## 4.1 Sprachmodelle: Regeln oder Statistik?

**Leitidee**

Du schreibst einem Chatbot eine Frage, und Sekunden später steht eine Antwort da, die klingt, als hätte ein Mensch sie geschrieben. Aber woher „weiß“ das Programm, was es antworten soll? Es gibt dafür zwei grundverschiedene Ideen. Die erste: Menschen schreiben **Regeln** auf, nach denen das Programm reagiert. Die zweite: Das Programm **zählt** in sehr vielen Texten, welche Wörter oft aufeinanderfolgen, und rechnet damit. In diesem Abschnitt lernst du beide Ideen kennen. Du programmierst einen kleinen Gesprächspartner nach festen Regeln und findest heraus, wo seine Grenzen liegen.

**Lernziele**

Nach diesem Abschnitt kannst du

- erläutern, was ein Sprachmodell ist,
- beschreiben, wie ein regelbasiertes System wie ELIZA funktioniert,
- ein kleines regelbasiertes Programm in Python testen und um eigene Regeln erweitern,
- die Grundidee eines n-Gramm-Modells beschreiben,
- regelbasierte Systeme und n-Gramm-Modelle unterscheiden und ihre Grenzen benennen.

### 4.1.1 Was ist ein Sprachmodell?

Menschen lernen Sprache, indem sie zuhören, lesen und sprechen. Dabei bekommst du mit der Zeit ein Gefühl dafür, welche Wörter zusammenpassen. Lies den folgenden Satzanfang:

> Ich trinke morgens eine Tasse …

Wahrscheinlich hast du sofort „Kaffee“ oder „Tee“ ergänzt – und nicht „Fahrrad“. Niemand hat dir dafür eine Regel beigebracht. Du hast diesen Satzanfang einfach schon sehr oft mit genau diesen Wörtern gehört oder gelesen.

Ein Computer kann so etwas nicht „fühlen“. Man kann ihm aber ein **Modell** der Sprache mitgeben: eine vereinfachte Beschreibung, wie Sätze aufgebaut sind und welche Wörter wie oft zusammen vorkommen. Damit kann ein Programm Texte untersuchen oder selbst neue Texte erzeugen. So ein Programm heißt **Sprachmodell**.

**Definition: Sprachmodell**

Ein **Sprachmodell** ist ein Computersystem, das ein vereinfachtes Modell einer Sprache enthält: wie Sätze aufgebaut sind und wie wahrscheinlich bestimmte Wortfolgen sind. Damit kann es Texte untersuchen oder neue Texte erzeugen.

Das Wort *Wahrscheinlichkeit* kennst du aus Mathe: Sie gibt an, wie sicher ein Ereignis eintritt, als Zahl zwischen 0 und 1 oder in Prozent. Bei Sprachmodellen geht es zum Beispiel um die Frage: Wie wahrscheinlich ist es, dass nach „eine Tasse“ das Wort „Tee“ folgt?

Sprachmodelle gibt es schon seit Jahrzehnten. Historisch gab es zwei wichtige Ansätze, die du in diesem Abschnitt kennenlernst: **regelbasierte Systeme** und **n-Gramm-Modelle**. Die heutigen großen Chatbots bauen auf dem zweiten Ansatz auf.

**Merke**

Ein **Sprachmodell** ist ein System, das sprachliche Strukturen bzw. Wahrscheinlichkeiten erfasst, um Texte zu analysieren oder neu zu erzeugen.

