בית. תוכנה. חברה.

info@zion-net.co.il

Generative AI in C#

Harnessing Large Language Models for Enhanced Development

Is Generative AI the End for Developers?

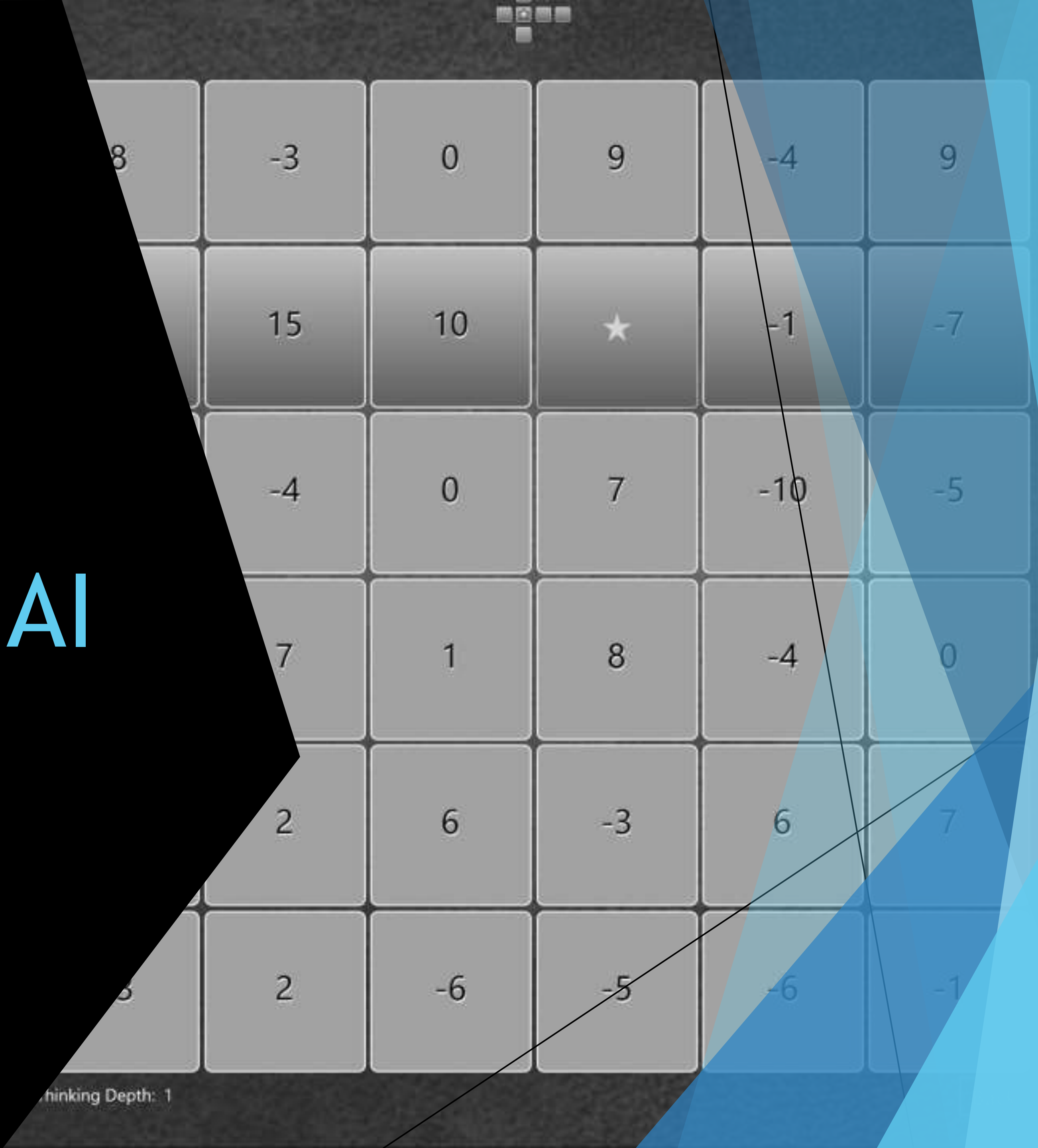
About Me

- Alon Fliess:
  - CTO of ZioNet
  - More than 30 years of hands-on experience
  - Microsoft Regional Director & Microsoft Azure MVP

About ZioNet

- A Professional Software service provider company
- The home for hi-potential novice developers and expert leaders
  - We support our developers' growth, provide them with professional and personal mentoring
- ZioNet management has over 20 years of experience
  - We strive to fulfill the need by ensuring developers have the best first-job experience!

