



The Assistant Project

BY MATEUSZ JADASZ

What is it?

By Mateusz Jadasz

The What

By Mateusz Jadasz

An overlay tool that assists, suggests and corrects code

How will it work?

By Mateusz Jadasz

The How

Idea:

- ▶ Capture text from the screen/image using Optical Character Recognition tools like Tesseract
- ▶ Identify if the text is code using libraries like Pygments which can detect programming languages based on syntax or guesslang which is an AI-based language detection library
- ▶ If the detected text is labelled as code, we feed it into Code Llama to see if there are more optimal ways of writing it

Why will this be beneficial?

The Why

- ▶ There is a need for tools that can intelligently optimize existing code, not just complete it
- ▶ Convenience
- ▶ Offline usage
- ▶ Accessibility
- ▶ Tries to address the issue of Collaboration over Dependency [9] [10] [11]

Related Works

Related works

The study by Rozière et al. [1] introduces Code Llama, a suite of open-source LLMs optimized for code generation, trained on public datasets with fine-tuning for infilling and instruction-following. On the other hand, Arkhangelskaya et al. [2] provide a survey of deep learning in NLP, reviewing key neural architectures and their impact on tasks like translation and sentiment analysis. Additionally, Touvron et al. [3] present Llama, a family of efficient foundation models trained exclusively on public data, demonstrating competitive performance against larger proprietary models. Laskar et al. [4] critically examine LLM evaluation challenges, identifying inconsistencies in methodologies and proposing improvements for reproducibility and reliability. Naveed et al. [5] offer a comprehensive overview of LLM advancements, discussing architectures, training strategies, and emerging trends such as multimodal models. Furthermore, Zacharias et al. [6] focus on scene-text detection and recognition using Tesseract, applying machine learning for text identification in images. Verma et al. [7] explore programming language identification with machine learning models, enhancing software development tools with automated language detection. Izadi et al. [8] evaluate code completion models in practical environments, identifying challenges and assessing real-world performance using large-scale data.

Related works

The eight papers offer a diverse look at the current landscape of language models and machine learning applications. Studies by Rozière et al. [1], Touvron et al. [3], and Izadi et al. [8] focus on specific applications, including code generation and completion, using advanced LLMs trained on open data. Arkhangel'skaya et al. [2] and Naveed et al. [5] provide theoretical insights into the evolution of NLP and LLMs, while Laskar et al. [4] emphasize the importance of rigorous evaluation techniques to ensure the reliability of these models in practice. Zacharias et al. [6] and Verma et al. [7] explore the practical use of machine learning in tasks like text recognition and programming language identification, addressing real-world challenges. Overall, while the papers vary in scope—from theoretical overviews to practical applications—they all underscore the significance of improving model performance, enhancing evaluation standards, and applying AI techniques to real-world problems across different domains.

