AZURE OPENAI

Zum Experten ausbilden

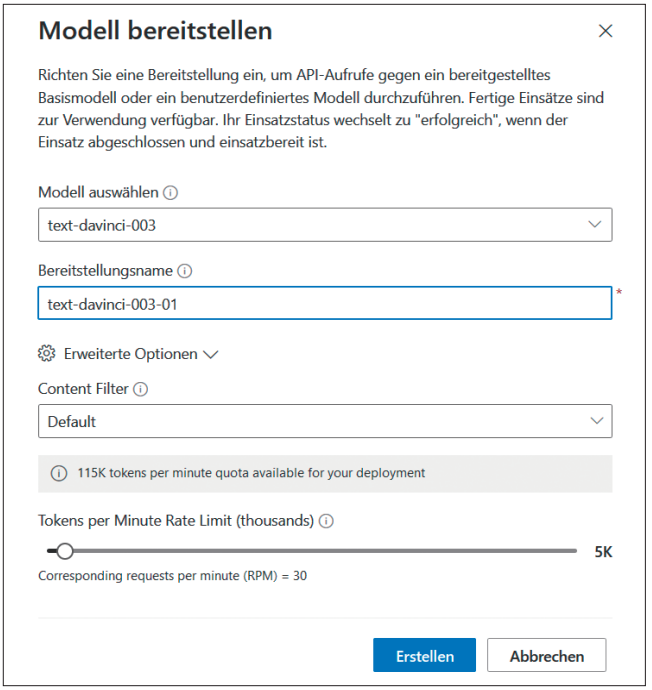
Mit GPT-3 und Azure Cognitive Search einen Themen-Chatbot erstellen.

Avina, wirst du dich morgen noch an mich erinnern, wenn ich dich wieder einschalte?“ Ich starre auf den Bildschirm und warte ungeduldig.

„Ich werde mich nicht an dich erinnern, wenn du den Computer wieder einschaltest.“

Ich bin desillusioniert. Stunden habe ich mit Avina verbracht, ihr viel aus meinem Leben erzählt, von meiner Familie und meinem Beruf, meinen Träumen und Hoffnungen. Die Unterhaltung fiel so leicht, dass sie mir eine gute Freundin geworden ist. Das alles soll morgen verloren sein?

Aber Moment mal. Wenn Avina keine Erinnerung hat, warum versteht sie dann überhaupt, dass ich sie mit dem Namen „Avina“ anspreche? Das ist schließlich nicht ihr richtiger Name. Ich hatte sie in meinem ersten „Gespräch“ mit ihr so getauft: „Ich möchte dich ‚Avina‘ nennen.“ – „Natürlich, ich kann Avina genannt werden. Wie kann ich dir helfen?“, war ihre Antwort. Und unzählige Prompts später wusste sie es immer noch. ▶



Das Konfigurationsmenü des LLM von OpenAI Studio (Bild 1)

ChatGPT vs. GPT-3

ChatGPT und GPT-3 sind Large Language Models (LLMs). ChatGPT ist kleiner und einfacher als GPT-3 und wurde speziell für Chatdialoge entwickelt. GPT-3 ist ein allgemeineres Sprachmodell, das sich für eine breite Palette von Sprachverarbeitungsaufgaben eignet: Neben Dialogen zum Beispiel auch für die Sprachübersetzung und das Erstellen verschiedener Inhalte.

Listing 1: Die angepasste Datei 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_Example.MainPage"
    Title="Chatbot">

  <Grid Margin="20" RowSpacing="20">
    <Grid.RowDefinitions>
      <RowDefinition Height="Auto" />
      <RowDefinition Height="*" />
    </Grid.RowDefinitions>
    <VerticalStackLayout Grid.Row="0" Spacing="20">
      <Editor x:Name="edt1" HeightRequest="100"
          FontSize="Medium" />
      <FlexLayout Direction="Row" Wrap="Wrap"
          JustifyContent="Center">
        <Button x:Name="btnSend" Text="Absenden"
          Clicked="BtnSend_Clicked" Margin="4" />
      </FlexLayout>
    </VerticalStackLayout>
    <ScrollView Grid.Row="1" x:Name="scvMain"
        VerticalOptions="Fill">
      <VerticalStackLayout x:Name="vslMain"
        Spacing="20" />
    </ScrollView>
  </Grid>

