4.6 Kapitel 4 kompakt: der Lernzettel

Leitidee

Hier steht das Wichtigste aus Kapitel 4 auf wenigen Seiten, zum Wiederholen vor einem GLN. Wenn dir etwas unklar ist, lies im passenden Abschnitt nach: Dort ist alles ausführlich erklärt, mit Beispielen und Aufgaben.

4.6.1 Sprachmodelle: Regeln oder Statistik? (Abschnitt 4.1)

- Ein **Sprachmodell** ist ein System, das sprachliche Strukturen bzw. Wahrscheinlichkeiten erfasst, um Texte zu analysieren oder neu zu erzeugen.
- **Regelbasierte Systeme** (z. B. **ELIZA**, 1966) reagieren nach **fest programmierten Regeln** und Mustererkennung: „Wenn der Satz ‚ich fühle mich …' enthält, antworte ‚Warum fühlst du dich …?' ". Die Regeln werden von oben nach unten geprüft, die erste passende gewinnt.
- **n-Gramm-Modelle** zählen in großen Textmengen, wie häufig Wortfolgen vorkommen, und sagen damit das **nächste Wort** voraus.
- Moderne **LLMs** (Large Language Models) nutzen komplexe neuronale Netze, schätzen im Kern aber ebenfalls **Wahrscheinlichkeiten für die Fortsetzung eines Textes**.

	regelbasiertes System	**n-Gramm-Modell**
Woher kommt das Wissen?	Menschen schreiben Regeln.	Der Computer zählt in Texten.
Grenze	Für jede Formulierung bräuchte man eine Regel.	Kann nur, was im Korpus vorkam.
Versteht es die Bedeutung?	nein	nein

Typische Fehler: Meinen, ELIZA „verstehe" den Satz. Vergessen, dass die Reihenfolge der Regeln entscheidet, wenn mehrere passen.

4.6.2 n-Gramme zählen, Wahrscheinlichkeiten berechnen (Abschnitte 4.2 und 4.3)

- Ein **n-Gramm** ist eine Folge von n aufeinanderfolgenden Wörtern (oder Zeichen). Die Stücke überlappen sich.

Name	**n**	**aus „der hund bellt laut"**
Unigramm	1	der · hund · bellt · laut
Bigramm	2	der hund · hund bellt · bellt laut
Trigramm	3	der hund bellt · hund bellt laut

- Der **Textkorpus** ist der Trainingstext. Bei der **Textkörperanalyse** zählt man, wie oft jedes n-Gramm darin vorkommt.
- Ein Text mit 9 Wörtern hat 8 Bigramme: Das letzte Wort hat keinen Nachfolger.
- Der **Kontext** ist das, worauf das Modell schaut: beim Bigramm ein Wort, beim Trigramm zwei Wörter.
- **Wahrscheinlichkeit** eines Folgewortes:

$$\text{Wahrscheinlichkeit} = \frac{\text{Anzahl des n-Gramms „Kontext Folgewort"}}{\text{Anzahl aller n-Gramme, die mit dem Kontext beginnen}}$$

- Kontrolle: Für jeden Kontext ergeben die Wahrscheinlichkeiten zusammen 1 (100 %).

Beispiel zum Korpus „ich mag tee ich mag kaffee ich trinke tee“ (9 Wörter, 8 Bigramme):

Kontext	Folgewort	Anzahl	Wahrscheinlichkeit
ich	mag	2	$\frac{2}{3} \approx 67\,\%$
ich	trinke	1	$\frac{1}{3} \approx 33\,\%$
mag	tee	1	$\frac{1}{2} = 50\,\%$
mag	kaffee	1	$\frac{1}{2} = 50\,\%$
tee	ich	1	$\frac{1}{1} = 100\,\%$
kaffee	ich	1	$\frac{1}{1} = 100\,\%$
trinke	tee	1	$\frac{1}{1} = 100\,\%$

- Große Korpora zählt man **automatisiert** mit Software (z. B. SoekiaGPT oder ein eigenes Python-Programm).
- Die Ergebnisse werden **tabellarisch** verwaltet: zu jedem Kontext die Folgewörter mit Anzahl. Gängige Datenstruktur: ein **Dictionary von Dictionaries** `{kontext: {folgewort: anzahl}}`, für das Beispiel oben:

```
{"ich": {"mag": 2, "trinke": 1}, "mag": {"tee": 1, "kaffee": 1}, ...}
```

Alle Bigramme eines Textes ausgeben:

```
woerter = korpus.split()                # Text -> Liste von Wörtern
for i in range(len(woerter) - 1):       # bis zum vorletzten Wort!
    print(woerter[i], woerter[i + 1])
```

Typische Fehler: Durch die Anzahl **aller** Bigramme teilen statt durch die Bigramme, die mit dem Kontext beginnen. Das **letzte Wort** mitzählen, obwohl ihm kein Wort folgt. `range(len(woerter))` statt `range(len(woerter) - 1)` führt zum `IndexError`. „Der“ und „der“ sind für den Computer verschiedene Wörter, deshalb alles kleinschreiben.

