DS 31 – n-Gramm-Analyse von Textkorpora (von Hand)

Informatik 10 – MINT-Zweig | Künstliche Intelligenz

Lernziel

Ich simuliere die Textkörperanalyse durch das Ermitteln von n-Gramm-Wahrscheinlichkeiten in gegebenen Textkörpern.

🕘 Stundenverlaufsplan (Doppelstunde, 90 Minuten)

Orientierungsrahmen für 11 SuS: Die ersten 80 Minuten bilden den Kern. Die letzten 10 Minuten sind bewusst flexibel und werden nicht mit neuem Pflichtstoff belegt.

Phase	Zeit	Inhalt/Aktion	Sozialform
Einstieg und Aktivierung	10'	Einen kurzen Satz in aufeinanderfolgende Wortgruppen zerlegen.	Plenum; kurze Einzeldenkzeit
Fragend-entwickelnde Erarbeitung	25'	Uni-, Bi- und Trigramme sowie Häufigkeiten aus den Schülerbeispielen entwickeln.	Unterrichtsgespräch; Beiträge der SuS werden sichtbar festgehalten
Gemeinsame Umsetzung und Sicherung	20'	Die Bigramme des Beispielkorpus gemeinsam vollständig auszählen.	Plenum; Live-Coding, Tafelbild oder gemeinsame Simulation
Kurzer Transfer	15'	Für einen kurzen Korpus Häufigkeiten und Folgewortwahrscheinlichkeiten bestimmen.	Einzelarbeit mit anschließendem Tandemabgleich (4 Tandems + 1 Trio)
Auswertung	10'	Die Berechnung einer bedingten Folgewortwahrscheinlichkeit sprachlich erklären lassen.	Plenum; Lösung erklären und Fachsprache sichern
Reserve / Vertiefung	10'	Python-Kontrolle oder Trigramme als Vertiefung.	Flexibel; bei Bedarf für Fragen, Fehlersuche oder Abschluss nutzen

🗣️ Sprachhilfe

- „Auf den Kontext … folgt mit der Häufigkeit/Wahrscheinlichkeit … das Wort …“
- „Der erzeugte Text verändert sich, weil …“
- „Im Vergleich zum n-Gramm-Modell kann das LLM …, jedoch …“

📌 Merksätze

- Ein **n-Gramm** ist eine Folge von `n` aufeinanderfolgenden Wörtern (oder Zeichen) aus einem Text.
  - **Unigramm** (n=1): einzelne Wörter
  - **Bigramm** (n=2): Wortpaare
  - **Trigramm** (n=3): Wortdreiergruppen
- Bei der **Textkörperanalyse** zählt man, wie oft jedes n-Gramm in einem **Textkorpus** (Trainingstext) vorkommt.
- Aus diesen Häufigkeiten lässt sich die **Wahrscheinlichkeit** ableiten, mit der auf ein bestimmtes Wort (bzw. eine Wortfolge) ein bestimmtes Folgewort folgt – das ist die Grundlage für die spätere Textgenerierung (DS 33).

💡 Beispiel: Bigramme von Hand zählen

Textkorpus: **"der hund läuft der hund bellt der hund schläft"**

Bigramme (Wortpaare): `(der, hund)`, `(hund, läuft)`, `(läuft, der)`, `(der, hund)`, `(hund, bellt)`, `(bellt, der)`, `(der, hund)`, `(hund, schläft)`

Zählt man die Häufigkeiten, ergibt sich für das Wort **"der"**, gefolgt von **"hund"**: 3-mal – das ist das mit Abstand häufigste Bigramm mit "der" als erstem Wort.

📝 Übung 1 – Von Hand zählen

Gegeben ist der Textkorpus:

"die katze schläft die katze spielt die katze frisst die katze schläft"

Zählt **von Hand** (auf Papier oder in dieser Zelle) alle Bigramme und deren Häufigkeit. Welches Wort folgt im Korpus am häufigsten direkt auf "die"?

(Eure Auszählung hier, z. B. als Tabelle)

🐍 Kontrolle mit Python

Überprüft eure Handauszählung mit dem folgenden Code.

```python
def erstelle_bigramme(text):
    woerter = text.split()
    bigramme = []
    for i in range(len(woerter) - 1):
        bigramme.append((woerter[i], woerter[i + 1]))
    return bigramme

def zaehle_haeufigkeiten(bigramme):
    haeufigkeiten = {}
    for bigramm in bigramme:
        haeufigkeiten[bigramm] = haeufigkeiten.get(bigramm, 0) + 1
    return haeufigkeiten

korpus = "die katze schläft die katze spielt die katze frisst die katze schläft"
bigramme = erstelle_bigramme(korpus)
haeufigkeiten = zaehle_haeufigkeiten(bigramme)

for bigramm, anzahl in haeufigkeiten.items():
    print(bigramm, ":", anzahl)
```

```
('die', 'katze') : 4
('katze', 'schläft') : 2
('schläft', 'die') : 1
('katze', 'spielt') : 1
('spielt', 'die') : 1
('katze', 'frisst') : 1
('frisst', 'die') : 1
```

📝 Übung 2 – Wahrscheinlichkeiten berechnen

Berechnet aus den Häufigkeiten oben die **Wahrscheinlichkeit** für jedes Wort, das auf "die" folgt (Anzahl des jeweiligen Bigramms geteilt durch die Gesamtanzahl aller Bigramme, die mit "die" beginnen).

🚀 Zusatzaufgabe – Trigramme

Erweitert den Code so, dass **Trigramme** (Dreiergruppen) statt Bigramme gezählt werden. Verwendet euren eigenen, etwas längeren Beispielsatz.

Lösung

✅ Lösungen

Lösung

```
# Lösung Übung 2
folgewoerter_nach_die = {b: h for b, h in haeufigkeiten.items() if b[0] == "die"}
gesamt = sum(folgewoerter_nach_die.values())
for bigramm, anzahl in folgewoerter_nach_die.items():
    print(bigramm[1], ":", round(anzahl / gesamt, 2))
```

```
katze : 1.0
```

Lösung

```
# Lösung Zusatzaufgabe
def erstelle_trigramme(text):
    woerter = text.split()
    trigramme = []
    for i in range(len(woerter) - 2):
        trigramme.append((woerter[i], woerter[i + 1], woerter[i + 2]))
    return trigramme

trigramme = erstelle_trigramme(korpus)
print(zaehle_haeufigkeiten(trigramme))
```

```
{('die', 'katze', 'schläft'): 2, ('katze', 'schläft', 'die'): 1, ('schläft', 'die', 'katze'):
1, ('die', 'katze', 'spielt'): 1, ('katze', 'spielt', 'die'): 1, ('spielt', 'die', 'katze'):
1, ('die', 'katze', 'frisst'): 1, ('katze', 'frisst', 'die'): 1, ('frisst', 'die', 'katze'):
1}
```