Aims and Objectives

By Mateusz Jadasz

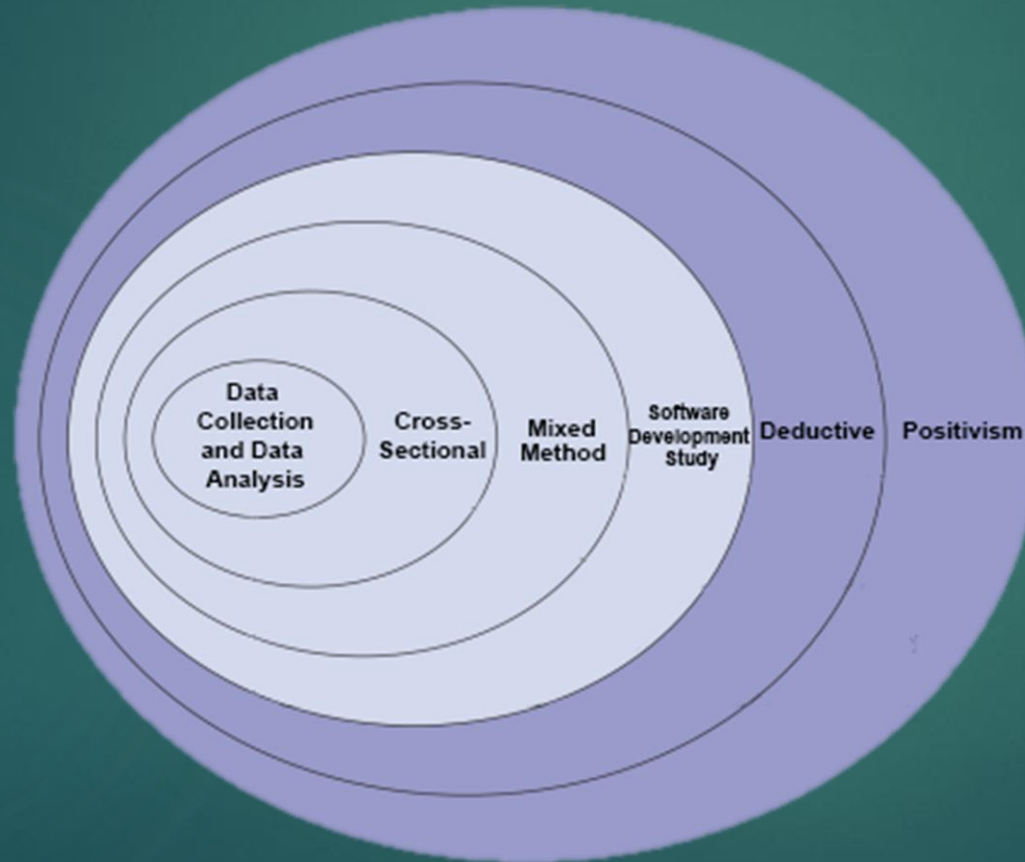
Aims and Objectives

The Aim: To develop the basis for an AI tool which promotes collaboration on programming projects over dependency

- ▶ Objective 1: Create a way for the tool to capture the text on screen
- ▶ Objective 2: Determine if captured text is code
- ▶ Objective 3: Generate more optimal code suggestions

Methodology

Methodology



Methodology

Data Collection and Data Analysis

- ▶ The models used in the program will need minimal training/tuning, if any. This is because they're already trained to do what is required of them
- ▶ The training/testing data I intend on using is code from my own past projects in order to test the code optimisation and explanation functions
- ▶ In order to determine the success of the program, I will test metrics such as execution time, memory usage, and readability of optimized vs. original code

Methodology

Cross-sectional

- ▶ This study does not require a long-term observation or anything of the sort as all the data can be created/gathered within a short timeframe and from multiple sources

Methodology

Mixed Method

- ▶ I believe this fits the project best as although the program primarily works off qualitative data such text and images, I will be measuring quantitative and qualitative data in the form of previously mentioned performance metrics such as execution time, memory usage, and readability.

Methodology

Software Development Study

- ▶ The measure if the aim has been achieved is based entirely on the software that will be created

Methodology

Deductive

- ▶ This study is based on an already existing theory. The theory being that programmers, especially newer ones, are growing too dependant on generative AI tools to do the work for them, rather than assisting them with their own code

Methodology

Positivism

- ▶ The theory behind the study is based on an observable and growing problem covered in several studies and by multiple people within the industry

Project Planning and Resources

By Mateusz Jadasz

Project planning and resources

By Mateusz Jacek

Plan

Project Area	Estimated Timescale
Report	2 – 2.5 Months
Planning	1-2 Weeks
Program	1 – 1.5 Months
Testing	1 Week

- The report timescale covers the time from when the introduction is written to when the report is finished, which means the expected time overlaps other areas
- The planning stage will include the creation of resources such as user stories and kanban boards
- In between the report writing, the program itself is expected to take up the bulk of the time
- The testing stage should take the least time as testing will not only be done throughout the development but also generally it is expected to be the least time-consuming aspect of the project

