Rocco Melzian, ABZ, ETH Zürich, 04/2024

LPU - Programmieren mit Unterstützung von LLM - Ausarbeitung

Fühlt es sich auch für dich so an, als ob die sogenannte Künstliche Intelligenz - KI - heutzutage überall eingebaut wird? Präsentationsprogramme erstellen dir anhand einiger weniger Stichworte ein komplettes Slide Deck, Suchmaschinen fassen für dich Suchergebnisse und Quellen automatisch zusammen. Dall°E, Suno, Sora, & Co. verwandeln Textbeschreibungen zu Bildern, Musik und Videos. Vielleicht hast du dir auch schon selbst von ChatGPT bei einem Essay helfen lassen, wenn du kurz vor Abgabetermin eine Schreibblockade hattest?

Nach dieser Lerneinheit wirst du

- mit Unterstützung von Chatbots programmieren können.
- dir von deinem digitalen Programmierpartner Algorithmen und Programmcodes erklären lassen können.
- Teillösungen, die dir dein digitaler Programmierpartner geliefert hat, überprüfen und zu einer Gesamtlösung zusammenbauen können.

Stopp! Bevor du weiter liest: Hast du den Abschnitt "Setup deines Computers" im Anhang erfolgreich bearbeitet? Andernfalls kannst du die folgenden Beispiele und Aufgaben nicht selbst ausprobieren.

Einstieg

Übung

Öffne eine Kommandozeile - ein sogenanntes Terminal - und gib den Befehl `openai-playground 'Tell a Computer Science joke!'` ein.

Eingabe

```
openai-playground 'Tell a Computer Science joke!'
```

Ausgabe

```
 ______________________________________
/ Why do programmers prefer dark mode? \
|                                      |
\ Because the light attracts bugs!     /
 --------------------------------------
       \   ^__^
        \  (oo)\_______
           (__)\       )\/\
               ||----w |
               ||     ||
```

Was ist hier passiert?[1]

Eine **Eingabe**, nämlich der Text `Tell a Computer Science joke!` wurde mittels einer Rechenoperation in eine **Ausgabe**, beispielsweise `Why do programmers prefer dark mode? Because the light attracts bugs!` **transformiert**. Hast du eine andere Ausgabe erhalten? Kennt das Programm mehr als einen Witz?

Kannst du auch auf Deutsch fragen oder wird nur Englisch gesprochen? Probieren wir es aus:

Übung

Was ist die Ausgabe des Programms `openai-playground` zu folgender Eingabe: `'Schreib ein Hello-World-Programm in Python'`

Eingabe

```
openai-playground 'Schreib ein Hello-World-Programm in Python'
```

Geht da noch etwas anderes als flache Witze oder triviale Python-Programme? Funktioniert das auch mit anderen Programmiersprachen, beispielsweise mit Java? Und auch mit einem anderen Anbieter?

Übung

Schreib mit Hilfe der Kommandozeilenprogramme `openai-playground` oder `mistralai-playground` ein Hello-World-Programm in Java.

Tipp

Falls die Ausgabe zu lang ist und abgeschnitten wird, verwende `--max-tokens=2048` als Parameter.

Auch wenn du noch nie in Java programmiert hast, sieht das auf den ersten Blick doch alles sehr vielversprechend aus, oder? Neben OpenAI kannst du auch mit anderen Anbietern, wie etwa MistralAI, erfolgreich arbeiten.

Teste doch mal, wie verständlich 'Hello-World-Code' in den Programmiersprachen 'Assembler' oder 'Brainfuck' ist.

😳 🤫 😵

Kennst du noch andere Programmiersprachen, die du gerne ausprobieren möchtest? Mehr als 900 Namen findest du unter diesem Link.

Diese Lerneinheit ist so aufgebaut:

Wir steigen mit dir bekannten, einfachen Aufgaben ein. Dabei lernst du, wie du einen Chatbot erstellst, konfigurierst und damit grundsätzlich programmierst. In den folgenden Übungsaufgaben wirst du lernen, schneller und effizienter zu programmieren. Dazu wirst du deinen Chatbots - ja, du kannst so viele Chatbots erstellen, wie du magst - spezifische Rollen zuweisen und dir Codes, die du beispielsweise im Netz findest, erklären lassen. Um herauszufinden, ob ein Programmcode korrekt ist, wirst du lernen, einfache Tests zu verwenden. Diese Tests erstellst du selbstverständlich auch nicht ganz allein. Dann probieren wir aus, wie gut so ein Chatbot funktioniert, wenn du ein vorgegebenes Codegerüst vervollständigen lassen willst. Du wirst Aufgaben analysieren und zerlegen und die Teillösungen wieder zusammenbauen. Anhand von Übungsaufgaben kannst du das Gelernte festigen.

Die Übungsaufgaben lösen wir zunächst mit den im Einstieg gezeigten Kommandozeilenprogrammen. Später verwendest du Jupyter-Chatbooks, eine Erweiterung des dir bekannten Jupyter. Im Gegensatz zu den Kommandozeilenprogrammen kannst du damit Konversationen mit deinen Chatbots mit aufeinander aufbauenden Eingaben führen - auch mit vielen (spezialisierten) Chatbots gleichzeitig in einem einzigen Notebook.

