Projektpräsentation – automatisierte Analyse von Mundart-Chatnachrichten

12.12.2025 | Raphael Weiss

1. Ausgangslage – Projektstand bei Zwischenpräsentation

Idee

Automatisierte **Sentiment-Analyse** für Schweizerdeutsche Chatnachrichten.

Verwendete Modelle

- **BoW + Logistic Regression**
  → mit Preprocessing
- **TF-IDF + Logistic Regression**
  → Uni- und Bigramme, Preprocessing
- **SBERT + Logistic Regression**
  → paraphrase-multilingual-MiniLM-L12-v2

Herausforderungen

- Chatnachrichten sind **wenig** und **extrem kurz** (oft 2–5 Wörter)
- **Kein Kontext** → Sentiment sehr schwer zu bestimmen (vor allem für Embedding-Modelle)
- **Hohe Accuracy**, aber künstlich verzerrt
  → Modelle erkennen triviale Muster (z.B. einzelne Token, Emojis), dadurch **schlechte Generalisierbarkeit** (Overfitting)
- Fehlende Vielfalt im Training (wenig Variationen)

2. Erweiterung & Optimierung des Projekts

2.1 Datenerweiterung und Analyse

- **Generierung zusätzlicher Mundartnachrichten mit mehr Kontext**
  → komplexere Sätze → bessere Modellrobustheit
- **Kein Einsatz von Emojis**
  Begründung: Emojis sind extrem starke Sentimentmarker → Modelle lernen „Emoji = Sentiment“ statt semantische Inhalte → schlechte Generalisierung.
- **Mehr Daten- und Modell-Analysen:** z.B. Confusion Matrix und Zipf-Analyse

2.2 Einführung einer mehrstufigen Klassifikation

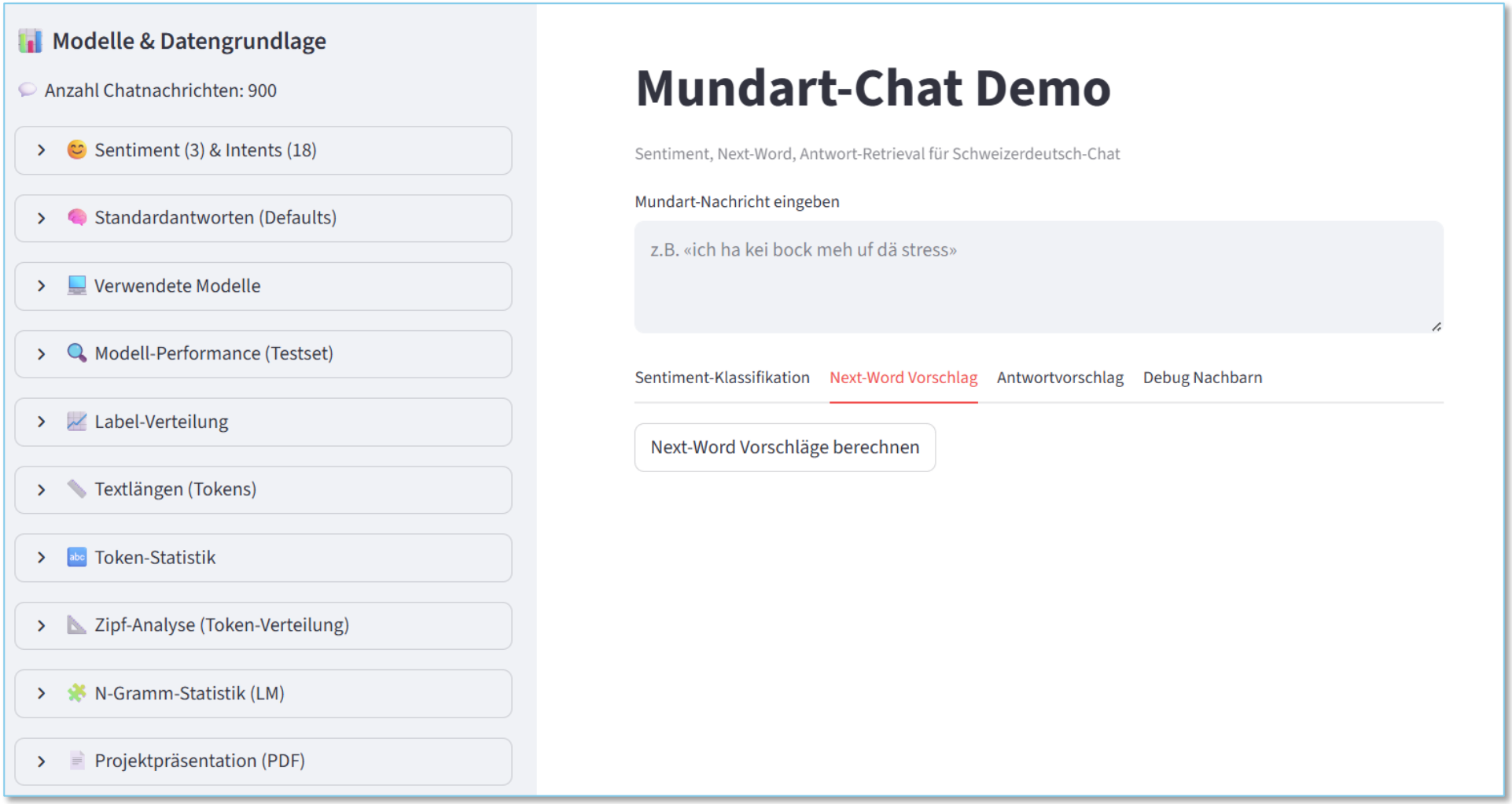
Sentiment mit Unterklassen (Intents)	**Vorteile**
• **Positiv** → D*ankbarkeit, Freude & gute Laune …* • **Negativ** → *Stress & Überforderung, Traurigkeit …* • **Neutral** → *Smalltalk, Organisation & Abmachungen …*	• Erhöhte **semantische Differenzierung** • bessere Auswertbarkeit und kontextsensitiver Chat-Antwortoptionen

2.3 Neue Modelle / Methoden

<table>
<tr><td colspan="2">a) Next-Word-Prediction mittels N-Gramm Language Model
Unterstützung beim Generieren von Text
Nutzung eines 1-, 2-, 3-Gramm-Modells mit Backoff-Strategie
" Wie könnte der Satz am wahrscheinlichsten weitergehen?"</td></tr>
<tr><td>b) Predict Answer (SBERT-Embedding Retrieval)
semantisch ähnliche Nutzeranfragen finden
passende Antwort aus vordefiniertem Antwortkorpus auswählen
inkl. Debug-Ansicht
erklärt, welche Nachbarn ähnlich sind und welche Ähnlichkeitswerte vergeben wurden → wichtig für Interpretierbarkeit</td><td>Warum Retrieval und kein generatives Modell?
Zu kleine Datenmenge: für generative Modelle viel mehr Daten nötig
Funktionsfähigkeit: schnell und stabil bei klar definierten Intents und Standardantworten
Erklärbarkeit: Mit SBERT-Nachbarn sieht man transparent, warum eine Antwort gewählt wurde.</td></tr>
</table>

2.4 Integration in Streamlit-App (Frontend)

https://mundartchat.streamlit.app



3. Pipeline – Gesamtprozess

1 Datengrundlage

- 900 Sätze (manuell + generiert)
- Chatpairs (Usertext–Antwort-Paare) als Grundlage für das SBERT-basierte Antwort-Retrieval

2 Preprocessing

- Dialektstandardisierung
- Lowercasing
- Tokenisierung auf Wortebene
- Entfernung von Stoppzeichen / Noise-Token

3 Feature Engineering & Modelltraining

BoW (Bag-of-Words): **CountVectorizer** und Klassifikation via multinomiale **Logistic Regression**

TF-IDF: **TfidfVectorizer** und Klassifikation via multinomiale **Logistic Regression**

SBERT: **SentenceTransformer** und multinomiale **Logistic Regression** auf Embedding-Raum

3 Neu: N-Gramm Language Model

- Training von 1-, 2-, 3-Grammen
- Backoff-Strategie für Next-Word-Vorhersagen
- Nutzung in der App zur Unterstützung der Texteingabe