Introduction to AI

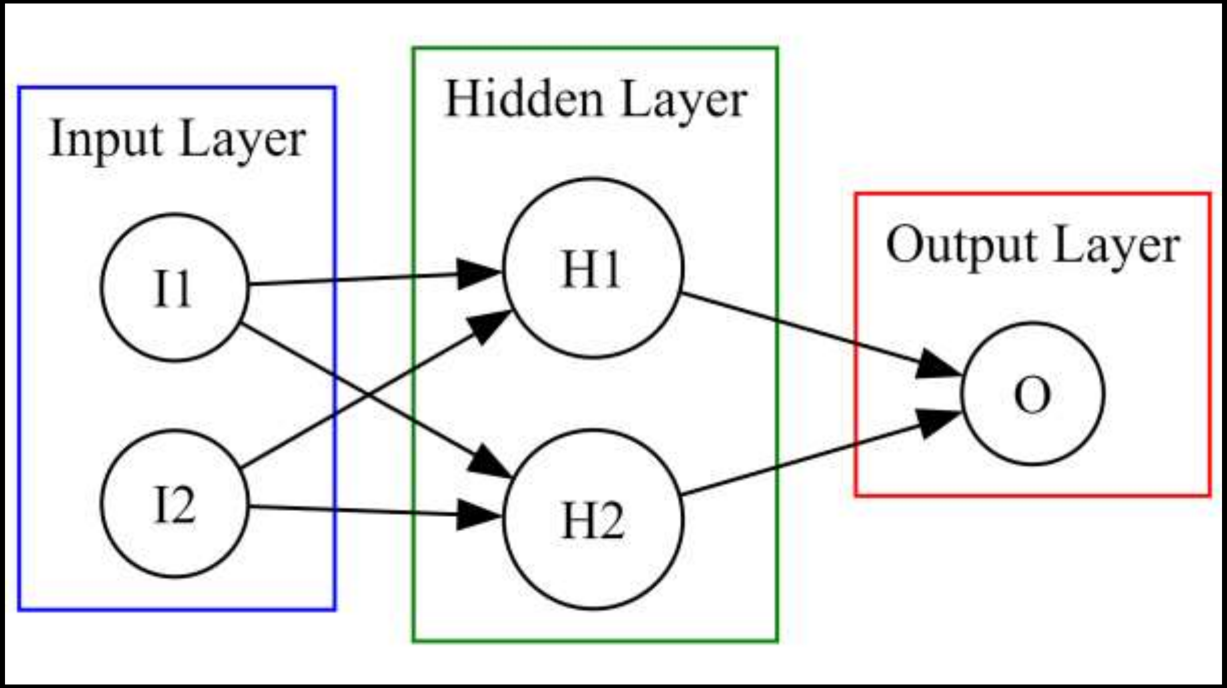


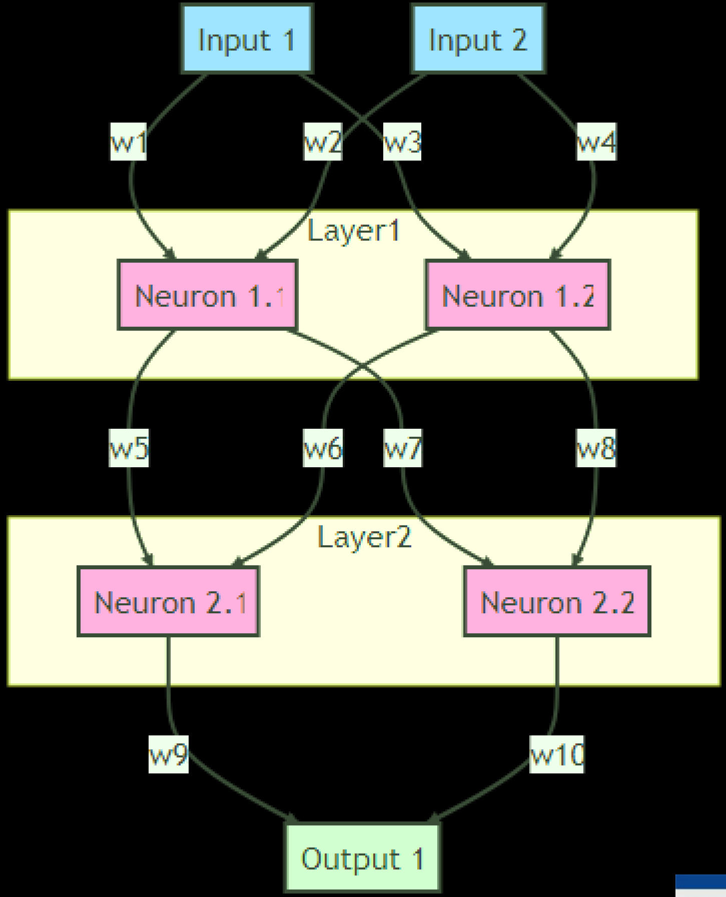
Overview of Different Types of AI

- **Algorithm-based AI:** Uses rule-based decision-making systems
- **Supervised Learning:** Learns from a labeled dataset
- **Unsupervised Learning:** Finds hidden patterns in data
- **Reinforcement Learning:** Improves via reward-based feedback
- **Hybrid AI:** Combines different AI methodologies
- **Large Language Models**: Generates text by learning from a tremendous amount of text

What is a Neural Network?

- Inspired by the human or animal brain
  - Far from being the same

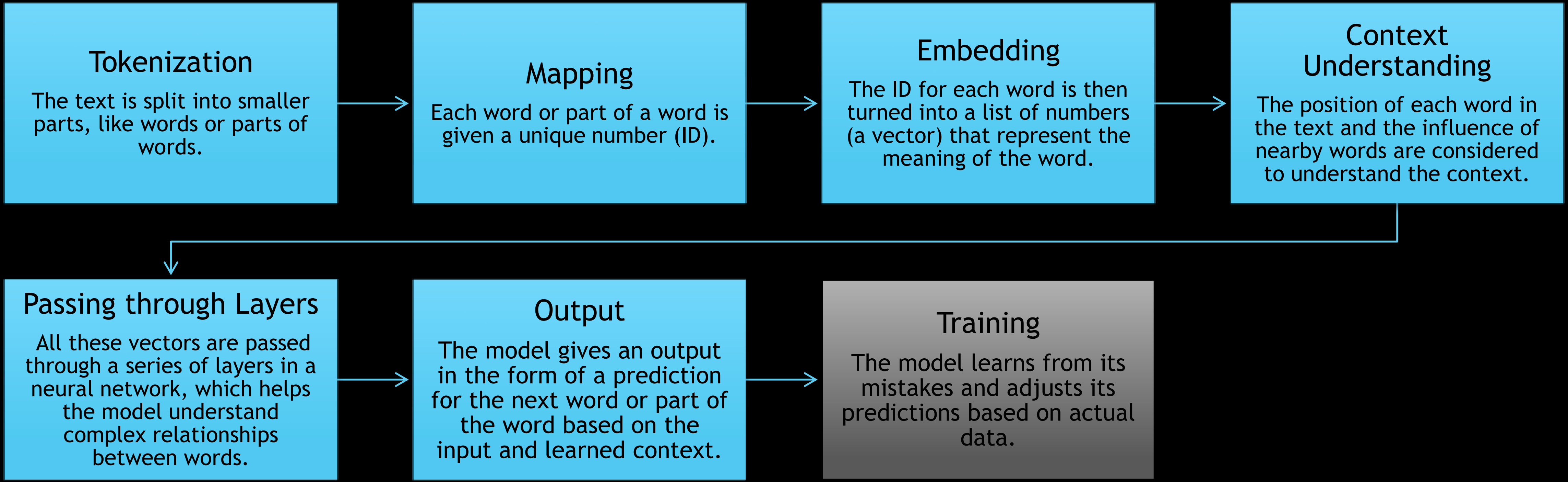




Large Language Model Overview (GPT, LLaMA, LaMDA, PaLM)

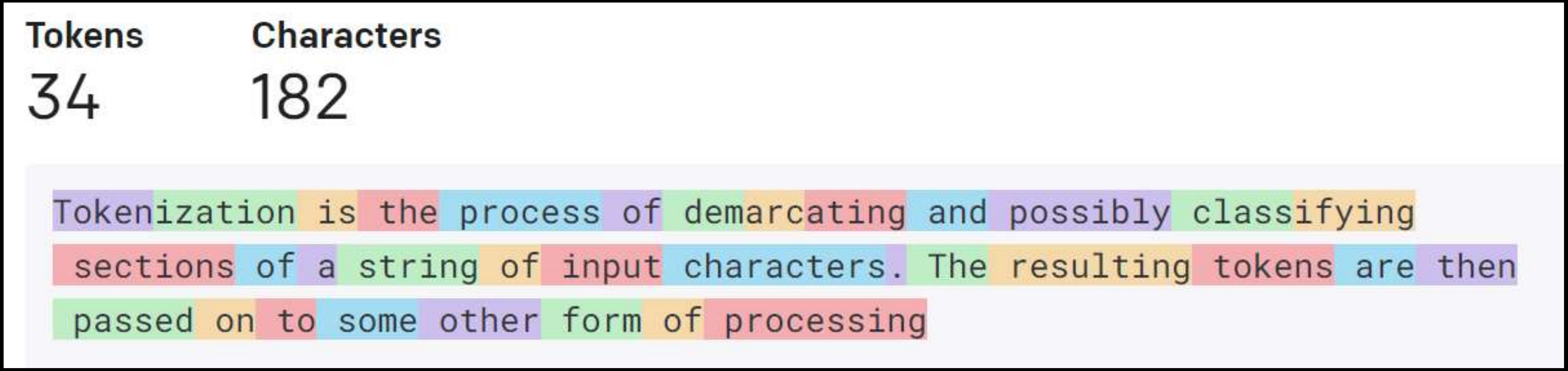
- Capture the semantics of a language
- Trained with very large amount of data
- All you need is Attention… and position (context)
- The model predicts the next world using probability, based on the input text
- The next word (token) is predicted using the original input and all the word that were generated before

