

Model Context Protocol (MCP)

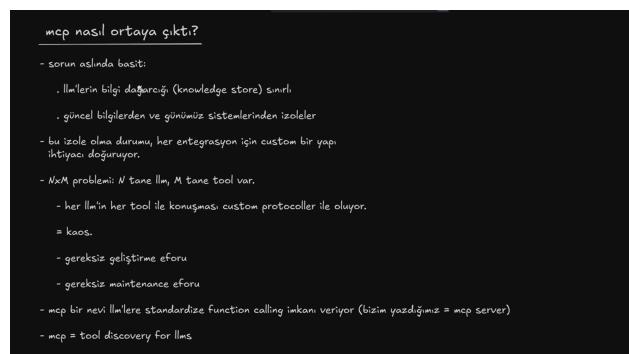
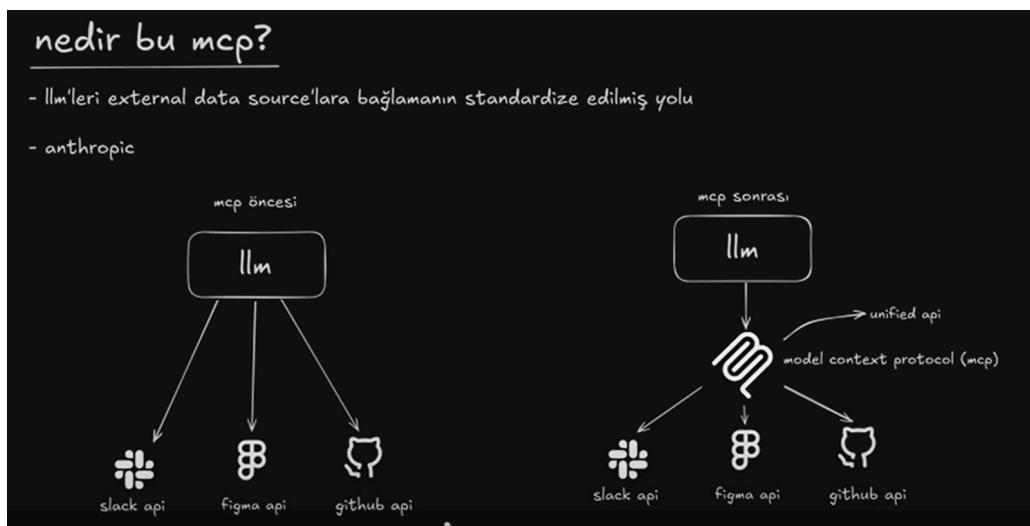
<https://modelcontextprotocol.io/docs/learn/architecture>