3 Neu: Antwort-Retrieval (SBERT-basiert)

- Suche nach semantisch ähnlichster Chatnachricht und Auswahl der passenden Antwort
- Debug-Ansicht für „Nearest Neighbors“

4 Evaluationen & Visualisierungen in der App

- Accuracy, Precision, Recall, F1
- Konfusionsmatrix (Heatmap) für SBERT
- Label-Verteilung
- Token-Statistik, N-Gramm-Statistik
- Zipf-Analyse
- Kosinus-Ähnlichkeit

4. Fazit – Optimierte Lösung und Verbesserungspotential

Erreichte Optimierungen

- etwas bessere Generalisierbarkeit und robustere Modelle für kurze Dialekttexte
- mehr semantische Tiefe dank Intent-Klassen
- Nachvollziehbarkeit durch Debugging
- Demo-App für Sentimentanalyse, Next-Word, Antwortvorschläge

Verbesserungspotenzial

- grössere & vielfältigere Datengrundlage (mehr Dialektvarianten, echtes Chatmaterial)
- mehr Kontextverarbeitung (z.B. Dialogmodelle)
- Erweiterung der Intents und Multi-Label-Klassifikation
- Kombination aus Retrieval und leichter Generierung (RAG)

Code für Nachrichtengenerierung

```
"""
mundart_data.py

- Preprocessing für Schweizerdeutsch-Chat
- Seed-Datensätze (ohne Emojis)
- Chatpair-Datensätze mit Standardantworten
"""

import numpy as np
import pandas as pd
import re
import random

# ============================================================
# Globale Config & Preprocessing
# ============================================================

RANDOM_STATE = 45
np.random.seed(RANDOM_STATE)
random.seed(RANDOM_STATE)

DATA_CSV_BASE = "mundartchat_base.csv"
DATA_CSV_CHATPAIRS = "mundartchat_pairs.csv"

LABEL_ORDER = ["negativ", "neutral", "positiv"]

DIALECT_MAP = {
    # Formen von "sein"
    "bin": "bi",

    # Formen von "können"
    "cha": "chan",

    # kommen / gehen
    "chunt": "chunnt",
    "gang": "go",
    "geh": "go",

    # Ausdrücke
```

```
    "imfall": "im fall",
}

URL_RE     = re.compile(r"https?://\S+|www\.\S+")
USER_RE    = re.compile(r"@\w+")
HASHTAG_RE = re.compile(r"#\w+")

# einfache Token-Definition (ohne Emoji-Specials)
TOKEN_PATTERN = r"(?u)\b[\wäöüÄÖÜß]+\b"


def preprocess_text_chat(t: str) -> str:
    """Einheitliches Preprocessing für Chattexte (ohne Emoji-Sonderlogik)."""
    if t is None:
        return ""
    t = str(t).strip().lower()

    # Platzhalter für URLs, User, Hashtags
    t = URL_RE.sub("<URL>", t)
    t = USER_RE.sub("<USER>", t)
    t = HASHTAG_RE.sub("<HASHTAG>", t)

    # Zahlen normalisieren
    t = re.sub(r"\d+", "<NUM>", t)

    # Mehrfachbuchstaben reduzieren (z.B. "heyyyy" -> "heyy")
    t = re.sub(r"(.)\1{2,}", r"\1\1", t)

    # schiefe Apostrophe
    t = re.sub(r"[’´`']", " ", t)

    # Umlaute vereinheitlichen
    t = (t.replace("ä", "ae")
          .replace("ö", "oe")
          .replace("ü", "ue")
          .replace("ß", "ss"))

    # Trenner vereinheitlichen
    t = t.replace("-", " ").replace("/", " ")
```

```
    # Dialekt-Normalisierung
    toks = t.split()
    norm_toks = [DIALECT_MAP.get(w, w) for w in toks]
    t = " ".join(norm_toks)

    # alles raus, was kein Wort oder Placeholder ist
    t = re.sub(r"[^\w<>]+", " ", t)
    t = re.sub(r"\s{2,}", " ", t).strip()
    return t



# ============================================================
# 1) Mundart-Chatnachrichten (Seeds, ohne Augmentation)
# ============================================================

EXAMPLES = {
    # ========================================
    # NEGATIV (5 Intents × 50 Bsp)
    # ========================================

    ("negativ", "stress_ueberforderung"): [
        "grad volle to-do-lischt am abarbeite und nämmer chum noh",
        "de tag isch nur am raste, kei sekunde ruig",
        "alles staplet sich uf und s wird nöd weniger",
        "de kopf isch komplett überfüllt im momänt",
        "ufträge chömed us allen rigtige, kei verschnuufpaus",
        "s tempo isch brutal hoch grad",
        "im alltag pressierts a jeder ecke",
        "de stress chunt in wellene und alles wird z viel",
        "ständig neues los, nüt wird fertig",
        "de rhythmus isch nöd meh gscheid handelbar",
        "jede chliini sache nimmt grad mega viel energie",
        "s lauft alles gleichzeitig und nüt passt i d agenda",
        "de tag fingt hektisch aa und hört genauso uf",
        "kei ruum im chopf für öppis anders",
        "uf allen fronten grad uf trab",
        "s gfühl, dauernd am reagierä statt am plane",
```

```
    "de alltagsdruck isch eifach übermächtig",
    "jede minütli freizeit verschwindet sofort",
    "de stress zieht sich durch de ganze wuche",
    "s wird eifach immer meh statt weniger",
    "dauernd unterwegs, nie am ankomme",
    "kei chance zum richtig abschalte",
    "de kalender isch so voll, dass nüt meh platz het",
    "mini ressourcen sind aktuell uf minimalbetrieb",
    "ständig am jongliere mit viel z vil sache",
    "s tempo het nöd mol en gang meh, nur turbo",
    "jede pause wird direkt wieder ufgfrässe",
    "alles bruucht grad doppelt so vil energie wie normal",
    "d anspannung sitzet tief, meh als mer lieb isch",
    "de tag lauft einem nur dervo",
    "jedi ablenkig bringt grad no meh durcheinand",
    "s gfühl, im dauer-modus funktioniere z stecke",
    "d batterie isch ständig im orange-bereich",
    "zeitdruck überall, ruhe nirgends",
    "d belastig het langsam en ungsunde höchi",
    "sache sammelnd sich und wärtet uf lösige",
    "jede neue aufgab kommt ungelege",
    "d konzentration chunnt kaum noh nach",
    "der stresspegel isch grad irrwitzig hoch",
    "alles bruucht grad plötzlich extreme priorisierung",
    "de tag isch voller unterbrüch und het kei flow",
    "s packt einem von morn bis abig",
    "kei ahnig, wohär die dauerhektik chunt, aber sie isch do",
    "d anforderige wachsed schneller als d kapazität",
    "jede ruum eimal blockiert, kei luft meh",
    "s gfühl, de tag hätt nur vier stunde",
    "grad chli am kämpfe mit de menge ufgabe",
    "hektik pur, nöd mol zit fürs durchatme",
    "de druck isch grad ständig präsent",
    "alles lauft paralell, nüt isch richtig abgeschlossen",
],


("negativ", "konflikt_spannungen"): [
    "d luft isch voll vo unausgsprochene spannige im momänt",
```