LLM Processing (GPT)


Tokenization
The text is split into smaller parts, like words or parts of words.
Mapping
Each word or part of a word is given a unique number (ID).
Embedding
The ID for each word is then turned into a list of numbers (a vector) that represent the meaning of the word.
Context Understanding
The position of each word in the text and the influence of nearby words are considered to understand the context.
Passing through Layers
All these vectors are passed through a series of layers in a neural network, which helps the model understand complex relationships between words.
Output
The model gives an output in the form of a prediction for the next word or part of the word based on the input and learned context.
Training
The model learns from its mistakes and adjusts its predictions based on actual data.

ZioNet ציונט

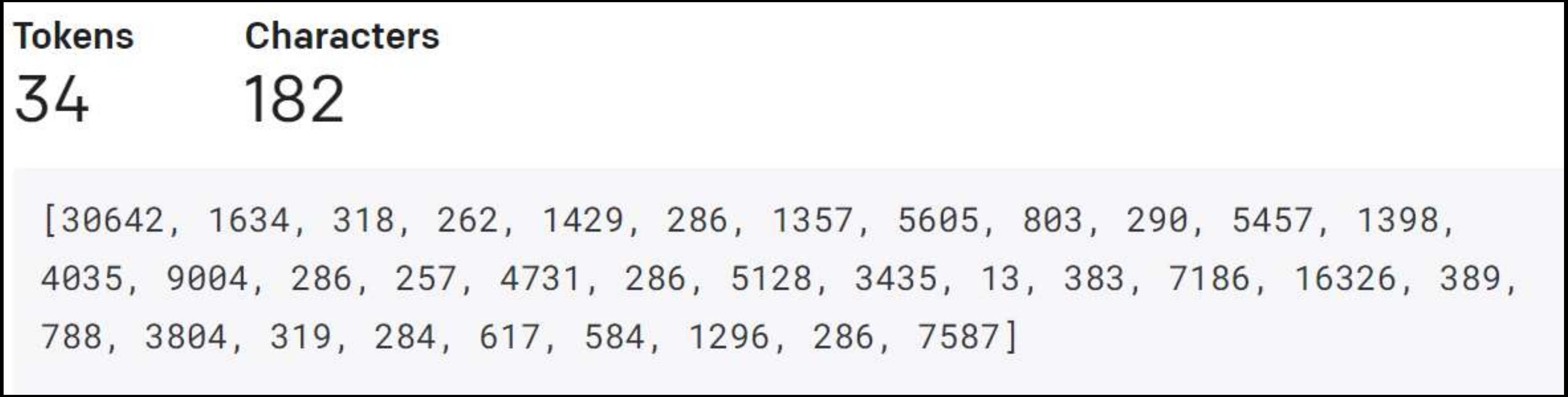
Tokenization - Explained



Tokens
34

Characters
182

Tokenization is the process of demarcating and possibly classifying sections of a string of input characters. The resulting tokens are then passed on to some other form of processing



Tokens
34

Characters
182

```
[30642, 1634, 318, 262, 1429, 286, 1357, 5605, 803, 290, 5457, 1398,
4035, 9004, 286, 257, 4731, 286, 5128, 3435, 13, 383, 7186, 16326, 389,
788, 3804, 319, 284, 617, 584, 1296, 286, 7587]
```

What are the Base Models (OpenAI, Azure)

- Some predefined models were trained to do specific tasks
- There are variations:
  - The size (input + output) tokens
  - The speed
  - The usage price
  - The ability to be fine-tuned

Can I have my Fined-Tuned Models?

- Why Fine-Tune?
  - Improve the model performance on specific tasks
  - Adapt the model to new data
  - Customize the model's behavior
- When to Fine-Tune?
  - When the Pre-Trained Model is not performing well, or the grounding (system) prompt is too large
  - When you have a specific task or data
- How to Fine-Tune?
  - Use tools like Azure Machine Learning Studio or OpenAI's Python client for fine-tuning
  - Specify the Base Model and Dataset
  - Monitor Fine-Tuning Job and retrain
  - Use Fine-Tuned Model for Predictions

Learn how to Generate or Manipulate Text

- Classification, such as sentiment
- Generation – Create a text or formatted text (JSON, XML)
- Conversation – Chat to get information, give commands, or generate and manipulate the result
- Transformation:
  - Language Translation
  - Text to emoji
  - Any format to any format
- Summarization – reduce the size of text
- Completion – complete a statement
- Code - Use Codex to generate, complete, or manipulate source code

LLM for Code is like SoC for Hardware

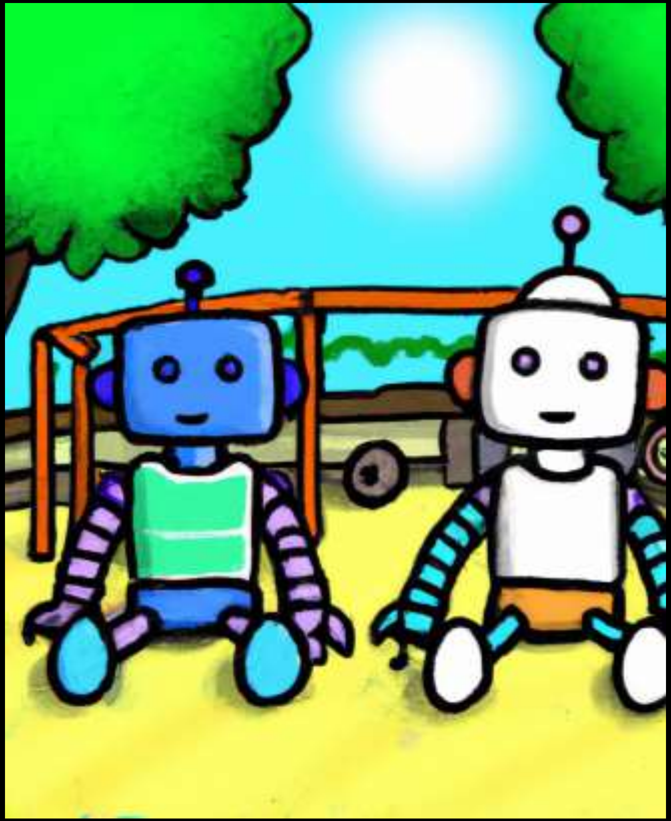
Introduction to Prompt Engineering

- What is the **first** thing that comes to your mind when I say "<prompt>?"
- Prompt Engineering is the art of crafting inputs to get desired outputs from AI models
- It's a crucial part of using AI models effectively
  - The design of the prompt can greatly influence the model's response.
- Examples:
  - If you want a list, start your prompt with a numbered list.
  - If you want a specific format, provide an example in that format.
- It often involves a lot of trial and error
  - Different prompt strategies may work better for different tasks
- Less is more!

