

Investigando o Uso de Modelos de Linguagem de Grande Escala para o Escalonamento de Tarefas

Lívia Mayumi Kawasaki Alves¹, Guilherme Piêgas Koslovski¹

¹Universidade do Estado de Santa Catarina (UDESC) – Joinville, SC – Brasil

livia.alves@edu.udesc.br, guilherme.koslovski@udesc.br

Resumo. *Os Large Language Models (LLMs) revolucionaram a área de Processamento de Linguagem Natural (PLN), sendo atualmente possível adaptar tais modelos para realizar atividades complexas em diversas áreas. Especificamente, a técnica de fine-tuning permite que os modelos pré-treinados aprimorem o conhecimento para funções específicas. Em ambientes de alto desempenho, o escalonamento é uma atividade complexa e consiste no processo de alocar recursos do sistema para diversas tarefas, atendendo a objetivos específicos. A escolha da ordem de alocação é fundamental para a eficiência desse ambiente. Nesse sentido, este artigo visa propor uma análise do uso de LLMs ajustados com fine-tuning para auxiliar no escalonamento de tarefas comunicantes.*

1. Introdução

Large Language Models (LLMs) representam um salto significativo em sistemas computacionais capazes de compreender e gerar textos em linguagem humana. Eles são ferramentas poderosas em Processamento de Linguagem Natural (PLN), capazes de realizar tarefas como tradução, resumo e conversação. Avanços na arquitetura dos *transformers*, no poder computacional e nos dados impulsionaram seu sucesso. Esses modelos aproximam-se do desempenho do nível humano, tornando-os valiosos para pesquisas e implementações práticas. O rápido desenvolvimento dos LLMs estimulou inovações nas estratégias e nas técnicas de *fine-tuning* [Parthasarathy et al. 2024]. Especificamente, o *fine-tuning* usa um modelo pré-treinado como base. O processo envolve um novo treinamento em um conjunto de dados menor e específico. Esta abordagem baseia-se no conhecimento pré-existente do modelo, melhorando o desempenho em tarefas específicas com dados e requisitos computacionais reduzidos.

O presente trabalho investiga a aplicação de LLMs como ferramenta auxiliar na realização de uma atividade gerencial comum em ambientes de alto desempenho e que demanda alto custo operacional: o escalonamento de tarefas. O escalonamento de tarefas refere-se a ação de alocar servidores para atender as solicitações dos usuários, considerando vários parâmetros como tempo de execução, memória e interdependências de tarefas, definindo o sequenciamento da execução [Brucker 2006]. Entre os algoritmos de escalonamentos conhecidos estão FCFS (*First Come First Serve*), *Shortest Remaining Time*, *Shortest-Area First*, entre outros, sendo que a escolha impacta diretamente no tempo de espera e no uso dos servidores.

Por isso, é um tema amplamente estudado. A otimização dessa tarefa é computacionalmente desafiadora, pois os métodos tradicionais apresentam limitações em termos de escalabilidade. Motivando, dessa forma, o estudo de outras alternativas utilizando técnicas de inteligência artificial [Abgaryan et al. 2024]. Nesse contexto, o objetivo deste

trabalho é propor a integração de *Fine-tuning* nos modelos pré-treinados para otimizar o processo de escalonamento de tarefas a fim de analisar e comparar os resultados.

2. Unindo Escalonamento e LLMs

Como o estudo foca na resposta gerada pelos modelos para decidir o escalonamento, tais modelos devem seguir a estrutura *Generative Pre-trained Transformer* (GPT) que apresenta três componentes principais: (i) generativo, eles aprendem as relações entre as variáveis de um conjunto de dados para gerar novos dados; (ii) pré-treinado, ou seja, são treinados usando uma grande base de dados, podendo assim economizar tempo e melhorar o desempenho; e (iii) *transformer*, uma arquitetura de rede neural artificial capaz de lidar com dados sequenciais [Yenduri et al. 2024]. Especificamente, a arquitetura do *transformer* apresenta dois componentes principais: o codificador e o decodificador. O primeiro processa a entrada e o segundo gera a saída. Ambos utilizam mecanismos de atenção para capturar relações entre palavras, e de *feedforward*, que permite que o modelo capture e transforme informações complexas de maneira não linear. Entre esses processos, estão as etapas de soma e normalização que somam a saída de uma camada à sua entrada original e aplicam a normalização para estabilizar o treinamento. Antes de executar todos os mecanismos citados, é necessário converter as palavras em representações numéricas, através do que é chamado de *embedding*, e usar a técnica de codificação posicional que fornece informações sobre a ordem das palavras. As últimas etapas de um *transformer* são a transformação linear, que mapeia a saída para o espaço dimensional do vocabulário, e a aplicação da função *softmax*, que transforma os valores em probabilidade [Prince 2023].