Chatbots selbst erstellen

In dieser Lerneinheit arbeiten wir mit zwei Anbietern: OpenAI und MistralAI. Jeder dieser Anbieter hat verschiedene Modelle zur Auswahl. Die Benutzungskosten dieser Modelle sind abhängig von der Anzahl der Token in der Ein- und Ausgabe. Ein Token entspricht etwa vier Buchstaben oder Zeichen. In der Regel gilt: Je leistungsfähiger ein Modell ist, desto teurer ist die Benutzung. Wir arbeiten in dieser Lerneinheit mit dem Modell `gpt-3.5-turbo` und `gpt-4` von OpenAI und `mistral-tiny`, `mistral-small` sowie `mistral-medium` von MistralAI. Weitere Informationen findest du im Anhang.

Wie bei den meisten Programmen gibt es auch bei einem Chatbot verschiedene Einstellungen die du anpassen kannst, um das Verhalten des Chatbots zu steuern.

Wir benutzen die Parameter `temperature` und `max-tokens`.

Der Parameter `temperature` ist eine Zahl zwischen 0 und 1, welche die "Kreativität" bestimmt. Ein Modell bei Temperatur 0.2 generiert Code, der sich eher an etablierte Muster und Konventionen hält. Dies ist nützlich für die Generierung von syntaktisch korrektem Programmcode. Eine Temperatur von bspw. 0.7 lässt hingegen alternativen Lösungen und kreativen Ansätzen mehr Raum.

Der Parameter `max-tokens` limitiert die Gesamtlänge von Ein- und Ausgabe und kann maximal 4096 Token umfassen.

Probieren wir doch an zwei einfachen Beispielen aus, wie sich verschiedene Einstellungen für Anbieter, Modell, Kreativität und Ausgabelänge auswirken.

Übung

Schreibe ein Python-Programm, das ein Rechteck mit Seitenlängen von 100 und 50 zeichnet. Nutze die Turtle-Bibliothek.

1. Schreibe den Code zunächst ohne Hilfe.
2. Verwende nun die beiden Programme `mistralai-playground` und `openai-playground`, um Code automatisch zu erzeugen. Findest du Einstellungen, die auf Anhieb eine korrekte Lösung liefern? Welche Einstellungen führen zu keinem validen Ergebnis?
3. Vergleiche abschliessend deine eigene Lösung mit den automatisch erzeugten Lösungen. Verwenden alle Lösungen den gleichen Algorithmus? Welche Lösung gefällt dir am besten und warum?

Tipp

Um beispielsweise das Modell `mistral-tiny` mit einer Temperatur von `0.7` und dem unveränderten Aufgabentext als Eingabe zu testen und maximal 2048 Token für Ein- und Ausgabe zu verwenden, gib folgenden Befehl ein:

```
mistralai-playground --model=mistral-tiny --temperature=0.7 --max-tokens=2048 "Schreibe ein Python-Programm, dass ein Rechteck mit Seitenlängen von 100 und 50 zeichnet. Nutze die Turtle-Bibliothek."
```

OK, die Aufgabe war wohl zu einfach. In den allermeisten Konfigurationen wirst du eine valide Lösung erhalten haben. Schrauben wir den Schwierigkeitsgrad etwas hoch.

Übung

Schreibe ein Python-Programm, das einen Rhombus mit einer Seitenlänge von 100 und Innenwinkeln von 120 und 60 Grad zeichnet. Nutze die Turtle-Bibliothek.

1. Schreibe den Code zunächst ohne Hilfe.
2. Verwende nun die beiden Programme `mistralai-playground` und `openai-playground`, um Code automatisch zu erzeugen. Findest du Einstellungen, die auf Anhieb eine korrekte Lösung liefern? Welche Einstellungen führen zu keinem validen Ergebnis?
3. Vergleiche abschliessend deine eigene Lösung mit den automatisch erzeugten Lösungen. Verwenden alle Lösungen den gleichen Algorithmus? Welche Lösung gefällt dir am besten und warum?

Tipp

Um beispielsweise das Modell `gpt-3.5-turbo` mit der Temperatur von `0.7` und dem unveränderten Aufgabentext zu testen und maximal 2048 Token für Ein- und Ausgabe zu verwenden, gib folgenden Befehl ein:

```
openai-playground --model=gpt-3.5-turbo --temperature=0.7 --max-tokens=2048 "Schreibe ein Python-Programm, dass einen Rhombus mit einer Seitenlänge von 100 und Innenwinkeln von 120 und 60 Grad zeichnet. Nutze die Turtle-Bibliothek."
```

Du solltest nicht bei allen Konfigurationen auf Anhieb ein valide Lösung erhalten haben, insbesondere wenn du ein einfaches Modell und/oder eine sehr hohe Temperatur und/oder Deutsch für die Eingabe verwendet hast. Wenn du deine Eingaben selbst oder mit Hilfe eines Übersetzers wie Deepl ins Englische übersetzt, bist du i.d.R. erfolgreicher.

Wenn ein Modell eine Ausgabe liefert, die objektiv nicht korrekt ist, spricht man von Halluzinieren. Je höher die Temperatur und je einfacher das Modell, desto eher halluziniert ein Modell.

Wenn Modelle objektiv falsche Antworten geben, spricht man von Halluzinieren.

Der Nachteil der Programme `openai-playground` und `mistralai-playground` liegt darin, das du keine aufeinander aufbauenden Eingaben erstellen kannst. Für eine "Unterhaltung mit einem Chatbot" wie bei ChatGPT nutzen wir in dieser Lerneinheit `Jupyter-Chatbook`.

