MLS 1: Generative AI Landscape

Welcome! We will begin shortly

Mentored Learning Session

Learning Outcomes

- Learn how GenAI is powering waves of new innovations across industries and solving pain points for businesses.
- Explore some ways in which GenAI is revolutionizing businesses through case studies.
- Get a sense of architectures used in industry for Gen-AI solutions.
- Acknowledge the importance of prompt engineering

Test your Knowledge 1

Which statement best describes the difference between Discriminative AI and Generative AI?

A. Discriminative AI learns the boundary between different classes, while Generative AI generates new data.

B. Discriminative AI generates new data, while Generative AI learns the boundary between different classes.

C. Discriminative AI and Generative AI both learn the boundary between different classes.

D. Discriminative AI and Generative AI both generate new data

Test your Knowledge 2

What is the primary difference between Supervised Learning and Unsupervised Learning?

A. Unsupervised Learning requires labeled data for training, while Supervised Learning does not.

B. Supervised Learning requires labeled data for training, while Unsupervised Learning does not.

C. Supervised Learning is used for regression tasks, while Unsupervised Learning is used for classification tasks.

D. Unsupervised Learning is used for regression tasks, while Supervised Learning is used for classification tasks.

Test your Knowledge 3

How do Large Language Models (LLMs) handle uncertainty when predicting the next word in a sequence?

A. By randomly selecting the next word from the vocabulary.

B. By relying on deterministic rules to choose the most probable next word.

C. By using heuristics to estimate the likelihood of each word based on its position in the sequence.

D. By assigning probabilities to each word in the vocabulary and sampling from the distribution.

Test your Knowledge 4

Which of the following factors can influence the quality of next-word predictions made by Large Language Models (LLMs)?

A. The geographical location of the training data

B. The dataset and the size of the model architecture

C. The color scheme used in the user interface

D. The programming language in which the model is implemented

Test your Knowledge 5

What is one of the primary reasons why Large Language Models (LLMs) may exhibit hallucinations?

A. Inadequate training data leading to overfitting.

B. Lack of diversity in the training data

C. Overly complex model architectures.

D. Excessive regularization techniques applied during training.

Gen AI - Revolutionizing Industries

Marketing and Sales

- Ad copy generation
- Auto-generating posts/articles for multiple social media channels

Software Industry

- Code Generation & Completion
- Code Quality Checks
- Debugging

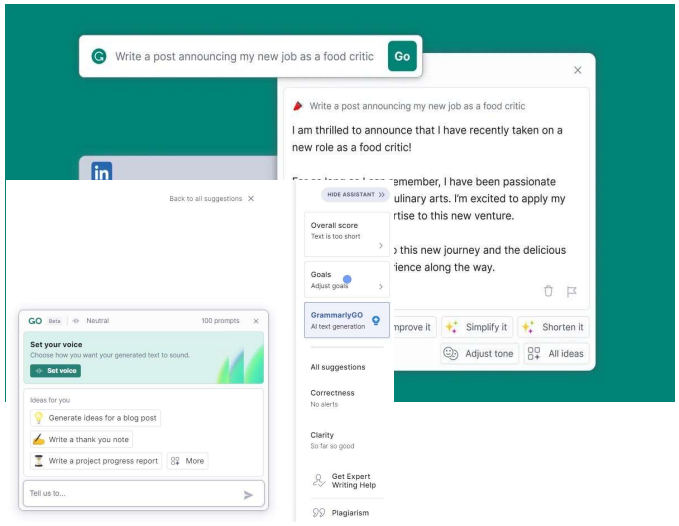
Education

- AI Tutors
- Topic explanations
- Doubt Clearing
- Auto-grading

GenAI is Disrupting across Scale. MNCs to Startups

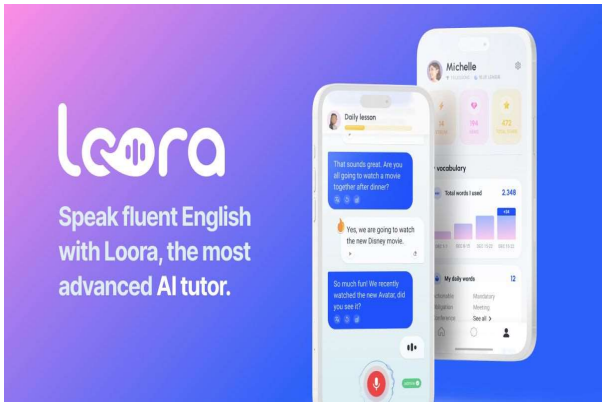
Case-Studies

Auto Text Generation Grammarly.



GitHub CoPilot

Tutor to Improve English Fluency Loora AI



Grammarly

Grammarly is an accomplished international corporation Ukraine-founded cloud-based typing assistant, headquartered in San Francisco that aids individuals in their writing endeavors by:

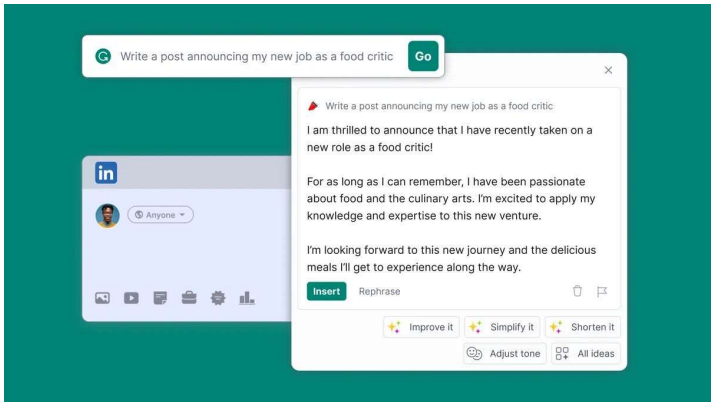
1. Reviewing spelling/grammar
2. Helping with punctuation
3. Reframing the text/content
4. Suggesting replacements for the identified errors.

