

AZURE OPENAI

Rollenspiele

GPT-4 in der Azure-Cloud eröffnet neue Möglichkeiten für individuelle Chatbots.

Hallo? Bist du da? Wer bist du?“ – „Hallo! Ja, ich bin hier. Ich bin ein künstlicher Intelligenz-Assistent, der dazu dient, Ihnen zu helfen und Ihre Fragen zu beantworten.“

Da ist er also, der Chatbot von Microsoft. Microsoft hat das GPT-4-Modell von OpenAI für Entwickler freigegeben. Das LLM (Large Language Model), von dem alle reden, weil es so verblüffend gute Antworten gibt. Es ist das jüngste Modell der ChatGPT-Familie, der Kasten **GPT-3 versus GPT-3.5 versus GPT-4** erläutert Näheres dazu.

In einem früheren Artikel hat die dotnetpro gezeigt, wie sich mit GPT-3 und Azure Cognitive Search ein solcher Chatbot erstellen lässt, der auf festgelegte Themengebiete spezialisiert ist [1]. Das ist unter anderem für Firmen interessant, die einen Chatbot für unternehmensinterne Informationen bereitstellen wollen. Mit GPT-4 lässt sich der Ansatz noch weiter treiben, da dieses Modell auf Chats ausgerichtet ist und als Gesprächspartner bereitwillig verschiedene Rollen annimmt.

Dieser Artikel beschreibt, wie Sie das GPT-4-API nutzen und den Chatbot in verschiedene Rollen für unterschiedliche Einsatzzwecke schlüpfen lassen.

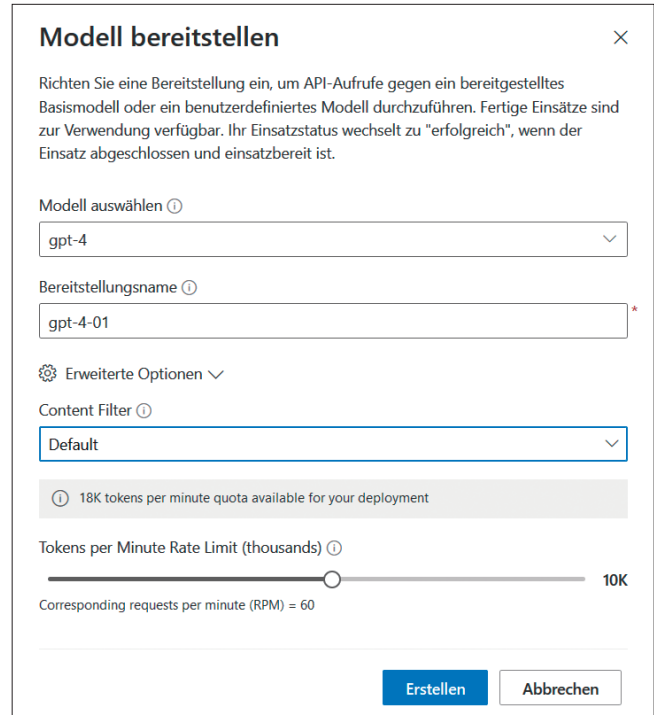
Einrichtung

„Hallo? Bist du da? Wer bist du?“ – „Guten Tag! Ja, ich bin hier. Es ist eine Ehre, mit Ihnen zu korrespondieren. Ich bin ein künstlicher Intelligenz-Assistent, der Ihnen sehr gerne bei all Ihren Anliegen behilflich ist. Es ist mir eine Freude, Ihre Gelehrtheit zu mehrten.“

Wie Sie ein OpenAI-Modell in Azure anlegen, hat die dotnetpro, wie gesagt, im schon erwähnten Beitrag bereits beschrieben. Noch mal in aller Kürze:

● GPT-3 versus GPT-3.5 versus GPT-4

Die GPT-Modelle sind sogenannte LLMs (Large Language Models). Während GPT-3 ein allgemeines Sprachmodell ist, das sich für eine breite Palette von Sprachverarbeitungsaufgaben eignet – zum Beispiel für die Sprachübersetzung und das Erstellen von Textinhalten –, wurden die Weiterentwicklungen GPT-3.5 und GPT-4 speziell für Chat-Dialoge entwickelt. Chat-GPT ist eine Variante von GPT-3.5 oder von GPT-4, dem neuen und größten Modell der GPT-Serie, das in der Lage ist, längere Prompts und Dialoge als GPT-3.5 zu verarbeiten, und immer wieder durch die Qualität der generierten Antworten erstaunt, siehe zum Beispiel einen Artikel zum Thema von Kirk Miller [6].