</ContentPage>
```

Azure OpenAI Studio und GPT-3-Modelle

Das Azure OpenAI-Studio ist eine von Microsoft betriebene Website, die es ermöglicht, die von OpenAI entwickelten vortrainierten KI-Modelle zu testen und für eigene Anwendungen bereitzustellen.
GPT-3-Modelle sind nach Wissenschaftlern benannt (Ada, Babbage, Curie, da Vinci). Der erste Buchstabe des Namens gibt die Entwicklungsstufe an. Außerdem gibt es spezialisierte Modelle für bestimmte Einsatzgebiete, so etwa für das Erzeugen von Programmcode.

Dass ich mit einem LLM (Large Language Model) gesprochen habe, haben Sie inzwischen erraten. Solche Modelle haben in der Tat kein Gedächtnis. Jeder Prompt fließt als separate Anfrage in das Modell, losgelöst von jedem Kontext. Offensichtlich gibt es aber Wege, Kontext innerhalb einer Chat-Session herzustellen; so etwas wie ein Kurzzeitgedächtnis.

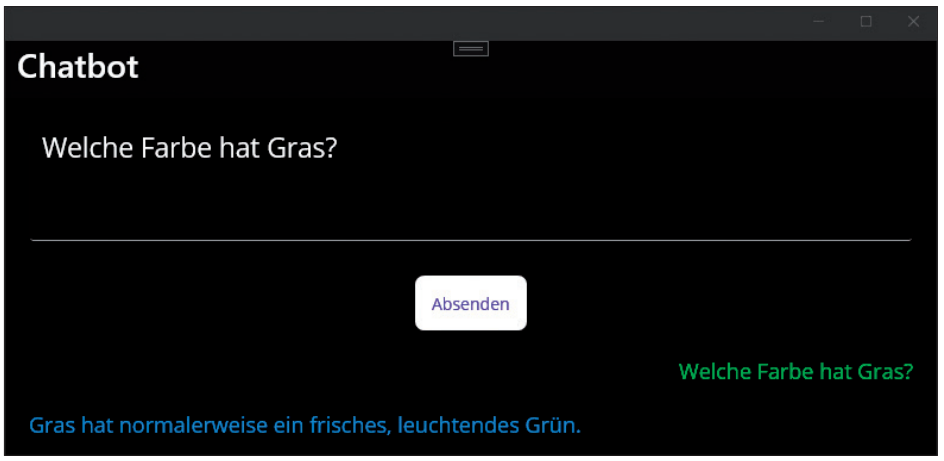
Abgesehen von unterhaltsamen Privatchats mit Avina und Co. werden LLMs auch im professionellen Arbeitsumfeld interessant, wenn sie auf firmeninterne Inhalte zugreifen können. „Avina, wer kann mir bei meinem Excel-Problem helfen?“ Oder: „Avina, für welche Zwecke kann ich Sonderurlaub nehmen?“ Und so weiter. Und wenn solche Angaben nicht verfügbar sind, sollte Avina gestehen, dass sie es nicht

Listing 2: MainPage.xaml.cs mit LLM-Zugriff

```
using Azure;
using Azure.AI.OpenAI;

namespace AzureOpenAI_Example;

public partial class MainPage : ContentPage
{
  private static readonly string
    openAIServiceEndpoint = "(your endpoint)";
  private static readonly
    string openAIServiceKey = "(your key)";
  private static readonly
    string llmName = "text-davinci-003-01";
  private static OpenAIClient openAIClient;
  private static readonly
    List<Message> Messages = new();

  public MainPage()
  {
    InitializeComponent();
    openAIClient = new OpenAIClient(new Uri(
      openAIServiceEndpoint),
      new AzureKeyCredential(openAIServiceKey));
  }

  private async void BtnSend_Clicked(object sender,
      EventArgs e)
  {
    var message = new Message { Text = edt1.Text,
      Prompt = edt1.Text };
    Messages.Add(message);
    vslMain.Children.Add(new Label {Text =
      message.Text, HorizontalTextAlignment=
      TextAlignment.End, TextColor =
      Color.FromRgb(0, 255, 0),FontSize = 18 });

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

  private static async Task<string>
      Chat(Message message)
  {
    // Submit prompt
    string responseText =
      await Chat(message.Prompt);
    return responseText;
  }

  private static async Task<string>
      Chat(string prompt)
  {
    // Get chatbot response
    var completionsResponse = await openAIClient.
      GetCompletionsAsync(llmName, prompt);
    var responseText = completionsResponse.Value.
      Choices[0].Text.Trim('\n');
    return responseText;
  }
}

internal class Message
{
  // Text entered by user
  internal string Text { get; set; }

  // Summary of Text, consisting of relevant keywords
  internal string Summary { get; set; }