4.6.3 Texte erzeugen: next word prediction und Sampling (Abschnitt 4.4)

- **Textgenerierung (next word prediction):** Ab einem Startwort wird mithilfe der n-Gramm-Tabelle immer wieder das **nächste Wort** bestimmt und angehängt. Hat das letzte Wort keine Folgewörter, endet der Text.
- Die **Sampling-Strategie** legt fest, **wie** aus mehreren Folgewörtern eines ausgewählt wird:

Strategie	Auswahl	Folge
greedy	immer das **wahrscheinlichste** Folgewort	deterministisch, oft eintönig und sich wiederholend
random	**zufällig, gewichtet** nach Wahrscheinlichkeit	abwechslungsreicher, manchmal unsinnig
top-k	nur die k wahrscheinlichsten, darunter gewichtet zufällig	Kompromiss zwischen Vorhersagbarkeit und Abwechslung

- **Top-k neu rechnen:** Folgewörter 4-mal, 3-mal, 2-mal, 1-mal. Bei random $\frac{4}{10}, \frac{3}{10}, \frac{2}{10}, \frac{1}{10}$. Bei top-2 bleiben zwei Wörter mit $4 + 3 = 7$ Vorkommen: $\frac{4}{7}$ und $\frac{3}{7}$.

- **Länge des n-Gramms:** Größere n (z. B. Trigramme statt Bigramme) ergeben **kohärentere**, aber **weniger variantenreiche** Texte. Der längere Kontext ist spezifischer und kommt seltener im Korpus vor. Bei kleinem Korpus bricht die Generierung schneller ab.
- **Trainingsdaten:** Das Modell kann nur Wörter und Wortfolgen aus seinem Korpus erzeugen. Ein anderer Korpus ergibt ganz andere Texte. Ein Sprachmodell erzeugt deshalb **nicht automatisch neutralen** Text.

Typische Fehler: Bei top-k weiter durch die alte Gesamtzahl teilen. Glauben, random wähle alle Wörter gleich oft: Häufige Wörter werden öfter gezogen. Vergessen, dass greedy immer denselben Text liefert.

4.6.4 Halluzination und moderne LLMs (Abschnitt 4.5)

- Eine **Halluzination** ist ein Text, der **grammatikalisch bzw. statistisch plausibel**, aber **inhaltlich falsch oder unsinnig** ist.
- Ursache beim n-Gramm-Modell: Es hat **kein Verständnis** der Bedeutung. Es reiht Wörter nach **statistischer Wahrscheinlichkeit** aneinander und schaut nur auf den kurzen Kontext. Jedes Wortpaar stammt aus dem Korpus, die Kombination kann trotzdem falsch sein.

Kriterium	n-Gramm-Modell	modernes LLM
Sprachqualität	oft holprig	flüssig, grammatisch meist korrekt
Kontextberücksichtigung	nur 1–2 vorherige Wörter	sehr lange Kontexte
Korrektheit	Inhalt oft zufällig falsch	häufig zuverlässiger, kann aber **weiterhin halluzinieren**
Vielseitigkeit	setzt nur Text im Stil seines Korpus fort	übersetzen, zusammenfassen, programmieren …

- Das n-Gramm-Modell ist für Halluzinationen bei LLMs nur eine **vereinfachte Analogie**. Die Ursachen bei LLMs sind wesentlich komplexer.
- **Generative KI und Berufe:** Manche Tätigkeiten werden unterstützt oder verändert, es entstehen neue Aufgaben (z. B. Prompt Engineering), manche Tätigkeiten sind schwer zu ersetzen. Zu einer Bewertung gehören Chancen **und** Risiken.

Typische Fehler: „Das Modell lügt“ – es hat keine Absicht, es rechnet nur. „LLMs halluzinieren nicht mehr“ – doch, auch sie können es. Eine Halluzination nur „falsch“ nennen, ohne die Ursache (Wahrscheinlichkeit statt Bedeutung, kurzer Kontext) zu erklären.