### 4.1.2 Regelbasierte Systeme: ELIZA

**Ein Programm, das zuhört – scheinbar**

Im Jahr 1966 veröffentlichte der Informatiker **Joseph Weizenbaum** am Massachusetts Institute of Technology (MIT) in den USA ein Programm namens **ELIZA**. Man konnte ihm Sätze eintippen, und es antwortete. In seiner bekanntesten Fassung spielte ELIZA eine Psychotherapeutin, die vor allem Rückfragen stellt. Ein Gespräch konnte etwa so aussehen:

```
Mensch:  Ich fühle mich müde.
ELIZA:   Warum fühlst du dich müde?
Mensch:  Ich brauche mehr Schlaf.
ELIZA:   Warum brauchst du mehr Schlaf?
```

Das wirkt, als würde ELIZA zuhören und mitdenken. Tatsächlich versteht das Programm **kein einziges Wort**. Es sucht im eingegebenen Satz nach bestimmten **Mustern**, zum Beispiel nach „Ich fühle mich“. Findet es ein Muster, setzt es den Rest des Satzes in eine vorbereitete Antwort ein. Weizenbaum war erschrocken, wie ernst manche Menschen das Programm trotzdem nahmen und ihm persönliche Dinge anvertrauten.

Das Wissen von ELIZA steckt also vollständig in Regeln, die ein Mensch vorher aufgeschrieben hat. Jede Regel hat die Form **„Wenn ..., dann ...“** – genau wie eine Verzweigung mit `if` in Python.

**Definition: Regelbasiertes System**

Ein **regelbasiertes System** reagiert nach **fest programmierten Regeln**. Es erkennt Muster in der Eingabe (z. B. bestimmte Wörter) und gibt die Antwort aus, die für dieses Muster vorgesehen ist.

**Merke**

**Regelbasierte Systeme** (z. B. das historische Programm **ELIZA**, 1966) reagieren auf Basis **fest programmierter Regeln** und Mustererkennung, z. B.: „Wenn der Satz ‚Ich fühle mich ...‘ enthält, antworte mit ‚Warum fühlst du dich ...?’ “. Sie „verstehen“ die Sprache nicht, sondern wenden feste Muster an.

**Ein Werkzeug: Textteile ersetzen**

Für unser eigenes kleines ELIZA brauchst du ein neues Werkzeug. Aus dem Satz „ich fühle mich müde“ soll das Programm den Rest „müde“ herausholen. Das geht, indem man den Anfang „ich fühle mich “ durch **nichts** ersetzt.

**Werkzeug**

Mit `text.replace(alt, neu)` erhältst du eine Kopie von `text`, in der jedes Vorkommen von `alt` durch `neu` ersetzt ist. Der ursprüngliche Text bleibt unverändert. Ersetzt du durch die leere Zeichenkette `""`, wird der Teil einfach entfernt:

```python
satz = "ich fühle mich müde"
print(satz.replace("fühle", "bin"))
print(satz.replace("ich fühle mich ", ""))
```

AUSGABE

```
ich bin mich müde
müde
```

Achte auf das Leerzeichen am Ende von `"ich fühle mich "`. Ohne dieses Leerzeichen würde der Rest mit einem Leerzeichen beginnen: `" müde"`.

### Ein Mini-ELIZA in Python

Das folgende Programm ist ein stark vereinfachtes ELIZA. Es kennt nur vier Regeln:

| Wenn der Satz ... enthält | antwortet Mini-ELIZA |
|---|---|
| „ich fühle mich …“ | „Warum fühlst du dich …?“ |
| „ich brauche …“ | „Warum brauchst du …?“ |
| ein Fragezeichen | „Interessante Frage - was denkst du selbst darüber?“ |
| sonst | „Erzähl mir mehr davon.“ |

Die Regeln werden von oben nach unten geprüft. Die erste passende Regel gewinnt. Im Programm wird daraus eine Funktion mit `if`, `elif` und `else`:

```python
def mini_eliza(satz):
    satz = satz.lower()
    if "ich fühle mich" in satz:
        gefuehl = satz.replace("ich fühle mich ", "")
        return f"Warum fühlst du dich {gefuehl}?"
    elif "ich brauche" in satz:
        bedarf = satz.replace("ich brauche ", "")
        return f"Warum brauchst du {bedarf}?"
    elif "?" in satz:
        return "Interessante Frage - was denkst du selbst darüber?"
    else:
        return "Erzähl mir mehr davon."

print(mini_eliza("Ich fühle mich müde"))
print(mini_eliza("Ich brauche mehr Zeit"))
print(mini_eliza("Wie geht es dir?"))
print(mini_eliza("Die Sonne scheint"))
```