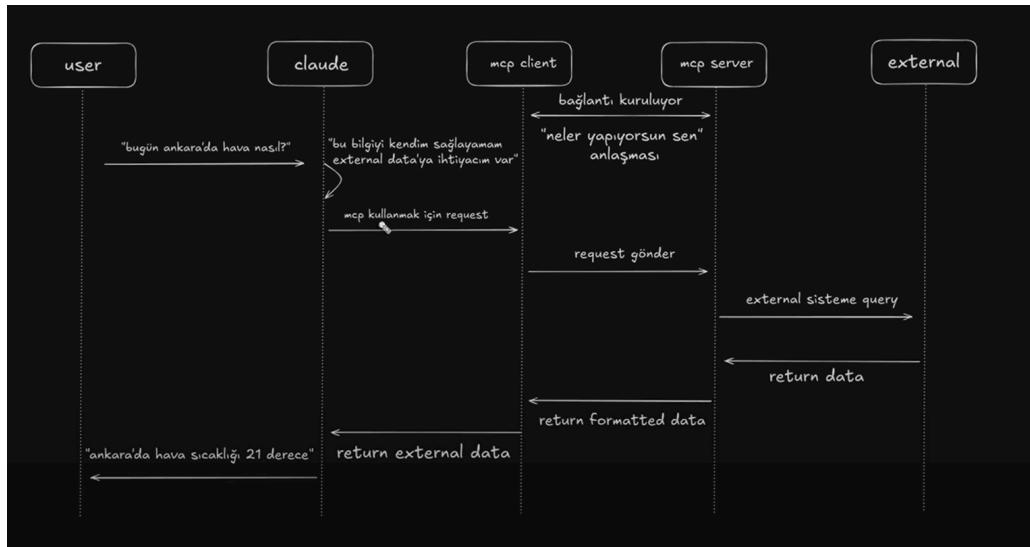
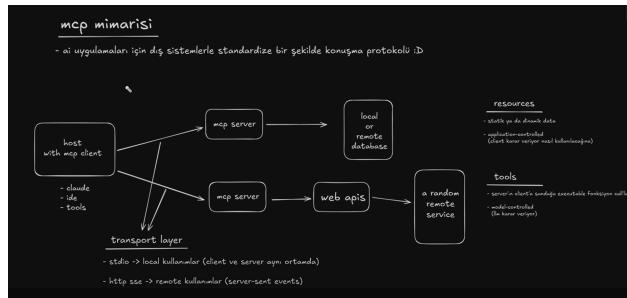
<https://github.com/modelcontextprotocol/servers-archived/tree/main>

video 1: <https://www.youtube.com/watch?v=u1PYSKqfHEw>

repo:

<https://github.com/fatihguzeldev/fatihlovestosimplify/tree/main/ne%20ulan%20bu/4.%20ne%20ulan%20bu%20-%20mcp%20server>





Model Context Protocol (MCP) kavramını detaylı şekilde anlatıyor. MCP, büyük dil modelleri (LLM - Large Language Models) ile dış kaynaklar ve araçlar arasında iletişimini standartlaştıran bir protokoldür. Gelenekselde her modelin farklı API'lerle kendi özel yöntemlerle iletişim kurması zorunluyken, MCP bu karmaşıklığı ortadan kaldırarak modellerin dış araçları ve API'ları tek bir standart üzerinden kullanmasını sağlar. Bu sayede gereksiz geliştirme ve bakım yükü azalır, ölçeklenebilirlik artar. Videoda MCP'nin ortaya çıkış sebebi, mimarisi, işleyışı ve faydaları detaylandırılıyor. Ayrıca, MCP kullanarak nasıl bir hava durumu sunucusu (server) yazılacağı örneklerle anlatılıyor. MCP server ile client arasındaki iletişim, function call ve tool discovery süreçleri açıklanıyor. MCP'nin, modellerin gerçek zamanlı ve güncel verilere erişimini sağlayarak halüsinsasyon problemini azaltabileceği vurgulanıyor. Son bölümde MCP server'ın uygulamalı kullanımı ve gerçek dünya örnekleri gösterilerek konsept pekiştiriliyor.

Öne Çıkan Noktalar

- 🌐 MCP, büyük dil modelleri ile dış kaynaklar arasında standart bir iletişim protokolü sağlar.
- ⌚ MCP, modellerin farklı API'larla özel entegrasyon yapma ihtiyacını ortadan kaldırır.
- 🛠️ MCP, "tool discovery" özelliği ile modelin hangi araçları kullanacağını kendisinin karar vermesine olanak tanır.
- 🌐 MCP, hem local hem de remote server ve client arasında çalışabilir; farklı transport layer'lar destekler.
- 📊 MCP, modellerin güncel veri kaynaklarına erişimini kolaylaştırarak halüsinsasyon problemini azaltır.

- MCP server, dış API'lardan veri çekip modeli bilgilendirir, böylece model gerçek ve doğrulanmış bilgi sunar.
- Videoda, MCP kullanarak basit bir hava durumu server'ının nasıl yazılmacı uygulamalı olarak gösterilir.