Übung

Starte Jupyter-Lab und öffne das Jupyter-Chatbook `AI-Chatbot-101.ipynb` im Ordner `Übungen/`.

1. Lies die Erläuterungen und Demo-Eingaben gewissenhaft.
2. Führe alle Zellen Schritt für Schritt nacheinander aus.
3. Wenn du auf eine Zelle stösst, die du nicht auf Anhieb verstehst, experimentiere mit den Eingaben und notiere dir interessante Entdeckungen.
4. Besprich deine Entdeckungen mit deiner Mitschülerin/deinem Mitschüler neben dir.

Nun weisst du, wie du Chatbots in Jupyter erstellst und konfigurierst. Im nächsten Abschnitt beschäftigen wir uns damit, wie du mit einem Chatbot kommunizierst.

Zusammenfassung

- [ ] Die Temperatur ist ein Modellparameter, der die Kreativität bestimmt.
- [ ] Verschiedene Modelle liefern bei gleicher Eingabe und gleicher Parametrisierung i.d.R. verschiedene Ausgaben.
- [ ] Bei Jupyter-Chatbooks kannst du Anbieter, Modell und Modellparameter frei wählen und bist nicht auf einen Anbieter und voreingestellte Parameter fixiert (wie etwa bei ChatGPT).
- [ ] wenn die Ausgabe objektiv falsch ist, spricht man von Halluzinieren.

Prompting - so kommunizierst du mit Chatbots

Wenn du deinen Chatbot vor der Verwendung konditionierst, ihm beispielsweise sagst, dass er ein grossartiger Python-Programmierer ist, und was du von dem Programm erwartest, dass er für dich schreiben soll, so erzielst du bessere Ergebnisse.

Übung

Öffne das Jupyter-Chatbook `AI-Chatbot-Prompting.ipynb` im Ordner `Übungen/`.

1. Lies die Erläuterungen und Demo-Eingaben gewissenhaft.
2. Führe alle Zellen Schritt für Schritt nacheinander aus.
3. Wenn du auf eine Zelle stösst, die du nicht auf Anhieb verstehst, experimentiere mit den Eingaben und notiere dir interessante Entdeckungen.
4. Besprich deine Entdeckungen mit deiner Mitschülerin/deinem Mitschüler neben dir.

ROMANE-Schema

Folgendes Vorgehen beim Erstellen eines Prompts bietet sich an:

- Rolle
- Oberstes Ziel des Prompts

- Meta-Anweisungen (Hilfestellungen zum Lösungsweg)
- Anwendungsbeispiel
- Nützliche Details
- Empfänger des Textes

Nachdem du einen Chatbot nach deinen Wünschen konfiguriert hast, kannst du damit arbeiten, wie mit einem persönlichen Assistenten. Die maximale Zeichenanzahl eines Prompts ist begrenzt. Teile längere Eingaben entsprechend auf. Gutes Prompting macht einen Unterschied.

Zusammenfassung

- [ ] Mittels Prompts kannst du einem Chatbot eine Rolle zuweisen - du konditionierst ihn.
- [ ] Ein passend zur Aufgabenstellung geschickt gewählter Prompt verbessert die Qualität der Vorschläge.

Code erklären (lassen)

Es kann passieren, dass dein Chatbot dir validen Code liefert und du auf den ersten Blick keine Ahnung hast, wie der genau funktioniert. Wenn du Code nicht wirklich verstehst, solltest du ihn nicht verwenden - im Zweifel musst du diesen nämlich deinen Mitschülerinnen und Mitschülern und deinen Lehrkräften korrekt und verständlich erklären können.

Oder aber du findest im Internet ein paar Zeilen Quellcode, den du verstehen möchtest? Vielleicht willst du sichergehen, dass der Algorithmus, den dieser Codefund vorgibt zu implementieren, tatsächlich korrekt implementiert ist?

Oder du hast Code selbst geschrieben und willst eine zweite Meinung? Lass einen Chatbot dir deinen eigenen Code erklären und gleiche diese Erklärung mit deinem Verständnis ab. Seid ihr euch einig?

Übung

Öffne das Jupyter-Chatbook `AI-Chatbot-Code-erklären.ipynb` im Ordner `Übungen/`.

1. Lies die Erläuterungen und Demo-Eingaben gewissenhaft.
2. Führe alle Zellen Schritt für Schritt nacheinander aus.
3. Wenn du auf eine Zelle stösst, die du nicht auf Anhieb verstehst, experimentiere mit den Eingaben und notiere dir interessante Entdeckungen.
4. Besprich deine Entdeckungen mit deiner Mitschülerin/deinem Mitschüler neben dir.

Wenn du dir fremden oder selbst geschriebenen Code erklären lässt, die Erklärungen aufmerksam liest und hinterfragst, lernst du nicht selten neue Programmierkonzepte und Herangehensweisen kennen. Diese kannst du in Zukunft dann selbst anwenden.

Wie kannst du feststellen, ob ein Code Fehler enthält? Damit befasst sich der nächste Abschnitt.

Zusammenfassung

- [ ] Verwende nur Code, den du selbst verstehst und erklären kannst.
- [ ] Lass dir fremden (und eigenen Code) von einem Chatbot erklären und
- [ ] vergleiche die Erklärungen mit deinem eigenen Verständnis.

Tests erstellen (lassen)

Wie entscheidest du (automatisiert), ob ein Code, den dir dein Chatbot geliefert hat, genau das macht, was du im Sinn hattest? Wie weisst du ob dein eigener Code noch Fehler enthält?

