PUC Minas

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

Tim Van Dam; Frank Van der Heijden; Philippe De Bekker; Berend Nieuwschepen; Marc Otten; Maliheh Izadi

Caio Diniz

Giuseppe Cordeiro

Vinícius Miranda

PUC Minas

Dados do Artigo

1 Introdução

- **Título:** Investigating the Performance of Language Models for Completing Code in Functional Programming Languages: A Haskell Case Study.

- **Autores:** Tim van Dam, Frank van der Heijden, Philippe de Bekker, Berend Nieuwschepen, Marc Otten, Maliheh Izadi

- **Ano:** 2024

- **Publicação:** First International Conference on AI Foundation Models and Software Engineering (Forge) Conference Acronym:

PUC Minas

Introdução

2 Contexto e Objetivo

- Importância crescente dos modelos de linguagem na conclusão de código.
- Muitos estudos focam em linguagens imperativas (ex: Python, JavaScript), resultando em desempenho inferior em linguagens funcionais como Haskell.
- O objetivo do estudo é investigar o comportamento dos modelos CodeGPT e UniXcoder em tarefas de completude de código em Haskell, identificando limitações e oportunidades de melhoria.

PUC Minas

Trabalhos Relacionados

3 Revisão na Literatura

- A maioria dos modelos de completude de código foi desenvolvida com foco em linguagens imperativas, levando a uma falta de representação para linguagens funcionais.

- Pesquisas anteriores analisaram a eficácia de modelos de linguagem em diferentes contextos, mas há uma lacuna significativa na pesquisa sobre linguagens funcionais.

- Limitações na aplicação de modelos treinados em linguagens imperativas a linguagens funcionais.

PUC Minas

Arquitetura de Transformer

3.1 Visão Geral da Arquitetura

- Revolucionou o processamento de linguagem natural, destacando os componentes codificador e decodificador.
- Diferenças entre modelos de codificador, decodificador e codificador-decodificador.
- Importância da arquitetura para a compreensão semântica e sintática do código.

PUC Minas

Metodologia

4 Abordagem Metodológica

- Seleção de modelos de linguagem: CodeGPT e UniXcoder, ajustando-os com um conjunto de dados específico de Haskell cuidadosamente preparado.

- Avaliação dos modelos foi realizada por meio de tarefas de completude de código, com análises das saídas geradas em comparação com implementações corretas.

- Métricas de avaliação como Exact Match (EM) e Edit Similarity (ES) para medir o desempenho, considerando precisão e relevância das previsões.

PUC Minas

Perguntas de Pesquisa

4 Questões e Análises

- Qual é o desempenho dos modelos de conclusão de código em Haskell?

R: Quase toda a literatura se concentra em linguagens imperativas e não funcionais, então não está claro se Haskell (e linguagens funcionais em geral) são mais difíceis para modelos de conclusão de código.

- Quais são as armadilhas mais comuns para conclusão de código em Haskell?

R: O artigo aponta que não foram encontradas armadilhas específicas comuns para conclusão de código em Haskell durante as avaliações dos modelos CodeGPT e UniXcoder. Ambos os modelos demonstraram um desempenho geral insatisfatório em todas as categorias analisadas, sugerindo que, em vez de problemas específicos, há uma falta de suporte adequado para a linguagem como um todo.

PUC Minas

Resultados

5 Descobertas do Estudo

- Modelos ajustados em Haskell superaram os modelos base.
- O desempenho foi avaliado em termos de precisão e relevância, mostrando que o conhecimento em linguagens imperativas não se transfere bem para Haskell.
- A análise manual revelou que o CodeGPT tende a produzir previsões vazias e comentários desnecessários, enquanto o UniXcoder apresenta erros de sintaxe e previsões incompletas.

PUC Minas

Discussão

6 Implicações dos Resultados

- Propõe que a comunidade de pesquisa considere a implementação de linguagens funcionais no treinamento de modelos de linguagem.
- A falta de dados de treinamento de alta qualidade para Haskell é uma barreira significativa para o desempenho dos modelos.
- Sugere que mais pesquisas são necessárias para entender como características das linguagens funcionais podem ser integradas nos modelos de linguagem.

PUC Minas

Conclusão

7 Resumo e Direções Futuras

- Os modelos de linguagem precisam ser ajustados especificamente para linguagens funcionais para melhorar seu desempenho.
- A criação de novos conjuntos de dados de Haskell e a exploração de diferentes abordagens de treinamento são essenciais para o avanço da pesquisa.
- Uma compreensão mais profunda dos conceitos fundamentais da programação funcional beneficiará não apenas Haskell, mas outras linguagens que adotam paradigmas funcionais.

PUC Minas

Destaques

8 Destaque de cada Integrante

- **Caio**

A exploração da relação entre desempenho de software e sustentabilidade energética, com foco na otimização de programas Haskell. Destacando a importância de medições detalhadas, como o impacto da DRAM, para melhorar a eficiência.

- **Giuseppe**

A investigação o desempenho de modelos de linguagem na conclusão de código em Haskell, destacando desafios específicos dessa linguagem funcional pura. Os resultados mostram que, com ajustes, esses modelos podem ser eficazes, abrindo novas possibilidades para o uso de IA no desenvolvimento em Haskell.

- **Vinícius**

O artigo como um todo, mas principalmente saber o fato que os modelos de conclusão de código têm grande dificuldade na realização de sua tarefas devido ao não treinamento dos modelos para as linguagens funcionais.

PUC Minas

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.

PUC Minas

MUITO OBRIGADO!

Caio Diniz

Giuseppe Cordeiro

Vinícius Miranda