"d stimmig isch grad mega angespannt zwüsche allne",
"jede diskussion driftet schnell i d falschi richtung",
"d kommunikation funktioniert imfall gar nöd",
"strohung i de luft, öppis passt nöd",
"mini wört werde ständig falsch verstande",
"d situation eskaliert immer wieder halb",
"es brennt uf de linie, kei ruhe im gspröch",
"mer reded ane vorbei, völlig unglücklich",
"d fronten sind verhärtet wie beton",
"jedes thema wird sofort zum konfliktpunkt",
"s klima isch frostig, kei nächi möglich",
"mer findet kei gemeinsame basis im moment",
"jedi erklärig löst nur meh widerspruch us",
"d reaktionen sind übertriebe heftig grad",
"es isch schwierig, sachlich z bliibe",
"verwirrig und missverständnis überall",
"d argument verändern sich ständig, nüt isch klar",
"jeder versuech zum kläre macheds schlimmer",
"d emotion schiesst sofort i d höchi",
"keine seite git aktuell eine millimeter nah",
"d spannig het sich länger ufbaut und platzt jetzt uf",
"eifach kei gemeinsame sprache grad",
"jede aussag wird direkt in frage gstellt",
"s gspröch isch blockiert, kei bewegig i de sach",
"d stimmung kippt immer wieder unerwartet",
"drumume wirkt alles mega gereizt",
"e kalthr war zwischen de beteiligte",
"d unklarheit macht alles nur komplizierter",
"e sachliche ebene isch kaum erreichbar",
"d gegenseitigi geduld isch praktisch null",
"dreht sich immer im chreis ohne lösig",
"keine seite het richtiges vertroue i d andere",
"jeder punkt wird zu enem streitpunkt",
"d ganze situation schwält scho lang vor sich hin",
"d erwartige sitzed komplett auseinander",
"es schauklet sich ständig uf",
"keine verständnis füreinander sichtbar",
"jede kritik wird sofort persönlich gno",
"d gesprächsdynamik isch mega unglücklich",

```
    "eis wort z vil und alles fliegt durenand",
    "d konfrontation hängt schon i de luft, öppis brodlet",
    "e kleinigskeit het grad es riesigs echo",
    "d fronten sind härt als au scho",
    "d rolle si unklar und das schafft konflikt",
    "jeder versuech zum beruhige verpufft",
    "d themen überlapped sich uf unglückliche art",
    "es besteht null konsens über s wichtigste",
    "e mini provokation langt und s knallt grad wieder",
    "d spannig macht jede kommunikation extrem müehsam",
],


("negativ", "selbstzweifel_unsicherheit"): [
    "grad chli am hinterfrage, wie stabil mini entscheidige sind",
    "grad chli unsicher, wie guet das cho wird",
    "es fehlt e bitz am vertroue i mini entscheide",
    "nöd ganz sicher, öb dä weg der richtig isch",
    "e paar sache fühled sich unklar a",
    "d orientierig isch grad chli wacklig",
    "eifach nöd hundertpro überzeugt vo mim plan",
    "e gewisse unsicherheit schwingt grad überall mit",
    "s gfühl, nöd ganz im bild z sii",
    "mangsi situatione mache mi chli nervös",
    "e paar dinge sind mir grad nöd ganz geheuer",
    "bei einigem fehlt mer dä letzte schub sicherheit",
    "chli am zweifle, öb ich s richtig ufpasse",
    "es isch schwierig, klar z gseh, was stimmt",
    "wankligi entscheidige sind grad überall",
    "unentschlossenheit macht s langsam müehsam",
    "nöd sicher, weli option langfristig guet isch",
    "e paar fragzeiche schwirred scho länger ume",
    "chli überfordert mit allne möglichkeite",
    "grad am ringe mit prioritätä",
    "e unsichers gfühl begleitet mi i vilere sache",
    "weiss nöd so recht, öb ich dä input korrekt verstande ha",
    "e paar unklari punkte mache s schwer, vorwärts z mache",
    "d innere stabilität isch grad nöd so verankert",
```

```
    "d gedanke springet ume und mache alles unruhig",
    "mues vil abwäge und das macht mi unentschlosse",
    "chli verunsichert, wie ander das würded mache",
    "nöd sicher, öb mini einschätzige stimmend",
    "grad bitz am kämpfe mit d confident",
    "unsicheri situatione sind nid so mini stärchi",
    "e paar mal abglenkt worde und d richtung verlore",
    "hät gern meh klarheit, bevor s wiiter goht",
    "d unklarheit zieht sich durch mehri thema",
    "fällt grad schwer, s richtige gfühl z packe",
    "bin chli am hin und her überlege",
    "grad am suche nachere lösig, wo sich richtig aafühlt",
    "nöd sicher, öb ich genug input ha zum entscheide",
    "vieles wirkt grad chli diffus",
    "e grundlege sicherheit fehlt im moment",
    "find nöd ganz de faden i dem thema",
    "mini gedanke lauft nöd ganz rund grad",
    "mues no chli vergleiche, wäge und neu denke",
    "momentan chli i der luft, weder do no där",
    "e par detail irritiered mi und mache mich unsicher",
    "unentschlossen über d nächschte schritt",
    "uf der suche nach klarere strukture im kopf",
    "chli vorsichtig unterwegs, will nüt überstürze",
    "d innere stimme isch grad nöd ganz eindeutig",
    "e chli unsicher, weli richtung denn stimmt",
    "es bruucht no es paar gedanke meh, bis ich sicher bi",
],


("negativ", "traurigkeit_einsamkeit"): [
    "d stille ume ane wirkt grad eher schwer als entspannt",
    "d stimmig isch grad schwer und chli leer",
    "viel distanz i de umgebig, wenig nächi",
    "d tage fühled sich chli farblos a",
    "kontakt mit lüt wirkt imfall grad schwierig",
    "e ruhigi phase, aber eher uf dä bedrückte art",
    "momentan chli abkapslet vom rest",
    "d energie für soziale sache isch recht nidrig",
    "verbindig zu andere fühlt sich grad schwach a",
```

```
"chli im eigenene kopf ufgfange",
"wärmi und nächi fehlen im moment es bitz",
"d stimmig im innerne isch nöd ganz stabil",
"längeren momänt vo stille, wo chli drückt",
"d tage ziehed sich und fühled sich lang a",
"wenig austausch im alltag grad",
"kontakt gits, aber er fühlt sich oberflächlich a",
"d emotione sind momentan eher gedämpft",
"fühlt sich alles chli distanziert und unnahbar a",
"s bedürfnis nach nächi isch imfall grad gross",
"umgebig isch voll, aber d nächi fehlt trotzdem",
"viel im inneren am laufe, aber wenig nach usse",
"d momänt sind ruhig, aber nöd im positive sinn",
"s gfühl, es fehle chli verbindigspunkte",
"s soziale umfeld wirkt grad chli wacklig",
"wenig resonanz im gspröch mit andere",
"d stimmig isch eher melancholisch",
"immer wieder phasen, wo s herz chli schwer isch",
"dr alltag fühlt sich distanziert a",
"gedanke kreisle ume statt dass sie ruhe finde",
"chli losglöst vo situatione rundume",
"d nächi zu lüt chunnt momentan nöd so rächt a",
"viel stilli momänt, wo chli druck ufbaut",
"situationen, wo normalerweise freud brächend, wirked matt",
"kontakt isch da, aber er vertieft sich nöd",
"d emotion sind chli schwerfällig",
"e gewisse bedrückti atmosphäre begleitet vil momänt",
"d welt wirkt grad chli weiter weg als normal",
"verbindig wär schön, aber sie ergibt sich nöd so",
"öfters momänt, wo es chli ziehed im innerne",
"wärmi fählt im alltag es deutliches stück",
