Artigo Haskell - Síntese e Destaques

Caio Faria

Giuseppe Cordeiro

Vinícius Miranda

SITES PARA PESQUISAR ARTIGOS:

- IEEE Xplore
- SciELO - Brasil
- CAPES - Portal
- Biblioteca Digital de Teses e Dissertações da USP
- Google Acadêmico
- arXiv

ARTIGOS (LINKs):

Helping Developers Write Energy Efficient Haskell through a Data-Structure Evaluation | IEEE Conference Publication | IEEE Xplore

Haskell before Haskell: an alternative lesson in practical logics of the ENIAC | OUP Journals & Magazine | IEEE Xplore

Investigating the Performance of Language Models for Completing Code in Functional Programming Languages: A Haskell Case Study | IEEE Conference Publication | IEEE Xplore

Verifying replicated data types with typeclass refinements in Liquid Haskell (capes.gov.br)

ARTIGOS (PDFs):

ARTIGOS

ARTIGO SELECIONADO:

en: Investigating the Performance of Language Models for Completing Code in Functional Programming Languages: A Haskell Case Study

pt: Investigando o Desempenho de Modelos de Linguagem para Completar Código em Linguagens de Programação Funcional: Um Estudo de Caso Haskell

---

SLIDES:

https://www.canva.com/design/DAGScqplv8A/Xu8mz_Lspp8qm9bOwrbAbw/edit?utm_content=DAGScqplv8A&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

---

ROTEIRO/SÍNTESE:

Introdução // vini

A introdução do artigo estabelece o contexto da pesquisa, enfatizando a crescente utilização de modelos de linguagem para auxiliar na programação. Com o avanço dos Large Language Models (LLMs), muitos desenvolvedores têm se beneficiado de ferramentas como o GitHub Copilot, que demonstram eficácia em prever a intenção dos programadores ao fornecer sugestões contextuais. Apesar do sucesso dessas ferramentas em linguagens imperativas, como Python e JavaScript, as linguagens funcionais, especialmente Haskell, têm sido menos exploradas, resultando em um desempenho inferior dos modelos ao completar código nessas linguagens. Este estudo tem como objetivo investigar como os modelos CodeGPT e UniXcoder se comportam em tarefas de completude de código em

Haskell, buscando identificar as limitações atuais e as oportunidades de melhoria que podem ser implementadas para otimizar o desempenho desses modelos em contextos funcionais.

Trabalhos Relacionados // caio

Na seção de trabalhos relacionados, o artigo revisa a literatura existente sobre a aplicação de modelos de linguagem na programação. Ele discute como a maioria dos modelos de completude de código foi desenvolvida com foco em linguagens imperativas, resultando em uma escassez de representações adequadas para linguagens funcionais. O artigo menciona pesquisas anteriores que abordaram a eficácia desses modelos em diferentes contextos, como a utilização de datasets variados e a análise de métricas de desempenho. No entanto, ressalta-se a lacuna significativa na pesquisa voltada para a aplicação de modelos em linguagens funcionais, como Haskell, o que justifica a necessidade de um estudo mais aprofundado nesse campo, proporcionando uma base para futuras investigações.

Arquitetura de Transformer // gius

O artigo fornece uma visão abrangente da arquitetura de Transformer, que revolucionou o campo do processamento de linguagem natural. A arquitetura é composta por dois componentes principais: o codificador e o decodificador, que trabalham em conjunto para processar sequências de dados. O artigo elogia a capacidade do Transformer de lidar com dependências de longo alcance em dados sequenciais, algo crucial para a compreensão da estrutura do código. Embora os modelos de codificador-decodificador tenham sido amplamente utilizados em tarefas de tradução, o artigo destaca que a arquitetura Transformer se mostrou eficaz em várias outras aplicações, incluindo a geração de código e a completude de código. A flexibilidade e a eficiência dessa arquitetura fazem dela uma escolha ideal para o desenvolvimento de modelos voltados para linguagens de programação.

Metodologia // vini

A metodologia do estudo é apresentada de forma detalhada, começando pela seleção dos modelos de linguagem, CodeGPT e UniXcoder. O artigo descreve o processo de ajuste dos modelos utilizando um conjunto de dados específico de Haskell, que foi cuidadosamente preparado para garantir a qualidade das amostras. Isso inclui a filtragem de dados irrelevantes e a seleção de implementações que refletem a complexidade e a diversidade do código Haskell. A avaliação dos modelos foi realizada por meio de tarefas de completude de código, onde as saídas geradas pelos modelos foram analisadas em

comparação com implementações corretas. O artigo também discute a importância de métricas de avaliação adequadas, como Exact Match (EM) e Edit Similarity (ES), para medir o desempenho dos modelos, levando em consideração tanto a precisão das previsões quanto a sua relevância em um contexto prático.