Prompt Engineering Recommendations

- **Goal**: Define what you want from the model
- **Instructions**: Be clear and explicit
- **Examples**: Use them for specific formats or styles
- **Iterate**: Experiment with different prompts
- **Guidance**: Use system-level and user-level instructions
- **Settings**: Adjust temperature and max tokens as needed

Playground – ChatGPT and Azure



- To get started, use ChatGPT or Azure Open AI playground - Demo
- You can generate your boilerplate code for grounding and examples
- You can play with fine-tuning parameters

LLM Meets C#

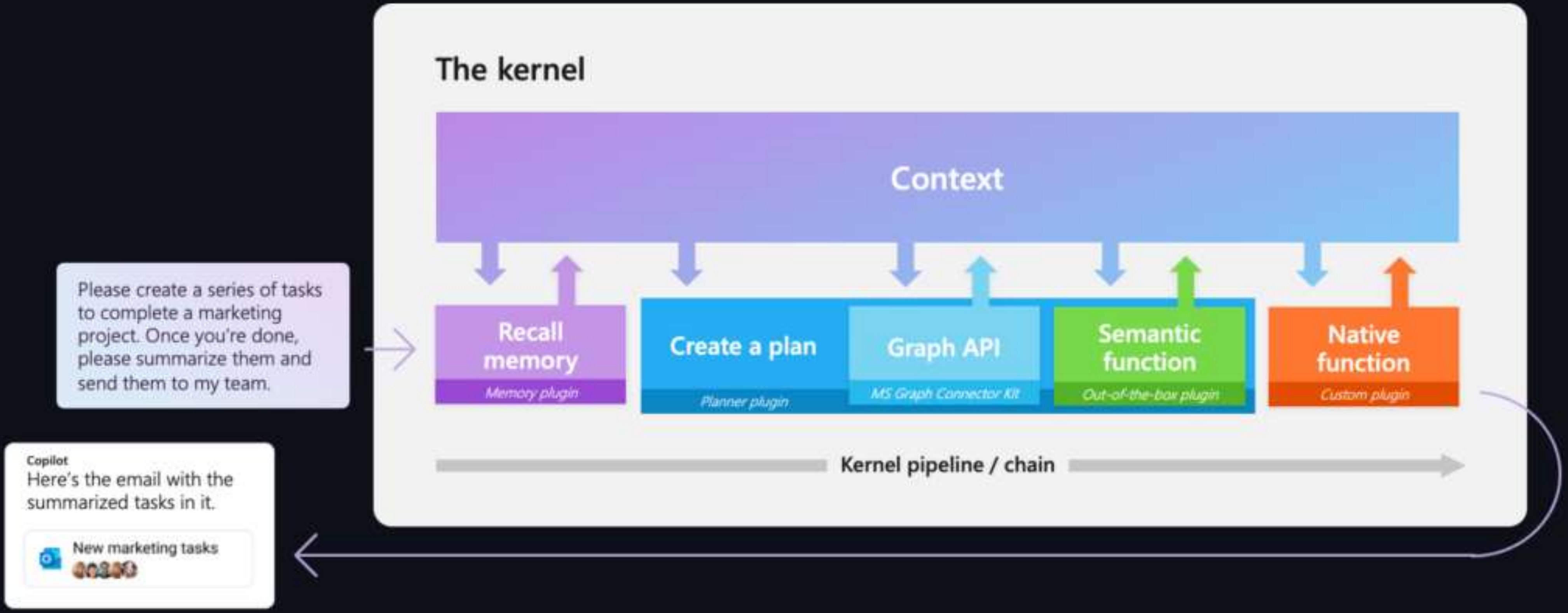
Semantic Kernel

pypi v0.4.7.dev0 Nuget package dotnet-ci-docker passing dotnet-ci-windows passing license MIT Discord 380 online

Semantic Kernel is an SDK that integrates Large Language Models (LLMs) like OpenAI, Azure OpenAI, and Hugging Face with conventional programming languages like C#, Python, and Java. Semantic Kernel achieves this by allowing you to define plugins that can be chained together in just a few lines of code.

What makes Semantic Kernel *special*, however, is its ability to *automatically* orchestrate plugins with AI. With Semantic Kernel planners, you can ask an LLM to generate a plan that achieves a user's unique goal. Afterwards, Semantic Kernel will execute the plan for the user.