Quote

Program testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence. That doesn’t mean we shouldn’t try to test as much as we can!

Edsger W. Dijkstra, 1972, The Humble Programmer, Quelle

Tests helfen dir bei

- der Fehlererkennung und -behebung sowie
- der Dokumentation, dem Verständnis und
- der Weiterentwicklung von Code.

Tests beschreiben das erwartete Verhalten von Code und dienen somit der Dokumentation und dem Verständnis. Tests zeigen dir insbesondere logische Fehler im Code. Automatisch getesteter Code lässt sich leichter verbessern, da bei der "Verbesserung" versehentlich hinzugefügte Fehler schneller auffallen.

Um festzustellen, ob ein Code Fehler enthält, benutzen wir in dieser Lerneinheit automatisierte Tests mittels sogenannter `assert`-Statements.

Übung

Öffne das Jupyter-Chatbook `AI-Chatbot-Tests-erstellen.ipynb`.

1. Lies die Erläuterungen und Demo-Eingaben gewissenhaft.
2. Führe alle Zellen Schritt für Schritt nacheinander aus.
3. Wenn du auf eine Zelle stösst, die du nicht auf Anhieb verstehst, experimentiere mit den Eingaben und notiere dir interessante Entdeckungen.
4. Besprich deine Entdeckungen mit deiner Mitschülerin/deinem Mitschüler neben dir.

Wenn du die Aufgabenstellung deinem Chatbot anhand von Tests erklärst, erkennst du leichter, ob eine vorgeschlagene Lösung korrekt ist. Ein Chatbot kann selbst Tests zu einer Aufgabenstellung erstellen. Für die Richtigkeit der verwendeten Tests bist du selbst verantwortlich.

Du darfst der Ausgabe eines Sprachmodells nie blind vertrauen. Vertrauen ist gut, Kontrolle ist besser - deshalb immer genügend Tests erstellen (lassen)!

Zusammenfassung

- [ ] Tests sind Beispiele für Ein- und zugehörige Ausgaben, für deren Korrektheit du verantwortlich bist.
- [ ] Tests beschreiben die Funktionalität und dokumentieren Code.
- [ ] Tests zeigen automatisch Fehler in Code und helfen bei deren Behebung.
- [ ] Tests ermöglichen die sichere Weiterentwicklung von Code.

Code vervollständigen (lassen)

Manchmal hast du eine Idee im Kopf, wie du ein gegebenes Programmierproblem prinzipiell lösen möchtest, weisst aber nicht genau, wie du deinen Algorithmus bzw. geplanten Programmablauf in Code ausdrücken kannst?

Prinzipiell kannst du einem Chatbot den Auftrag auf verschiedene Arten erklären, z.B.:

- Beschreibung in natürlicher Sprache
- durch Pseudocode
- durch Struktogramme bzw. Nasi-Sneiderman-Diagrams
- durch Code-Fragmente

Wir haben bisher die zu lösenden Aufgaben meist in natürlicher Sprache unseren Chatbots beschrieben. Pseudocode ist schon etwas formaler; meist eine Mischung aus natürlicher Sprache und Anweisungen höherer Programmiersprachen. Struktogramme sind - wie der Name schon vermuten lässt - sehr strukturiert und lassen sich deshalb häufig automatisch in validen Code übersetzen.

Pseudocode und Struktogramme sind Themen für eine eigene Lerneinheit. In dieser Lerneinheit beschäftigen wir uns nur mit der 1. und der 4. Möglichkeit, also natürlicher Sprache und Code-Fragmenten.

Übung

Öffne das Jupyter-Chatbook `AI-Chatbot-Code-vervollständigen.ipynb`.

1. Lies die Erläuterungen und Demo-Eingaben gewissenhaft.
2. Führe alle Zellen Schritt für Schritt nacheinander aus.
3. Wenn du auf eine Zelle stösst, die du nicht auf Anhieb verstehst, experimentiere mit den Eingaben und notiere dir interessante Entdeckungen.
4. Besprich deine Entdeckungen mit deiner Mitschülerin/deinem Mitschüler neben dir.

Tipp

In Python kann `pass` verwendet werden, um einen Funktionskörper zu markieren, der noch nicht implementiert ist.

```
def funktionsname(parameter1, parameter2):
    pass
```

Wenn du das gewünschte Ergebnis einer Programmieraufgabe deinem Chatbot mit Hilfe von Code-Fragmenten und zugehörigen Tests detailliert erklärst, erhältst du häufig gute Lösungsvorschläge.

Zusammenfassung

- [ ] Code vervollständigen lassen funktioniert sehr gut in Kombination mit Tests.
- [ ] Nutze `pass` für Funktionskörper, die dein Chatbot für dich implementieren soll.