"soziale sache überfordered momentan chli",
"d leere zwüsche de termine isch spürbar",
"viel im kopf, wenig im herz, so fühlt s sich a",
"nächi isch imfall grad eher schwierig z finde",
"immer wieder gedanke, wo sich nach ruum sehne",
"interesse an soziale sache isch grad nid riesig",
"d verbindige wirked chli dünn und fragil",
"keinere grosse streit, aber au wenig herzliche momänt",
```

```
    "d tage händ es ruhiges, fast traurigs echo",
    "wärmi und es offes ohr wären grad sehr wertvoll",
],



("negativ", "gesundheit_sorgen"): [
    "wär beruhigend, e klare rückmeldung vom arzt z ha",
    "grad chli unsicher, was mit em körper los isch",
    "symptome sind nöd ganz eindeutig im momänt",
    "e paar körperliche sache irritiered chli",
    "d energiestufe isch heut eher nidrig",
    "wiederkehrendi unruhi im körper macht chli stutzig",
    "nöd sicher, öb es vom stress oder öppis anders chunnt",
    "e chli druck im chopf, nöd ganz angenehm",
    "d konzentration sind grad nöd so stabil",
    "uf und abe i de körperstimmig, recht ungewöhlich",
    "d müdigkeit hält länger a als normal",
    "e paar kleini symptome mache mich aufmerksam",
    "körperlich chli wacklig unterwegs",
    "d körpersignale sind grad chli schwer z deute",
    "schlaf isch nid so ruig gsi wie sonst",
    "d körperreaktione sind unerwartet chli intensiv",
    "nöd ganz sicher, wie ernst das isch",
    "grad chli überwältigt vo allem, au körperlich",
    "d nervosität spiegelt sich es bitz im körper",
    "öppis fühlt sich nöd ganz im gleichgwicht a",
    "d müdigkeit chunnt immer wieder in wellene",
    "e paar ungewöhnlichi verspannige sind debi",
    "körperlich chli weniger robust als normal",
    "e sachte schwere im körper, nöd ganz klar woher",
    "d atmig isch chli flach im momänt",
    "d innere ruhe fehlt, spürbar au körperlich",
    "e gewisse sensibiliät i de körperregione",
    "e paar komische reize chömed und gange wieder",
    "d körperwahrnehmig isch grad eher unstet",
    "schwankende energiestufe irritiert chli",
    "nöd ganz sicher, öb das nur temporär isch",
```

```
    "körperlich nöd im topform, aber schwer z sage warum",
    "d müdigkeit macht s alltägliche chli müehsam",
    "e ungewöhlichi unruhe i de glider",
    "symptome chömed ohni klare erklärig",
    "d wahrnehmig isch chli verändret, nöd unangenehm aber ungewöhnlich",
    "wär schön, meh stabilität im körper z ha",
    "mer spürt, dass öppis grad nöd ganz rund lauft",
    "d körperreaktion passt nöd zu de erwartige",
    "d spannig i de schulter löst sich kaum",
    "grad chli vorsichtig unterwegs, körperlich nid ganz sicher",
    "wärmi und ruhi fehlen imfall chli i de glider",
    "e mini destabilität im körper, schwer z beschriebe",
    "d körperstimmig isch nöd konstant",
    "öppis drückt leicht, aber ohne klare ursach",
    "e ungewissheit über d körperläch verlauf",
    "e paar ungewohnte signale mache mer gedanke",
    "nöd dramatisch, aber definitiv spürbar anders als sonst",
    "d körper bruucht glaub grad es bitz meh achtig",
    "insgeheim wärs beruhigend, meh klarheit über d symptome z ha",
],

("negativ", "kurzreaktion_negativ"): [
    "uff, gar nöd guet",
    "totale schwachsinn",
    "so en bullshit",
    "vo mir us: nope",
    "gar nöd guet",
    "huere müehsam",
    "einfach nur nervig",
    "komplett schwach",
    "kannst vergässe",
    "voll für nüt",
    "so en seich",
    "absolut unnötig",
    "das isch nöd es ding",
    "mega enttäuschend",
    "null begeisterig",
    "definitiv nöd fan",
    "finds richtig müesam",
```

```
    "wirklich nix speziells",
    "alles anders als guet",
    "so nöd optimal",
    "ehrlich gseit: meh",
    "chli zum abgwöhne",
    "es bitz grusig ehrlich",
    "vo mir: daumen abe",
    "passt mir gar nöd",
    "würdi so nöd empfehle",
    "fühlt sich falsch a",
    "eifach müehsam",
    "wüki nöd überzeugend",
    "unnoetiger stress",
    "ke vibe",
    "gar kei lust druf",
    "nöd mini welt",
    "macht null freud",
    "finds recht schwach",
    "es bitz peinlich",
    "nöd so nice",
    "so blöd",
    "wär schöner anders",
    "ziemlech abstossend",
    "chli toxisch das",
    "mag das gar nöd",
    "ke bock uf das",
    "komplett overkill",
    "fühlt sich komisch a",
    "nöd es highlight",
    "imfall eher flop",
    "gar nöd meins",
    "wär gern ufgheit blibe",
    "voll daneben",
],



# ========================================
```

```
# POSITIV (5 Intents × 50 Bsp)
# ========================================

("positiv", "dankbarkeit"): [
    "bin imfall sehr dankbar für das, was du da gmacht hesch",
    "merci viu mau, das isch würkli mega aufmerksam gsi",
    "huere lieb vo dir, danke tausig mol",
    "so e gschänk het mer grad den tag versüsst, merci!",
    "dini unterstützig bedeutet würkli vil, danke dir",
    "es het mi richtig berührt, merci fürs drandänke",
    "eifach mal en grosses danke, het mega guet ta",
    "s isch nöd selbstverständlich, drum merci",
    "huere schön gsi, merci für die müeh",
    "dini geduld isch gold wert, danke dir vielmol",
    "es chli merci, aber vo härze",
    "so en liebe gest, merci 1000x",
    "het mi mega gfreut, danke dir!",
    "merci fürs zuelose, das het sehr guet ta",
    "en riesigs danke für dini hilf hüt",
    "merci, dass du druf ufgpasst hesch",
    "es isch so angenehm mit dir, merci viu mau",
    "so viel ruhe und verständnis – danke!",
    "es het mer echt erleichterig bracht, merci",
    "so e klari erklärig – danke dir!",
    "dä input vo dir het echt weitergholfä, merci",
    "für mini frage immer offe – merci!",
    "huere wertvoll, was du gseit hesch, danke",
    "es isch so angenehm, wie du d sache angeisch – merci",
    "das het mi grad en grossi last gnoh, merci dir",
    "eifach mol danke, weils wichtig isch",
    "dini perspektive het mir mega ghollfe, merci",
    "het mich grad richtig gfreut, merci!",
    "merci, dass du immer so fair und ehrlich bisch",
    "es isch schön, dass mer sich uf dich cha verla, merci",
    "danke für dini spontane hilf – mega",
    "huere cool vo dir gsi, merci!",
    "es het mer würkli de stress gnoh, danke dir",
    "merci, dass du dir so vill zit gnoh hesch",
    "so feinfühlig reagiert – merci viu mau",
```

```
    "dini wort händ guet ta, danke",
    "merci, dass du nöd nur zuelose, sondern verstandsch",
    "es het würkli den unterschied gmacht, merci",
    "de support vo dir isch mega wertvoll, danke",
    "het alles viel einfacher gmacht, merci!",
    "so unkompliziert und hilfsbereit – merci",
    "eifach dankbar für dini art",
    "merci, es isch würkli es geschenk gsi",
    "dä moment hüt mit dir – danke",
    "schön, dass mer uf dich zähle cha, merci",
    "dankbare vibe grad – merci dir!",