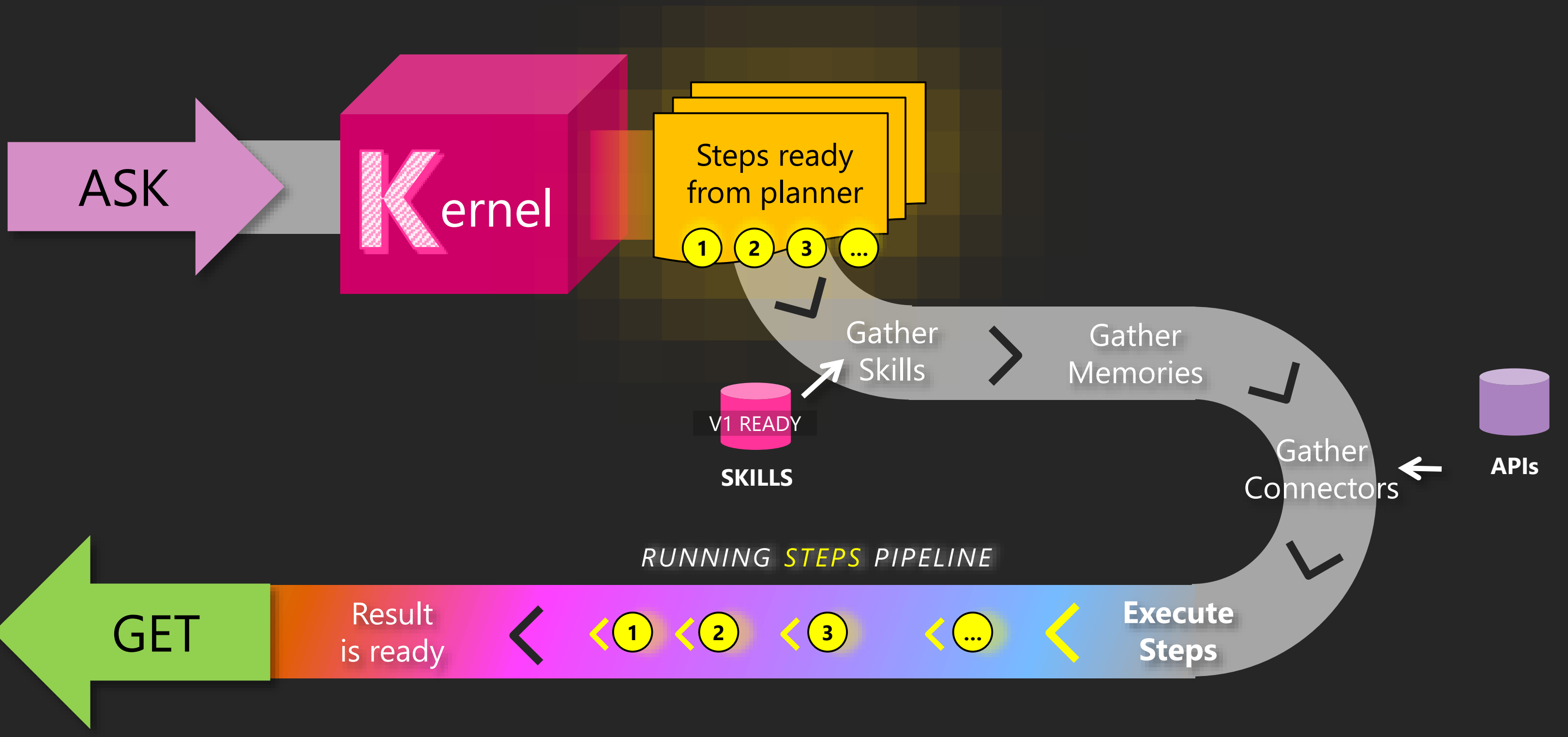
Please star the repo to show your support for this project!



Goals-First AI

It all starts with a user's AI ask …

… resulting in new productivity

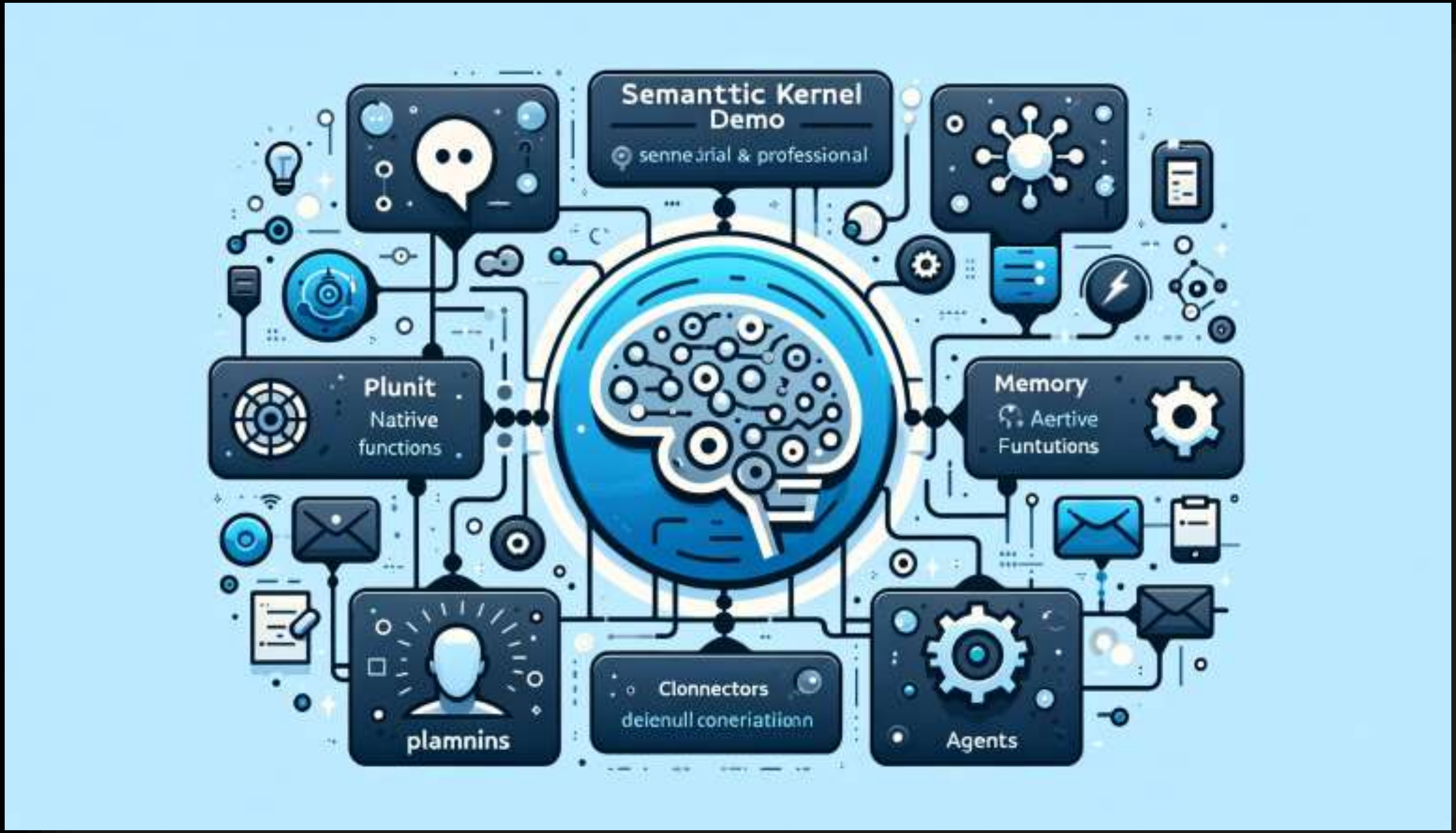


aka.ms/semantic-kernel

Semantic Kernel Main Concepts

- Prompt Functions:
  - Define interaction patterns with LLMs
- Native Functions:
  - Enable direct code execution by AI
- Plugins:
  - Custom-built, modular elements enabling specialized LLM task handling
- Memory:
  - Contextual data repository; supports key-value pairs and semantic embeddings
- Connectors:
  - Interface with external data and APIs
- Planners:
  - Orchestrate tasks, manage complex tasks through intelligent LLM planning
- Agents:
  - Autonomous entities executing orchestrated tasks