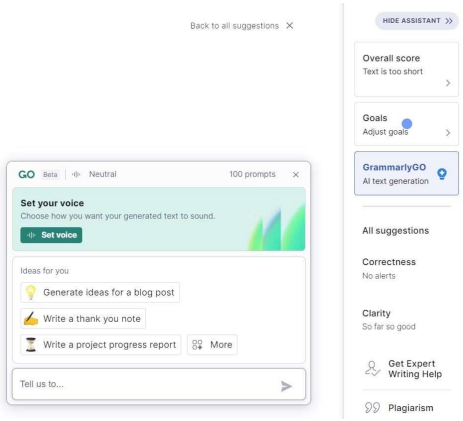
Threat to Grammarly

- The advent of tools like ChatGPT posed a threat to Grammarly.
- Being experts of text generation, apart from correcting user's grammar and tone, they could also enhance user's text and even generate it from scratch.
- Failure to adapt could have led to a loss of user base to these platforms.

Auto Text Generation - Grammarly

Grammarly started offering an unique service for its premium users where they can quickly generate content using Gen AI.





How Grammarly Benefitted

- By prioritizing innovation and staying abreast of emerging technologies, Grammarly maintained its relevance and competitiveness in the market.
- This strategic move simplified workflow and enhanced the overall user experience.
- Altogether Grammarly successfully countered the ChatGPT threat and stayed relevant for its user base

Auto Text Generation - Grammarly

Tools like ChatGPT posed a threat to Grammarly. Being experts of text generation, apart from correcting user's text they could also enhance user's text and even generate it from scratch.

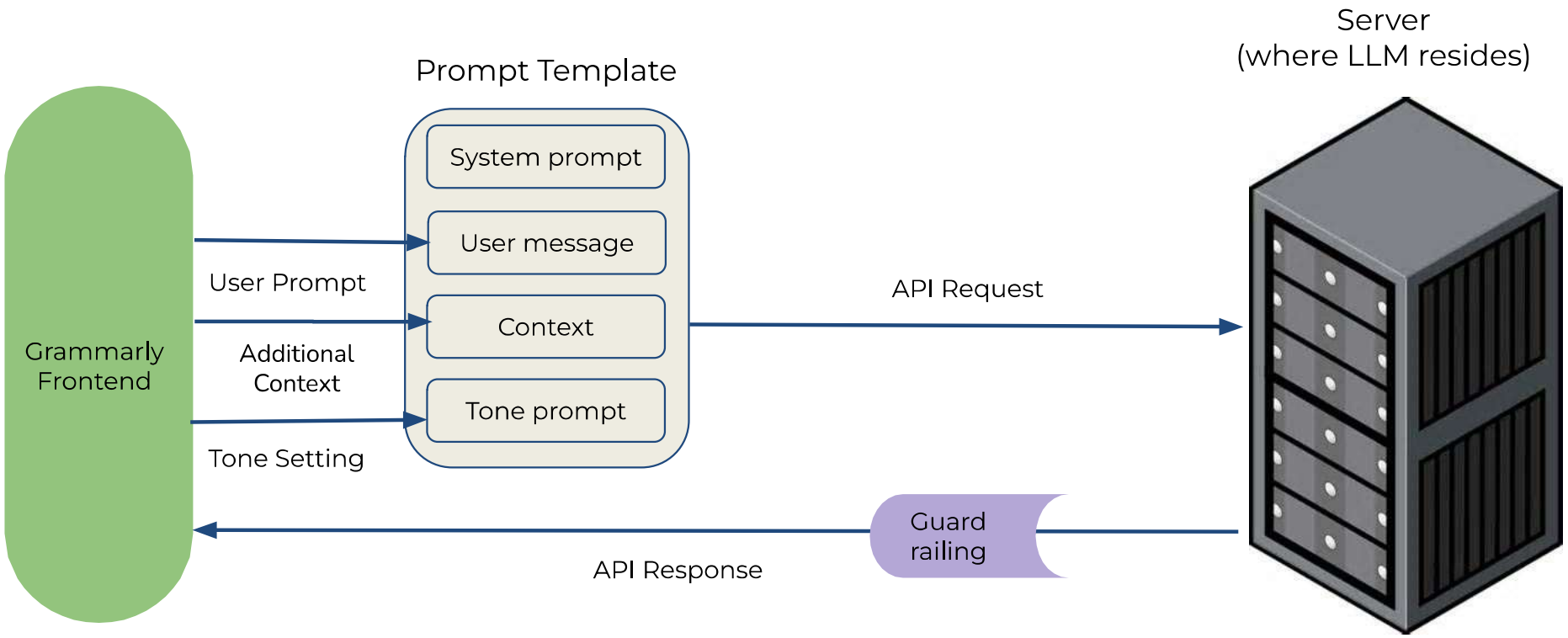
Grammarly has integrated Gen AI to significantly enhance its functionality.

Failure to adapt could have led to a loss of user base to these platforms.

The introduction of text generation capabilities streamlines content creation for users. Users can now generate text directly within their preferred text editor or web browser.

Auto Text Generation for Grammarly.

