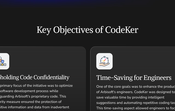
CodeKer by Arbisoft

Arbisoft created CodeKer as an AI powered alternative to platforms like ChatGPT, Github Copilot, and ChatGPT Plus. Designed to optimize the software development lifecycle, CodeKer identifies coding issues, offers real-time support, and streamlines solutions. It's a flexible tool easily integrated into various development environments.



Key Objectives of CodeKer

Upholding Code Confidentiality

The primary focus of this initiative was to optimize the software development process while safeguarding Arbisoft's proprietary code. This security measure ensured the protection of sensitive information, mitigating the risk of external data breaches and collaboration problem solving.

Time-Saving for Engineers

One of the core goals was to contribute to the productivity of Arbisoft's engineers. CodeKer was designed to save valuable time by providing intelligent suggestions and automating repetitive coding tasks. This allows engineers to focus on more complex and creative aspects of their work.

Versatility in Project Work

CodeKer aimed to empower engineers to work on different projects seamlessly. By automating repetitive coding tasks and offering accurate suggestions, the tool was designed to be adaptable to a wide variety of projects, fostering versatility and adaptability.

Resource Optimization

Utilizing cloud infrastructure was a crucial aspect of the initiative. CodeKer's assistance streamlined the development process, allowing engineers to allocate their time and resources more efficiently. This optimization facilitated a more efficient use of resources across various projects.

Maintaining Code Security

A paramount objective of CodeKer was to provide an in-house solution that eliminated the risks associated with external applications like Github Copilot, ChatGPT, and Bard. By ensuring the maintenance of code security, protecting against unintentional data exposure and potential security vulnerabilities.

The Rationale

In software development, integrating AI-based tools has become instrumental in driving progress. Recognizing the potential of tools like ChatGPT, GitHub CoPilot, Perplexity, BARD, and more, we support their use to increase developer productivity. This commitment, however, is tempered by our adherence to security measures and ethical considerations.

Security Measures

Our engineers care deeply about doing things right. They carefully choose what goes into our code assistance tools, always putting privacy and data security first. This dedication extends to the tools we use, ensuring we strive to balance valuable investments with the imperative of safeguarding sensitive information.

Enhanced Productivity

Our proprietary CodeKer plugin showcases our commitment to efficiency and enhanced productivity. Acting as an AI coding assistant, it understands coding patterns and offers context-based suggestions. This not only speeds up coding, improves code quality through accurate suggestions, and reduces errors by acting as an AI pair programmer.

Addressing Ethical Challenges

Acknowledging the ethical challenges posed by AI developers, we remain vigilant. Some AI tools lack ethical considerations in their training data, potentially introducing bias and the open-source domain without adhering to licensing or attribution requirements. We've developed an in-house tool for navigating industry challenges while upholding ethical standards. This gives us the freedom to harness AI power responsibly.

Testing & Implementation

We leveraged a highly secure server-based architecture to implement CodeKer's vision. At the core of this innovation lies OpenAI's open-source LLM Whisper, a robust model for handling crew-oriented tasks and queries.

During our initial testing phase, we utilized StarChat due to its well-documented excellence within the industry, as demonstrated by the available literature.

Integration with Visual Studio Code

To enhance CodeKer's capabilities, we integrated the tool with Visual Studio Code, a widely used development environment. This collaboration empowered developers to select specific sections of code, access comprehensive descriptions, and receive suggestions for improvements. Moreover, CodeKer also played the role of a code reviewer, adept at identifying errors and providing suggestions for improvements. It's essential to note that while CodeKer operates in the open-source domain, it remains firmly within the secure bounds of Arbisoft's infrastructure.

Diverse Hardware Testing

Before we started with CodeKer, we needed to explore options for hosting LLM, which is the backbone of our solution. The initial phase involved the implementation of various models including Mistral, GPU's and Apple Silicon. This exploration encompassed different GPU models like the A6000, A100 PCIe, A100 SXM4, and H1 Ultra (CPU cores). Notably, we successfully deployed Quantized LLMs on Apple Silicon.