Resultados // caio

Os resultados do estudo revelam que os modelos ajustados para Haskell superaram significativamente os modelos base, que não foram especificamente treinados para essa linguagem. A avaliação do desempenho dos modelos em termos de precisão e relevância das previsões indicou que o conhecimento adquirido em linguagens imperativas não se transfere bem para Haskell. Além disso, a análise manual das saídas geradas pelos modelos revelou que o CodeGPT tende a produzir um número maior de previsões vazias e comentários desnecessários, enquanto o UniXcoder apresentou problemas mais fundamentais, como erros de sintaxe e previsões incompletas. Essa análise qualitativa fornece insights valiosos sobre as fraquezas dos modelos e indica áreas específicas para melhorias futuras.

Discussão // gius

Na discussão, o artigo analisa as implicações dos resultados, sugerindo que a falta de dados de treinamento de alta qualidade para Haskell representa uma barreira significativa para o desempenho dos modelos. O artigo propõe que a comunidade de pesquisa deve considerar a implementação de linguagens funcionais ao treinar modelos de linguagem, especialmente à medida que muitas linguagens de programação orientadas a objetos estão incorporando conceitos funcionais. Além disso, enfatiza a necessidade de mais pesquisas para entender como as características das linguagens funcionais, como a imutabilidade e a ausência de efeitos colaterais, podem ser melhor integradas nos modelos de linguagem, visando otimizar suas capacidades de completude de código.

Conclusão //os 3

A conclusão resume as principais descobertas do estudo, enfatizando que os modelos de linguagem precisam ser ajustados especificamente para linguagens funcionais para melhorar seu desempenho. O artigo sugere que a criação de novos conjuntos de dados de Haskell, que sejam ricos em diversidade e qualidade, e a exploração de diferentes abordagens de treinamento, são essenciais para avançar a pesquisa nesta área. Além disso, destaca a importância de uma compreensão mais profunda dos conceitos fundamentais da programação funcional, que não apenas beneficiarão Haskell, mas também poderão ser aplicados em outras linguagens que adotam paradigmas funcionais. O

estudo conclui que, ao abordar essas lacunas, é possível promover avanços significativos na capacidade dos modelos de linguagem de completar código de forma eficaz em ambientes funcionais.

DESTAQUES:

- **Caio:**
  O artigo foi muito bem fundamentado e relevante, especialmente pela conexão entre desempenho de software e sustentabilidade energética. Ele oferece orientações práticas para desenvolvedores que desejam otimizar o consumo de energia em seus programas Haskell, além de reforçar a importância de medições detalhadas, como o impacto da DRAM, para atingir um software mais eficiente.

- **Gius:**

  O artigo "Investigando o desempenho de modelos de linguagem para completar código em linguagens de programação funcional: um estudo de caso Haskell" explora como esses modelos podem ser eficazes na conclusão de código em Haskell, uma linguagem funcional pura. A pesquisa destaca que, enquanto os modelos têm se mostrado eficientes em linguagens imperativas como Python e Java, Haskell apresenta desafios únicos devido à sua ausência de estruturas tradicionais de controle de fluxo, como loops "for" e "while".

  Os resultados indicam que, com ajustes específicos, os modelos de linguagem podem alcançar um desempenho competitivo em Haskell, abrindo novas possibilidades para o uso de IA em ferramentas de desenvolvimento, como busca e reparo de código. O estudo também identifica dificuldades, como a necessidade de uma melhor compreensão da semântica funcional, e sugere que a integração de IA no desenvolvimento de linguagens funcionais pode trazer avanços significativos, um campo ainda pouco explorado em comparação com as linguagens imperativas.

- **Vini:** O artigo, de modo geral, mas especialmente ao apontar que os modelos de conclusão de código têm grande dificuldade em suas tarefas, destaca que isso se deve à falta de treinamento adequado para linguagens funcionais.

REFERÊNCIA

VAN DAM, T.; VAN DER HEIJDEN, F.; DE BEKKER, P.; NIEUWSCHEPEN, B.; OTTEN, M.; IZADI, M. **Investigating the Performance of Language Models for Completing Code in Functional Programming Languages: A Haskell Case Study**. In: 2024 IEEE/ACM First International Conference on AI Foundation Models and Software Engineering (Forge). Lisbon, Portugal: IEEE, 2024. p. 91-102. DOI: 10.1145/3650105.3652289.