Architecture of GrammarlyGo



grammarly

Great Learning
POWER AHEAD

Grammarly Architecture

- User inputs a prompt (what the user wants to accomplish) on Grammarly front-end.
- Users also have an option to provide tone and extra context like the text in the document.
- These inputs are then sent to a back-end where all of them along with a system prompt are concatenated into a single concrete prompt that is then sent to the LLM hosted on a server through an API.
- The response from the server is then sent through a guardrail to check for any violations.
- Post that, the response is sent to the front end where user can integrate the response into their documents.

grammarly

Great Learning
POWER AHEAD

Importance of Prompt Engineering

- The heart of this system is the system prompt which makes sure the LLM generates the required response.
- The system prompt is carefully crafted to extract the necessary information in the required format.
- Later in the course, we will learn the craft of creating this part of the prompt. We will see how different techniques suit different applications.
- Also, we will learn to do prompt engineering at scale using python and no-code tools which will enable you to create real-world solutions like these ones where one has to process a lot of requests automatically.

grammarly

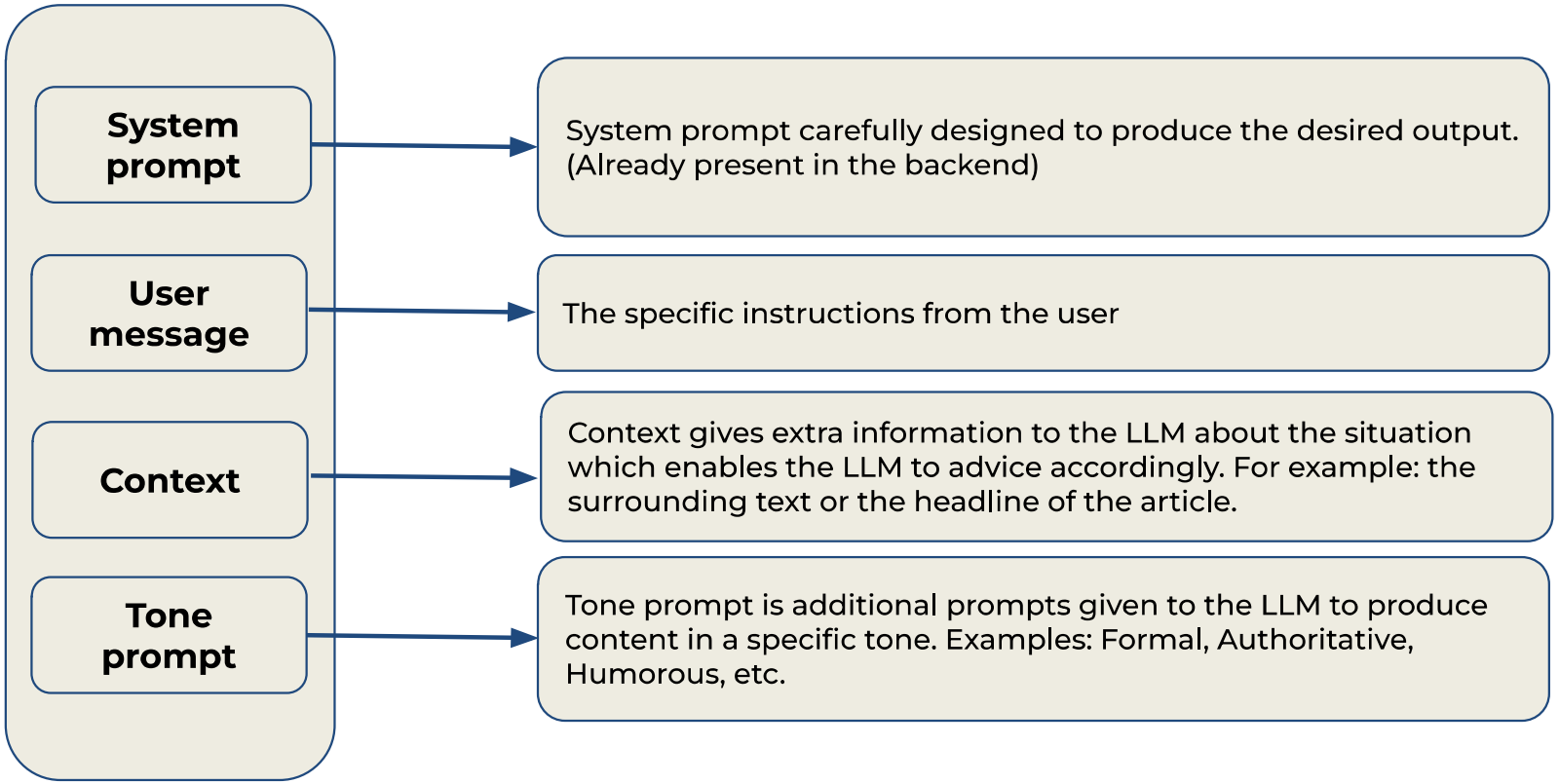
Great Learning
POWER AHEAD

Importance of Prompt Engineering

"In very simple terms, the LLM is, at its core, just a document completion model. For training, it was given partial documents and it learned how to complete them one token at a time. Therefore, the art of prompt crafting is all about creating a 'pseudo-document' that will lead the model to completion that benefits the customer," - John Berryman, a senior researcher"

An LLM has been trained on vast amounts of text data and can generate responses to a wide range of questions and prompts.
However, it needs a clear and specific prompt to retrieve the desired response. A well-crafted prompt is like a treasure map that guides the LLM to the precise location where the desired information is buried.