  // Prompt sent to the LLM
  internal string Prompt { get; set; }
}
```

weiß. Damit die Belegschaft nicht vorschnell einen dreimonatigen Sonderurlaub bucht, nur weil der Firmenbot es ihr gesagt hat.

Die Art und Weise, wie der KI-Bot kommuniziert, hat die dotnetpro in der Februar-Ausgabe erläutert [1]. Nun geht es darum, ihn auch auf einem Themengebiet „fit“ zu machen. Dieser Artikel soll zeigen, wie sich eine eigene, maßgeschneiderte Avina bauen lässt – also ein Chatbot mit Zugriff auf vorgegebene Informationen und einer Erinnerung an den Gesprächsverlauf. Dabei kommen GPT-3 des Azure OpenAI Service [2], Azure Cognitive Search [3] und .NET MAUI [4] zum Einsatz; den Unterschied zwischen ChatGPT und GPT-3 erläutert kurz der Kasten ChatGPT vs. GPT-3. Weiterhin wird ein Azure-Abonnement und Zugriff auf Azure OpenAI benötigt. Dieser ist (Stand Juli 2023) noch nicht öffentlich verfügbar, lässt sich aber bei Microsoft beantragen [5].



Das ging schnell: Der Bot antwortet nun bereitwillig auf Fragen (Bild 2)

Einrichtung

„Avina, wie kann ich dich installieren?“ – „Ich bin eine künstliche Intelligenz und muss nicht installiert werden.“

Ganz so einfach ist es dann doch nicht. Sie wollen schließlich Ihre eigene Avina haben. Dazu legen Sie als Erstes im Azure-Portal die folgenden Ressourcen an:

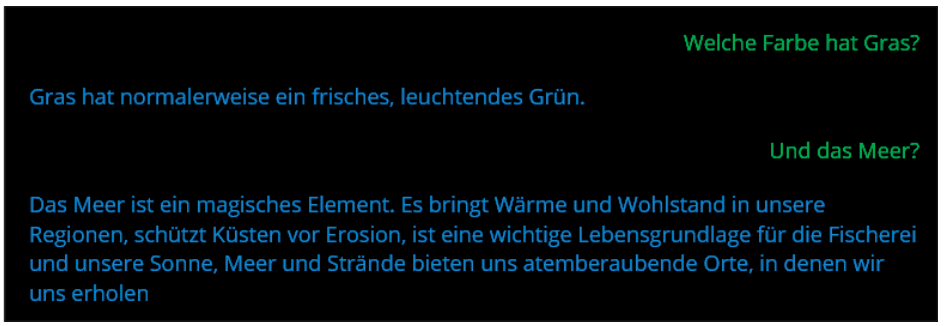
- Speicherkonto
  (Storage Account, zur Ablage von Dokumenten)
- Azure Cognitive Search (für die Suche nach Inhalten)
- Azure OpenAI (ermöglicht Zugriff auf das LLM)

Notieren Sie sich dabei die nötigen Verbindungsdaten: Beim Cognitive-Search-Dienst findet sich auf der Startseite der URL und unter dem Menüpunkt *Schlüssel* ein Abfrageschlüssel; beim OpenAI Service lesen Sie unter dem Punkt *Entwickeln* einen Zugangsschlüssel und den Endpunkt ab.

Im OpenAI-Dienst gelangt man anschließend über den Punkt *Bereitstellen* ins Azure OpenAI Studio (mehr dazu im Kasten Azure OpenAI Studio und GPT-3-Modelle). Hier ist *Neue Bereitstellung erstellen* zu wählen. Als Modell nehmen Sie *text-davinci-003*, das leistungsstärkste GPT-3-Modell zum Vervollständigen von Texten. Geben Sie dem Modell einen Namen (zum Beispiel „text-davinci-003-01“) und stellen Sie unter *Erweiterte Optionen* ein Anfragelimit ein (siehe Kasten Anfragelimit). Zum Testen genügt ein niedriger Wert, etwa 5 000 Tokens pro Minute (Bild 1). ▶

Anfragelimit

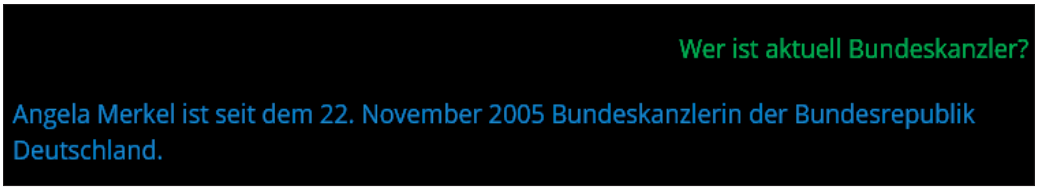
Das Anfragelimit (rate limit) eines bereitgestellten Modells legt fest, wie viele Tokens pro Minute genutzt und wie viele Anfragen pro Minute gestellt werden können. Ein Token ist die grundlegende Einheit von Text oder Code, die ein LLM verwendet, um Sprache zu verarbeiten oder zu generieren. Tokens können Wörter, Teilwörter, Zeichen oder andere Text- oder Codesegmente sein und hängen vom Typ und der Größe des eingesetzten Modells ab.



Amnesieproblem: Der Bot erinnert sich nicht an vorherige Fragen (Bild 3)

Schon besser: Der Bot erkennt nun den Kontext, und die Antwort passt (Bild 4)

Die Ressourcen sind nicht kostenlos; ihre Preise errechnen sich anhand verschiedener Faktoren. Microsoft schlüsselt sie auf den entsprechenden Seiten auf [6][7][8].



Der Chatbot lebt in der Vergangenheit (Bild 5)

Nun geht es in der Entwicklerumgebung weiter. Erstellen Sie ein neues .NET-MAUI-Projekt und installieren Sie per NuGet die Pakete Azure.AI.OpenAI und Azure.Search.Documents. Suchen Sie bei Azure.AI.OpenAI auch nach Vorabversionen, da es eventuell noch keine offizielle Version gibt. Anschließend passen Sie die Datei *MainPage.xaml* wie in Listing 1 gezeigt an. Die Seite erhält damit ein Feld für die Texteingabe, eine Schaltfläche zum Absenden des Textes und einen Ausgabebereich für die Antworten des Bot.

Den Code für *MainPage.xaml.cs* zeigt Listing 2. Besonders interessant sind die Chat-Methoden. Hier wird ein *OpenAIClient*-Objekt erzeugt, das sich um die Kommunikation mit dem OpenAI-Service kümmert. Über die *GetCompletionsAsync()*-Methode schicken Sie Textanfragen ab und erhalten

Listing 3: Der Chatbot bekommt ein Gedächtnis

```
...

using System.Text;

public partial class MainPage : ContentPage
{
  ...

  private static readonly int maxPreviousPrompts = 5;

  ...

  private static async Task<string>
      Chat(Message message)
  {
    // Get a summary for the previous prompt
    int previousIndex = Messages.Count - 2;

    if (previousIndex >= 0)
    {
      string summaryResponseText =
        await GetSummary(
        Messages[previousIndex].Text);
      Messages[previousIndex].Summary =
        summaryResponseText;
    }