A entrada provida aos modelos nesse estudo foi um arquivo de texto que simula a alocação de tarefas em um *Data Center* (DC) composto por servidores, tarefas agendadas e tarefas na fila, incluindo as informações dos recursos de cada um deles. O arquivo fornece detalhes representando um recorte temporal de métricas e demais informações gerenciais, que podem ser obtidas de rastros de execução publicamente disponíveis. Assim, é possível reproduzir as conclusões do presente estudo.

Inicialmente, os modelos assimilam os padrões do comportamento das tarefas submetidas e escalonadas em servidores do DC, e determinam uma política de escalonamento, gerando uma saída da ordem de execução. A partir da saída, são recortadas as informações pertinentes, ou seja, quais tarefas foram escalonadas e como foram escalonadas (o resumo dos recursos após a decisão). Esses resultados são então passados para um simulador que gerencia a alocação das tarefas em servidores ao longo do tempo, registrando as métricas de desempenho, como o tempo de execução, a utilização dos recursos do DC, e o número de tarefas na fila e em execução.

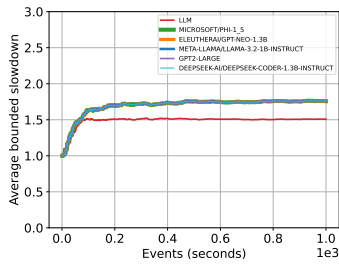
3. Desenvolvimento e resultados preliminares

Inicialmente, realizou-se o treinamento de um modelo GPT próprio usando um fragmento da entrada dada aos modelos já treinados para alimentar o modelo. O código realiza todos os processos desde processamento de dados até a geração de texto. Além desse modelo, os testes foram executados com outros já consolidados e publicamente disponibilizados.

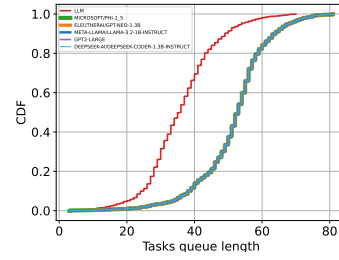
Através do *Hugging Face*, plataforma que fornece ferramentas de inteligência artificial como modelos e *datasets* para treino, foi possível realizar o teste para

os modelos descritos: *gpt2-large*, *EleutherAI/gpt-neo-1.3B*, *meta-llama/Llama-3.2-1B-Instruct*, *microsoft/phi-1.5* e *deepseek-ai/deepseek-coder-1.3b-instruct*.

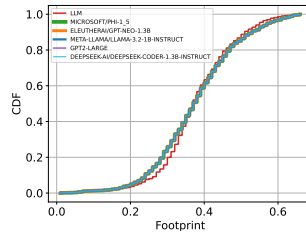
O teste consistiu em executar cada modelo com os mesmos contextos, gerando saídas em arquivos *json* utilizados posteriormente pelo simulador que gerencia o escalonamento. Após a simulação foram obtidos os resultados ilustrados pelos gráficos da Figura 1, onde as linhas vermelhas (LLM) representam o modelo treinado pelos autores e as outras cinco, os modelos do *Hugging Face* citados anteriormente.