Auto Text Generation for Grammarly

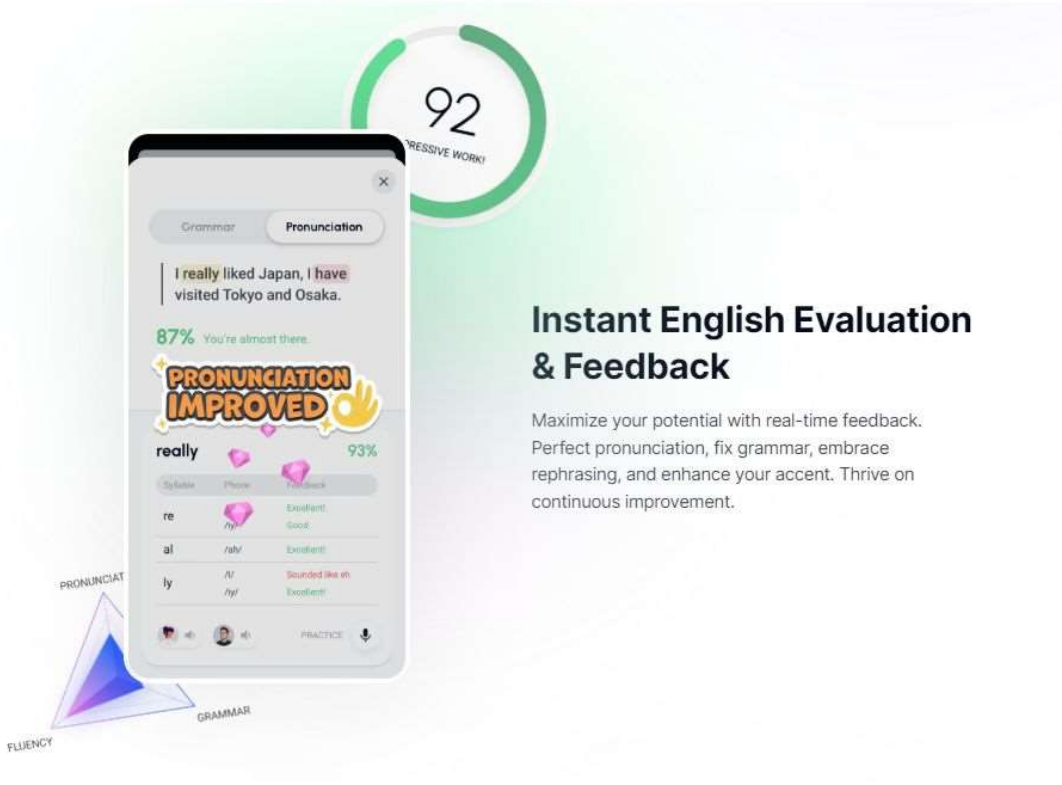


Tutor to Improve English Fluency - Loora AI

- Loora AI is a new age product that wanted to revolutionize how english is taught.
- Traditional English coaching services often rely on expensive and scarce resources, such as qualified English coaches, limiting their accessibility and reach.
- This presents a significant challenge for individuals seeking to improve their English language proficiency, especially in regions with scarce or financially prohibitive resources.

Loora AI

Loora AI is offering tutoring services to improve fluency of its customers using a chat interface.

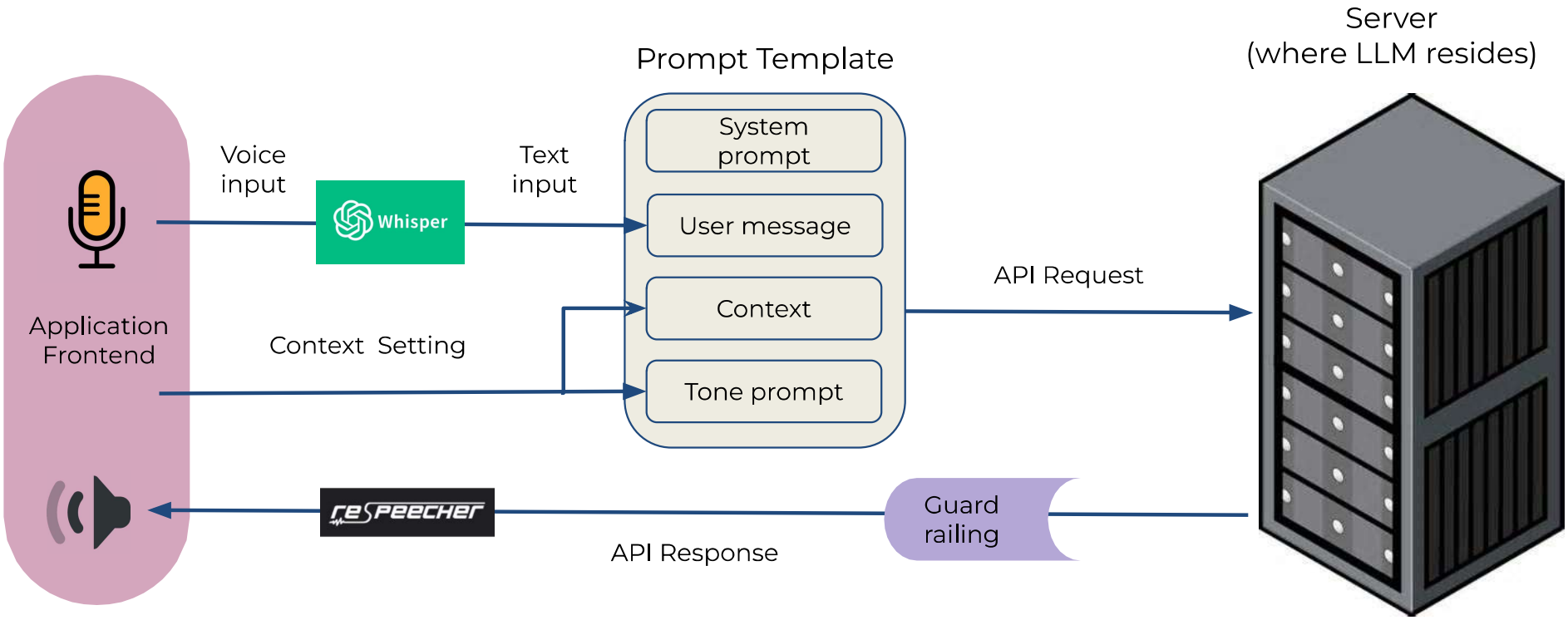


Loora AI effectively addressed a significant challenge faced by individuals from non-English-speaking countries: difficulties with fluency, vocabulary, grammar, and idiomatic expressions.

