows* de complexidade $N2$, ainda utilizando o *prompt* $P1$, foram encontrados desafios. O Nextflow se destacou negativamente por gerar códigos que divergiam estruturalmente da tarefa proposta, indicando um maior nível de imprecisão na compreensão do LLM quanto a linguagem Groovy. Em contraste, o Airflow, o Parsl e o Dask apresentaram problemas localizados, como uso incorreto de módulos ou falhas na etapa de salvamento dos resultados. Com a transição para o *prompt* $P2$, observou-se uma melhora no desempenho, especialmente nos três sistemas citados, que passaram a apresentar *workflows* funcionais. O Nextflow continuou apresentando problemas, sugerindo que o LLM ainda não domina a estrutura declarativa e os paradigmas de fluxo de dados característicos dele.

Por fim, os testes com *workflows* de complexidade $N3$, mesmo com o *prompt* $P2$, apresentaram limitações dos LLMs. O Command-A apresentou maior propensão a produzir especificações incoerentes, enquanto os demais LLMs geravam resultados parcialmente corretos. Os erros mais comuns incluíram importações incorretas, má definição de variáveis e, especialmente no caso do Nextflow, dificuldades na estruturação de dependências entre tarefas. No Dask, identificou-se um desafio relacionado ao encadeamento de resultados entre tarefas, principalmente no uso do *dask.delayed*, o que indica que os LLMs ainda não compreendem plenamente esse padrão de programação assíncrona. O Parsl também apresentou problemas na sincronização das tarefas e na manipulação de dependências, mesmo após a introdução de exemplos. Esses resultados sugerem que, embora *prompts* mais informativos melhorem o desempenho dos LLMs na tarefa de especificação dos *workflows*, ainda existem desafios para a geração automática de *workflows* de maior complexidade.

6. Conclusão

Este artigo avaliou o uso de LLMs na tarefa de especificação automatizada de *workflows* a partir de descrições em linguagem natural. A análise foi conduzida considerando diferentes variáveis: (i) tipos de LLMs, (ii) versões de *prompts* com diferentes níveis de detalhamento, e (iii) sistemas de *workflow* com distintos paradigmas de execução, aplicados em três níveis graduais de complexidade de *workflows*. Os resultados mostraram que a formulação dos *prompts* exerce um

impacto direto sobre a qualidade dos *workflows* gerados. O *prompt* $P2$, que inclui exemplos, apresentou resultados superiores em praticamente todos os cenários avaliados. Dentre os LLMs, o GPT-4o apesesentou maior capacidade de interpretar e gerar códigos em conformidade com as especificações fornecidas, enquanto o Command-A apresentou os piores resultados, especialmente para *workflows* mais complexos. Quanto aos sistemas de *workflow*, o Dask destacou-se pela maior taxa de códigos sintaticamente corretos, o que pode ser atribuído ao uso do Python, mais comum no corpus de treinamento dos LLMs. Em contrapartida, o Nextflow apresentou mais dificuldades, possivelmente devido ao uso da linguagem Groovy. Apesar dos resultados promissores, a geração de *workflows* com estruturas complexas de paralelismo continua sendo um desafio. Além disso, a avaliação revelou que muitos códigos gerados requerem ajustes manuais, ainda que mínimos, para que possam ser executados corretamente.

Referências

Babuji, Y. N. et al. (2019). Parsl: Pervasive parallel programming in python. In *HPDC'19*, pages 25–36. ACM.

Choi, H. K. and Li, Y. (2024). PICLe: Eliciting Diverse Behaviors from Large Language Models with Persona In-Context Learning. In *PMLR'24*, pages 8722–8739.

de Oliveira, D. et al. (2019). *Data-Intensive Workflow Management: For Clouds and Data-Intensive and Scalable Computing Environments*. Morgan & Claypool Publishers.

Di Tommaso, P. et al. (2017). Nextflow enables reproducible computational workflows. *Nature Biotechnology*, 35(4):316–319.

Dong, Q. et al. (2024). A Survey on In-context Learning. In *Proc. of EMNLP'24*, pages 1107–1128, Miami, Florida, USA. ACL.

Duque, A., Syed, A., Day, K. V., Berry, M. J., Katz, D. S., et al. (2023). Leveraging large language models to build and execute computational workflows.

Koziolek, H. et al. (2024). Llm-based and retrieval-augmented control code generation. In *LLM4Code '24*, LLM4Code '24, page 22–29, New York, NY, USA. ACM.

Matthew Rocklin (2015). Dask: Parallel Computation with Blocked algorithms and Task Scheduling. In *Proc. of the 14th Python in Science Conference*, pages 126 – 132.

Paiva, L. et al. (2025). Domínio delimitado, Ódio exposto: O uso de prompts para identificação de discurso de Ódio online com llms. In *SBBD'25*, Fortaleza, Brasil.

Sänger, M. et al. (2024). A qualitative assessment of using chatgpt as large language model for scientific workflow development. *GigaScience*, 13.

Vaswani, A. et al. (2017). Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Wei, J. et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. In *NIPs'22*, Red Hook, NY, USA. Curran Associates Inc.

Xu, J. et al. (2024). Llm4workflow: An llm-based automated workflow model generation tool. In *ASE'24*, ASE '24, page 2394–2398, New York, NY, USA. ACM.

Yildiz, O. and Peterka, T. (2025). Do large language models speak scientific workflows?

Zhang, X. et al. (2024). Massw: A new dataset and benchmark tasks for ai-assisted scientific workflows. *arXiv preprint arXiv:2406.06357*.