Construção de um Modelo de Desenvolvimento/Geração de Código com IA

Gustavo Orlando Costa dos Santos Henriques - 64361
Estudo Orientado
Mestrado em Engenharia Informática
Faculdade de Ciências, Universidade de Lisboa
fc64361@fc.ul.pt

Abstract

A Inteligência Artificial está a tornar-se cada vez mais comum no dia a dia da sociedade, atuando como um recursco para simplificar e automatizar certas tarefas, como o desenvolvimento de software. Com isto surgiram os Large Languages Models, que têm a capacidade de gerar vários tipos de output a partir de inputs variados. Contudo, estes modelos ainda enfrentem desafios como a coerência e fiabilidade. A presente tese tem como objetivo desenvolver um modelo capaz de gerar código a partir de descrições textuais e visuais, tendo como caso de uso a criação de uma wiki interna na Trust Systems, permitindo aos trabalhadores gerir as suas tarefas. Este trabalho irá avaliar como estratégias de prompt engineering serão capazes de influenciar a qualidade do código gerado, contribuindo para uma melhor compreensão da automatização do desenvolvimento de software com LLMs.

Keywords Inteligência Artificial, Large Language Models, Prompt Engineering, Geração Automática de Código, Engenharia de Software

1 Introdução

Graças aos avanços da inteligência artificial nos últimos anos, a forma como é desenvolvido o software tem evoluído significativamente. Uma grande revolução que se deu em IA por volta de 2018, quando começaram a surgir os primeiros Large Language Models, como o BERT [2] e o GPT-1[6], que se baseavam na arquitetura dos transformers, conceito este que foi introduzido em 2017[7]. Desde então estes modelos têm sofrido alterações no sentido de se tentar otimizar a sua performance, sendo hoje em dia capazes de gerar e completar código executável[3].

Com estas contínuas melhorias hoje existem ferramentas como o GitHub Copilot ou Amazon CodeWhisperer que conseguem auxiliar o trabalho de um programador, causando um impacto significativo na sua produtividade[5].

Apesar destes modelos e ferramentas, assistidos por IA, trazerem consigo beneficios para o ambiente empresarial, é importante referir que têm as suas limitações no que toca a segurança, qualidade, robustez e confiabilidade. Estudos feitos à robustez do GitHub Copilot indicam que, pequenas alterações de input podem originar alterações significativas no código gerado em, aproximadamente, 46% dos casos[4]. Além disso, Nam Huynh and Beiyu Lin[3], concluiram que 40% do código gerado pelo Copilot continha vulnerabilidades na segurança. Com isto é possível concluir que mesmo com toda a automatização existente, continua a ser essencial uma análise humana que seja rigorosa e, que é vital o modo como construimos o input que será recebido pelos modelos. Com isto, surgiram técnicas mais recentes, designadas de prompt engineering, que estudam como o design dos prompts pode influenciar o desempenho dos LLMs e diminuir o impacto das limitações referidas acima.

Como foram apresentados progressos substanciais no campo da IA durante as últimas décadas, as expectativas do que realmente se pode atingir aumetaram. Apesar disso, há ainda muitos domínios que permanecem em estágio inicial, com incertezas quanto ao seu real potencial, como o prompt engineering e o desenvolvimento autônomo de software. Além disso, os algoritmos existentes, para avaliação de modelos, ainda não são suficientes para tratar de um modo sistemático e comparável as suas capacidades e os seus riscos potenciais [1]. Portanto, considerando o impacto que estas técnicas podem causar na redução de custos e na otimização de recursos, conclui-se que a pesquisa dos limites dos LLMs pode trazer consequências significativas para a sociedade atual.

Neste trabalho especificamente, a finalidade é estudar o estado da arte dos Large Language Models e das técnicas de prompt engineering no domínio da geração automática de código. O principal intento é analisar como estas abordagens podem ser implementadas de modo a produzir código, com a melhor qualidade disponível. Qualidade essa que será avaliada através de métricas automáticas e manuais, tentando assim identificar o potencial atual da IA para o desenvolvimento de software.

Outline. How is the rest of the document structured?
The remainder of this document is organised as follows. Section 2 presents bla bla bla. ...

2 Background

In this section, you should describe the scientific or technological context of your project. Provide enough information so that a reader unfamiliar with the topic can understand the problem you are addressing and the rationale for your

project. This may include relevant concepts and definitions in the area, particularly those that will be used throughout this document.

3 Related Work

This section should present the state of the art on the topic of your project. It should discuss relevant related work and existing solutions, highlighting their main contributions as well as their limitations, and identifying the gaps or opportunities that motivate your project.

Preparing this section will require you to include references to academic papers, books, and possibly online resources. The next paragraph exemplifies how to do it.

In this work, you are expected to follow the guidelines on document preparation presented in Lamport's book on LaTeX [**?**]. For editing, you may use tools such as the online platform Overleaf [**?**]. There is also a good chance that your project will build upon some of Lamport's many scientific contributions, such as the concept of logical clocks [**?**].

4 «Other Section(s) as Appropriate»

The report should include one or more sections providing a detailed description of the problem you are addressing in the project and your plan to tackle it. Use appropriate section titles for what is presented.

You should explain the methods you are planning to use, or have already started to apply, in your project. This discussion should be grounded in the related work, your own understanding of the problem, and, when available, preliminary results.

In case you already have some preliminary results, consider to include a section devoted to them. This section should describe the work already carried out, what data has already been collected, what analysis and designs have already been done, what methods have been used, what programs and/or preliminary results already exist, etc.

5 Forthcoming Work and Conclusions

This section should include subsections describing the work to be carried out during the remainder of the school year and its objectives. It should also present a chronological plan for the completion of the project. Finally, include a concluding subsection that summarizes the contributions already made, provides a preliminary self-assessment of the progress achieved so far, and discusses the main difficulties encountered.

References

[1] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2024. A Survey on Evaluation of Large Language Models. *ACM Transactions on Intelligent Systems and Technology* 15 (3 2024). Issue 3. doi:10.1145/3641289

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova Google, and A I Language. [n. d.]. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Technical Report. 4171–4186 pages. https://github.com/tensorflow/tensor2tensor

[3] Nam Huynh and Beiyu Lin. 2025. Large Language Models for Code Generation: A Comprehensive Survey of Challenges, Techniques, Evaluation, and Applications. (4 2025). http://arxiv.org/abs/2503.01245

[4] Antonio Mastropaolo, Luca Pascarella, Emanuela Guglielmi, Matteo Ciniselli, Simone Scalabrino, Rocco Oliveto, and Gabriele Bavota. 2023. On the Robustness of Code Generation Techniques: An Empirical Study on GitHub Copilot. (2 2023). http://arxiv.org/abs/2302.00438

[5] Suresh Babu Nettur, Shanthi Karpurapu, Unnati Nettur, Likhit Sagar Gajja, Sravanthy Myneni, and Akhil Dusi. [n. d.]. *The Role of GitHub Copilot on Software Development: A Perspec-tive on Productivity, Security, Best Practices and Future Directions.* Technical Report.

[6] Alec Radford Openai, Karthik Narasimhan Openai, Tim Salimans Openai, and Ilya Sutskever Openai. [n. d.]. *Improving Language Understanding by Generative Pre-Training*. Technical Report. https://gluebenchmark.com/leaderboard

[7] Ashish Vaswani, Google Brain, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. [n. d.]. *Attention Is All You Need.* Technical Report.