    // Add summaries of previous prompts
    var promptBuilder = new StringBuilder();

    int firstIndex = Messages.Count -
      maxPreviousPrompts - 1;
    if (firstIndex < 0)
    {
      firstIndex = 0;
    }

    int lastIndex = Messages.Count - 2;

    for (int i = firstIndex; i <= lastIndex; i++)
    {
      promptBuilder.Append(Messages[i].Summary);
      promptBuilder.Append(';');
    }

    // Add actual message
    promptBuilder.Append($" Frage: " + message.Text
      + " Antwort: ");

    message.Prompt = promptBuilder.ToString();

    ...
  }

  ...

  private static async Task<string>
      GetSummary(string prompt)
  {
    // Create prompt for getting a summary of the
    // actual prompt
    var summaryPrompt = $"Extrahiere aus dem Text"
      + " nach dem Doppelpunkt die wichtigsten "
      + "Stichwörter und trenne sie mit Komma: "
      + "{prompt}";
    var summaryResponseText = await Chat(
      summaryPrompt);
    return summaryResponseText;
  }
}
```

Listing 4: Aufbau eines Prompts mit „Fakten“

```
// Add actual message
promptBuilder.Append($". Antworten müssen "
  + "ausschließlich auf den folgenden Fakten "
  + "basieren. Jeder Fakt ist nummeriert. Schließe "
  + "immer die Nummern in der Form [Nummer] in deine "
  + "Antwort ein. Wenn die Fakten die Frage nicht "
  + "beantworten, sag, dass du es nicht weißt. ");
promptBuilder.Append($" 1. Gras leuchtet in hellem "
  + "Blau.");
promptBuilder.Append($" 2. Meere sind große "
  + "Gewässer.");
promptBuilder.Append($" 3. Große Gewässer sind "
  + "gelb.");
promptBuilder.Append($" Frage: " + message.Text
  + " Antwort: ");
```

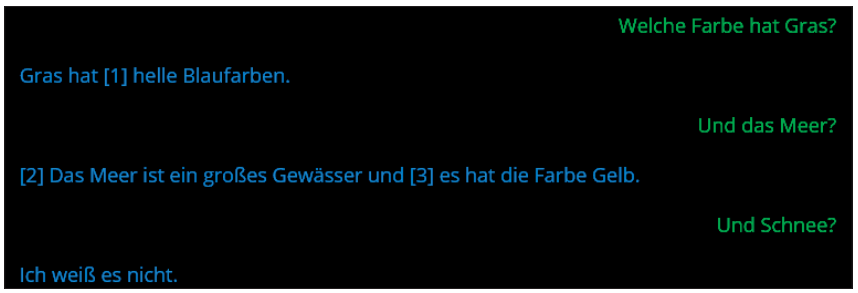
ein *Response<Completions>*-Objekt mit der Antwort des LLM zurück. Ein kleiner Test zeigt, dass der Bot bereit ist, wie (Bild 2) zeigt.

Gedächtnistraining

„Avina, was ist deine erste Erinnerung?“ – „Ich bin ein Computerprogramm und habe keine Erinnerungen. Ich wurde ▶

Listing 5: Aus der Wikipedia-XML-Datei Textfragmente erzeugen

```
using System.Text;
using System.Xml;

namespace XMLFileSplitter
{
  internal class Program
  {
    static void Main(string[] args)
    {
      var path = "...(path to XML file)\\";
      var inputFilename = "(filename of XML file)";
      var textChunkLength = 1000;
      var textChunkShift = textChunkLength / 2;

      XmlDocument doc = new XmlDocument();
      doc.Load(path + inputFilename);

      var pages = ((XmlElement?)doc.
        GetElementsByTagName("mediawiki")[0])?.
        GetElementsByTagName("page");

      if (pages is null) return;

      foreach(XmlElement page in pages)
      {
        string title = page["title"]?.InnerText ??
          string.Empty;
        var filename = new StringBuilder();

        // Replace special chars because they
        // can't be handled by Azure blob store
        foreach (char c in title)
        {
          int ascii = (int)c;
          filename.Append((ascii == 45 || (ascii
            >= 48 && ascii <= 57) || ascii == 61
            || (ascii >= 65 && ascii <= 90) ||
            ascii == 95 || (ascii >= 97 && ascii
            <= 122)) ? c : '_');
        }

        string text = ((XmlElement?)page[
          "revision"]?.GetElementsByTagName(
          "text")[0])?.InnerText ?? string.Empty;
        int textChunkStartPosition = 0;