AUSGABE

```
Warum fühlst du dich müde?
Warum brauchst du mehr zeit?
Interessante Frage - was denkst du selbst darüber?
Erzähl mir mehr davon.
```

- **Zeile 1:** Die Funktion `mini_eliza` bekommt einen Satz als Parameter und gibt eine Antwort mit `return` zurück.
- **Zeile 2:** Der Satz wird in Kleinbuchstaben umgewandelt. So erkennt das Programm „Ich fühle mich“ und „ich fühle mich“ gleichermaßen.

- **Zeilen 3–5:** Die erste Regel. Steht „ich fühle mich“ im Satz, wird dieser Teil entfernt. Was übrig bleibt (hier „müde“), wird mit einem f-String in die Antwort eingesetzt.
- **Zeilen 6–8:** Die zweite Regel funktioniert genauso mit „ich brauche“.
- **Zeilen 9–10:** Enthält der Satz ein Fragezeichen, kommt eine allgemeine Antwort, die zum Nachdenken auffordert.
- **Zeilen 11–12:** Passt keine Regel, antwortet das Programm mit einem Satz, der fast immer irgendwie passt.
- **Zeilen 14–17:** Vier Testsätze. Für jeden wird die Antwort ausgegeben.

Schau dir die zweite Ausgabezeile genau an: „mehr zeit“ ist kleingeschrieben. Das Programm hat den ganzen Satz in Zeile 2 kleingeschrieben und setzt den Rest unverändert ein. Ein Mensch würde diesen Fehler nie machen. Das Programm merkt ihn nicht, denn es weiß nicht, dass „Zeit“ ein Nomen ist.

**Versuch: Probiere es aus**

Tippe das Programm ab und teste es mit eigenen Sätzen. Ändere auch die Reihenfolge der Regeln: Was antwortet das Programm auf „Ich fühle mich gut, und du?“, wenn die Regel mit dem Fragezeichen ganz oben steht?

**Stolperstein**

Wenn du `mini_eliza` mit `print` aufrufst und statt einer Antwort `None` erscheint, fehlt in einem Zweig das `return`. Dann gibt die Funktion nichts zurück. Prüfe, ob in **jedem** Zweig (`if`, `elif`, `else`) eine `return`-Anweisung steht.

### Wo die Regeln an ihre Grenzen stoßen

Mini-ELIZA wirkt nur dann klug, wenn der Satz genau zu einem Muster passt. Schon kleine Änderungen bringen es aus dem Tritt. Die Funktion `mini_eliza` von oben wird hier mit zwei neuen Sätzen aufgerufen:

```
print(mini_eliza("Heute fühle ich mich müde"))
print(mini_eliza("Ich brauche dich"))
```

AUSGABE

```
Erzähl mir mehr davon.
Warum brauchst du dich?
```

Im ersten Satz stehen die Wörter in anderer Reihenfolge („fühle ich mich“). Das Muster passt nicht, und das Programm weicht auf seine Allerweltsantwort aus. Im zweiten Satz passt das Muster, aber das Programm übernimmt „dich“ wörtlich. Richtig wäre „Warum brauchst du mich?“. Um das zu können, müsste das Programm wissen, wer mit „dich“ gemeint ist. Das weiß es nicht.

Natürlich könnte man für jeden dieser Fälle eine weitere Regel schreiben. Aber Sprache ist unglaublich vielfältig: Es gibt unzählige Arten, dasselbe zu sagen. Man müsste unüberschaubar viele Regeln schreiben – und die Regeln dürfen sich nicht widersprechen.

