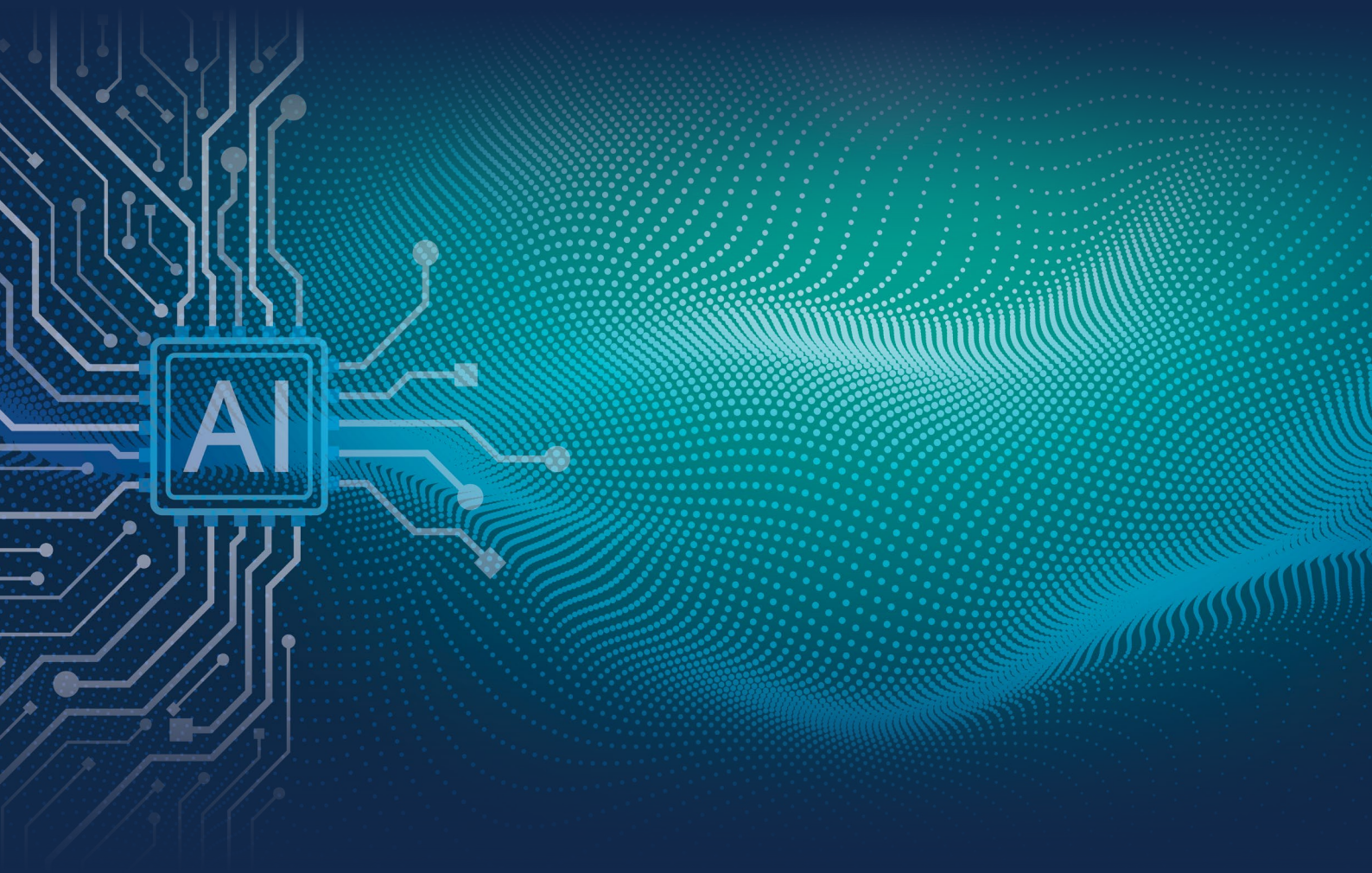
AdaCore CASE STUDY

Copilot for Ada programming language to boost developer productivity



How can we boost Ada software developers' productivity? We teamed up with deepsense.ai to create a proof-of-concept copilot solution for the Ada programming language.