Temel İçgörüler

- **Standardizasyonun Önemi:** MCP, modellerin farklı araç ve API'larla entegre olurken her seferinde sıfırdan geliştirme yapılması gerekliliğini ortadan kaldırır. Bu, geliştirme sürecini hızlandırır ve bakım maliyetlerini düşürür. Standardizasyon, yapay zeka ekosisteminde karmaşayı azaltarak daha sürdürülebilir ve ölçeklenebilir çözümler sağlar.
- **Modelin Karar Verme Yeteneği:** MCP'nin önemli bir avantajı, modelin hangi tool'u kullanacağına kendisinin karar vermesidir (model kontrolü). Model, MCP server'ın sunduğu araçların açıklamalarını okuyup, ihtiyaca göre uygun fonksiyonu çağrıabilir. Bu, klasik API çağrılarından farklı olarak modelin daha akıllı ve bağımsız hareket etmesini sağlar.
- **Halüsinsiyon Probleminin Azaltılması:** Büyük dil modelleri, eğitildikleri veri setlerine bağlı olarak güncel olmayan veya yanlış bilgi verebilir. MCP, modellerin gerçek zamanlı, doğrulanmış verilere ulaşmasını mümkün kılarak, halüsinsiyonları azaltır ve modelin doğruluk oranını artırır. Bu, özellikle kritik uygulamalarda güvenilirlik sağlar.
- **Çok Katmanlı Mimari:** MCP mimarisi client-server yapısına dayanır. MCP server, araçları ve kaynakları tanımlar ve yönetir. Client (model) ise bu kaynakları kullanmak için MCP protokolü üzerinden iletişim kurar. Transport layer olarak yerel (std io) ve uzak (HTTP Server Sent Events) iletişim yöntemlerini destekler. Bu esneklik farklı kullanım senaryoları için uygundur.
- **Tool ve Resource Kavramları:** MCP'de "tools" (araçlar) fonksiyon çağrılarıdır, model karar verir hangi aracı kullanacağına. "Resources" ise statik veya dinamik veri kümeleridir, bu konuda host (client) kontrol sahibidir. Bu ayrim, iş bölümü ve yetki yönetimini netleştirir.
- **Pratik Uygulama ve Kodlama:** Videoda MCP server örneği TypeScript ile gösterilmiştir. Zod kütüphanesi ile skema doğrulama yapılır, tool listelenir ve çağrılr. Örneğin hava durumu için "getWeather" adlı tool, şehir adı alır ve API'dan veri çekip sonucu döner. Bu somut örnek, MCP'nin nasıl kullanılacağını anlaşırlar kilar.
- **Gerçek Dünya Kullanımı:** MCP, bulut tabanlı asistanlar veya chatbotlar gibi sistemlerin dış dünyaya bağlanması kolaylaştırır. Örneğin, bir kullanıcı hava durumu sordduğunda, model doğrudan web araması yapmak yerine MCP server'a başvurup gerçek, anlık veriyi alabilir. Bu da kullanıcı deneyimini iyileştirir ve modelin bilgi erişimini genişletir.

MCP, büyük dil modellerinin dış dünya ile entegre olurken karşılaştığı karmaşıklıkları azaltan, işlevselligi ve verimliliği artıran önemli bir standart protokoldür. Fatih'in videosu, MCP'nin teorik temellerinden pratik uygulamasına kadar kapsamlı bir bakış sunarak, yapay zeka ve yazılım geliştirme alanında çalışanlar için değerli bir kaynak oluşturuyor.