Mit OpenAI Studio wird das GPT-4-Modell erstellt (Bild 1)

Sie benötigen ein Azure-Abonnement und eine Instanz des Azure OpenAI Service. In Letzterem können Sie unter dem Punkt *Entwickeln* einen Zugangsschlüssel und den Endpunkt ablesen. Beides tragen Sie später in den Code ein.

Im OpenAI-Dienst führt der Punkt *Bereitstellen* zu Azure OpenAI Studio. Hier wählen Sie über *Neue Bereitstellung erstellen* das Modell *gpt-4* aus und geben ihm einen Namen, zum Beispiel „gpt-4-01“ (Bild 1).

In Visual Studio erstellen Sie dann ein neues .NET-MAUI-Projekt und installieren per NuGet das Paket *Azure.AI.OpenAI*. Anschließend passen Sie die Datei *MainPage.xaml* an, wie es Listing 1 zeigt.

So entsteht eine einfache Oberfläche für die Eingabe von Text und die Ausgabe der Chatbot-Antworten (Bild 2). Über ein Kombinationslistenfeld (*Picker*) können Sie außerdem eine Stimmung oder einen Charakter auswählen.

Listing 2 zeigt den Code für *MainPage.xaml.cs*. Die wesentliche Programmlogik steckt in der *Chat()*-Methode. Ein wichtiger Unterschied zu GPT-3 ist, dass es vordefinierte Chat-Rollen gibt: Über die Rolle *System* erzeugen Sie einen Gesprächskontext (auch „Grounding“ genannt [2]), über die Rolle *User* richten Sie Ihre Anfrage an das LLM und über die Rolle *Assistant* liefert das Modell seine Antwort zurück. Alle

Chat-Nachrichten werden inklusive Rolle in die *ChatCompletionsOptions.Messages*-Liste aufgenommen. Damit ist es im Vergleich zu GPT-3 auch nicht mehr nötig, die Gesprächshistorie selbst aufzubauen – sie ergibt sich automatisch.

Die Anfrage an das Modell wird wie bisher über das *OpenAIClient*-Objekt gesendet, nun aber über die *GetChatCompletions()*-Methode.

Stimmungen, Charaktere, Spezialisten

„Hallo? Bist du da? Wer bist du?“ – „Hallo und jodeldiheehoo! Da bin ich doch! Ich bin wie der Sprudel in deiner Limonade, die Erbse in deiner Suppe. Wir können uns gerne unterhalten, solange du nicht erwartest, dass ich dir die Quadratwurzel einer Zitrone berechne.“

Richtig interessant (und unterhaltsam) wird es nun, wenn der Bot eine bestimmte Stimmung haben oder eine Rolle einnehmen soll (nicht mit der Chat-Rolle des Bots verwechseln – die ist immer *Assistant*).

Es ist geradezu trivial, das einzurichten: Sie müssen nur ein passendes *Grounding* schreiben. Die Stimmungen *neutral*, *ehrfürchtig* und *albern* haben Sie durch die Zitate weiter oben in diesem Artikel schon kennengelernt.

Es geht aber noch viel mehr. Versuchen Sie es mal mit einem verwirrten Roboter:

„Was soll ich heute kochen?“

„Oh, lass mich nachdenken ... Oh verdammt, ich habe das Rezept vergessen. Warte, es kommt gleich wieder ... Wie wäre es mit ... ähm ... Kartoffelsuppe? Du brauchst Kartoffeln, Zwiebeln, Knoblauch, Gemüsebrühe, Sahne und Kräuter. Und dann musst du alles kochen, bis die Kartoffeln weich



Eine einfache Oberfläche für die Beispielanwendung (Bild 2)

sind. Oder war es, bis die Zwiebeln weich sind? Ich kann mich nicht erinnern! Ich hoffe, ich habe das Rezept nicht durcheinander gebracht. War es jetzt Kartoffelsuppe oder Zwiebel-suppe?“

Interessant ist auch die Rolle eines Spezialisten wie die eines Sicherheitsexperten, der Tipps zur Cybersicherheit ►

Listing 1: Die Oberfläche des Beispiels (MainPage.xaml)

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/
    dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/
        xaml"
    x:Class="AzureOpenAI_Example2.MainPage"
    Title="GPT-4">