adacore.com

Project Overview

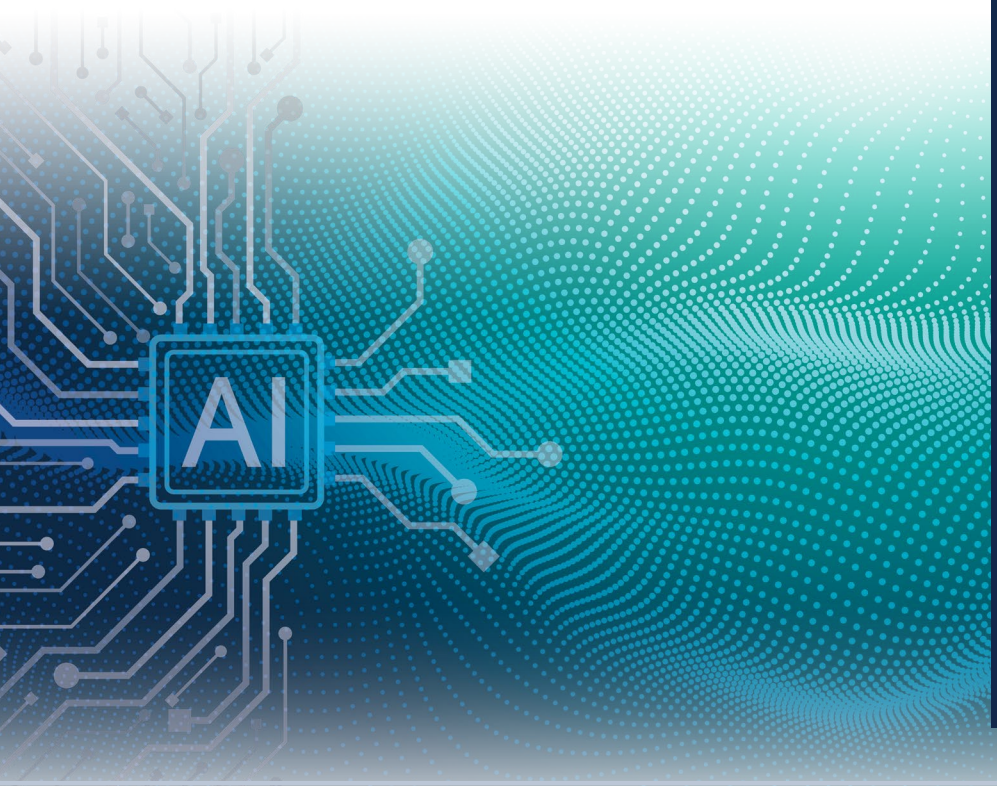
The Copilot for Ada programming language project aimed to research and develop a proof-of-concept code completion tool and evaluate its performance on the Ada code generation task. Its idea is to boost Ada software developers' productivity by providing intelligent code completions and suggestions, improving the pace of task automation, and saving significant amounts of time on repetitive and boilerplate code.

Key objectives

The solution aimed to prepare the groundwork for significantly boosting the effectiveness of Ada developers in the future by developing an intelligent code completion and suggestions solution.

Key outcomes

- Demo application allowing to test code completion LLM-based solution on any Ada language samples,
- Fine-tuned checkpoints of the selected coding LLMs, such as StarCoder and CodeGen, and their comparison with the baseline pre-trained models,
- Recommendations on further steps for fine-tuning and enhancing Ada-specific code completion models.



deepsense.ai

Partner:
deepsense.ai specializes in providing cutting-edge AI solutions tailored to enterprises and mid-sized companies. With a focus on driving forward-thinking approaches, they collaborate with global leaders to optimize their products and services while delivering additional value to their partners. Since 2014 they have successfully delivered over 200 AI projects, engaging with organizations such as Zebra Technologies, Sky, Volkswagen, Brainly, Hexagon, Getinge, Johnson & Johnson, Hitachi, AdaCore, LogicMonitor, Juniper, Whirlpool, L'Oréal, Nielsen, Nvidia, Intel, Google and many more.

Challenge:
How can we boost the productivity of Ada software developers? We teamed up with deepsense.ai to create a proof-of-concept copilot solution for the Ada programming language.

Solution:
deepsense.ai quickly delivered a Proof of Concept for a code completion tool using a state-of-the-art technological stack, including the newest available LLMs and libraries.

Results and Benefits:
The co-pilot for the Ada programming language project aimed to enhance Ada developer productivity with an intelligent code completion tool. deepsense.ai overcame challenges in LLM evaluation, training, and data resource constraints through problem decomposition and choosing the suitable evaluation scheme. The project marked significant progress for AdaCore in leveraging ML efforts to improve Ada developers' productivity.

Challenges and solutions

This project encountered several challenges while developing the copilot solution for the Ada programming language.