Demo

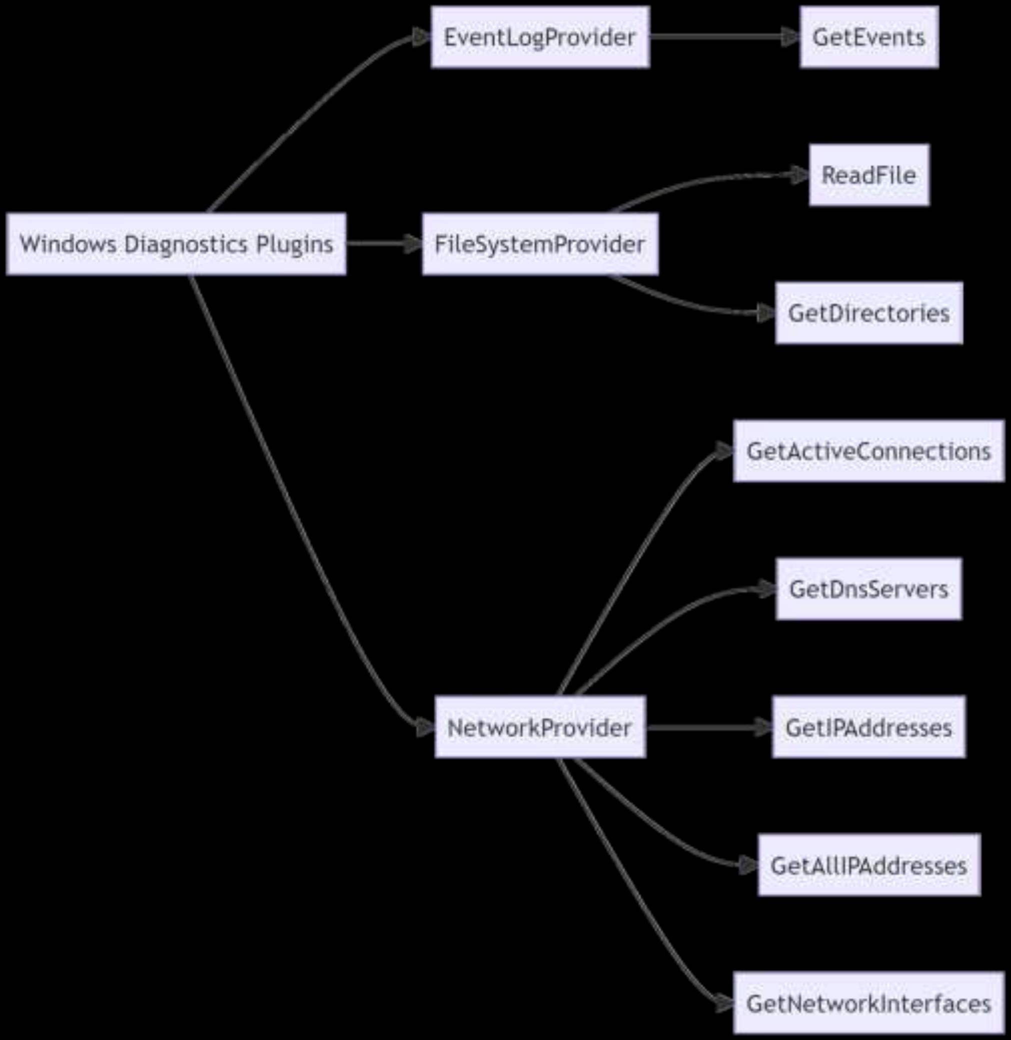


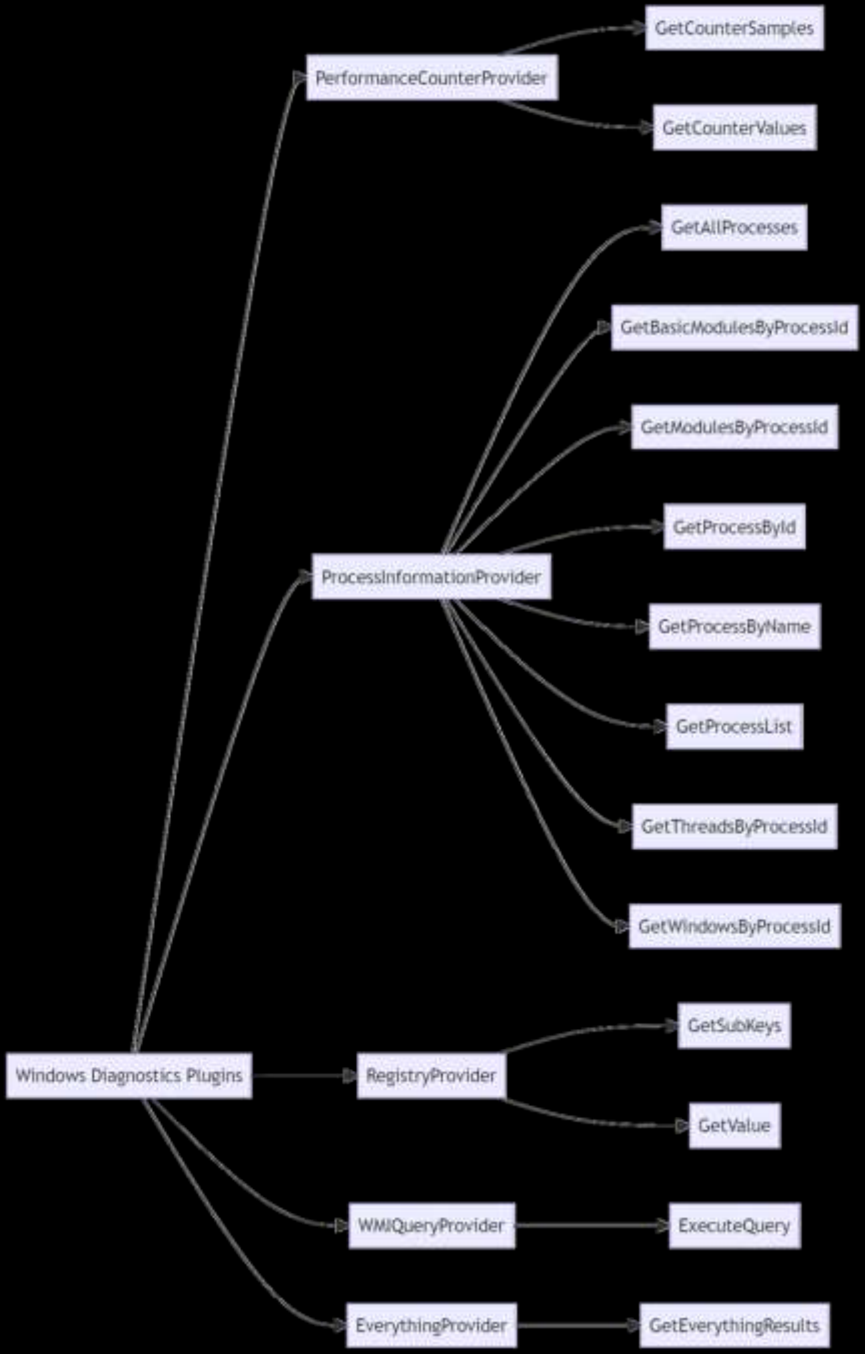
ZioNet ציונט

Windows Troubleshooting Plugin Capabilities

- **Event Log**: Query Windows Event Logs
- **File System**: Read and search files
- **Registry**: Access and query keys
- **Performance**: Monitor system metrics
- **Network Info**: Retrieve connection details
- **Process Info**: List running processes
- **Service Info**: Query system services
- **WMI Query**: Execute WMI commands
- **Everything**: Advanced file search
- **Paging**: Supports result pagination
- **Tenant/PC ID**: Requires GUIDs.
- **Multi-Function**: Multiple queries in one call
- **Parameter Flex**: Customizable parameters
- **Truncation**: Handles truncated results
- **Max Token**: Customizable data size.

The Windows Troubleshooting Plugin





Lesson Learned

- The first project iteration: An OpenAI ChatGPT plugin
  - Before Semantic Kernel
  - Had to bridge LLM to a C# Code
    - Simple APIs, Json, Reflection, Default Values
    - Needed to reduce the description load dew to token limitation
    - No Planner
- The second project iteration: Use multiple LLM and SK
  - Work in progress
  - Requires lots of fine tuning – debug while you go
  - Expensive
  - For agents → Use Asynchronous Model or a batch processing

Lessons Learned from Developing Plugin

- ***Plugin API Design***
  - **Be Systematic**: Stick to one or a few APIs for consistency
  - **OpenAPI Description**: Provide a comprehensive Open API specification
- ***Plugin Manifest***
  - **Examples**: Include examples for each query method
  - **Instructions**: Update examples and instructions if ChatGPT calls with the wrong data schema
- ***Data Handling***
  - **Paging**: Implement paging capabilities