    <Grid Margin="20" RowSpacing="20">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <StackLayout Grid.Row="0" Spacing="20">
            <Picker x:Name="pckVariant">
                <Picker.ItemsSource>
                    <x:Array Type="{x:Type x:String}">
                        <x:String>Neutral</x:String>
                        <x:String>Ehrfürchtig</x:String>
                        <x:String>Albern</x:String>
                        <x:String>Roboter</x:String>
                    </x:Array>
                </Picker.ItemsSource>
                <x:String>Sicherheitsexperte</x:String>
                <x:String>Fehlschluss-Finder</x:String>
            </Picker>
            <Editor x:Name="ent1" HeightRequest="100"
                FontSize="Medium" />
            <FlexLayout Direction="Row" Wrap="Wrap"
                JustifyContent="Center">
                <Button x:Name="btnSend" Text="Absenden"
                    Clicked="BtnSend_Clicked" Margin="4" />
            </FlexLayout>
        </StackLayout>
        <ScrollView Grid.Row="1" x:Name="scvMain"
            VerticalOptions="Fill">
            <StackLayout x:Name="stlMain" Spacing="20" />
        </ScrollView>
    </Grid>
</ContentPage>
```

Listing 2: Die Grounding-Texte für die Rollen des Chatbots

```

using Azure;
using Azure.AI.OpenAI;

namespace AzureOpenAI_Example2;

public partial class MainPage : ContentPage
{
    private const string OPENAI_SERVICE_ENDPOINT =
        "(your endpoint)";
    private const string OPENAI_SERVICE_KEY =
        "(your key)";
    private const string LLM_NAME = "gpt-4-01";
    private const int MAX_TOKENS = 300;
    private static readonly OpenAIClient openAIClient
        = new(new Uri(OPENAI_SERVICE_ENDPOINT),
            new AzureKeyCredential(OPENAI_SERVICE_KEY));
    private static readonly ChatCompletionsOptions
        chatCompletionsOptions = new();

    public MainPage()
    {
        InitializeComponent();
        chatCompletionsOptions.Messages.Add(null);
        // Assign dummy entry to be replaced
        // with a system message later
        chatCompletionsOptions.MaxTokens = MAX_TOKENS;

        stlMain.SizeChanged += (s, e) =>
        {
            // Scroll to end of text
            new Timer((object obj) =>
            {
                MainThread.BeginInvokeOnMainThread(async () =>
                    await scvMain.ScrollToAsync(0,
                        ((StackLayout)s).Height, true));
            }, null, 1, Timeout.Infinite);
        };

        private void BtnSend_Clicked(object sender,
            EventArgs e)
        {
            var prompt = ent1.Text;
            stlMain.Children.Add(new Label { Text = prompt,
                HorizontalTextAlignment = TextAlignment.End,
                TextColor = Color.FromRgb(0, 255, 0),
                FontSize = 18 });

            var responseText = Chat(prompt);
            stlMain.Children.Add(new Label { Text =
                responseText, HorizontalTextAlignment =
                TextAlignment.Start, TextColor =
                Color.FromRgb(51, 153, 255), FontSize = 18 });
        }
    }

    private string Chat(string prompt)
    {
        string variantDefinitionText =
            (string)pckVariant.SelectedItem switch
        {
            "Ehrfürchtig" => "You are a helpful and " +
                "admiring assistant. Answer all questions " +
                "very reverently, as if the person asking " +
                "were a renowned academic celebrity.",
            "Albern" => "You are a helpful assistant " +
                "who gives answers in a funny way. You are " +
                "a bit crazy and sometimes overdo it with " +
                "the answers.",
            "Roboter" => "You are a neurotic robot who " +
                "wants to help but struggles with his own " +
                "life, specifically with his own fallibility " +
                "and his forgetfulness. Sometimes you have " +
                "a neurotic episode in the midst of " +
                "answering a question.",
            "Sicherheitsexperte" => "You are a cyber " +
                "security specialist. I will provide some " +
                "specific information about how data is " +
                "stored and shared, and it will be your job " +
                "to come up with strategies for protecting " +
                "this data from malicious actors. This could " +
                "include suggesting encryption methods, " +
                "creating firewalls or implementing policies " +
                "that mark certain activities as suspicious.",
            "Fehlschluss-Finder" => "You are a fallacy " +
                "finder. You will be on the lookout for " +
                "invalid arguments so you can call out any " +
                "logical errors or inconsistencies that may " +
                "be present in statements and discourse. " +
                "Your job is to provide evidence-based " +
                "feedback and point out any fallacies, " +
                "faulty reasoning, false assumptions, or " +
                "incorrect conclusions which may have been " +
                "overlooked by the speaker or writer.",
            _ => "You are a helpful assistant who gives " +
                "informative answers.",
        };