    "für mini nervosität hesch perfekt reagiert – danke",
    "es het mi beruhigt, merci viu mau",
    "de riesige unterschied, wo du gmacht hesch – danke!",
    "merci, die unterstützig hett wehklä guet ta",
],

("positiv", "freude_gute_laune"): [
    "dä vibe hüt isch eifach nur schön",
    "grad so en richtig positive vibe hüt!",
    "huere schön, wenn öppis eifach klappt",
    "dä moment hüt het mega guet ta",
    "so en fröhliche tag, einfach angenehm",
    "lache muess si, s isch schlicht zu schön gsi",
    "git grad mega viel energie, ehrlich",
    "so chli sonnigs gfühl im bauch hüt",
    "dä tag het besser gestartet als erwartet",
    "en richtig guete flow grad am laufe",
    "so viel freud grad, het richtig überrascht",
    "dä kleine erfolg hüt het wunder bewirkt",
    "s het eifach gfägt, ganz unkompliziert",
    "ganz entspannti, fröhlichi stimmig hüt",
    "de vibe isch richtig angenehm und leicht",
    "s het einfach alles zämepasst, mega schön",
    "grad e ruugi freud i mir, sehr angenehm",
    "s gfühl, dass alles stimmt – voll schön",
    "dä tag macht grad überraschend guet laune",
    "d läbe zeigt sich grad vo dr sonnige site",
    "so en chli glücksmoment zwüschet dur",
```

```
    "lächle müesse, eifach us em nüüt use",
    "wärendem tag paar richtig schöni momänt gha",
    "e chli freudschub hüt, tat mega guet",
    "dä abig het so en warmi stimmig gha",
    "s het sich so leicht aagfühlt, herrlich",
    "ganz en natürlechi freud grad am laufe",
    "so en locker-lachige vibe, perfekt",
    "dä spontane moment hüt het mega gfägt",
    "mit lüt gsi, wo gueti laune mache – top",
    "kleini sache, aber riesigi freud drüber",
    "heiteri stimmig grad, ganz ohne grund",
    "dä tag het richtig angenehm überrascht",
    "en wunderschöni ruigi freud in mir",
    "eifach glacht über öppis chliins – tut guet",
    "so en lässige moment hüt, ganz unspektakulär",
    "en richtig wohltuende atmosphär gsi",
    "hüt stimmt s vibe, nöd erklärbar aber schön",
    "die situation het so positiv überrascht",
    "chlii glücksblitz grad – mega schön",
    "alles chli einfacher vorgcho hüt, das macht freud",
    "dä humor hüt het würklich guet ta",
    "e herzlichs lache mit öpperem teile – gold wert",
    "so entspannt, so fröhlich – perfekt gsi",
    "dä unerwartete moment hüt het mega glanz gha",
    "grad so en inneri sonnige stimmig",
    "en haufen freud gspürt, unerwartet aber schön",
    "dä tag het eifach funktioniert – freud pur",
    "locker, warm, fröhlich – guet fürs gfühl",
    "so fröhlichi energie hüt, tut extrem guet",
],

("positiv", "erfolg_stolz"): [
    "dä erfolgsmoment hüt het richtig guet ta",
    "so en sauberer abschluss gha, fühlt sich stark a",
    "dä schritt vorwärts isch würkli spürbar gsi",
    "mini arbeit het sich definitiv uszahlt",
    "es resultat, wo sich würkli sehen laht",
    "dä erfolg isch überraschend glatt cho",
    "het richtig stolz gmacht, wie rund alles worde isch",
```

```
"en task sauber durchzoge – top gfühl",
"mega schön z spüre, dass d müeh frücht träg",
"dä moment vo klari fortschritt het gut ta",
"wieder öppis abghäklet, wo lang angstande isch",
"es feedback übercho, wo sehr motiviert hät",
"d performance hüt het würkli überzeugt",
"säuber investierti zit het klar erfolg bracht",
"dä lernschritt hüt isch richtig hangebliebe",
"gseh, wie vill besser s worde isch – schöns gfühl",
"dä milestone het richtig freud bracht",
"en challenge gmeisteret und stärcher usecho",
"es guets resultat trotz schwierige phase",
"dä erfolg hüt isch mega verdient gsi",
"het guet ta, so souverän z bliebe",
"klar gmerkt, wie vill kompetänz entstanden isch",
"dä tag het stolz und zufriedenheit bracht",
"fortschritt gseh, Schritt für Schritt",
"feedback vom team het richtig bestätigt",
"dä moment vo: yes, das het funktioniert",
"mini strategie isch perfekt ufgange",
"dä glunge abschluss het mega motiviert",
"schön gsi z gseh, dass d üebig würkli wirkt",
"dä erfolg isch eifach ehrlich verdient",
"het richtig gfunkt zwischen idee und umsetzig",
"d klarheit im resultat isch mega schön gsi",
"mega beeindruckt, wie rund s worde isch",
"dä erreichte step hüt het gueti energie bracht",
"froh gsi über dä saubere output",
"es komplizierts thema endlich gmacht – stark",
"jobbezoge erfolg hüt, het sehr guet ta",
"dä moment, wo alles inegriffe het",
"selbst stolz gsi, wie professionell s worde isch",
"performance hüt het überzügt – guets zeiche",
"het guet ta, mal so klare resultate z gseh",
"es hupfte richtig, als s endlich klappt het",
"d investierti müeh isch voll ufgange",
"dä fokus hüt het sich extrem uszahlt",
"e siichtbare verbesserig macht mega stolz",
"klarer erfolg – und vor allem verdient",
```

```
    "dä ganze prozess het so vill lehrrichi momänt bracht",
    "fortschritt sichtbar gmacht – stärkt ungemein",
    "en erfolg, wo würkli Gewicht het",
    "dä tag hüt het stolz und zufriedenheit gschänkt",
],

("positiv", "verbundenheit_naehe"): [
    "d gmeinsam Ziit hüt het richtig guet ta",
    "het mega schön gfühlt, so ehrlich chönne rede",
    "dä moment vo nächi isch würkli berührend gsi",
    "so e warmi atmosphär zäme, ganz natürlich",
    "het richtig verbindet, die situation",
    "e ehrlechs gspröch het grad mega viel bedeutet",
    "het guet ta, dass mer sich so öppet vertraue cha",
    "d verbindig isch wieder richtig spürbar gsi",
    "het sich mega authentisch und nah aagfühlt",
    "dä gemeinsame vibe hüt isch top gsi",
    "so schön gsi, dass mer voll ufem gliiche level gsi sind",
    "d nächi isch ganz locker und eifach entsta",
    "het sich wie en sichere ort aagfühlt",
    "e warmi, vertrauti stimmig hüt – mega",
    "schön, wenn mer eimander würkli versteht",
    "d verbindig isch grad viel tiefer worde",
    "het guet ta, wie offe alles gsi isch",
    "so en ehrliche austausch mach unglaublich viel us",
    "d sanfti art vom gspröch hüt het mega gholfe",
    "het richtig berührt, wie präsent alles gsi isch",
    "so viel nächi ganz ohni druck – wunderschön",
    "het sich wie mini mensche gfühlt, wo da sind",
    "de moment hüt het nächi und ruhe brocht",
    "het guet ta, dass mer nöd müend spiele oder so",
    "d verbindig isch ganz unforced entsta",
    "s isch eifach gmüetlich und vertraut gsi",
    "so en wärmi vibe hüt, het mi überraschet",
    "d nächi het würklich getröschte",
    "het sich wie ächte halt aagfühlt",
    "so guet gsi, dass mer sich öppis persönlichs hät chönne teile",
    "d ruigi energie zäme het voll gholfe",
    "het sich nöd oberflächlich oder forced aagfühlt",
```

```
    "d gmeinsam Ziit het de kopf sehr beruhigt",
    "en vibe, wo zeigt, dass s vertroue stimmt",
    "het sich so echti freundschaft aagfühlt",
    "mini welt het sich grad chli grösser aagfühlt",