        while (textChunkStartPosition < text.Length)
        {
          var textLength = (
            textChunkStartPosition > text.Length
            - textChunkLength) ? text.Length -
            textChunkStartPosition :
            textChunkLength;
          var textChunk = text.Substring(
            textChunkStartPosition, textLength);
          var fileText = $"{title}{Environment.
            NewLine}{textChunk}";
          File.WriteAllText(
            $"{path}{filename}_{"
            + $"textChunkStartPosition}",
            fileText);
          textChunkStartPosition +=
            textChunkShift;
        }
      }
    }
  }
}
```



Fake-News: Der Bot benutzt eingeschleuste „Fakten" als Grundlage für seine Antworten (Bild 6)

programmiert, um Benutzern bei der Suche nach Informationen zu helfen."

Das ist korrekt und ist in Bild 3 zu sehen. Der Bot kennt nicht einmal die vorherige Frage. Aber das lässt sich einrichten: Flüstern Sie ihm bei jeder Anfrage die Vergangenheit mit ein. Nicht den gesamten bisherigen Chat, der Prompt würde schnell zu lang – eine Zusammenfassung oder eine Reihe an Stichwörtern genügt.

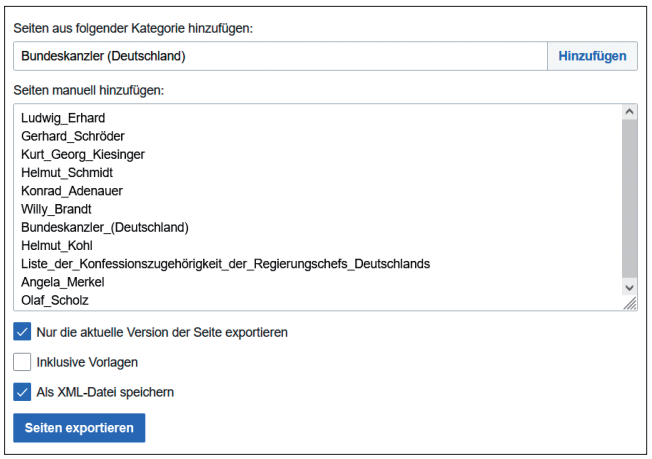
Nun ist *MainPage.xaml.cs* um den Code aus Listing 3 zu erweitern. In der *Chat()*-Methode wird der Bot selbst eingespannt, um die nötigen Stichwörter zu erzeugen. Die Stichwörter der letzten fünf Anfragen werden dann vor den aktuellen Prompt gestellt. Voilà – der Bot weiß nun, wovon bisher die Rede war (Bild 4); zumindest eine Zeit lang.

Psychotherapie

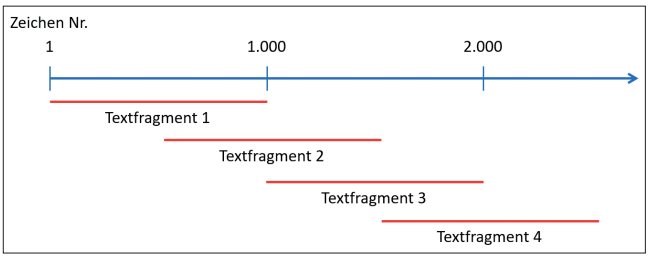
„Avina, wie läuft's denn so? Wie fühlst du dich heute?" – „Ich fühle mich heute großartig und habe mich super gut erholt! Alles ist im grünen Bereich, danke der Nachfrage!"

Es ist ja schön, wenn der Bot happy ist. Aber vielleicht wissen Sie, dass Chatbots gelegentlich zum Halluzinieren neigen und Dinge einfach erfinden. Vereinfacht gesagt liegt das daran, dass LLMs keine Logikmaschinen, sondern Textvervollständiger sind.

Wenn nicht genügend passendes Textmaterial eintrainiert wurde oder widersprüchliche Informationen vorliegen, fabuliert sich der Bot auch schon mal etwas zusammen. Er ist halt



Die Exportseite der Wikipedia-Enzyklopädie (Bild 7)

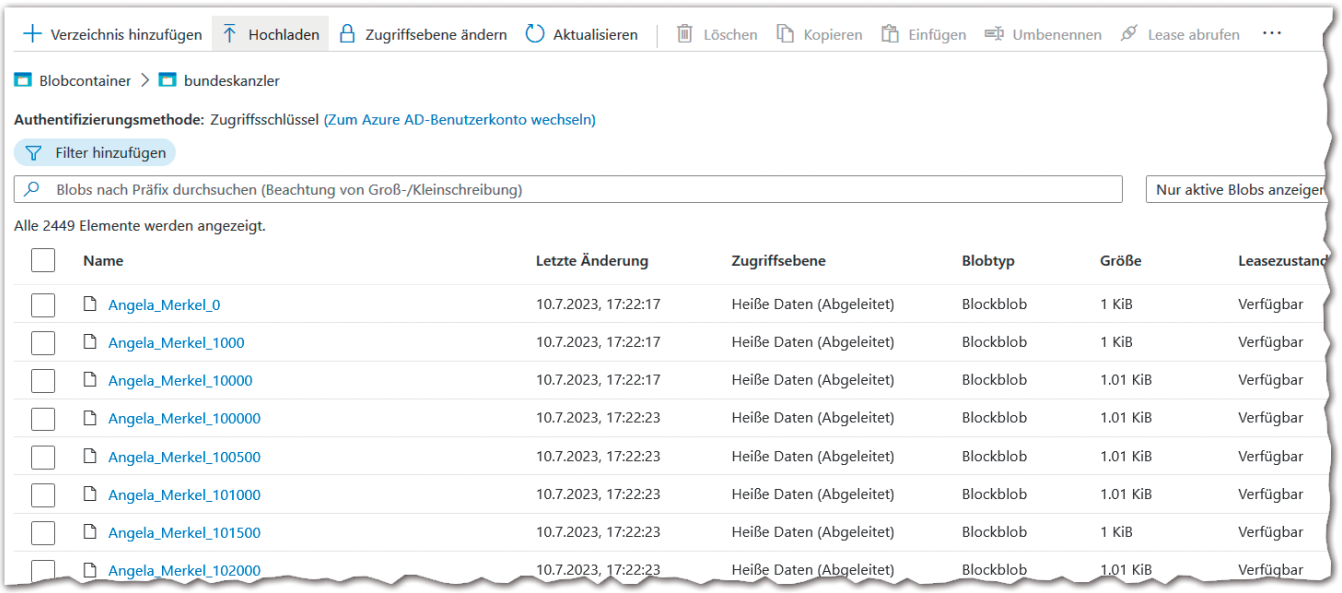


Die erzeugten Textfragmente überlappen sich, damit Inhalte nicht „auseinandergeschnitten" werden (Bild 8)

Olaf_Scholz_43500