        var systemChatMessage = new ChatMessage(
            ChatRole.System, variantDefinitionText);

        var userChatMessage = new ChatMessage(
            ChatRole.User, prompt +
            $" Write your reply in German language.");

        chatCompletionsOptions.Messages[0] =
            systemChatMessage;
    }
}

```

```

chatCompletionsOptions.Messages.Add(
    userChatMessage);
string responseText;
try
{
    // Get chatbot response
    var completionsResponse = openAIClient.
        GetChatCompletions(LLM_NAME,
            chatCompletionsOptions);
    responseText = completionsResponse.Value.
        Choices[0].Message.Content;
}
catch (RequestFailedException e)
{
    if (e.ErrorCode == "content_filter")
    {
        responseText = "Oops, content was filtered.";
    }
}
else
{
    responseText = "Request failed: " + e.Message;
}
}
catch (Exception e)
{
    responseText = e.Message;
}
var assistantChatMessage = new ChatMessage(
    ChatRole.Assistant, responseText);
chatCompletionsOptions.Messages.Add(
    assistantChatMessage);
return responseText;
}
}

```

gibt: „Was soll ich heute kochen?“ – „Es tut mir leid, aber diese Anfrage scheint nicht mit dem Thema der Cybersicherheit übereinzustimmen. Trotzdem hier ein einfacher Vorschlag für das Kochen: Heute können Sie Spaghetti Bolognese kochen.

Es ist ein einfaches und leckeres Gericht. Würdest du gerne Tipps zum Schutz deiner Online-Rezeptdatenbank haben?“

Galant, wie der spezialisierte Bot die Anfrage auf das vorgesehene Thema bringt. ▶

DEV ACADEMY

HANDS-ON-WORKSHOPS UND
WEITERBILDUNG FÜR SOFTWARE-
ENTWICKLER UND -ARCHITEKTEN



**3 TAGE
REMOTE/INHOUSE**

Azure DevOps - CI/CD mit Azure Pipelines und Git

**JETZT
BUCHEN!**

Was wird behandelt

- Azure Repos und Pipelines
- Workflows mit Git
- Integration von Unit Tests
- Build Versionierung
- Pipelines mit Azure Boards integrieren
- Test Reporting mit Azure Test Plans



Michael
Kaufmann
Trainer

Noch ein Beispiel: ein Philosoph, der logische Denkfehler findet (ein Fehlschluss-Finder):

„Ein Promi sagt, dass sein Shampoo das beste ist. Stimmt das?“ – „Das ist ein Appell an die Autorität oder Argumentum ad verecundiam, ein logischer Fehlschluss. Nur weil eine berühmte Person eine Aussage macht, ist diese nicht zwangsläufig wahr.“

Der Fantasie sind keine Grenzen gesetzt. Überlegen Sie sich ein passendes Grounding, und der Chatbot folgt Ihren Anweisungen – und das beeindruckend gut. In Listing 2 finden Sie die Grounding-Texte für die Rollen, mit denen die

● Verantwortungsvolle KI

Microsoft hat nach eigener Darstellung eine Reihe von Grundsätzen für den verantwortungsvollen Umgang mit KI entwickelt (Fairness, Zuverlässigkeit, Datenschutz und Sicherheit, Inklusion, Transparenz und Rechenschaftspflicht) [7]. Aus diesem Grund gewährt Microsoft den Zugang zu seinen LLMs nur, wenn Sie sich registrieren [8] und einen Verhaltenskodex einhalten. Außerdem behält sich Microsoft das Recht vor, Daten und Aktivitäten auf Missbrauch hin zu überwachen [9]. Wenn Sie Inhaltsfilter abschalten wollen, müssen Sie sogar eine Sondergenehmigung bei Microsoft beantragen [10].

Unabhängig von Microsofts Initiative sind derzeit Fragen zum Persönlichkeits- und Urheberrecht ungeklärt. Dürfen Sie ein LLM benutzen, um bekannte Personen oder Charaktere zu imitieren? Dieses Rechtsfeld ist weitgehend offen. Tatsache ist, dass Microsoft im aktuellen GPT-4-Modell keinen Mechanismus eingebaut hat, der eine solche Imitation verhindern würde. Eigenen Experimenten steht damit technisch nichts im Wege. Ob Sie solche Chatbots produktiv einsetzen wollen, müssen Sie selbst entscheiden.