  - **Truncation**: Use HTTP code 206 and a special message to indicate truncated results
  - **Important Data**: Always return key data like I do in the Windows Troubleshooting with Tenant Id and PC ID
- ***Performance & Limitations***
  - **Trial and Error**: Extensive testing is crucial
  - **Size Limits**: Be aware of total size and per JSON element limits
  - **Resource Management**: Be cautious of overusing ChatGPT 4 resources
- ***Dynamic Content***
  - For complex plugins, dynamically generate the Open API specification and plugin manifest

Lessons Learned from Embedding LLM into Applications

- ***Grounding & Serialization***
  - Provide accurate grounding and a JSON schema describing the result
  - Use Semantic Kernel Prompt and Native function – use string and Json parameters
- ***Model Selection***
  - Use GPT3/3.5/4 for chat functionalities
  - Use other models for embedding, image creation and recognition
- ***Performance & Cost***
  - GPT4 is more accurate but costly and slower
  - Consider fine-tuned models if applicable – high cost for a good result
- ***Message Handling***
  - Handle truncated messages with a continuation strategy – less important on 120K tokens
  - Manage message size by counting tokens and removing history
  - Use summary messages to replace original history if needed
    - Use Memory (Embedding/Vectorization)

Overcome the Model Limitation

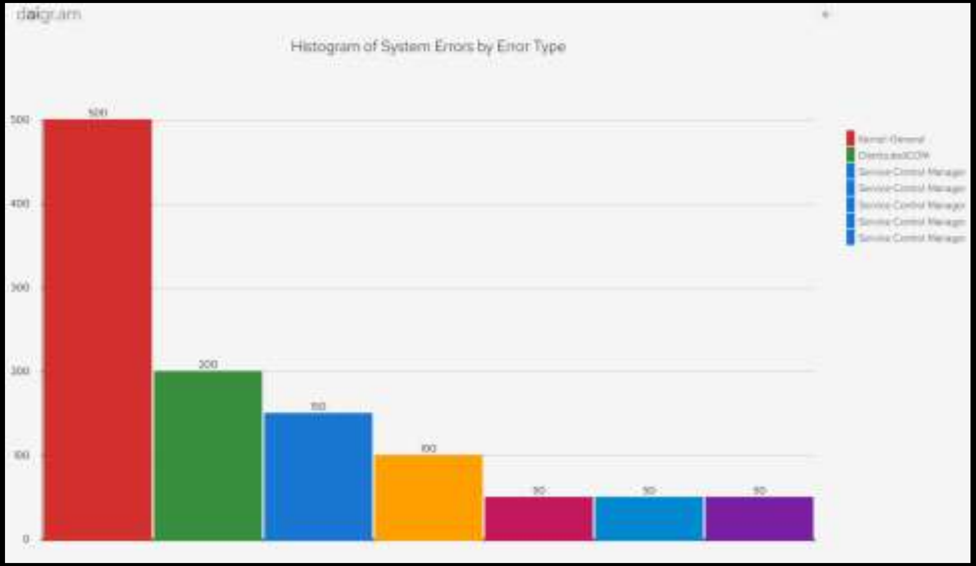
- Token Limitation
  - Use Summarization
  - Use Memory (store important facts)
  - Use Retrieval Augmented Generation (RAG) + Vector and other search
- Overcome the lack of current information
  - Use Bing, or other search Plugins
- Overcome cost and availability
  - Use GPT3.5 Turbo
  - Use Open-Source Models
  - Prompt engineer cheap models (3.5) with GPT 4