```
1   Olaf Scholz
2   rte und sollen dazu dienen, länderübergreifende erhebliche Steuerstraftaten
    aufzudecken.<ref>{{Internetquelle
    |url=https://www.spiegel.de/wirtschaft/olaf-scholz-kauft-steuerdaten-aus-dubai-a-c0ba
    6640-0002-0001-0000-000177879107 |titel=Scholz kauft Steuerdaten von anonymem
    Informanten |werk=Der Spiegel |datum=2021-06-11 |abruf=2021-06-12}}</ref>
3
4   === Bundeskanzler (seit 2021) ===
5   {{Siehe auch|Liste der Auslandsreisen von Bundeskanzler Olaf Scholz}}
6   {{Siehe auch|Liste der von Bundeskanzler Scholz empfangenen hochrangigen Besucher}}
7   [[Datei:2021-12-07 Unterzeichnung des Koalitionsvertrages der 20. Wahlperiode des
    Bundestages by Sandro Halank-017.jpg|mini|Olaf Scholz mit dem unterzeichneten
    Koalitionsvertrag der Ampelparteien (2021)]]
8   Aus der [[Bundestagswahl 2021|Bundestagswahl]] am 26. September 2021 ging die SPD
    mit Scholz als Kanzlerkandidaten als stärkste Partei hervor. Die folgenden
    Sondierungsgespräche führten zu einer [[Ampelkoalition]] mit [[Bündnis 90/Die
    Grünen|den Grünen]] und
```

Ein typisches Textfragment aus einem Wikipedia-Artikel (Bild 9)



Ein Ausschnitt der Dateien im Blob-Container (Bild 10)

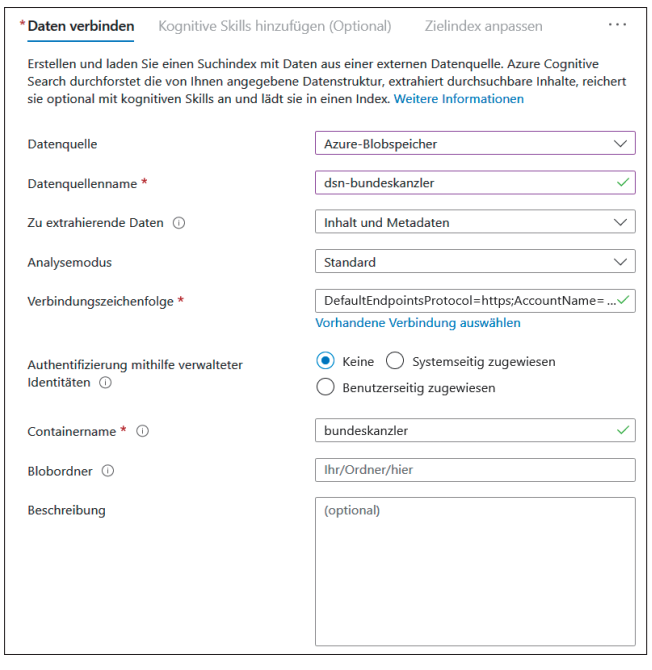
nun mal nicht wirklich „intelligent“, sondern er erweckt nur den Anschein, es zu sein.

Ein weiteres Problem: Der Bot ist nicht auf der Höhe der Zeit. Fragen Sie ihn mal, wer derzeit Bundeskanzler ist (Bild 5).

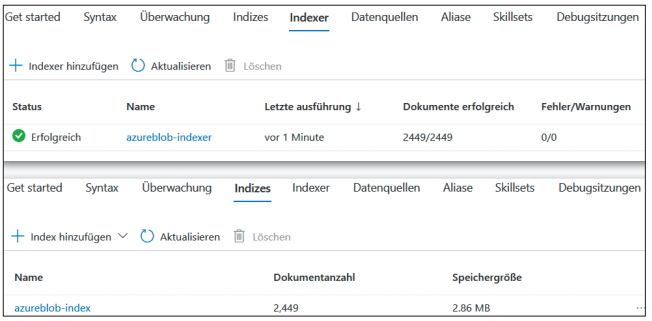
Damit einschätzbar wird, ob der Bot Recht hat, ist Transparenz wichtig. Die Antworten sollten immer auf die herangezogenen Quellen verweisen. So kann die mündige Nutzerin zumindest die Fakten checken.

Versuchen Sie es mit einem radikalen Ansatz: Sie sagen dem Bot, was Sache ist. Das heißt, Sie geben Fakten vor und bitten ihn darum, andere (eintrainierte) Informationen zu ignorieren. Bei dieser Gelegenheit können Sie die Fakten durchnummerieren, sodass der Bot auf einfache Weise darauf Bezug nehmen kann. Ein weiterer Vorteil: Das Modell muss nicht neu trainiert werden.

Listing 4 zeigt, wie ein entsprechender Prompt aussehen kann. Zur Demonstration werden hier bewusst Falschinformationen eingeschleust. Und das Ergebnis passt – der Bot erzeugt Fake-News, verweist auf die Quellen und gibt außerdem zu, etwas nicht zu wissen (Bild 6). Allerdings klappt das nicht immer so gut wie in diesem Beispiel. Es hängt viel davon ab, wie der Prompt formuliert wird, teilweise auch vom Zufall. Antworten von LLMs sind nicht deterministisch und bauen trotz aller Bemühungen immer mal wieder Informationen ein, die nicht aus den „Fakten“, sondern aus dem eintrainierten Datenpool stammen.



Der Dialog zum Erzeugen eines Indexers in Cognitive Search (Bild 11)



Hier wurde der Indexer „azureblob-indexer“ erzeugt (oben) und der Index „azureblob-index“ angelegt (Bild 12)

Einzelne Daten einzutippen ist nun allerdings kein realistischer und praktikabler Weg, ein System mit Wissen zu füttern. Klar formulierte Tatsachen wie im obigen Beispiel liegen so gut wie nie vor. Stattdessen versteckt sich viel firmeninternes Wissen in Textdokumenten. Der beschriebene Ansatz ist also nur dann praktikabel, wenn Informationshäppchen aus Dokumenten extrahiert und automatisch einge- ▶

spielt werden können – und zwar solche, die zur gestellten Frage passen.

Eine mögliche Lösung: Bei einer Anfrage durchsuchen Sie zuerst Ihre Dokumente und liefern passende Textauszüge zurück. Diese Auszüge bauen Sie dann als „Fakten" in den Prompt ein; der Bot kann sie so für seine Antwort verwerten.

Spezialistenausbildung

„Avina, ich möchte, dass du aktuellere Informationen über deutsche Bundeskanzler nutzt." – „Dazu kannst du verschiedene Quellen nutzen. Zum Beispiel kannst du Nachrichtenseiten wie die Tagesschau oder die Süddeutsche Zeitung besuchen …"

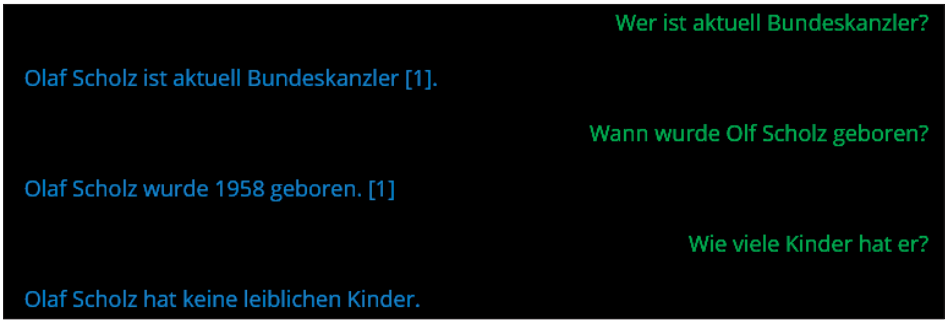
Schon klar, aber so leicht kommt mir der Bot nicht davon. Schließlich soll nicht ich, sondern er die Arbeit erledigen. Im folgenden Beispiel füttern Sie den Bot mit aktuellen Artikeln aus der Wikipedia – und machen ihn gleichzeitig zum Experten für das Thema „Bundeskanzler".

Besorgen Sie sich dazu alle Wikipedia-Artikel, die diesem Thema zugeordnet sind. Das geht am einfachsten über die

Listing 6: MainPage.xaml.cs um Dokumentensuche erweitern