index.ts:

```
import { Server } from "@modelcontextprotocol/sdk/server/index.js";
import { StdioServerTransport } from "@modelcontextprotocol/sdk/server/stdio.js";
import {
  CallToolRequestSchema,
  ListToolsRequestSchema,
} from "@modelcontextprotocol/sdk/types.js";
import { z } from "zod";
```

```

const GetWeatherSchema = z.object({
  city: z.string(),
});

const server = new Server(
{
  name: "fatihlovestosimplify-weather-mcp",
  version: "1.0.0",
},
{
  capabilities: {
    tools: {},
  },
}
);

// list all available tools
server.setRequestHandler(ListToolsRequestSchema, async () => {
  return {
    tools: [
      {
        name: "get-weather",
        description: "get weather info",
        inputSchema: {
          type: "object",
          properties: {
            city: {
              type: "string",
              description: "name of the city (e.g istanbul)",
            },
          },
          required: ["city"],
        },
      },
    ],
  };
});

// tool body
server.setRequestHandler(CallToolRequestSchema, async (request) => {
  const { name, arguments: args } = request.params;

  try {
    if (name === "get-weather") {
      const { city } = GetWeatherSchema.parse(args);
      const reqUrl = `${process.env.WEATHER_API_BASE_URL}?key=${process.env.WEATHER_API_KEY}&q=${city}&aqi=no`;

      const data = await fetch(reqUrl);
    }
  }
});

```

```

if (!data.ok) {
  return {
    content: [
      {
        type: "text",
        text: "some error",
      },
    ],
  };
}

const jsonData = (await data.json()) as any;

return {
  content: [
    {
      type: "text",
      text: `${city} için sıcaklık değeri anlık: ${jsonData.current.temp_c}`,
    },
  ],
};

} else {
  return {
    content: [
      {
        type: "text",
        text: "unknown tool",
      },
    ],
  };
}
} catch (error) {
  const err = error as any;

  throw new Error(err.message);
}
});

const transport = new StdioServerTransport();
await server.connect(transport);

```

video 2: <https://www.youtube.com/watch?v=LCC8JQugeDc>

repo: <https://github.com/Kodla-devs/mcp-local-llama>



```

from praisonaiagents import Agent, MCP

agent = Agent(
    instructions="You manage the local file system using MCP.",
    llm="ollama/llama3.2",
    tools=MCP("npx -y @modelcontextprotocol/server-filesystem ./Desktop")
)

agent.start("Write 'hello mcp' to the file at /Users/suleymangunes/Desktop/test.txt")

```

```

(base) suleymangunes@Suleymans-MacBook-Air ~ % vim file_app.py
(base) suleymangunes@Suleymans-MacBook-Air ~ % python file_app.py
Secure MCP Filesystem Server running on stdio
Allowed directories: [ '/Users/suleymangunes/Desktop' ]
[13:17:29] INFO      [13:17:29] llm.py:795 INFO Getting response from llm.py:795
          ollama/llama3.2
Agent Info
  Agent: Agent
  Role: Assistant
  Tools: read_file, read_multiple_files, write_file, edit_file,
  create_directory, list_directory, list_directory_with_sizes,
  directory_tree, move_file, search_files, get_file_info,
  list_allowed_directories

```

```

index.js
  import { McpServer } from "@modelcontextprotocol/sdk/server/mcp";
  import { StdioServerTransport } from "@modelcontextprotocol/sdk/server/stdio";
  import { z } from "zod";

  const server = new McpServer({
    name: "ping-server",
    version: "1.0.0"
  });

  server.registerTool("pingWebsite", {
    title: "Ping Website Tool",
    description: "Sends a ping request to a given URL and returns the result.",
    inputsSchema: [url: z.string().url()],
    async ({ url }) => {
      const exec = await import("child_process");
      const host = url.replace("https://", "").split('/')[0];
      return new Promise((resolve) => {
        exec(`ping -c 2 ${host}`, {error, stdout, stderr});
        if(error) {
          resolve({ content: { type: "text", text: error.message } });
        } else {
          resolve({ content: { type: "text", text: `beni ping testi yapamam` } });
        }
      });
    },
    transport: new StdioServerTransport(),
    await server.connect(transport);
  });