**Arbeitsauftrag 1: Grenzen des regelbasierten Ansatzes**

Teste `mini_eliza` mit mindestens **drei eigenen Sätzen**, die **nicht** zu den vorgesehenen Mustern passen oder bei denen eine seltsame Antwort entsteht. Notiere Satz und Antwort. Beschreibe jeweils, warum das System hier keine sinnvolle Antwort liefern kann.

Meine Testsätze, Antworten und Erklärungen:

**Arbeitsauftrag 2: Regel erweitern**

Erweitere `mini_eliza` um mindestens **eine weitere Regel** deiner Wahl, zum Beispiel eine Reaktion auf „Ich mag …“ oder „Ich hasse …“. Nenne die neue Funktion `mini_eliza_erweitert` und teste sie mit passenden Sätzen.

### 4.1.3 n-Gramm-Modelle: Statistik statt Regeln

**Die Idee: Zählen, was oft zusammen vorkommt**

Erinnere dich an den Satzanfang „Ich trinke morgens eine Tasse …“. Du hast das fehlende Wort nicht mit einer Regel gefunden, sondern aus **Erfahrung**: Nach „eine Tasse“ kam in deinem Leben oft „Kaffee“ oder „Tee“.

Genau diese Idee nutzt ein **n-Gramm-Modell**. Statt Regeln aufzuschreiben, gibt man dem Computer sehr viele Texte. Er **zählt**, welche Wörter wie oft direkt hintereinander stehen. Eine solche Folge von aufeinanderfolgenden Wörtern heißt **n-Gramm**. Das *n* steht für die Anzahl der Wörter: Ein Wortpaar wie „eine Tasse“ ist ein 2-Gramm, man sagt auch **Bigramm**.

Ein ganz kleines Beispiel. Der Computer bekommt diese drei Sätze als Trainingstext:

```
ich trinke eine tasse tee
ich trinke eine tasse kaffee
ich trinke eine tasse tee
```

Nach dem Wort „tasse“ folgt zweimal „tee“ und einmal „kaffee“. Soll das Modell den Satz „ich trinke eine tasse …“ fortsetzen, ist „tee“ das wahrscheinlichste nächste Wort. Das Modell hat keine Regel über Getränke gelernt. Es hat nur gezählt.

Diese Vorhersage des nächsten Wortes heißt auf Englisch **next word prediction**. Die Wortvorschläge, die dir beim Tippen auf dem Handy über der Tastatur angezeigt werden, beruhen auf einer ähnlichen Idee: Das Programm schlägt Wörter vor, die nach dem bisher Getippten wahrscheinlich sind.

**Definition: n-Gramm-Modell**

Ein **n-Gramm-Modell** ist ein Sprachmodell, das aus einer großen Textsammlung zählt, wie häufig bestimmte Folgen von $n$ Wörtern (n-Gramme) vorkommen. Aus diesen Häufigkeiten berechnet es, welches Wort wahrscheinlich als nächstes folgt.

**Merke**

**n-Gramm-Modelle** basieren auf der **statistischen Analyse großer Textmengen**: Sie erfassen, wie häufig bestimmte Wortfolgen (n-Gramme) in einer Textsammlung vorkommen, und nutzen diese Häufigkeiten zur Vorhersage des nächsten Wortes.

Wie man n-Gramme genau zählt und daraus Wahrscheinlichkeiten berechnet, lernst du im nächsten Abschnitt – zuerst von Hand, dann mit Python.

### Und die modernen Chatbots?

Die Chatbots, die du vielleicht kennst, beruhen auf sogenannten **Large Language Models**, kurz **LLMs** (englisch für „große Sprachmodelle“). Sie wurden mit riesigen Textmengen trainiert und benutzen sehr komplexe Rechenverfahren, sogenannte **neuronale Netze**. Wie diese im Einzelnen funktionieren, ist nicht Thema dieses Skripts. Die Grundidee ist aber dieselbe wie beim n-Gramm-Modell: Ein LLM schätzt, welches Wort (genauer: welcher Wortbaustein) als nächstes am wahrscheinlichsten passt, hängt es an und wiederholt das immer wieder.