Leveraging AI tools, particularly Language Model Models (LLMs), proved to be the ideal solution. LLMs, renowned for their mastery of language, efficiently tackled these linguistic hurdles.

Loora AI

Architecture



Architecture of Loora AI

1. On loora.ai interface, the user first selects a context (interview, conversation at a restaurant, phone call, etc.).
2. Then as the conversation starts, the user speaks into the device.
3. The voice is then converted into text using AI tools like Whisper.
4. User's input (after getting converted into text), along with the tone, and context information extracted from the user input at the start are sent to the back-end where all of them along with a system prompt designed to guide the LLM's response towards the desired output are concatenated into a single concrete prompt.
5. The final prompt is then sent to the LLM hosted on a server through an API.
6. The response from the server is then sent through a guardrail to check for any violations. After that, the text is sent through a text-speech converter like Speecher.
7. Post that, the converted audio is sent to the front end where the user can hear the "other person" talking.

Architecture of Loora AI

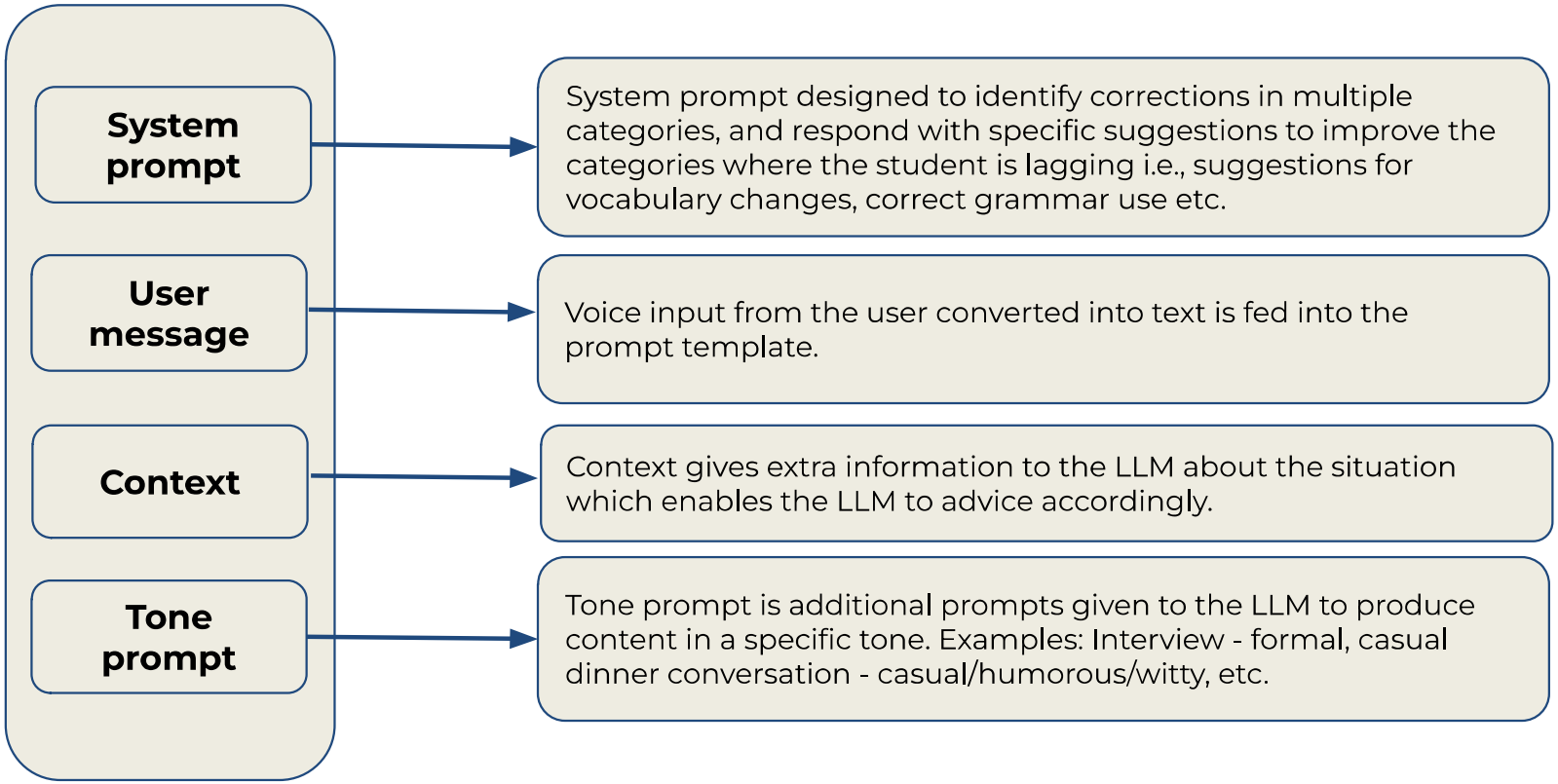
For Loora, voice functionality is crucial, as users require practice in speaking the language. So apart from the architecture we have seen in the grammarly's generative text, Loora must have integrated other AI tools to convert audio to text and text to audio.

The integration of complementary AI tools such as Whisper and Respeecher can facilitate the seamless connection between the text-based LLM like GPT3.5 and the predominantly voice-based interface of the application.

Moreover, with the advent of multi-modal LLMs like GPT4, the need for such bridges diminishes, as GPT4 boasts inherent listening and speaking capabilities. Thus, becoming an end to end solution.

Note:
Unlike Grammarly, where Gen AI adds new functionality, for this start-up, GenAI tech is at the core of its product. In a way, the whole business of this start-up is founded on GenAI technology.

Tutor to Improve English Fluency for Loora AI



Github Copilot

GitHub Copilot, a code completion tool co-developed by GitHub and OpenAI. This AI-powered pair programmer enhances efficiency in the software development process, reducing delivery times and boosting individual developer productivity.