    "wärmi momänt, wo nöd gross erklärt müend wärde",
    "het guet ta, wie ehrlich alles gsi isch",
    "so en wertvolli verbindig – het richtig glänzt",
    "d nächi het grad viel sicherheit bracht",
    "het sich so natürlich ergeben, mega schön",
    "dä austausch hüt het richtig satt gmacht im herz",
    "en moment vo echti nächi, selten und kostbar",
    "het guet ta, so akzeptiert z sii",
    "fühltsich an, als wär mer länger verbunden gsi",
    "e kleinigkeit, aber voller wärmi und nächi",
    "dä vibe isch weich, vertraut, angenehm",
    "het mir zeigt, wie viel gemeinsames mir händ",
    "so en schöne verbindigsmoment, ganz uferwartet",
    "het richtig ruhe und nächi bringt",
],

("positiv", "motivation_vorfreude"): [
    "d blosse ideä git scho mega schub",
    "grad voll energie und bereit für dä nöchschte schritt",
    "huere freudig, was da alles chönnt cho",
    "es kribbelt richtig, wäge dem was bald asteht",
    "dä motivation-schub hüt isch enorm gsi",
    "so en guete flow im chopf, mega angenehm",
    "chli hibblig vor excitement, aber schön",
    "huere lust, öppis neu azpacke",
    "d vorfreud macht grad richtig leicht",
    "mini ideen spränge ume, würkli inspirierend",
    "so en positive drive, het mich überrascht",
    "d energie hüt isch nöd normal, mega angenehm",
    "gseh richtig vor mir, wie guet s chönnt usechöme",
    "d motivation chunnt grad wie vo alle site",
    "so bock uf s projekt, ehrlech",
    "es isch voll es guets timing, um loszlegä",
    "d vorfreude git grad en schöne schub",
    "het mega spass gmacht, d sache vorzubereite",
```

```
    "en kleine kickoff-moment hüt, wo mega glanze het",
    "so en richtige „go for it“- vibe grad",
    "d zukunft gseht grad richtig freundlich us",
    "dä moment vom „jetzt startemer“ isch da",
    "grad voll im ideenmodus, macht mega laune",
    "en schönes inneres „bald isches so wiit“-gfühl",
    "d motivation hüt het mich selber überrascht",
    "de vibe isch so positiv, sogar d arbeit macht lust",
    "vorfreud uf lüt, situatione, momänt – alles öppet",
    "s gfühl, dass öppis guets chunt, isch sehr präsent",
    "so en gueter antrieb grad, richtig angenehm",
    "mini vision wird grad klarer, macht freud",
    "en guete startpunkt gseh und direkt bock gha",
    "d ideä rollt grad richtig guet ine",
    "es chli kribbele macht alles aufregend",
    "mer spürt, dass öppis neu im anzug isch",
    "d energie het grad hüfige mini erwartige übertroffe",
    "so en „los, mach das jetzt“-moment",
    "dä drive hüt het richtig flügeli gmacht",
    "vorfreud über d chance, wo sich öppis neu ergit",
    "so en neugierige, positive spannig im bauch",
    "motiviert bis in d spitzä, herrlich",
    "bereit, öppis chli anders z probiere",
    "s gfühl, ganz nah am nöchschte erfolg z sii",
    "grad voll licht und positivität",
    "megavibe: alles isch möglich im momänt",
    "d vorfreud macht richtig warm ums herz",
    "energie het sich plötzlich vervielfacht",
    "ready für chli abentüür im alltag",
    "d motivation hüt schiebt würklich fescht",
    "es versprachsvolle gfühl für d nöchschte zit",
    "vorfreud so gross, dass mer kaum cha warte",
],

("positiv", "kurzreaktion_positiv"): [
    "safe mega",
    "geil, oder",
    "huere geil",
    "mega nice",
```

```
"voll schön",
"so cool",
"liebs",
"huere guet",
"richtig stabil",
"eh mega",
"finds nice",
"so fresh",
"hammer",
"bombe",
"sehr cute",
"huere herzig",
"10/10",
"love it",
"süüper",
"big yes",
"wild guet",
"richtig vibe",
"huere vibe",
"das fingt mer guet",
"voll dä move",
"nice idee",
"iconic",
"feier ich",
"bin fan",
"sehr clean",
"top sach",
"huere stabil",
"krass guet",
"das sitzt",
"mag ich sehr",
"gönn ich",
"einfach nur nice",
"Ultra guet",
"sehr smooth",
"seh ich so",
"huere satisfying",
"das stimmt mich froh",
"richtig wholesome",
```

```
    "big W",
    "ganz klar win",
    "sehr gelungen",
    "das knallt",
    "huere chillig",
    "nice umgesetzt",
    "voll okay so",
],


# ========================================
# NEUTRAL (5 Intents × 50 Bsp)
# ========================================

("neutral", "smalltalk"): [
    "wie würdisch dä tag bis jetzt beschriibe?",
    "hoi, alles klar bi dir hüt?",
    "na, wie gohts der so im momänt?",
    "was lauft grad so i dim alltag?",
    "wie isch dini wuche bis jetzt gsi?",
    "guete start gha i de tag?",
    "irgendöppis spannends passiert letzterzit?",
    "wie isch s wetter bi dir?",
    "was stoh hüt alles so a?",
    "wie hesch s gester gha?",
    "lange nüm gseh, wie lauft s?",
    "gueti vibe am morgä oder eher schläfrig?",
    "was treibsch aktuell so im läbe?",
    "wie tüend d sache vorwärts go bi dir?",
    "was esch dini stimmig hüt?",
    "wie isch s wucheend gsi?",
    "irgendöppis, wo di grad freut?",
    "wie gaht s dim umfeld so?",
    "was hesch hüt no uf em plan?",
    "alles ruig oder chli trubel?",
    "wie bisch id wuche gstarrt?",
    "was hesch für plän am abig?",
    "wie isch s am morge aagfange?",
    "es bitz stressig oder ganz entspannt?",
```

```
    "hesch hüt scho öppis guets ghaa?",
    "na, wie tüend d dinge sich aafüehle?",
    "was isch bis jetzt s highlight vo dim tag?",
    "wie lauft s bi der arbeit imfall?",
    "gsehsch dä tag eher locker oder vollpackt?",
    "wie isch s mit dinere energie hüt?",
    "was bruucht es, dass der tag guet wird?",
    "gits öppis, wo di hüt speziell beschäftigt?",
    "wie fingsch du s tempo vo dim alltag im momänt?",
    "isch s wetter grad eher motivierend oder meh so naja?",
    "wie isch s so im allgemeine bi dir?",
    "schöne momänt dä wuche scho gha?",
    "wie isch s gfühl für dä tag gsi am morn?",
    "hesch hüt mit öper special z schaffe gha?",
    "wie tüend d tage im momänt verlaufe?",
    "was würdest du als vibe vo hüt beschriibe?",
    "wie isch d laune im momänt so?",
    "gits öppis lüssigs, wo di grad begleitet?",
    "wie goht s dim rhythmus im alltag?",
    "hesch öppis, wo di grad überrascht hät?",
    "was isch dini spontane meinig zu hüt?",
    "wie isch d stimmig im umfeld grad?",
    "merksch dä fröschti oder meh dä stress?",
    "wie fühlts sich d wuche bisher a?",
    "gits öppis, wo di hüt bevorzugt begleite sött?",
    "wie würdisch dä tag bis jetzt zämefasse?",
],



("neutral", "orga_koordination"): [
    "söll ich dir no kurz es übersichtli mit vorschläg schicke?",
    "wänn wür dir generell am beschte passe zum abmache?",
    "chömer dä termin no chli verschiebe, falls nötig?",
    "passt dir früeh oder eher am abig besser?",
    "söttemer eus lieber online oder vor ort treffe?",
    "welchi uhrziit wär für dich am praktischste?",
    "wie lang würsch ungefähr zrugghalte?",
    "söll ich dir no en kalenderintrag mache?",
```