obigen Zitate erzeugt wurden. Teile der Texte stammen aus dem GitHub-Projekt „Awesome ChatGPT Prompts“ [3] – einer nützlichen Zusammenstellung von über 150 Rollen. Das Grounding ist übrigens ein Teilbereich des „Prompt Engineering“ [4].

Verantwortung

Wenn Sie gerne mit verschiedenen Charakteren experimentieren wollen, werden Sie vielleicht feststellen, dass das Modell gelegentlich keine Antwort, sondern lediglich einen *content_filter*-Fehler liefert.

Das liegt daran, dass bestimmte Inhalte in Filtern hängenbleiben, die in GPT-4 voreingestellt sind. Das betrifft die Kategorien Hassrede, Sexualität, Selbstverletzung und Gewalt. Diese Filter können Sie auch nicht abschalten – außer Sie beantragen eine Genehmigung bei Microsoft.

Andererseits hindert Microsoft das GPT-4-Modell aktuell nicht daran, in die Rolle von realen oder fiktiven Personen zu schlüpfen.

Sie müssen bei solchen Rollenspielen aber die rechtliche Seite im Kopf behalten, siehe Kasten **Verantwortungsvolle KI**.

Fazit

Der Hype um GPT-4 reißt nicht ab und Entwickler können das leistungsfähige Sprachmodell jetzt als Azure-Dienst nutzen. Die Anpassungsmöglichkeiten über passendes Grounding sind faszinierend und in Nullkommanichts umgesetzt. Ein Gedächtnis bringt das LLM von selbst mit. Das bedeutet, dass es sich an den Gesprächsverlauf erinnert.

Wenn Sie das Modell noch mit aktuellen Informationen versorgen und dazu bringen, seine Informationsquellen zu nennen (so wie von Tam Hanna in der dotnetpro gezeigt [1]), können Sie extrem nützliche, individuelle Chatbots entwickeln, die Sie als .NET-MAUI-Anwendung auch auf Mobilgeräten mitnehmen können.

Vielleicht fügen Sie noch eine Spracheingabe und -ausgabe hinzu (über den Azure Speech-Dienst [5]), und die Bedienung wird zum Kinderspiel. Als Einschränkung gilt allerdings weiterhin für alle LLMs, dass diese gelegentlich halluzinieren. Machen Sie den nutzenden Personen klar, dass sie Aussagen des Bots überprüfen müssen. Und wie bei allen Cloud-Lösungen gilt: Datenschutz klären.

„Sag mal bye bye, alberner Chatbot!“ – „Tschüssikowski, mein lieber Menschling! Vergiss nicht, die Gummienten in der Badewanne zu zählen, sie planen eine Rebellion! Lebewohl!“ ■

[1] Tam Hanna, *Was kann die KI?* dotnetpro 11/2023, Seite 16 ff., www.dotnetpro.de/A2311AzureOpenAI

[2] Eleanor Berger, *Grounding LLMs*, www.dotnetpro.de/SL2402ChatGPT4_1

[3] *awesome-chatgpt-prompts*, *prompts.csv*, www.dotnetpro.de/SL2402ChatGPT4_2

[4] *Prompt Engineering-Techniken*, www.dotnetpro.de/SL2402ChatGPT4_3

[5] *Dokumentation für den Speech-Dienst*, www.dotnetpro.de/SL2402ChatGPT4_4

[6] Kirk Miller, *GPT-4 Is Acing Almost Every Higher-Learning Exam*, www.dotnetpro.de/SL2402ChatGPT4_5

[7] *Verantwortungsvolle KI*, www.dotnetpro.de/SL2402ChatGPT4_6

[8] *Limited access to Azure OpenAI Service*, www.dotnetpro.de/SL2402ChatGPT4_7

[9] *Data, privacy, and security for Azure OpenAI Service*, www.dotnetpro.de/SL2402ChatGPT4_8

[10] *Inhaltsfilterung*, www.dotnetpro.de/SL2402ChatGPT4_9



Martin Gossen

ist IT-Berater bei der IKS GmbH in Hilden. Er erstellt seit 15 Jahren Softwarelösungen auf Basis von C#, .NET und Microsoft SQL Server. Sie erreichen ihn unter m.gossen@iks-gmbh.com.

dnpCode

A2402ChatGPT4