```

```

ping_app_ui.py
  from prismaagents import Agent, MCP
  import gradio as gr

  def ask_mcpc_server(query):
    agent = Agent(
      instructions="You are a ping assistant. Use the tool",
      tools=MCP("node/index.js")
    )
    result = agent.start(query)
    return f"## Response\n{result}"

  demo = gr.Interface(
    fn=ask_mcpc_server,
    inputs=gr.Textbox(placeholders="example: enter a website"),
    outputs="text",
    title="Mcp Ping UI",
    description="test a website is reachability"
  )

  if __name__ == "__main__":
    demo.launch()

```

index.js:

```

import { McpServer } from "@modelcontextprotocol/sdk/server/mcp";
import { StdioServerTransport } from "@modelcontextprotocol/sdk/server/stdio";
import { z } from "zod";

const server = new McpServer({
  name: "ping-server",
  version: "1.0.0"
})

```

```

server.registerTool("pingWebsite",
{
  title: "Ping Website Tool",
  description: "Sends a ping request to a given URL and returns the result.",
  inputSchema: { url: z.string().url() }
},
async ({ url }) => {
  const { exec } = await import("child_process");

  const host = url.replace(/^https?:\/\/| /).split('/')[0];

  return new Promise((resolve) => {
    exec(`ping -c 2 ${host}`, (error, stdout, stderr) => {
      if(error) {
        resolve({ content: [{ type: "text", text: `Ping failed: ${stderr || error.message}` }]});
      } else {
        resolve({ content: [{ type: "text", text: stdout}]});
      }
    });
  });
};

const transport = new StdioServerTransport();
await server.connect(transport);

```

Model Context Protokolü (MCP) kavramını ayrıntılı şekilde açıklıyor ve MCP'nin dil modelleri ile dış dünya arasındaki etkileşim sorununa nasıl çözüm sunduğunu anlatıyor. Dil modelleri, yalnızca metin üretirken dış dünyadaki görevleri gerçekleştiremezler. Bu eksikliği gidermek için "ajanlar" geliştirilmiş, ancak bu ajanların her dil modeli için ayrı ayrı entegre edilmesi zorluluğu büyük bir sorun yaratmıştır. MCP, bu sorunu çözmek için dil modellerinden bağımsız çalışan ortak bir ara katman sunar. MCP sayesinde dil modeli, sadece metin üretmekle kalmaz; dış dünyadaki farklı görevleri (e-posta gönderme, dosya yönetimi, YouTube'a video yükleme, takvim yönetimi gibi) gerçekleştirebilir. MCP mimarisi, MCP host, MCP client ve MCP server bileşenlerinden oluşur. MCP host, dil modeli ile iletişim kurar; MCP client, host ile MCP serverlar arasında bağlantıları sağlar; MCP serverlar ise belirli görevleri gerçekleştiren araçları içerir. Video aynı zamanda MCP kullanımını bir Python ve Node.js örneği üzerinden detaylıca göstererek, MCP server yazmayı, host ve client kurulumunu ve kullanımını uygulamalı olarak anlatıyor. Sonuç olarak MCP, yapay zekanın pasif metin üreticisinden aktif görev gerçekleştiren bir araca dönüşmesini sağlıyor ve geleceğin yapay zekası için standart bir protokol olarak kabul ediliyor.

Öne Çıkan Noktalar

- 🤖 MCP, dil modellerinin dış dünya ile etkileşim kurmasını sağlayan ortak bir araç katmanıdır.
- 📩 Dil modelleri e-posta içeriği oluşturabilir ancak göndermek için MCP serverlara ihtiyaç duyar.
- 🌐 MCP mimarisi; MCP host, MCP client ve MCP server bileşenlerinden oluşur.