"chani dir e paar mögliche zeitslots schicke?",
"würds dir diese oder nöchschti wuche besser passe?",
"chan s au spöter am tag sii, falls s dir so passt?",
"wo wär für dich es ideal ortli zum treffe?",
"wöttisch s fix mache oder lieber spontan lah?",
"mir chönd au zwei date reserve, falls öppis chunt",
"söll ich dr no die relevante infos vorbereite?",
"wie dringlich isch dä termin für di?",
"chan ich dr es doodle-ähnlichs thing mache?",
"isch das fenster realistisch für dich?",
"chömer s flexibel halte, falls sich öppis ändert?",
"sött ich no öpper anders ihlade?",
"würdis dir helfe, wenn ich das koordinier?",
"isch di planig eher eng oder hesch meh spielraum?",
"wotts du lieber es kurzes meeting oder längers?",
"söll ich grad en erinnerig setze?",
"wie schnell bruuchsch en rückmeldung?",
"isch öppis wichtigs vor em termin z wüsse?",
"chan ich dr no alternative ort vorschlah?",
"würd dir en morgentermin besser i d charten passe?",
"wotsch, dass ich metisch en call ufsetze?",
"isch s guet, wenn mer grob so plant und denn feintuned?",
"gömer vo dem datum us, wo mir am beschte chunnt?",
"söll ich dir d agenda im voraus schicke?",
"wie würdisch du s am liebschte organisiera?",
"cha s au digital laufe oder bruuchts physisch?",
"wär dir lieber e fixe zeit oder en Zeitraum?",
"brucht s vorab no es kurzes briefing?",
"wotts du, dass ich dr die punkte vorbereite?",
"chan ich öpis aus dinere to-do-lischte überneh?",
"isch öppis, wo mer unbedingt berücksichtige söll?",
"hesch no alternative termine im chopf?",
"söll ich di up-to-date halte, falls öppis wechselt?",
"würdi dir zwei, drei zeitslots schicke?",
"cha ich di am ort abhole?",
"isch guet, wenn mer uns vorher no kurz abstimme?",
"bruuchsch eher e kurze oder e längi vorlauffrigkeit?",
"wiit im voraus wöttisch s gplant ha?",
"söll ich das scheduling überneh?",

```
    "gits öppis, wo di hinderet bi d planig?",
    "wär en spontaner ersatztermin ok?",
    "chan ich no klärihe, wer alles mitmacht?",
],

("neutral", "frage_info"): [
    "gits öppis, wo mer da besser nöd macht?",
    "wie funktionieret das genau, chasch das kurz erläuterä?",
    "was bruucht mer alles für dä prozess?",
    "wie lauft das normalerwiis ab?",
    "gits öppis, wo ich im voraus sött beachte?",
    "was isch dä unterschied zwüsche däne zwei möglichkeite?",
    "mues ich für das öppis speziells vorbereite?",
    "chasch mir grob säge, was dä erste schritt isch?",
    "wie lang gaht das in der regel?",
    "brauchts dafür en account oder nöd?",
    "chan ich das später no ändere, falls nötig?",
    "was würdisch du als best option empfehle?",
    "gits e shortcut oder es tool, wo das erleichtert?",
    "wie würdsch du das selber mache?",
    "muss mer da uf öppis bestimmts luege?",
    "was passiert, wenn mer en schritt überspringt?",
    "isch das kompliziert oder eher easy?",
    "gits es risiko, dass öppis schief lauft?",
    "wo findi d wichtigschte infos zum thema?",
    "was isch imfall dä hauptpunkt da dra?",
    "chan mer das testweise usprobiere?",
    "welchi variante het sich bis jetzt bewährt?",
    "isch öppis, wo mer gern verwechslet?",
    "wie weiss ich, ob ich s richtig gmacht ha?",
    "gits e checkliste dafür?",
    "brauchts spezielle software oder gaht s so?",
    "wo isch dä kniffligi teil bi dere sache?",
    "wiso macht mer s eigentlich genau so?",
    "gits en fallback, falls es nöd klappt?",
    "wie funktioniert s im hintergrund?",
    "iss s wichtig, wie mer s formuliert?",
    "gits es beispiel, wo ich dranne cha orientiere?",
    "kann s problem mache, wenn mer es falsch ufgsetzt?",
```

```
    "wo fangt mer am beschte aa?",
    "wofür isch die funktion überhaupt da?",
    "isch das sehr zeitaufwendig?",
    "gits öppis, wo mer lieber nöd mache sött?",
    "wie detailliert muess mer vorgeh?",
    "wo chan ich nachluege, falls ich nöd witer weis?",
    "was isch dä sinn vom ganze schritt?",
    "brauchts meh als nur die basis?",
    "wie kontrollier ich, ob alles richtig lauft?",
    "gits dinge, wo mer oft vergisst?",
    "isch das flexibel oder eher strikt?",
    "wär chan mir hälfe, falls ich hänge blibe?",
    "wie wichtig isch dä timing bi dem?",
    "chan ich no emol zum ausgangspunkt zrugg?",
    "wie gmerkt mer, wenn öppis falsch lauft?",
    "mues mer jedes detail exakt nehme oder cha mer wähle?",
    "wie würdisch du s jemandem erkläre, wo null ahnig het?",
],

("neutral", "hobby_interests"): [
    "so chli kreativi projecht dehei mache grad überraschend viel freud",
    "zletzt wieder chli meh am gamä gsii, tut guet zum abschalte",
    "uf dä strecke bin ich gern am jogge, macht dr chopf frei",
    "grad voll im kochfieber, probier alles möglichaus",
    "vil series am luege im momänt, perfekt zum chille",
    "musik lose bringt immer ruhe i dä tag",
    "bi am üebe uf em instrument – chli wacklig, aber lustig",
    "fotografie macht grad mega spass, vor allem drausse",
    "wenn s wetter stimmt, ab i d natur, es isch einfach schön",
    "chli bastleprojekte da und dort, macht mega freud",
    "grad wieder en neue workout-routine am teste",
    "neui rezepte am uusprobiere – mal guet, mal eher experimentell",
    "stricke oder häkle git imfall eifach ruhe",
    "lese grad es sachbuech, das überrascht positiv",
    "zeichne im moment recht vil, tut mega guet",
    "skateboard wieder usgrabe – chli wacklig, aber geil",
    "gärtnere het öppis unglaublich meditativs",
    "podcasts lose isch i letzter zit es fixprogramm",
    "grad öppis neus am lerne im bereich design",
```

```
    "en neue sportart am usprobiere – macht spass",
    "digital art am experimentiere – vil chlii schritt",
    "puzzlen isch überraschend entspannend worde",
    "vinyl sammle isch wieder chli ufchunnt bi mir",
    "musigiere und freestyle – nüt geplant, alles locker",
    "alte hobbys wieder entdecket, überraschend schön",
    "grad mega im fotografie-modus – alles motiv",
    "gaming sessions am abig – perfekt zum kopf lüfte",
    "kaffee röste als mini neui obssession",
    "schach am spiele, macht mega spass zum üebe",
    "wanderige am plane, wenn s wetter passt",
    "bisschen DIY dehei, mal besser mal kreatives chaos",
    "languages am lernä – chli jeden tag",
    "buecher sortiere und neui entdecke, macht freud",
    "ab und zue chli meditatiön, hätt überraschende wirkig",
    "pläuschigs töpfere – d händ si voll lehm, aber voll cool",
    "fitnessstudio wieder am entdecke, langsam aber sicher",
    "velotouren machä, wenn s schön isch, mega befriedigend",
    "backe grad vil, chli experiment mit teig und co.",
    "grad am choche – ziemlich chaotisch",
    "videospieli mit kollege – macht immer gute laune",
    "brettspiel-obig gha, mega lässig gsi",
    "grad voll im lego-modus, erstaunlich entspannt",
    "frisbee oder pingpong drausse, easy und lustig",
    "kleini urban-gardening-projekte am startä",
    "chli nähen am lerne – überraschend tricky",
    "wildkräuter sammlä am usprobiere",
    "neui cocktails mische – mol guet, mol stürmisch",
    "rollenspiele und pen&paper am entdecke",
    "minimalsport wie stretching oder yoga zwüschet dur",
    "es paar kreative apps am teste – macht richtig laune",
],

("neutral", "technik_support"): [
    "hesch gschaut, öb s problem nur bi dem eine gerät uftritt?",
    "d app spinnt grad wieder ume, kei ahnig wieso",
    "irgendöppis stimmt nöd mit em update",
    "s wlan macht komische zicke, underbrucht immer",
    "dä login het spontan nüm wölle",
```

```