(a) bsld time consolidation



(b) tasks queue consolidation



(c) footprint consolidation

Figure 1. Resultados do experimento

A Figura 1(a) apresenta no eixo y o valor médio do *bounded slowdown* (*bsld*) pelo tempo em segundos no eixo x. O *bsld* é medido por $bsld_t = \max \left\{ \frac{w_t + p_t}{\max(p_t, \tau)}, 1 \right\}$, sendo t tarefa, w_t tempo de espera, p_t tempo de processamento solicitado e τ uma constante definida em 10 segundos para evitar que tarefas muito pequenas acarretem em valores excessivamente altos [Feitelson and Rudolph 1998]. Notou-se que o LLM se saiu ligeiramente melhor com *bsld* menor comparado aos outros modelos. A Figura 1(b) mostra no eixo x a quantidade de tarefas na fila e no eixo y o *Cumulative Distribution Function* (CDF) que representa a porcentagem de valores que estão inseridos nos intervalos específicos de x. Nessa métrica, o LLM se mostrou melhor com menores comprimentos de fila, e com os valores de CDF crescendo mais rapidamente. Já na Figura 1(c) é possível entender a utilização de recursos dos servidores pelas tarefas, com o eixo x representando o *footprint* e o eixo Y, o CDF. Nesse caso, os seis modelos apresentaram comportamento parecido.

Em geral, o modelo treinado apresentou resultados relativamente melhores que os outros modelos pré-treinados. Porém, é conhecido que estes foram treinados em dados de uso geral como livros e sites, portanto, eles podem não ter sido expostos, durante o pré-treinamento, a dados semelhantes ao contexto apresentado, e por isso ter tido dificuldades em gerar resultados significativos e diferentes entre si.

Em vista disso, propõe-se a implementação da etapa de *fine-tuning* no estudo. Esse passo é importante para adaptar o conhecimento dos modelos para tarefas específicas, melhorando o desempenho deles de forma mais rápida e com recursos computacionais reduzidos [Parthasarathy et al. 2024]. A implementação dessa técnica mostra melhoria no desempenho dos LLMs efetivamente nos problemas de escalonamento, podendo até superar as abordagens tradicionais [Abgaryan et al. 2024]. Para a futura execução, faz-se necessário discutir e definir as configurações e os critérios ideais do processo considerando o cenário do estudo. Aplicando essa técnica tanto no modelo treinado quanto nos pré-treinados, será possível analisar o desempenho e a capacidade real de cada um, comparando as vantagens e desvantagens em cada caso.

4. Conclusão

Este trabalho contribuiu para a exploração da aplicação de LLMs no escalonamento de tarefas comunicantes. Analisou-se que o treinamento de um modelo próprio para aplicações na área de escalonamento pode superar, em alguns casos, as limitações de outros modelos pré-treinados. Ademais, propôs-se a integração da técnica de *fine-tuning* para adaptar os modelos para contextos específicos e melhorar seu desempenho. Os resultados parciais indicam que os LLMs apresentam potencial para isso. Os trabalhos futuros envolvem a implementação da solução proposta, para o estudo dos resultados com o intuito de realizar uma análise comparativa. Espera-se, dessa forma, resultados que possam contribuir na busca por soluções eficientes para escalar tarefas em ambientes de alto desempenho.

Agradecimentos: Este trabalho recebeu apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Fundação de Amparo à pesquisa e Inovação (FAPESC), desenvolvido no Laboratório de Processamento Paralelo e Distribuído (LabP2D).

Referências

- Abgaryan, H., Harutyunyan, A., and Cazenave, T. (2024). Llms can schedule. *arXiv preprint arXiv:2408.06993*.
- Brucker, P. (2006). *Scheduling Algorithms*. Springer.
- Feitelson, D. G. and Rudolph, L. (1998). Metrics and benchmarking for parallel job scheduling. In Feitelson, D. G. and Rudolph, L., editors, *Job Scheduling Strategies for Parallel Processing*, pages 1–24, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Parthasarathy, V. B., Zafar, A., Khan, A., and Shahid, A. (2024). The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities.
- Prince, S. J. (2023). *Understanding Deep Learning*. MIT Press.
- Yenduri, G., Ramalingam, M., Selvi, G. C., Supriya, Y., Srivastava, G., Maddikunta, P. K. R., Raj, G. D., Jhaveri, R. H., Prabadevi, B., Wang, W., Vasilakos, A. V., and Gadekallu, T. R. (2024). Gpt (generative pre-trained transformer)— a comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions. *IEEE Access*, 12:54608–54649.