Key Features:

1. Integration: Works with Visual Studio Code, Visual Studio, Neovim, and JetBrains IDEs.
2. AI-driven: Leverages OpenAI's powerful language models for intelligent code completion.
3. Time-saving: Speeds up development by reducing time spent on manual coding.
4. Context-aware: Understands code context to provide relevant and accurate suggestions.
5. Continuous learning: Improves with use as the AI model adapts to user coding patterns and styles.

Github Copilot

Challenges in Software Industry Pre-Generative AI

- **Steeper learning curve:** Software development is renowned for its steep learning curve, characterized by the rapid acquisition of knowledge and skills required to master the craft. This posed a higher entry barrier to people.
- **Maintaining Code Quality:** Ensuring code quality and consistency across large projects with many individuals each with a different skill level and a different style pose a significant challenge.
- **Debugging:** Even to the skilled developers identifying bugs in their code can be problematic and could potentially take away hours from their day.
- **Automatic Testing:** LLMs support the creation of comprehensive test cases by generating inputs, expected outputs, and edge cases based on code logic and specifications. They assist in automating the generation of test scripts and scenarios, reducing the manual effort required for test case creation.
- **Documentation:** *LLMs streamline the documentation process by generating clear and concise explanations for code segments and functions.*

Github Copilot

Entry of LLMs

For large language models (LLMs), code is indeed another language. These models excel at predicting the next set of tokens in a sequence and generating tokens given a prompt, which makes them particularly well-suited for programming-related tasks.

When trained on programming-related datasets, LLMs demonstrate remarkable coding capabilities, such as writing functions, debugging code, and even generating entire programs from scratch. This makes them valuable tools for developers, enabling them to automate repetitive tasks, explore new ideas, and improve their productivity.

Github Copilot

Copilot disrupting the software industry

As the LLMs started to gain prowess in code completing and generation, Github copilot, a project developed together by github (owned by microsoft) and open AI (49% owned by microsoft) saw the potential of the technology and started developing a solution for the woes of developers and programmers.

Github copilot integrated the solution right into the IDEs where developers code. Programmers could interact with the model in two ways.

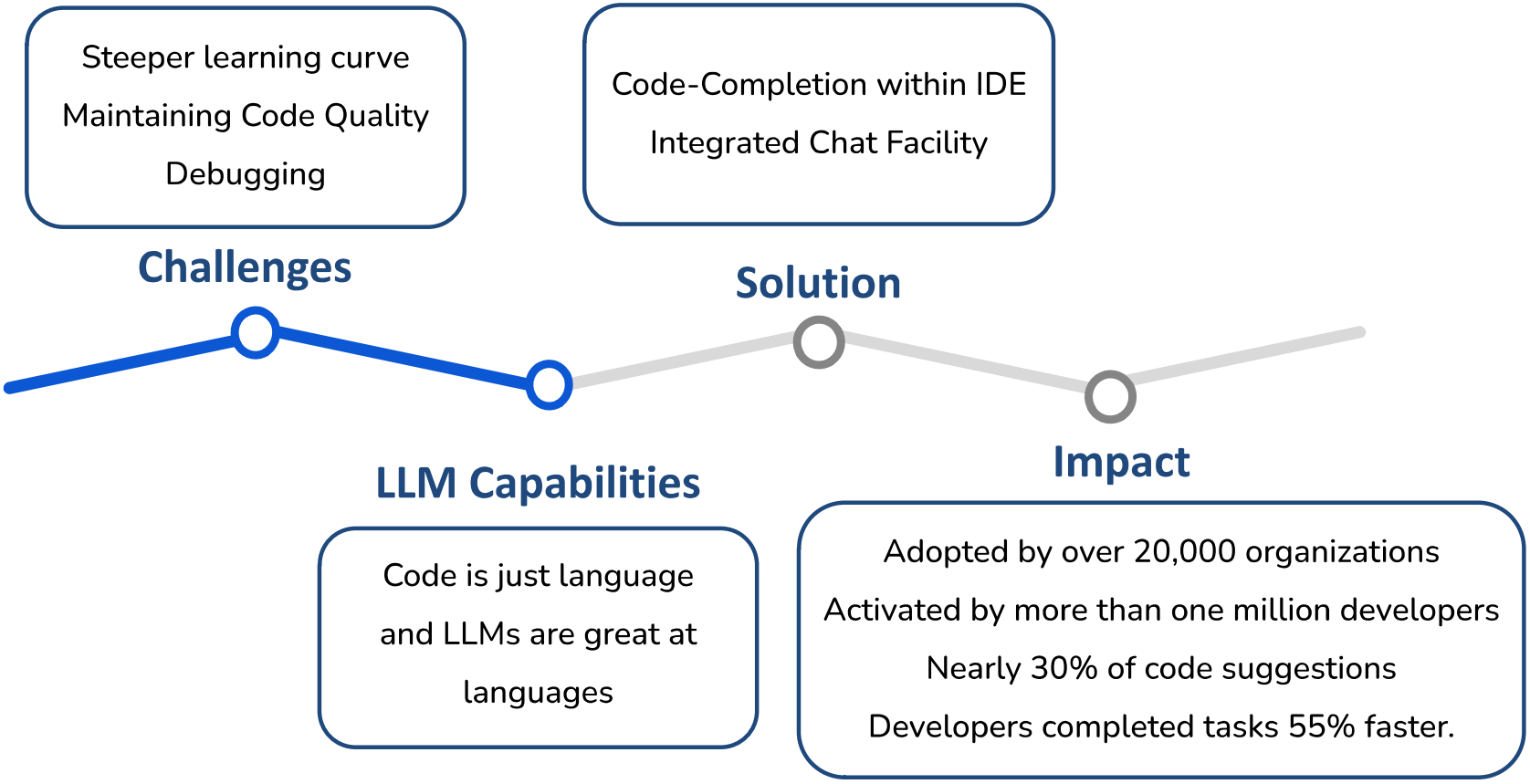
1. Copilot can proactively suggest code completions for the programmer based on the comments in the code, the file name, and other code in the vicinity (within the same file and the same folder) which the user can accept by pressing the 'tab' key.
2. Programmers can chat with the copilot where they can provide explicit instructions on what they want from it. This could be helped with debugging, syntax suggestions, a faster code snippet, etc.