Project planning and resources

By Mateusz Jadasz

Resources

- ▶ Python
- ▶ VS Code
- ▶ Tesseract
- ▶ Pygments/Guesslang
- ▶ Code Llama
- ▶ Benchmarking Frameworks
- ▶ Microsoft Office

Project Impact and Limitations

By Mateusz Jadasz

Project impact and limitations

Impact

- ▶ Automates performance optimization, reducing manual effort
- ▶ Works only off existing code, offering only assistance in optimising instead of writing software for the user
- ▶ Creates a collaborative environment, even for solo programmers offering a more flexible and convenient way to learn

Limitations

- ▶ May not always produce optimal solutions; requires benchmarking for validation
- ▶ Potentially may be misused and therefore may not serve its purpose of being a source of assistance as opposed to being the creator of the source code
- ▶ It will need further development which would be outside of the time-scope in order to make the software as user-friendly as it should be

References

Reference

- [1] B. Rozière et al., 'Code Llama: Open Foundation Models for Code', Meta AI, Aug. 2023.
- [2] E. O. Arkhangelskaya and S. I. Nikolenko, 'Deep Learning for Natural Language Processing: A Survey', J. Math. Sci., vol. 273, no. 4, pp. 533–582, Jul. 2023, doi: 10.1007/s10958-023-06519-6.
- [3] H. Touvron et al., 'Papers with Code - LLaMA: Open and Efficient Foundation Language Models', Meta AI, Feb. 2023, Accessed: Feb. 22, 2025. [Online]. Available: <https://paperswithcode.com/paper/llama-open-and-efficient-foundation-language-1>
- [4] M. T. R. Laskar et al., 'A Systematic Survey and Critical Review on Evaluating Large Language Models: Challenges, Limitations, and Recommendations', in Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds., Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 13785–13816. doi: 10.18653/v1/2024.emnlp-main.764.
- [5] H. Naveed et al., 'A Comprehensive Overview of Large Language Models', Oct. 17, 2024, arXiv: arXiv:2307.06435. doi: 10.48550/arXiv.2307.06435.
- [6] E. Zacharias, M. Teuchler, and B. Bernier, 'Image Processing Based Scene-Text Detection and Recognition with Tesseract', Apr. 17, 2020, arXiv: arXiv:2004.08079. doi: 10.48550/arXiv.2004.08079.
- [7] A. Verma, R. Saha, G. Kumar, A. Brighente, M. Conti, and T.-H. Kim, 'Exploring the Landscape of Programming Language Identification With Machine Learning Approaches', IEEE Access, vol. 13, pp. 23556–23579, Feb. 2025, doi: 10.1109/ACCESS.2025.3538108.
- [8] M. Izadi, J. Katzy, T. van Dam, M. Otten, R. M. Popescu, and A. van Deursen, 'Language Models for Code Completion: A Practical Evaluation', Feb. 25, 2024, arXiv: arXiv:2402.16197. doi: 10.48550/arXiv.2402.16197.
- [9] S. Peng, E. Kalliamvakou, P. Cihon, and M. Demirer, 'The Impact of AI on Developer Productivity: Evidence from GitHub Copilot', Feb. 13, 2023, arXiv: arXiv:2302.06590. doi: 10.48550/arXiv.2302.06590.
- [10] J. Prather et al., 'The Widening Gap: The Benefits and Harms of Generative AI for Novice Programmers', May 28, 2024, arXiv: arXiv:2405.17739. doi: 10.48550/arXiv.2405.17739.
- [11] R. Ulfesnes, N. B. Moe, V. Stray, and M. Skarpen, 'Transforming Software Development with Generative AI: Empirical Insights on Collaboration and Workflow', Feb. 12, 2024, arXiv: arXiv:2405.01543. doi: 10.48550/arXiv.2405.01543.