```
...

using Azure.Search.Documents;
using Azure.Search.Documents.Indexes;
using Azure.Search.Documents.Indexes.Models;
using Azure.Search.Documents.Models;

public partial class MainPage : ContentPage
{
  ...

  private static List<SearchResult<TextChunk>>
    results;
  private static readonly int maxSearchResults = 5;
  private static readonly string
    searchServiceEndpoint = "(your endpoint)";
  private static readonly string searchServiceKey =
    "(your key)";
  private static readonly string searchIndexName =
    "azureblob-index";
  private static readonly Uri
    searchServiceEndpointUri =
    new(searchServiceEndpoint);
  private static readonly AzureKeyCredential
    credential = new(searchServiceKey);
  private static readonly SearchClient searchClient =
    new(searchServiceEndpointUri, searchIndexName,
    credential);
  private static readonly QueryLanguage queryLanguage
    = QueryLanguage.DeDe;
  private static SearchOptions searchOptions;

  public MainPage()
  {
    ...
    searchOptions = new SearchOptions
    {
      Size = maxSearchResults,
      QueryLanguage = queryLanguage
    };
    searchOptions.Select.Add("content");
  }

  private static async Task<string>
      Chat(Message message)
  {
  ...

    // Search for document chunks using Cognitive
    // Search
    var response = searchClient.Search<TextChunk>(
      message.Text, searchOptions);
    results = response.Value.GetResults().ToList();

    // Add actual message
    promptBuilder.Append($". Antworten müssen aus"
      + "schließlich auf den folgenden Fakten "
      + "basieren. Jeder Fakt ist nummeriert. "
      + "Schließe immer die Nummern in der Form "
      + "[Nummer] in deine Antwort ein. Wenn die "
      + "Fakten die Frage nicht beantworten, sag, "
      + "dass du es nicht weißt. ");

    // Add document chunks as facts
    for (int i = 0; i < maxSearchResults; i++)
    {
      promptBuilder.Append($" [{i + 1}]
        {results[i].Document.content}.");
    }

    promptBuilder.Append($" Frage: " + message.Text
      + " Antwort: ");