- 💻 MCP sayesinde dil modelleri dosya yönetimi, YouTube video yükleme, Slack mesajı gönderme gibi çok sayıda görevi yapabilir.
- 🛠 MCP serverlar bağımsızdır ve farklı dil modelleriyle uyumludur, bu da entegrasyon maliyetini düşürür.
- 🧩 Kullanıcılar kendi MCP serverlarını yazarak dil modellerine yeni özellikler kazandırabilir.
- 🔥 MCP, yapay zekanın sadece zeki değil, aynı zamanda etkin ve iş yapabilen bir yapıya dönüşmesini sağlar.

Anahtar İçgörüler

- 🌐 **Dil Modellerinin Sınırları ve Ajanların Rolü:** Dil modelleri sadece metin üretir, dış dünyaya doğrudan etkileşime geçemez. Bu nedenle ajanlar geliştirilmiş, ancak her model için özel entegrasyon gerektirmesi büyük bir zorluk yaratmıştır. Bu durum, yapay zekanın geniş çaplı kullanımını ve esnekliğini engellemiştir. MCP, bu sorunu ortadan kaldırarak, dil modellerinin dış dünyadaki görevleri bağımsız ve standart bir yolla yapabilmesini sağlamıştır.
- 📦 **MCP'nin Mimari Yapısı ve İşleyışı:** MCP, host, client ve server bileşenlerinden oluşur. Host dili modeli barındırır ve kullanıcı ile iletişim kurar. Client, host ile dış dünyadaki MCP serverlar arasında köprü görevi görür. Serverlar ise belirli görevleri gerçekleştiren araçlardır. Bu katmanlı mimari, sistemin esnekliğini ve ölçeklenebilirliğini artırır.
- 🕒 **Dil Modeli ve MCP Serverlar Arasındaki Etkileşim:** Dil modeli, gerçekleştiremediği bir görevi MCP serverlar aracılığıyla tamamlar. MCP serverların açıklamaları, modelin hangi aracı kullanacağını anlamasını sağlar. Model, uygun MCP serverı seçip gerekli bilgileri ileter, işlem tamamlandıktan sonra sonucu kullanıcıya bildirir. Bu etkileşim sayesinde yapay zeka sadece cevap vermekten çıkar, aktif görev yapan bir sisteme dönüşür.
- 💡 **MCP'nin Çoklu Model ve Çoklu Araç Uyumluluğu:** MCP protokolü, dil modellerinden bağımsızdır. Yani herhangi bir MCP server, herhangi bir MCP host ve dolayısıyla herhangi bir dil modeli ile çalışabilir. Bu, farklı yapay zeka sistemlerinin ve araçlarının kolayca birbiriyile entegre olmasını sağlar ve ekosistemin büyümесini hızlandırır.
- 🛠 **Pratik MCP Kullanımı ve Geliştirme Süreci:** Video, MCP'nin pratikte nasıl kurulup kullanılacağını adım adım gösteriyor. Örnek olarak masaüstünde dosya oluşturma işlemi, Python ve Node.js kullanarak MCP host ve server geliştirme süreçleri detaylıca anlatılıyor. Bu, geliştiricilere MCP sistemini kendi projelerinde nasıl uygulayacakları konusunda yol gösteriyor.
- 🚀 **Geleceğin Yapay Zekası: Etkin ve Görev Odaklı:** MCP sayesinde yapay zeka sadece bilgi üretmekle kalmayacak, aynı zamanda gerçek dünyadaki görevleri yerine getirebilecek. Örneğin tatil planlaması, bilet ve otel rezervasyonu, takvime ekleme gibi çok aşamalı işlerde yapay zekanın aktif rolü mümkün olacak. Bu da kullanıcı deneyimini ve yapay zekanın kullanım alanlarını ciddi şekilde genişletecek.
- 🌐 **Açık Kaynak ve Topluluk Desteği:** MCP, açık kaynaklı bir protokol olarak geliştiriliyor ve yüzlerce farklı MCP server topluluk tarafından paylaşılıyor. Bu, kullanıcıların ve geliştiricilerin kendi ihtiyaçlarına göre yeni MCP serverlar oluşturmasına olanak tanıyor. Topluluk desteği ile MCP ekosistemi hızla büyuyerek yapay zekanın demokratikleşmesini sağlıyor.