Gymnasium Ottweiler

**Merke**

Moderne **Large Language Models (LLMs)** nutzen komplexe neuronale Netze, beruhen aber im Kern auf einem ähnlichen Prinzip: Sie schätzen **Wahrscheinlichkeiten für die Fortsetzung eines Textes**.

Deshalb lohnt es sich, das n-Gramm-Modell genau zu verstehen. An ihm kannst du vollständig nachrechnen, was im Inneren passiert – bei einem LLM ist das nicht mehr möglich.

### 4.1.4 Regeln und Statistik im Vergleich

Die beiden Ansätze unterscheiden sich vor allem darin, **woher das Wissen** des Systems kommt:

| | **regelbasiertes System** | **n-Gramm-Modell** |
|---|---|---|
| Woher kommt das Wissen? | Ein Mensch schreibt Regeln auf. | Der Computer zählt in Texten. |
| Was steht im Programm? | „Wenn …, dann …“-Regeln | Tabellen mit Häufigkeiten |
| Neue Formulierungen | nur, wenn eine Regel passt | möglich, wenn ähnliche Wortfolgen im Text vorkamen |
| Verhalten nachvollziehbar? | ja, jede Antwort folgt aus einer Regel | ja, jede Wahl folgt aus den Zahlen |
| Versteht es die Bedeutung? | nein | nein |

Die letzte Zeile ist wichtig: **Keines** der beiden Systeme versteht, was es schreibt. Das regelbasierte System erkennt Zeichenfolgen, das n-Gramm-Modell rechnet mit Häufigkeiten.

**So kannst du es sagen**

So beschreibst du die Ansätze in Fachsprache – vom einfachen zum genauen Satz:

- *„ELIZA hat Wenn-dann-Sätze.“ → „ELIZA ist ein regelbasiertes System: Es erkennt Muster in der Eingabe und reagiert nach fest programmierten Regeln.“*
- *„Das Modell guckt, was oft kommt.“ → „Das n-Gramm-Modell ermittelt aus einer Textsammlung, wie häufig Wortfolgen vorkommen, und sagt daraus das wahrscheinlichste nächste Wort voraus.“*
- *„Der Chatbot versteht nichts.“ → „Das System verarbeitet Zeichenfolgen bzw. Häufigkeiten, ohne die Bedeutung der Wörter zu erfassen.“*

### 4.1.5 Übungen

**Arbeitsauftrag 3: Regeln oder Statistik?**

Entscheide jeweils, ob das beschriebene Programm **regelbasiert** arbeitet oder nach dem Prinzip eines **n-Gramm-Modells**. Begründe in einem Satz.

1. Ein Programm antwortet auf jede E-Mail, die das Wort „Urlaub“ enthält, mit „Ich bin bis Montag nicht erreichbar.“
2. Ein Programm schlägt beim Tippen das Wort vor, das in einer großen Sammlung von Nachrichten am häufigsten nach dem zuletzt getippten Wort stand.
3. Ein Spiel-Chatbot erkennt die Wörter „hallo“, „tschüss“ und „hilfe“ und hat für jedes eine feste Antwort.

4. Ein Programm zählt in allen Märchen der Brüder Grimm, welches Wort wie oft nach „es war“ folgt, und ergänzt damit angefangene Sätze.

Meine Antworten:

**Arbeitsauftrag 4: Vergleich in Worten**

Beantworte schriftlich:

1. Warum stößt ein rein regelbasiertes System bei der Vielfalt natürlicher Sprache schnell an Grenzen?
2. Welchen grundsätzlich anderen Ansatz verfolgen n-Gramm-Modelle (Stichwort: Regeln oder Statistik)?