Programmieraufgaben lösen

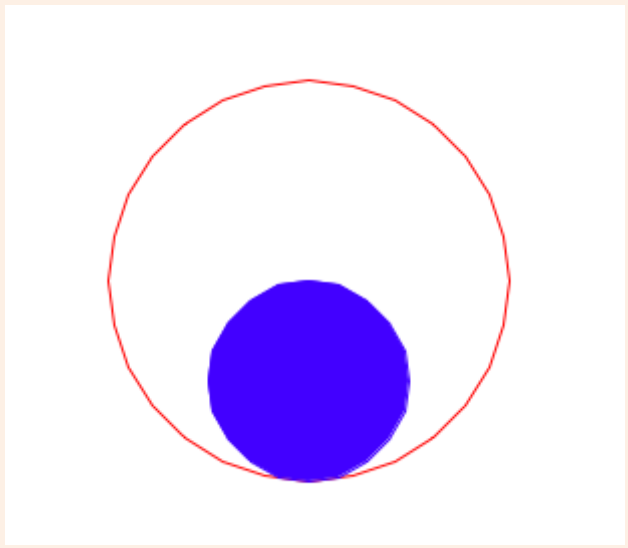
Du hast gelernt, wie du Chatbots selbst erstellst, diese konfigurierst und mit diesen kommunizierst. Du weisst jetzt, wie nützlich Tests sind und wie man diese (automatisch) erzeugt. Ausserdem weisst du, wie du ein Code-Gerüst von einem deiner Chatbot vervollständigen lassen kannst. Damit hast du das technische Rüstzeug, um mit Hilfe eines Chatbot effizient und schnell Programmieraufgaben korrekt zu lösen.

Fehlt noch etwas? Herausfordernd wird es für dich immer dann, wenn eine Programmieraufgabe komplex und nicht eindeutig beschrieben ist und somit Spielraum für Interpretation lässt. Diese beiden Aspekte des Programmierens werden wir in diesem Abschnitt behandeln.

Informatik erfordert von dir insbesondere logisches Denken und Abstraktion.

Aufgabe Kreis im Kreis

Erstelle mit Hilfe eines Chatbots Python-Code, der die folgende Figur erzeugt:



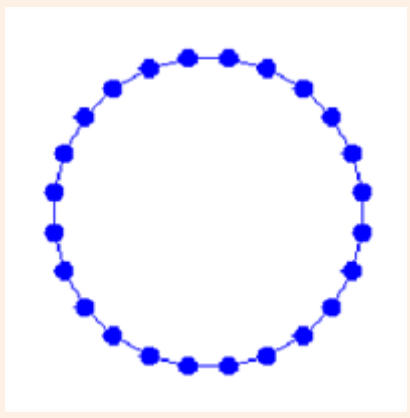
Tipp

Verwende als Eingabe eine präzise Beschreibung der einzelnen Elemente der Figur. Erzeuge die Elemente nacheinander mit separaten Eingaben. Übersetze die Eingaben ins Englische. Kombiniere dann beide Ausgaben zu (d)einer validen Lösung.

Du ziehst sowohl Nutzen aus den Vorschlägen des Chatbots, als auch aus der Interaktion mit diesem selbst. Warum? Das "Gespräch" mit dem Chatbot zwingt dich, deine Gedanken zu ordnen. Das ist so ähnlich wie beim sogenannten Rubber-Duck-Debugging, bei dem ein Programmierer sein Problem einer Gummiente auf seinem Schreibtisch erklärt und beim Erklären meist selbst darauf kommt, wo die Ursache seines Problems liegt.

Aufgabe Perlenkette

Programmieren mit Turtle: Verwende eine repeat- oder while-Schleife, um eine Perlenkette zu zeichnen. Die Anzahl Perlen kannst du beliebig wählen.



Vorgehen

- entwickle zunächst eine Lösung ohne Hilfe eines Chatbots
  - notiere dir deine Startzeit
  - notiere dir dein Vorgehen, insbesondere, wo du auf Hindernisse stösst
  - notiere dir deine Gesamtzeit
- entwickle nun eine Lösung mit Hilfe eines Chatbots
  - notiere dir deine Startzeit
  - öffne das Jupyter-Notebook `Perlenkette.ipynb` und verwende den vorkonfigurierten Chatbot `python-coder`
  - nutze für jede Anfrage an den Chatbot eine neue Zelle
  - starte deine Arbeit, indem du die Aufgabenstellung unverändert an den Chatbot übergibst
  - notiere dir, wo du auf Hindernisse stösst
  - notiere dir deine Gesamtzeit
- mit der Schülerin oder dem Schüler neben dir:
  - vergleicht eure Juypter-Chatbooks, wie seid ihr vorgegangen? Wieviele Schritte habt ihr bis zur Lösung benötigt?
  - besprecht, was euch beim Vergleichen aufgefallen ist, insbesondere wo Hindernisse auftraten und wie diese gelöst wurden.

Wie du bei den vorhergehenden Aufgaben erkennen konntest, solltest du deinem Chatbot eine Aufgabenstellung und das gewünschte Ergebnis möglichst präzise und ohne Zweideutigkeiten beschreiben. Häufig ist es für dich zielführend, eine komplexe Aufgabenstellung in einfachere Teilaufgaben zu zerlegen, diese lösen zu lassen und dann die Teillösungen zur Gesamtlösung zusammen zu setzen.

Zusammenfassung

- [ ] Beschreibe die Aufgabenstellung und das erwartete Ergebnis dem Chatbot präzise (genug) und ohne Zweideutigkeiten.
- [ ] Zerlege komplexe Aufgabenstellungen in einfacher zu lösende Teilaufgaben.
- [ ] Kombiniere abschliessend die Lösungen der Teilaufgaben zur Gesamtlösung.

Übungsaufgaben

Achtung

Ein Chatbot unterstützt dich beim Denken. Du bleibst derjenige, der die Aufgabenstellung verstehen muss. Du verantwortest die Lösungsstrategie und überprüfst das erstellte Programm. Fehler in deiner Lösung liegen in deiner Verantwortung, nicht beim Chatbot. Du musst den zugrundeliegenden Algorithmus und deinen Programmcode Zeile für Zeile verständlich erklären können.