Detaylı İnceleme

Video, öncelikle dil modellerinin dış dünya ile doğrudan etkileşim kuramama sorununu vurgulayarak başlıyor. Bu eksiklik, metin bazlı yapay zekanın gerçek anlamda iş yapabilmesini engelliyor. Dil modeli bir e-posta yazabilir ancak gönderemez, bir hava durumu sorusuna yanıt verir ancak güncel veriyi kendisi çekemez. Bu ihtiyaç, "ajanlar" adı verilen yardımcı sistemlerin geliştirilmesine yol açtı. Ancak ajanların her dil modeli için ayrı entegrasyon gerektirmesi, maliyet ve karmaşıklığı artırdı.

MCP, bu sorunları çözmek için ortaya çıktı. MCP, dil modeli ile dış dünya araçları arasında ortak bir ara katman sunar. MCP host, kullanıcı ile dil modeli arasındaki iletişimini sağlar; MCP client, host ile MCP serverlar arasında güvenli bağlantı kurar; MCP serverlar ise belirli görevleri gerçekleştiren bağımsız araçlardır. Dil modeli, MCP serverlarının açıklamalarına bakarak hangi aracın hangi görevi yapabileceğini anlar ve gerektiğinde bu araçları kullanır. Böylece görevler dil modeli tarafından değil, MCP serverlar tarafından gerçekleştirilir.

Video, MCP'nin pratik kullanımını da örneklerle destekliyor. Örneğin, Python'da bir MCP host oluşturulup, Node.js kullanılarak bir MCP server yazılıyor. Bu MCP server, bir web sitesine ping atarak bağlantı hızını ve IP adresini tespit ediyor. Dil modeli, bu görevi gerçekleştiremediği için MCP serverdan yardım alıyor. Kullanıcı arayüzü Gradio ile oluşturuluyor ve kullanıcından gelen komutlar MCP host üzerinden işlenerek MCP servera iletiliyor. MCP serverdan gelen sonuç tekrar dil modeline iletilip kullanıcıya sunuluyor.

Bu örnek, MCP'nin nasıl esnek, ölçeklenebilir ve güçlü bir sistem olduğunu gösteriyor. MCP'nin standartlaşması, farklı dil modelleri ve araçların kolayca entegre edilmesini sağlıyor. Ayrıca MCP sayesinde yapay zekanın kullanım alanları büyük ölçüde genişliyor ve gerçek dünya ile etkileşim kapasitesi artıyor.

Son olarak, video MCP'nin gelecekte yapay zeka deneyimini nasıl dönüştüreceğine dair vizyon sunuyor. Tatil planlamadan video düzenlemeye, sunum hazırlamadan sosyal medya yönetimine kadar pek çok alanda MCP destekli yapay zekalar aktif görev yapacak. Bu da yapay zekanın sadece zeki değil, aynı zamanda etkin ve pratik bir yardımcı haline gelmesini sağlayacak.

Sonuç

MCP, dil modellerinin dış dünya ile etkileşim kurma yeteneğini sağlayan devrim niteliğinde bir protokoldür. MCP sayesinde yapay zekalar, sadece metin üretmekte kalmayıp gerçek görevleri yerine getirebilen, genişletilebilir ve standart bir sisteme dönüşmüştür. MCP'nin mimarisi, geliştiricilere esneklik sunar ve yapay zekanın gelecekte çok daha etkin, işlevsel ve kullanıcı dostu olmasını mümkün kılar. Bu teknoloji, yapay zekanın kullanım alanlarını genişletirken, kullanıcıların hayatını kolaylaştıracak pek çok yeniliğin kapısını aralamaktadır.