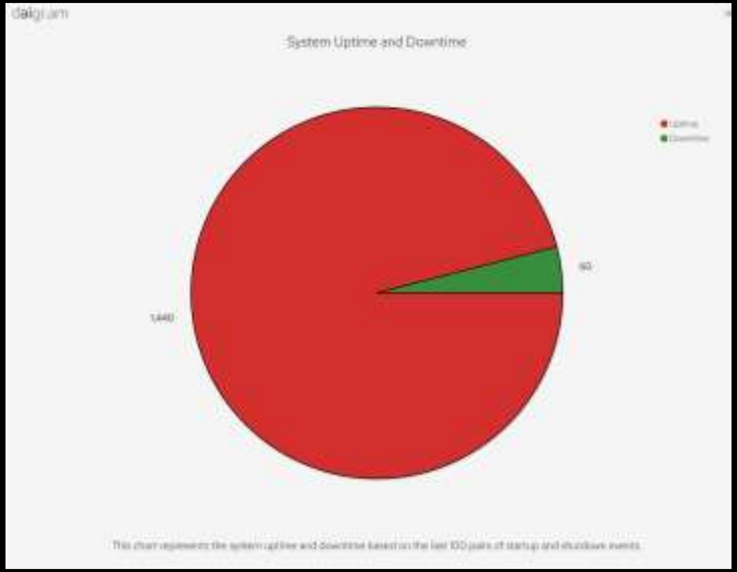
ציונטZioNet

The Windows Event Log Plugin

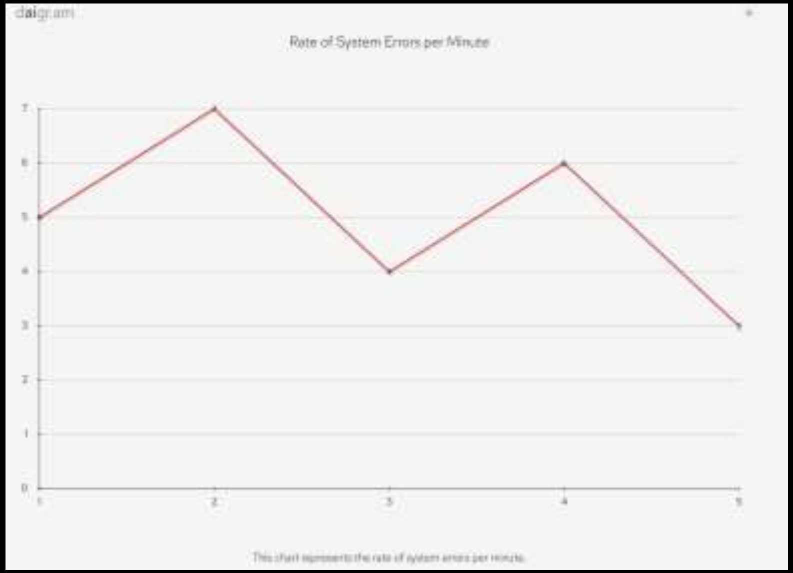
- https://github.com/alonf/WindowsEventLogChatGPTPlugIn.git
- Retrieve specific events from the Windows Event Log using XPath queries.
- Supports all major log names: Application, Security, Setup, System, and ForwardedEvents.
- Use it to solve problems and get information about your Windows system status.
- The plugin supports paging. It estimates the number of tokens and limits the result.

Examples





This chart represents the system uptime and downtime based on the last 100 pairs of startup and shutdown events.



This chart represents the rate of system errors per minute.

Developing ChatGPT Plugin Using C#

- Use ASP.NET Minimal or Controller based API
- Use YamlDotNet to convert Json Open API specification to yaml based
  - The Open API specification, the plugin manifest and the icon file can come from the file system, or as a HTTP query

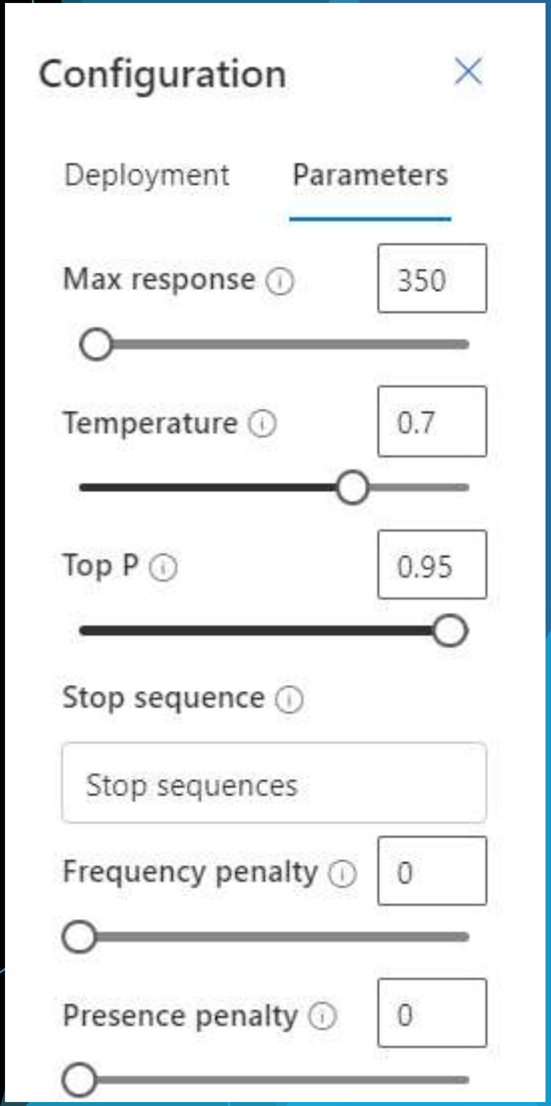
```
app.UseStaticFiles(new StaticFileOptions
{
    FileProvider = new PhysicalFileProvider(
        Path.Combine(app.Environment.WebRootPath, "OpenAPI")),
    RequestPath = "/OpenAPI"
});
```

Developing ChatGPT Plugin Using C#

- Route HTTP Request to a function
  - For a simple plugin, it is just a get request
  - For A complex plugin, use POST with a body and have your own route
- In the Windows Troubleshooting plugin, I use a map of providers and made the call to the specific function using reflection
  - I use validation to make sure the data size and type is correct
  - I provide extensive error message
- Use correlation id for local development (Tenant ID, PC ID)
- Use OAuth for released plugin

Controlling the Output layer

- Max response: defines the token limit for the model's response
  - One token is roughly equivalent to 4 English characters
- Temperature: Controls the randomness of the model's responses.
  - Lower values result in more deterministic responses, higher values lead to creativity
- Top P: Another parameter to control randomness
  - Lower values make the model choose more likely tokens
- Stop sequence:
  - Specifies a sequence at which the model should stop generating a response
- Frequency penalty: Reduces the likelihood of the model repeating the same text
  - by penalizing tokens that have appeared frequently
- Presence penalty:
  - Encourages the model to introduce new topics in a response
    - by penalizing any token that has appeared in the text so far.


Configuration

Deployment Parameters

Max response 350

Temperature 0.7

Top P 0.95

Stop sequence

Stop sequences

Frequency penalty 0

Presence penalty 0

ציונט ZioNet

Cost, Privacy & Security

- ChatGPT and Azure OpenAI services can be used without donating your data
  - For the Public ChatGPT you can ask to opt-out
  - For the API, You can ask to opt-in
- ChatGPT 4 is a very hi-cost model (become cheaper)
  - You can use ChatGPT 4 to create prompt and examples for ChatGPT 3.5
- You can host your model on-premise, however:
  - The Open-Source models do not contain the latest ChatGPT 3.5 and 4
  - You may train your own model - costly

Summary

- AI Types: Explored diverse AI forms
- Neural Networks & LLMs: Discussed their functionality
- Base Models & Fine-Tuning: Highlighted fine-tuning's importance
- Prompt Engineering: Introduced crafting effective prompts
- Semantic Kernel: Your LLM Swiss army tool
- Application Transformation: Extending applications with AI
- Autonomous Agents: The future of AI systems

LLM ←→ The Software System on a Chip

The End

Thank You!