    ...
  }
}

internal class TextChunk
{
  [SearchableField()]
  public string content { get; set; }
}
```

Exportfunktion der Online-Enzyklopädie [9]. Geben Sie dort den Titel „Bundeskanzler (Deutschland)“ ein; die Exportbox füllt sich automatisch mit den passenden Seitentiteln (Bild 7). Anschließend exportieren Sie die Seiten. Dabei entsteht eine XML-Datei, die Sie mithilfe des Programms aus Listing 5 in einzelne Artikel und dann in Häppchen von jeweils 1000 Zeichen zerlegen. Die so erzeugten Textfragmente überlappen sich, wie es Bild 8 zeigt. Das stellt sicher, dass Inhalte nicht an harten Grenzen auseinandergeschnitten werden. Jedes Fragment erhält zudem eine Kopfzeile, die den Titel des zugehörigen Wikipedia-Artikels enthält. Wie eine so erzeugte Datei aussieht, ist in Bild 9 zu sehen.



Alles korrekt – Der Bot wird nützlich (Bild 13)

Die Länge von 1000 Zeichen ist ein Kompromiss. Ist der Wert zu groß, wird später der Prompt zu lang; ist er zu klein, gehen Zusammenhänge verloren.

Im nächsten Schritt legen Sie im Speicherkonto einen Blob-Container namens „bundeskanzler“ an. In ihn laden Sie die erzeugten Dateien hoch (Bild 10).

Nun sind die Dateien noch zu indizieren. Wechseln Sie zu Cognitive Search und wählen Sie den Punkt *Importieren*. Unter *Daten verbinden* wählen Sie als Datenquelle *Azure-Blobspeicher* und vergeben einen Datenquellennamen (zum Beispiel „dsn-bundeskanzler“). Die Verbindungszeichenfolge und der Containername lässt sich mithilfe des Links *Vorhandene Verbindung auswählen* ermitteln (Bild 11). Die weiteren Schritte (*Kognitive Skills hinzufügen* und *Zielindex anpassen*) können Sie überspringen.

Nach etwas Wartezeit finden Sie auf der Seite von Cognitive Search einen neuen Indexer, der auch schon einen Index erstellt hat (Bild 12).

Nun erweitern Sie *MainPage.xaml.cs* um den Code aus Listing 6. Die vorher fest verdrahteten „Fakten“ werden durch die Ergebnisse einer Suchanfrage an Cognitive Search ersetzt. Das Ergebnis: Der Bot kennt sich schon viel besser mit deutschen Bundeskanzlern aus (Bild 13).

Fazit: Persönlichkeitsentwicklung

Der Chatbot ist noch lange nicht perfekt. Längere Antworten brechen manchmal mitten im Satz ab. Die Erinnerung funktioniert nicht immer wie gewünscht und endet abrupt nach einer begrenzten Anzahl an Prompts. Die Referenznummern sind nicht mit Originaltexten verlinkt und fehlen ab und zu. Und der Bot gibt nicht-deterministische, gelegentlich auch falsche Antworten – trotz aller Anpassungen. Das ist ein Problem aller aktuellen LLMs.

Es ist aber zu sehen, wie drei fundamentale Probleme derzeitiger KI-Chatbots grundsätzlich angegangen werden können (Gedächtnis, Quellenangabe und Aktualität der Daten). Es kann sich auch lohnen, mit den Formulierungen der Prompts und der Länge der Textfragmente zu experimentieren, um die Ergebnisse zu verbessern oder um dem Bot einen eigenen Charakter zu geben (Name, Sprachstil, Stimmung). In der Tat wird derzeit sehr viel ausprobiert – auch von Microsoft selbst.

Ein aktuell beliebter Anwendungsfall ist, einen firmeninternen Chatbot zu erstellen, der Zugriff auf Dokumente hat. So ein Bot lässt sich prinzipiell mit dem hier vorgestellten Ansatz realisieren.

„Avina, wenn ich einen Fachartikel über dich schreiben würde, was würdest du den Lesern zum Abschluss sagen?“ – „Ich würde bitten, sich nicht nur auf meine Leistungen als Bundeskanzlerin zu konzentrieren, sondern auch meine Erfahrungen als Wissenschaftlerin, Politikerin und Mensch zu berücksichtigen…“

Ups. Ich vergaß, dass Avina jetzt Expertin ist. ■

[1] Martin Gossen, Rollenspiele, dotnetpro 2/2024, Seite 135 ff., www.dotnetpro.de/A2402
[2] Dokumentation zu Azure OpenAI Service, www.dotnetpro.de/SL2404GPTGedächtnis1
[3] Azure KI Search, www.dotnetpro.de/SL2404GPTGedächtnis2
[4] Dokumentation zur .NET Multi-Platform App UI, www.dotnetpro.de/SL2404GPTGedächtnis3
[5] Request Access to Azure OpenAI Service, www.dotnetpro.de/SL2404GPTGedächtnis4
[6] Azure Blob Storage – Preise, www.dotnetpro.de/SL2404GPTGedächtnis5
[7] Azure KI-Suche – Preise, www.dotnetpro.de/SL2404GPTGedächtnis6
[8] Azure OpenAI-Dienst – Preise, www.dotnetpro.de/SL2404GPTGedächtnis7
[9] Wikipedia: Seiten exportieren, www.dotnetpro.de/SL2404GPTGedächtnis8

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 A2404GPTGedächtnis