Challenges faced

LLMs Evaluation

Evaluating LLMs is difficult because no performance metrics can be automatically calculated and nicely correlated with the model's performance in the program synthesis task without manually creating a set of programming challenges with unit tests. To circumvent this, they used text comparison metrics like BLEU and chrF measures to compare the ground truth code to the model's generation.

Training Objective

Standard autoregressive training would not suffice because information is needed from before and after the cursor to complete the code completion task correctly. As a result, deepsense.ai needed to understand how LLMs are trained to perform fill-in-the-middle tasks (FIM).

Resources

Despite the availability of memory-efficient training methods, memory requirements remain challenging when fine-tuning large models, particularly those with extended context. They had to rely on small batch sizes, which meant slow iteration speed (one training run could take several days to fine-tune on the largest dataset for more than one epoch).

The fast pace of the developments in the field of coding LLMs

The new versions of CodeGen and StarCoder models were released during the project. Because our training pipeline was designed with modularity in mind, deepsense.ai could seamlessly incorporate these models.

Development process

The project's goal in Data Science terms was to fine-tune the LLM, already pre-trained on code, for Ada code synthesis. The overall strategy included three crucial components as follows:

1. Training dataset preparation

deepsense.ai took an existing and cleaned corpus of GitHub repositories with permissive licenses called The Stack. After keeping only files with the correct Ada code and some additional preprocessing, they transformed it into a form that allows the models to be trained on the FIM task.

2. Evaluation

Due to the lack of better evaluation methods, they utilized the chrF metric, which measures the similarity of predicted text and the ground truth, and were fully aware of some shortcomings. deepsense.ai evaluated the checkpoints on the held-out Ada corpus and the dataset of short Ada programming challenges found on the AdaCore website.

3. Fine-tuning Runs

They ran and compared the performance of pre-trained and fine-tuned StarCoder and CodeGen models in different configurations, manipulating the number of model parameters, precision, context lengths, and memory-efficient fine-tuning methods, like LoRA and QLoRA.

Data

Three data sources were utilized throughout the project:

The Stack: A collection of code repositories for 358 programming languages available under permissive licenses. Only file extensions associated with Ada scripts were retained (.ads, .adb, .ada), their correctness was validated with the libadalang module, and files lacking Ada keywords in their contents were filtered out. In the end, 30,528 files remained, representing 2.4% of the original dataset.

Ada Course Labs: This consists of short Ada exercises with descriptions. deepsnese.ai utilized this dataset as a supplementary test set.

Ada code from GitHub, the code that was not in The Stack dataset, was used to improve the model's performance with a more extensive training corpus and generate a test set of Ada files unseen by pre-trained models during their training phase.

Outcomes and benefits

The fine-tuning improved the model's performance on the Ada code synthesis tasks compared to the pre-trained version. The project is a significant step forward. The generations from the delivered fine-tuned model outperformed the ground truth and GitHub Copilot even though Copilot uses a much larger model with a more extended context and a more complex prompting method.

> **"deepsense.ai quickly delivered a Proof of Concept for a code completion tool, using a state-of-the-art technological stack, including the newest available LLMs and libraries. They also led an excellent LLM discovery workshop that jump-started AdaCore's integration of LLM solutions into our business processes and products. Their technical knowledge and commitment to delivering tailored, top-notch services were evident throughout our collaboration. Partnering with deepsense.ai has helped us accelerate our understanding of AI, implement AI solutions, and gain a strategic edge in today's competitive landscape."**
>
> **- M. Anthony Aiello,**
> **Head of Product & Innovation at AdaCore**

Lessons learned

- The field of coding **LLMs** is constantly evolving, with more and more capable models being released at a fast pace. Even the foundation models pre-trained on publicly available code repositories have a decent level of understanding of how to program in the **Ada language**. The model's capabilities can be improved by fine-tuning an additional **Ada-specific corpus of code repositories**.
- For coding models, context length is more important than the model's size - even a 1B model with a context of 8k tokens can outperform a 15.5B model with only 2k tokens context.
- While useful as a proxy measure for the model's output quality, existing metrics and benchmarks for coding LMs are unreliable for differentiating models. The end-user subjective evaluation is still required to determine the model's usefulness.

Summary

The co-pilot for the Ada programming language project aimed to enhance Ada developer productivity with an intelligent code completion tool. deepsense.ai overcame challenges in LLM evaluation, training, and data resource constraints through problem decomposition and choosing the suitable evaluation scheme. The project marked significant progress for AdaCore in leveraging ML efforts to improve Ada developers' productivity.

AdaCore

adacore.com