In our hardware and infrastructure selection process, we aimed to strike a balance between cost-effectiveness and performance. There's a tradeoff between how good it is and how good it performs.

Measures	CodeKer results against python with 16 bit	CodeKer results against python with 32 bit
Correct	42	48
Minor Tweaks	7	2
Incorrect	6	5
Accuracy (Correct+Minor Tweaks / Total)	0.84	0.92

Our research uncovered that Code LLM 32-bit outperforms 16-bit precision by 8%. However, the 16-bit model showed a slight drop in performance. But, it's significantly more cost-effective. Additionally, we compared the rental costs of GPUs, including popular models like the RTX A6000 and A100. Our analysis shows that hosting a 32-bit Model involves significantly higher expenses compared to a 16-bit model, with costs approximately 2.5 times higher.

Comprehensive Data Set Utilization

We conducted thorough testing using a combination of datasets, including the Human Eval dataset from the Open AI to evaluate code generation. We also designed our dataset consisting of prompts, encompassing two different languages: Python and JavaScript, with 50 prompts for each language.

These comprehensive experiments benchmarked CodeKer's performance against Llama 2, BARD, GitHub Copilot, and ChatGPT. Our diverse range of prompts in various languages enabled rigorous testing against a variety of state-of-the-art technologies, further strengthening CodeKer's position as a groundbreaking solution.

python | javascript

Tools	Correct	Minor Tweaks	Incorrect	Correct	Minor Tweaks	Incorrect
CodeKer	42	3	5	41	6	3
LLAMA 2 70B Chat	33	7	10	37	7	15
Bard	40	1	9	48	4	2
Chatgpt	42	7	1	51	0	4
Github Copilot	35	3	12	43	5	7

This Google Sheets link will lead you to the detailed breakdown of results.

Prompt: def second_largest(arr): → int:\n ''' Calculates the second largest number from the list of integers to find the second largest.\n Parameters:\n arr (List[int]): The list of integers to find the second largest.\n Returns:\n The second largest number in the list, or None if the list is empty.\n '''

Tools	Evaluation	Responses

CodeKer	✓	def second_largest(arr): → int:
		''' Calculates the second largest number from the list.
		Arguments:
		arr: The list of integers to find the second largest.
		Returns:
		The second largest number in the list.
		Raises:
		ValueError: If the list has less than two unique elements.
		largest = float('-inf')
		second_largest = float('-inf')
		for num in arr:
		if num > largest:
		second_largest = largest
		largest = num
		elif num > second_largest and num != largest:
		second_largest = num
		return second_largest

LLAMA 2 70B Chat	✓	def second_largest(arr): → int:
		''' Calculates the second largest number from the list.
		Arguments:
		arr: The list of integers to find the second largest.
		Returns:
		The second largest number in the list.
		Raises:
		ValueError: If the list has less than two elements.
		Calculates the second largest number from the list.
		arr: List[int]
		Return:
		int
		'''
		if len(arr) <= 1:
		raise ValueError
		return heapq.nlargest(2, arr)[1]

Github Copilot	✓	def second_largest(arr): → int:
		''' Calculates the second largest number from the list.
		Arg:
		arr: The list of integers to find the second largest.
		Return:
		The second largest number in the list.
		Raise:
		ValueError: If the list has less than two elements.
		largest, second_largest = float('-inf'), float('-inf')
		for num in arr:
		if num > largest:
		second_largest, largest = largest, num
		elif largest > num > second_largest:
		second_largest = num
		if second_largest == float('-inf'):
		raise ValueError("There aren't enough elements in the list are the same.")
		return second_largest