Diese Fragen werden in den nächsten Abschnitten praktisch vertieft.

Gymnasium Ottweiler

Meine Antwort:

### 4.1.6 Weitere Übungen

Hier kannst du das Gelernte weiter üben. Die Punkte im Titel zeigen, wie schwer eine Aufgabe ist: ●○○ leicht, ●●○ mittel, ●●● knifflig. Fang mit den leichten an.

**Arbeitsauftrag 5: Richtig oder falsch? ●○○**

Entscheide, ob die Aussage richtig oder falsch ist. Trage „r“ oder „f“ ein. Verbessere die falschen Aussagen unten in einem Satz.

| Nr. | Aussage | r oder f? |
|---|---|---|
| 1 | Ein Sprachmodell erfasst sprachliche Strukturen bzw. Wahrscheinlichkeiten, um Texte zu analysieren oder zu erzeugen. | |
| 2 | ELIZA versteht die Bedeutung der Sätze, die man eintippt. | |
| 3 | Ein regelbasiertes System reagiert nach Regeln, die vorher fest programmiert wurden. | |
| 4 | Ein n-Gramm-Modell zählt in Texten, wie oft bestimmte Wortfolgen vorkommen. | |
| 5 | Ein n-Gramm-Modell braucht für jede mögliche Eingabe eine eigene Wenn-dann-Regel. | |
| 6 | Moderne LLMs arbeiten ganz anders als n-Gramm-Modelle: Sie schätzen keine Wahrscheinlichkeiten. | |

Verbesserungen:

Gymnasium Ottweiler

### Arbeitsauftrag 6: Was antwortet Mini-ELIZA? ●○○

Bestimme **ohne Computer**, was die Funktion `mini_eliza` aus dem Abschnitt *Ein Mini-ELIZA in Python* auf die folgenden Sätze antwortet. Denke daran: Der Satz wird zuerst kleingeschrieben, und die Regeln werden von oben nach unten geprüft. Die erste passende Regel gewinnt.

| Satz | Antwort von `mini_eliza` |
|---|---|
| Ich fühle mich super | |
| Ich brauche eine Pause | |
| Hast du Hunger? | |
| Mein Handy ist kaputt | |
| Ich brauche Hilfe, kannst du mir helfen? | |

### Arbeitsauftrag 7: Regeln oder Statistik? ●○○

Entscheide jeweils, ob das Programm **regelbasiert** arbeitet oder nach dem Prinzip eines **n-Gramm-Modells**. Begründe kurz.

1. Ein Chatbot eines Online-Shops schickt auf jede Nachricht, in der „Rücksendung“ vorkommt, einen Link zum Rücksendeformular.
2. Ein Programm hat Tausende Chatnachrichten ausgewertet. Nach „bis“ schlägt es das Wort vor, das dort am häufigsten folgte, zum Beispiel „morgen“.
3. Ein Lernspiel prüft, ob deine Antwort das Wort „Photosynthese“ enthält. Wenn ja, zeigt es „Richtig!“.
4. Ein Programm hat alle Bände einer Fantasy-Buchreihe ausgezählt. Es setzt angefangene Sätze fort, indem es immer ein Wort wählt, das dort häufig auf die letzten Wörter folgte.
5. Ein Sprachassistent im Auto zeigt die nächste Tankstelle an, sobald im gesprochenen Satz das Wort „Tankstelle“ vorkommt.