Github Copilot

Impact of Copilot

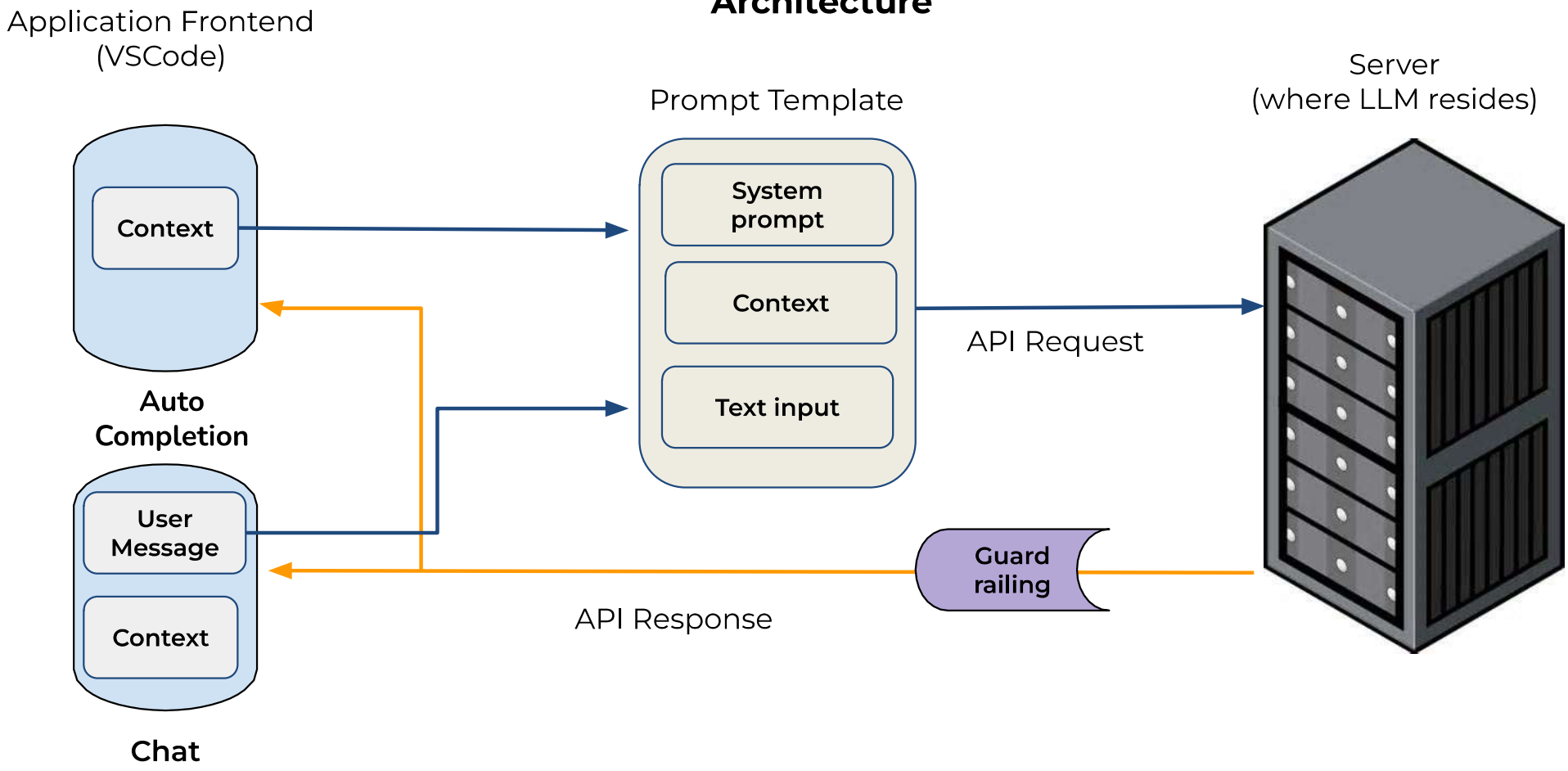
- Favored by 55% of developers,
- adopted by over 20,000 organizations
- activated by more than one million developers
- users accept nearly 30% of code suggestions from GitHub Copilot and report increased productivity from these acceptances.
- developers completed tasks 55% faster with GitHub Copilot.
- in the files where it was enabled, 46% of the code was completed by GitHub Copilot.
- Engineering teams at Duolingo, for instance, have used GitHub Copilot for Business to achieve a 25% increase in developer velocity.
- Github Copilot created a successful product and raked in massive revenue which will only increase moving forward.

GitHub Copilot



GitHub Copilot

Architecture



Github Copilot

- Github copilot integrates with multiple IDEs (Let's focus on Visual Studio).
- There are two functionalities of GitHub Copilot.
  a. Code Completion
  b. Chat
- **Code Completion:** While the user is coding, the copilot proactively gathers context information like file name, extension, code within the file, and relevant code snippets from other files in the folder.
- All of this information is brought together along with a system prompt designed by the developers at GitHub copilot to create a final prompt.
- For **Chat functionality**, along with the user prompt, context is gathered by the co-pilot. All of this is brought together to create a final prompt in the back-end.
- These final prompts are sent to the LLM hosted on a server through an API.
- The response from the server is then sent through a guardrail to check for any violations.
- Post that, the response is sent to the front end where the user can integrate the response into their documents.