Wie du in den vorhergehenden Abschnitten gelernt hast, kannst du dir die Arbeit erleichtern, indem du

- Aufgaben analysierst und in Teilaufgaben zerlegst.
- umfangreiche Tests erstellst.
- Codegerüste deiner Ideen vervollständigen lässt.
- Code und Algorithmen vom Chatbot erstellen und dir erklären lässt.

Um Programme auszuprobieren, verwende den Editor wie `Thonny` oder nutze ein Python-Jupyter-Notebook. Wichtig bei allen folgenden Aufgaben ist dein Lösungsweg, insbesondere aber, dass du deine Lösung wirklich verstehst und erklären kannst.

Für alle Übungsaufgaben findest du ein Jupyter-Notebook mit einer Musterlösung im Unterordner `Musterlösungen/`. Verwende eine neue Kopie des Jupyter-Notebooks `Vorlagen/Python-Coder.ipynb` als Startpunkt für jede der folgenden Aufgaben.

Aufgabe E ersetzen

Schreibe eine Funktion `ersetze_e_durch_E_in(eingabe)`, die das Zeichen **'e'** durch **'E'** in der Eingabe ersetzt. Überprüfe deine Lösung mit Hilfe von Tests.

```
def ersetze_e_durch_E_in(eingabe):
    pass
```

Beispiele:

- `assert ersetze_e_durch_E_in("Hallo Welt") == "Hallo WElt"`
- `assert ersetze_e_durch_E_in("Let me entertain you!") == "LEt mE EntErtain you!`

Aufgabe Buchstaben ersetzen

Schreibe eine Funktion, die drei Eingabeparameter besitzt: zwei Zeichen und eine Zeichenkette. Die Ausgabe bestehe aus der Zeichenkette in der Eingabe, jedoch seien alle Vorkommen des ersten Zeichens durch das zweite Zeichen ersetzt. Überprüfe deine Lösung mit Hilfe von Tests.

Beispiele:

- `assert ersetze("a", "x", "Hallo Welt") == "Hxllo Welt"`
- `assert ersetze("n", "p", "Let me entertain you!") == "Let me eptertaip you!"`
- `assert ersetze("!", "?", "Let me entertain you!") == "Let me entertain you?"`

Aufgabe Quersummenteilbarkeit

Schreibe eine Funktion welche bestimmt, ob eine natürliche Zahl ohne Rest durch ihre Quersumme teilbar ist. In diesem Fall soll diese Funktion `True` zurückgeben. Andernfalls `False`. Überprüfe deine Lösung mit Hilfe von Tests.

Beispiele:

- 12 => True
- 19 => False

Aufgabe 0-1-Vielfaches

Schreibe eine Funktion, die für eine natürliche Zahl als Eingabe das kleinste Vielfache berechnet, welches nur aus den Ziffern 0 und 1 besteht. Überprüfe deine Lösung mit Hilfe von Tests.

Beispielsweise wird für 55 als Eingabe 110 als Ausgabe erwartet.

Aufgabe Quersummenteilbarkeit-Reloaded

Schreibe eine Funktion, die alle natürlichen Zahlen kleiner 100 ausgibt, die ohne Rest durch ihre Quersumme teilbar sind.

Aufgabe Echte-Teiler-Teilbarkeit

Schreibe eine Funktion, die alle natürlichen Zahlen kleiner 100 ausgibt, die ohne Rest durch die Summe ihrer echten Teiler (incl. 1) teilbar ist. Überprüfe deine Lösung mit Hilfe von Tests.

Aufgabe Diverse-Quadratzahl

Schreibe eine Funktion, welche die erste Quadratzahl findet, die mindestens fünf unterschiedliche Ziffern enthält.

Aufgabe Freitagsfinder

Schreibe eine Funktion, welche für ein gegebenes Jahr die letzten Freitage eines Monats berechnet. Nutze diese Funktion, um alle Freitage für die Jahre 2000 bis 2100 zu berechnen.

Aufgabe 5-Wochenenden-Monate

Schreibe eine Funktion, die alle Monate mit fünf Wochenenden eines gegebenen Jahres findet. Nutze diese Funktion, um alle entsprechenden Monate für die Jahre 2000 bis 2100 zu berechnen.

Aufgabe Ursprung

Schreibe eine Funktion, die den Abstand eines Punktes vom Ursprung im euklidischen Koordinatensystem berechnet. Erweitere anschliessend die Funktion auf drei Dimensionen.

Aufgabe Schnittpunkt

Schreibe eine Funktion, die den Schnittpunkt von zwei Geraden im euklidischen Koordinatensystem berechnet.

Aufgabe Pangram

Du erhältst per E-Mail eine Textdatei (Daten/pangram.in), in der auf jeder Zeile eine Zeichenkette steht. Schreibe eine Funktion, die einen Dateipfad als Parameter erhält und für jede Zeile der Textdatei angibt, ob diese Zeile ein Pangram ist.

Pangram kommt aus dem Griechischen ("παν γράμμα") und bedeutet "jeder Buchstabe". Das wohl bekannteste Pangram in englischer Sprache lautet:

> "The quick brown fox jumps over the lazy dog."

da es jeden Buchstaben von a bis z enthält.