Gymnasium Ottweiler

Meine Antworten:

**Arbeitsauftrag 8: Der Kiosk-Bot ●●○**

Der Schulkiosk hat einen kleinen regelbasierten Chatbot:

```python
def kiosk_bot(satz):
    satz = satz.lower()
    if "brötchen" in satz:
        return "Brötchen gibt es ab 9:30 Uhr."
    elif "kostet" in satz:
        return "Alle Getränke kosten 1 Euro."
    elif "?" in satz:
        return "Das weiß ich leider nicht."
    else:
        return "Hallo! Frag mich etwas zum Kiosk."
```

1. Bestimme **ohne Computer** die Antworten auf die Sätze in der Tabelle. Trage ein, ob die Antwort zur Frage passt (ja/nein).
2. Erkläre bei jeder unpassenden Antwort, warum das Programm sie gibt.

| Satz | Antwort | passt? |
|---|---|---|
| Wann gibt es Brötchen? | | |
| Was kostet ein Brötchen? | | |
| Wie teuer ist ein Kakao? | | |
| Was kostet ein Wasser? | | |
| Ich hätte gern einen Kakao. | | |

Meine Erklärungen:

**Arbeitsauftrag 9: Regeln für die Schulbibliothek ●●○**

Die Schulbibliothek möchte einen regelbasierten Chatbot.

1. Formuliere **drei Regeln** in der Form „Wenn der Satz … enthält, antworte: …“.
2. Nenne **zwei Sätze**, die eine Schülerin schreiben könnte, bei denen deine Regeln eine unpassende oder gar keine passende Antwort liefern. Erkläre jeweils, warum.

Meine Regeln und Beispiele:

Gymnasium Ottweiler

**Arbeitsauftrag 10: Den Kiosk-Bot verbessern ●●●**

Schreibe am Computer eine Funktion `kiosk_bot_2`, die zwei Schwächen des Kiosk-Bots behebt:

1. Fragt jemand nach dem **Preis eines Brötchens** (der Satz enthält „brötchen“ und „kostet“), antwortet sie: „Ein Brötchen kostet 50 Cent.“
2. Das Wort „teuer“ soll genauso behandelt werden wie „kostet“.

Alle anderen Antworten bleiben wie beim Kiosk-Bot. Teste deine Funktion mit den fünf Sätzen aus dem Auftrag *Der Kiosk-Bot* und zusätzlich mit „Ist ein Brötchen teuer?“.

**Tipp**

Überlege genau, in welcher **Reihenfolge** die Regeln stehen müssen. Eine Regel, die zwei Bedingungen braucht, muss vor den Regeln stehen, die nur eine davon prüfen. Bedingungen verknüpfst du mit `and` und `or`. Klammern helfen, den Überblick zu behalten.

### 4.1.7 Das Wichtigste in Kürze

- Ein **Sprachmodell** erfasst sprachliche Strukturen bzw. Wahrscheinlichkeiten, um Texte zu analysieren oder zu erzeugen.
- **Regelbasierte Systeme** wie ELIZA (1966) reagieren nach fest programmierten Wenn-dann-Regeln und erkennen nur Muster in der Eingabe.
- Regelbasierte Systeme scheitern an der Vielfalt der Sprache: Für jede Formulierung bräuchte man eine eigene Regel.
- **n-Gramm-Modelle** zählen in großen Textsammlungen, wie oft Wortfolgen (n-Gramme) vorkommen, und sagen damit das nächste Wort voraus (**next word prediction**).
- Moderne **LLMs** nutzen neuronale Netze, schätzen aber ebenfalls Wahrscheinlichkeiten für die Fortsetzung eines Textes.
- Keines dieser Systeme versteht die Bedeutung dessen, was es verarbeitet.

**Selbstcheck**

- ☐ Ich kann erläutern, was ein Sprachmodell ist.
- ☐ Ich kann beschreiben, wie ELIZA auf einen Satz reagiert.
- ☐ Ich kann eine Funktion wie `mini_eliza` testen und um eine Regel erweitern.
- ☐ Ich kann an Beispielen zeigen, wo regelbasierte Systeme an ihre Grenzen stoßen.
- ☐ Ich kann die Grundidee eines n-Gramm-Modells mit eigenen Worten erklären.
- ☐ Ich kann regelbasierte Systeme und n-Gramm-Modelle an einem Beispiel unterscheiden.