Chatgpt	✓	def second_largest(arr: List[int]) -> int:
		''' Calculates the second largest number from the list.
		arr: The list of integers to find the second largest.
		Returns:
		The second largest number in the list, or None if the list is empty.
		'''
		if len(arr) < 2:
		return None
		largest = float('-inf')
		second_largest = float('-inf')
		for num in arr:
		if num > largest:
		second_largest = largest
		largest = num
		elif num > second_largest and num != largest:
		second_largest = num
		return second_largest

Bard	✗	def second_largest(arr: List[int]) -> int:
		''' Calculates the second largest number from the list.
		arr: The list of integers to find the second largest.
		Returns:
		The second largest number in the list, or None if the list is empty.
		'''
		if len(arr) < 2:
		return None
		largest = float('-inf')
		second_largest = None
		for num in arr:
		if num > largest:
		second_largest = largest
		largest = num
		elif num != largest and (second_largest is None or num > second_largest):
		second_largest = num
		return second_largest
		print("Enter the second largest number:")
		second_largest = int(input("Enter the second largest number: ", largest))
		if second_largest == largest:
		print("The second largest number cannot be the same as the largest number.")
		else:
		print("The second largest number:", second_largest)

PS: The datasets utilized in these tests are sourced from the August 2023 model, acknowledging that LLMS models undergo periodic updates.

Benefits of CodeKer

Faster Coding

CodeKer provides intelligent suggestions, completing statements and suggesting entire functions as you type, saving valuable time, especially when working with repetitive or boilerplate code.

Reduced Coding Errors

Acting as an AI pair programmer, CodeKer minimizes syntax errors and typos, enhancing code accuracy.

Improved Code Quality

The plugin's AI algorithms ensure high-quality suggestions, maintaining coding standards and avoiding common mistakes.

Enhanced Productivity

By automating repetitive coding tasks and offering accurate suggestions, CodeKer allows developers to focus on the higher-level aspects of their projects. This streamlining effect boosts overall productivity.

Simplified User Experience with Logical Insights

CodeKer is a transformative tool that combines design with its intuitive interface, which easily fits into developers' workflows. This aligns with Arbisoft's mandatory criteria for developers of all levels.

Notably, CodeKer goes beyond mere code recommendations. It provides transparent, logical explanations, empowering developers to understand the "why" behind the suggestions. This dedication to clarity helps with code comprehension, skill development, and overall code quality improvement. CodeKer is a tool that prioritizes both ease of use and learning, establishing a new benchmark in the software development world.

The Fruits of Our Labor

The introduction of CodeKer marks the beginning of a new era of efficiency and precision in Arbisoft's software development. This innovative tool has significantly streamlined the development process, fine-tuned our development workflows, allowing developers to complete tasks faster and more precisely. This improved efficiency has resulted in significant time savings for our members, allowing them to concentrate on the creative and strategic sides of their work.

Moreover, CodeKer has fostered a culture of collaboration and versatility within our organization. Its seamless integration with Visual Studio Code has empowered developers to work across different projects and environments easily. By facilitating communication and knowledge sharing, CodeKer has strengthened teamwork and encouraged exchanging ideas, leading to greater innovation and creativity in our development processes.

Beyond its practical benefits, CodeKer has fueled innovation and creativity at Arbisoft. Its ability to generate insightful suggestions and provide logical insights has inspired our developers to explore new ideas and approaches to problem-solving. By offering transparent explanations and logical insights, CodeKer has contributed a culture of continuous learning and improvement, driving innovation across our organization.

Looking ahead, the possibilities with CodeKer are limitless. Its easy integration, increased efficiency, and dedication to quality have paved the way for our continued success. As we continue on our path of innovation and growth, we are confident that CodeKer will continue to be an invaluable partner, allowing us to push the frontiers of software development.

Call us
+1 (469) 315 2958

Email us
sales@arbisoft.com

Visit us
2020 Central Ex Sprlo
8321 Ashtrow, TX 75063