Drei-Kubikzahlen-Und-Ein-Teiler

Von drei aufeinanderfolgenden natürlichen Zahlen wird die dritte Potenz berechnet. Ist die Summe dieser drei Kubikzahlen immer durch 9 teilbar?

Schreibe ein Programm, welches unter ersten 10 Millionen natürlichen Zahlen ein Gegenbeispiel sucht.

Perfekte Division

Die Zahl 9 lässt sich als Bruch zweier Zahlen darstellen, wobei jede Ziffer von 1 bis 9 genau einmal vorkommt.

$$\frac{57429}{6381} = 9$$

Was passiert, wenn wir die Ziffer 0 hinzunehmen? Gibt es dann auch eine Division aus zwei Zahlen, die exakt 9 ergibt?

Hinweis

Jede der zehn Ziffern muss genau einmal vorkommen. Eine Zahl darf nicht mit der Ziffer 0 beginnen.

Wurzel (oldschool)

Berechne die Quadratwurzel einer natürlichen Zahl, ohne die Python-Funktionen `sqrt` oder `exp` oder den Potenz-Operator `**` zu verwenden.

Eine-Billion-Zikaden-Jahr

In den USA gibt es Zikaden, die alle 17 Jahre schlüpfen – und solche, die es alle 13 tun. In insgesamt 16 Bundesstaaten im Mittleren Westen und im Südosten des Landes überschneidenden sich die Verbreitungsgebiete der Zikaden.

Im Jahr 1803, Thomas Jefferson war Präsident, schlüpften sowohl die »Brut XVII« sowie die »Brut XIII« genannten Zikadenpopulationen zur selben Zeit. Im Jahr 2024 ist es wieder so weit. Das Land kann sich auf das Auftreten von Billionen von Zikaden einstellen.

Schreibe eine Funktion, welche für ein gegebenes Jahr berechnet, ob dies ein Eine-Billion-Zikaden-Jahr ist, also ob die 13- und die 17-Jahr-Zikaden gleichzeitig schlüpfen. In welchem Jahr nach 2024 tritt dieses ganz besondere Ereignis wieder auf?

Noch mehr Übungen ...

Aufgabe

Schlage in deinen Unterlagen drei Aufgabenstellungen aus dem Bereich "Programmieren mit Turtle" nach, die dir damals sehr schwierig erschienen und löse diese nochmals mittels Chatbot. Findest du auf diesem Weg elegantere Lösungen?

Python-Profi

Absolviere den Python-Track auf Exercism.org.

Olympischer Gedanke

Schaffst du die erste Runde bei der SOI?

Mach dein Ding

Löse vier Wochen lang die aktuelle Wochenaufgabe von The Weekly Challenge in Python.

Hole in One

Was beim klassischen Golf die Anzahl der Schläge, ist beim Code-Golf die Anzahl der Zeichen. Bist du besser als deine Kollegen?

Steile Lernkurve

Wie viele Tage schaffst du es, beim Adventofcode durchzuhalten? Bis zum Samichlaustag am 6. Dezember?

International Combat

Schaffst du es bei Codeforces.com in ein Turnier?

Anhang

Setup deines Computers

Damit Jupyter-Chatbooks ohne grösseren Support (durch deine Lehrpersonen) laufen, nutzen wir in dieser Lerneinheit NixOS.

USB-Stick

Nutze einen USB-Stick mit NixOS als Betriebssystem. So kannst du NixOS verwenden, ohne Software auf deinem Computer zu installieren.

Installation

Hast du noch einen (alten) Computer, der die aktuellen Windows 11 Anforderungen nicht erfüllt? Dann installiere dort direkt NixOS.

Oder aktiviere unter Windows Windows Subsystem for Linux . Dann hast du auch dort die Möglichkeit, genau wie unter Linux und Mac, NixOS als Paketmanager zu installieren.

Virtualbox

Oder installiere Virtualbox, um NixOS auszuführen. Lade dir das NixOS-Image herunter und starte dieses.

Docker

Oder installiere Docker, um NixOS auszuführen. Um NixOS zu starten, verwendest du im Terminal den Befehl

```
docker run -ti ghcr.io/nixos/nix
```

Erfolgs-Check

Ob du NixOS erfolgreich installiert hast, kannst du im Terminal so überprüfen:

```
nix-shell --version
```

Wenn kein Fehler gemeldet wird, ist alles bereit, um Jupyter-Chatbook zu starten.

Starten von Jupyter-Chatbook

Öffne ein Terminal und führe folgende Befehle aus:

a.) ggf. Ordner wechseln

```
cd $HOME/Documents
```

b.) die Unterrichtsunterlagen clonen

```
nix-shell -p git
```

```
git clone --depth=1 https://github.com/zero-overhead/programmieren-mit-LLM
cd programmieren-mit-LLM
```

c.) API-Keys angeben (diese Werte natürlich mit validen Schlüsseln ersetzen)

```
export OPENAI_API_KEY=sk-VE5rCA0sesZppklctgcOT3BlbkFJAVjrnvtqRpM1njjWdZeqvq
export MISTRAL_API_KEY=mucG8sAMn8Lb8mRvTPJqsH3ElbDoowvwFw
```

d.) Jupyter starten

```
nix-shell --run "jupyter lab"
```

e.) Falls du Python-Code ausprobieren möchtest, nutze doch `thonny`:

```
nix-shell --run "thonny"
```

Was du erreicht hast

- [ ] Du hast NixOS installiert und der Befehl `nix-shell --version` funktioniert.
- [ ] Du kannst API-Keys für OpenAI und MistralAI hinterlegen.
- [ ] Du kannst Jupyter starten.
- [ ] Du kannst Thonny starten.