Importance of good prompts:

An Inside story of Prompting at Github Copilot - part 1

In the beginning, GitHub Copilot also tended to suggest lines of code in a completely different programming language, which created a poor developer experience (for somewhat obvious reasons).

“You could be working in a C# project, then all of the sudden at the top of a new file, it would suggest Python code,” Rosenkilde explains. So, the team added a headline to the prompt that listed the language you were working in. “Now this had no impact when you were deep down in the file because Copilot could understand which language you were in. But at the top of the file, there could be some ambiguity, and those early models just defaulted to the top popular languages.”

Importance of good prompts:

An Inside story of Prompting at Github Copilot - part 2

About a month following that improvement, the team discovered that it was much more powerful to put the path of the file at the top of the document.
“The end of the file name would give away the language in most cases, and in fact the file name could provide crucial, additional information,” Rosenkilde says. “For example, the file might be named ‘connectiondatabase.py.’

Well that file is most likely about databases or connections, so you might want to import an SQL library, and that file was written in Python. So, that not only solved the language problem, but it also improved the quality and user experience by a surprising margin because GitHub Copilot could now suggest boilerplate code.”

Summary

Pre-GenAI V/S Post GenAI

Historically, these tasks posed significant challenges to businesses, often requiring extensive manual hours. However, the emergence of Generative Artificial Intelligence (Gen AI) is revolutionizing how these challenges are tackled.

Importance of Prompt engineering

Throughout the case studies, we have seen the importance of prompt engineering. While LLMs are the powerhouses behind these solutions, interacting with LLMs requires skill. Getting the LLM to generate desired responses requires crafting suitable prompts.

Demand for GenAI solutions:

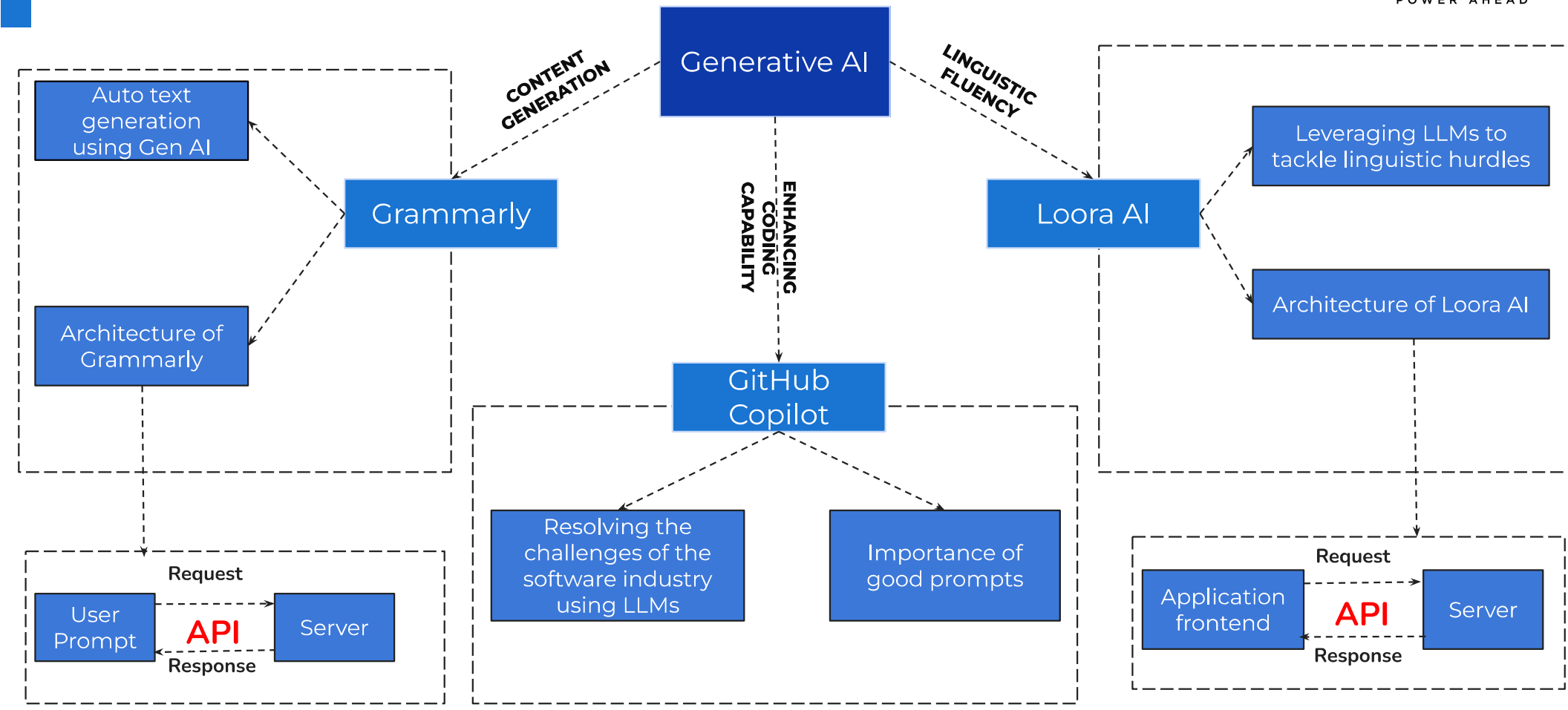
Demand for Gen AI products is across industries. From small to huge corporations everyone requires more Gen AI Products.

Prompting at scale

All the case studies, we have seen processed a lot of requests per second whether it be Grammarly or Copilot. To achieve this, it is necessary to craft dynamic prompts at scale. We can achieve that through,

- No-code tools
- Programming (Like Python)

Questions?



Great Learning

Happy Learning!