Datenschutz

In Jupyter verwenden wir alle den gleichen API-Key, um auf die Modelle zuzugreifen. Somit sind deine Eingaben pseudomisiert. Wenn du keine Bedenken bezüglich des Schutzes deiner Eingaben und persönlichen Daten hast, kannst du dich auch bei z.B. ChatGPT mit (d)einem Microsoft-, Google- oder Apple-Konto direkt anmelden.

GPT/LLM/API-Key - Was bedeutet das und wie funktioniert das grundsätzlich?

Dieser Abschnitt dient deinem Verständnis der Gesamtzusammenhänge. In dieser Lektion ging es aber nicht darum, wie LLM prinzipiell funktionieren und welche verschiedenen Ansätze es gibt. Wir fokussierten uns darauf, wie dir LLM ganz praktisch beim Programmieren helfen können. Vergleiche es mit der Fahrschule: Um ein Auto zu fahren, musst du nicht wissen, wie der Motor, die Radaufhängung und der Katalysator ganz genau funktionieren, oder?

Wenn wir von AI reden, musst du zwischen kommerziellen Anbietern wie OpenAI, MistralAI oder Google und den eigentlichen Rechenmodellen wie etwa `gpt-3.5-turbo` oder `mistral-tiny` und den Benutzerschnittstellen unterscheiden.

Die Schnittstelle zwischen Mensch und Computer wird als Benutzerschnittstelle oder User Interface (UI) bezeichnet. Dazu gehören grafische Benutzeroberflächen (GUI), Text-basierte Benutzeroberflächen (CLI), Touchscreens, Tastatur- und Maussteuerungen, sowie Spracherkennungssysteme.

Die verschiedenen Anbieter stellen über ein sogenanntes Application Programming Interface - API - den Zugriff auf ihre Modelle zur Verfügung und verlangen dafür i.d.R. Geld. Damit sich das auf deinem Rechner laufende `Jupyter-Chatbook` oder bspw. das Programm `openai-playground` - deine Benutzerschnittstellen - mit der API eines der Anbieter verbinden kann, benötigst du einen Schlüssel, einen sogenannten API-Key. Den bekommst du von deiner Lehrperson. Über diesen Schlüssel rechnet der Anbieter die Zugriffe ab. Wenn der Zugriff auf ein Modell von einem Anbieter kostenlos angeboten wird, frage dich, was oder wer das Produkt ist, welches das wirtschaftliche Überleben des Anbieters sichert.

Wo ist der Unterschied zwischen ChatGPT und Jupyter-Chatbook?

Der Unterschied zwischen ChatGPT und Jupyter-Chatbook ist, dass du mit ChatGPT nur auf Modelle von OpenAI zugreifen kannst. Bei Jupyter-Chatbooks bist du frei in deiner Wahl und kannst Modelle verschiedener Anbieter sogar gleichzeitig nutzen.

Das eingangs verwendete `openai-playground` funktioniert prinzipiell wie das bekannte `ChatGPT`. Beides sind Programme, die Eingaben von Usern an spezialisierte, i.d.R. in der Cloud laufende Modelle (`GPT` - Generative Pre-trained Transformer) weiterleiten, welche darauf trainiert wurden, aus einer Eingabe eine dazu passende Ausgabe zu berechnen. Für das Training wurden riesige Mengen von im Internet verfügbaren Texten verwendet. Ein `GPT`-Modell, man spricht allgemeiner auch von Large Language Models - `LLM`, transformiert eine Eingabe wie etwa "Erzähle einen Witz" in eine Zeichenkette, die im Zusammenhang mit ähnlichen Eingaben in den Trainingsdaten aufgetaucht ist. Ein LLM versteht nicht, was ein Witz ist und warum etwas lustig ist. Auch kann ein LLM implizit nur auf die Daten zurückgreifen, die beim Training zur Verfügung standen. Ausgaben werden als statistisch wahrscheinliche Fortsetzung der Eingabe errechnet, was dazu führt, dass prinzipiell zu jeder Eingabe eine Ausgabe gemacht werden kann, ohne jedoch Aussagen über Wahrheit oder Korrektheit machen zu können. In dem Zusammenhang spricht man dann auch von **Halluzinieren**.

Halluzinieren: Wenn Modelle objektiv falsche Antworten geben, spricht man von Halluzinieren.

Zusammenfassung

- [ ] GPT ist die Abkürzung für "Generative Pre-trained Transformer" und beschreibt Berechnungsmodelle, die bspw. hinter ChatGPT stehen.
- [ ] LLM bedeutet Large Language Model und ist ein Oberbegriff.
- [ ] LLM berechnen aus Text-Eingaben statistisch passende Text-Ausgaben, haben aber kein Verständnis im herkömmlichen Sinn über den Inhalt.
- [ ] API bedeutet Application Programming Interface und regelt den Zugriff auf bspw. ein LLM.
- [ ] Anbieter wie OpenAI oder Google geben über ein API i.d.R. kostenpflichtigen Zugriff auf ihre LLM.

---

1. OK, hier wurde ein bisschen geschummelt. Die sprechende Kuh wurde durch das Programm `cowsay` erzeugt. Hier ist der ganze Code. ↩