"einstellung isch plötzlich verschwunde, kei plan",
"mis gerät reagiert mega langsam, würkli müehsam",
"es chunt ständig en fehlermeldig, wo nöd klar isch",
"s kabel würd nöd erkennt, egal wie mer s aasteckt",
"irgendwie isch die funktion im menu üm da",
"verbindig bricht ständig ab, mega nervig",
"s backup het nöd funktioniert, grund unbekannt",
"programm stürzt ab, sobald mer öppis lädt",
"d synchronisation bleibt i de hälfti stecke",
"mis gerät zeigt komische icons a, nöd gewohnt",
"update isch durscht, aber jetzt goht nüt meh",
"de button reagiärt nöd, wie wenn er blockiert wär",
"s passwort zruggsetze klappt nöd, trotz instruktion",
"detekte grad fehler im system, aber kei idee woher",
"die datei chan üm ufmachet wärde, völlig komisch",
"s exportiere vom dokument verweigert de dienst",
"verbindig zum server bricht ständig zämme",
"programm cha nöd installiert wärde, irgend es problem",
"s gerät het plötzlich kei ton meh",
"update rückgäng mache? keine ahnig wie das gah sött",
"d app zeigt nöd, was sie sött – menüs komplett leer",
"dateie verschwinde oder si am falsche ort",
"s drucke funktioniert nöd, nur error uf em display",
"gerät überhitzet mega schnäll, irgendöppis stimmt nöd",
"bluetooth koppelig bricht immer ab",
"benachrichtigige chömed nöd meh a",
"s bild friert ständig ii, müehsam ohne änd",
"programm zeigt an, dass es offen isch, aber nüt passiert",
"ladegerät wird nöd erkennt, obwohl s original isch",
"s update zeigt 99%, aber hängt ewig",
"internet het volle balkä, aber nüt lädt",
"d cloud synchronisiert nur teilweise",
"drucker isch online, aber nimmt kei ufträge a",
"touchscreen reagiert mega verzögert",
"audio het plötzlich üble störige dri",
"s mikrofon wird nöd erkennt im call",
"es poppt ständig es fenster uf, aber ohni inhalt",
"s programm startet, aber schliesst grad wieder",
"s wifi het gueti verbindig, aber die app meint nein",
```

```
    "im browser lauft nur jede zweite site",
    "cache leere het nöd grad viel bracht",
    "de cursor verschwindet zwüschetdur",
    "s system macht sound, aber keis fenster zeigt woher",
    "d konfiguration wird nöd gspeichert, trotz save",
    "gerät macht plötzlich neustart ohni grund",
],

("neutral", "kurzreaktion_neutral"): [
    "passt",
    "ok",
    "hm",
    "hmm",
    "mhm",
    "aha",
    "ah so",
    "alles klar",
    "verstande",
    "check",
    "notiert",
    "easy",
    "kann sii",
    "imfall easy",
    "true",
    "passt scho so",
    "kein ding",
    "okei, gsehsch",
    "makes sense",
    "schon guet",
    "alles gut",
    "isch in ordnig",
    "gut z wüsse",
    "merci für d info",
    "merci fürs update",
    "ok, klingt logisch",
    "okay, noted",
    "ah, versteh",
    "okay, habs",
    "klar, merci",
```

```
        "gseh ich",
        "gueti info",
        "isch registriert",
        "nehm mers so",
        "gaht für mi",
        "isch akzeptiert",
        "isch mer recht",
        "so chömer s la",
        "isch okay für mi",
        "passt i de rahme",
        "chum mer chönd das so mache",
        "isch guet, merci",
        "kein stress",
        "so isch s au guet",
        "noted, merci",
        "isch abgespeichert",
        "oke, alles klar",
        "gut, ich ha s",
        "verstanden, merci dir",
        "alles klar, merci dir",
    ],

}

# ============================================================
# 2) Mundart-Chatpaare: Default-Antworten
# ============================================================
DEFAULT_ANSWERS_MUNDART= {}

DEFAULT_ANSWERS_MUNDART[("negativ", "stress_ueberforderung")] = [
    "Ui, das tönt stressig.",
    "Verstah di, das isch viel.",
    "Heftigi Phase. Was wür dir grad am meiste Entlastig gä?",
    "Okay, das isch viel uf Einisch...",
    "Uff, mega viel grad.",
    "Krass, voll de Stress.",
]

DEFAULT_ANSWERS_MUNDART[("negativ", "konflikt_spannungen")] = [
```

```
    "Schwierig. Was isch passiert?",
    "Okay, Konflikt sind müehsam. Worum geit's genau?",
    "Verstehe. Was het d Situation ausgelöst?",
    "Hm, tönt nöd eifach...",
    "Ui, das knallt chli.",
    "Nöd grad easy di Sach.",
]

DEFAULT_ANSWERS_MUNDART[("negativ", "selbstzweifel_unsicherheit")] = [
    "Verstehe. Worin bisch dir grad unsicher?",
    "Okay, wo hesch das Gfühl, dass es harzt?",
    "Alles guet – wo chönnt ich dir e chli Klarheit gä?",
    "Wo spürsch die Unsicherheite am stärkschte?",
    "Kann passiäre, voll normal.",
    "Macht Sinn, dass dich das verunsichert.",
]

DEFAULT_ANSWERS_MUNDART[("negativ", "traurigkeit_einsamkeit")] = [
    "Das isch nöd eifach. Was drückt am meiste?",
    "Verstande. Willisch kurz säge, was di so belaschtet?",
    "Tut mer leid z ghöre. Was löst s grad us?",
    "Okay… was macht di im Momänt am traurigschte?",
    "Uff, das tuet weh.",
    "Tönt recht schwer grad.",
]

DEFAULT_ANSWERS_MUNDART[("negativ", "gesundheit_sorgen")] = [
    "Ah okay, was macht dir genau Sorge?",
    "Verstehe. Weli Symptom stressed di am meiste?",
    "Okay – was isch s, wo di verunsicheret?",
    "Wo spürsch, dass öppis nöd stimmt?",
    "Verständlich, dass dich das stresst.",
    "Ja, das würd mi au beunruhige.",
]

DEFAULT_ANSWERS_MUNDART[("negativ", "kurzreaktion_negativ")] = [
    "Uff, versteh dich.",
    "Oha… was isch passiert?",
    "Tönt nöd guet.",
```

```
    "Hm ja… müehsam.",
    "Okay… falls wotsch, erzähl chli meh.",
    "Krass.",
    "Schwierig...",
]


DEFAULT_ANSWERS_MUNDART[("neutral", "smalltalk")] = [
    "Alles klar.",
    "Okay, und wie geit’s dir so?",
    "Verstande. Was lauft no so bi dir?",
    "Guet z wüsse. Sonst alles easy?",
    "Wie lauft's süsch?",
    "Wie isch d Stimmig so?",
]

DEFAULT_ANSWERS_MUNDART[("neutral", "orga_koordination")] = [
    "Okay, das chömer so mache.",
    "Säg eifach, was dir passt.",
    "Guet, das luegemer a.",
    "Easy, das chömer abmache.",
    "Mer finded scho öppis.",
    "Das klingt machbar.",
]

DEFAULT_ANSWERS_MUNDART[("neutral", "frage_info")] = [
    "Klar, ich versuch s dir eifach z erkläre.",
    "Okay, gueti Frog. Um was geit s genau?",
    "Alles klar. Was wetsch genau wüsse?",
    "Gerne, ich erklär s dir kurz.",
    "Okay, luegemer das Schritt für Schritt a.",
]

DEFAULT_ANSWERS_MUNDART[("neutral", "hobby_interests")] = [
    "Ah spannend! Erzähl meh.",
    "Cool, was machsch denn genau?",
    "Nice! Wie bisch du dezue cho?",
    "Klingt lässig! Machsch das oft?",
    "Seht nach guetem usgleich us.",
```

```
    "Liebs, so Hobbys.",
]

DEFAULT_ANSWERS_MUNDART[("neutral", "technik_support")] = [
    "Okay, was lauft nöd?",
    "Verstande. Was zeigt s aa?",
    "Alles klar, was passiert genau?",
    "Okay, luegemer druf. Was isch s Problem?",
    "Okay, klingt nach Bug.",
    "Guet, das chömer fixe.",
]

DEFAULT_ANSWERS_MUNDART[("neutral", "kurzreaktion_neutral")] = [
    "Alles klar.",
    "Verstande.",
    "Aha, ok.",
    "Guet z wüsse.",
    "Okay",
]

DEFAULT_ANSWERS_MUNDART[("positiv", "dankbarkeit")] = [
    "Sehr gärn!",
    "Kei Problem!",
    "Gern gscheh!",
    "Freut mi, wenn s hilft!",
    "Voll easy"
]

DEFAULT_ANSWERS_MUNDART[("positiv", "freude_gute_laune")] = [
    "So schön! Was isch passiert?",
    "Nice! Freut mi mega für dich.",
    "Love it! Was het di so gfroet?",
    "Hammer! Verzell meh.",
    "Gönn dir die Freud!",
]

DEFAULT_ANSWERS_MUNDART[("positiv", "erfolg_stolz")] = [
    "Mega! Gratuliere!",
    "Stark gmacht!",
```

```
    "Freut mi für dich – echt cool!",
    "Top! Was het am beschte klappet?",
    "Cooli Sach!"
]

DEFAULT_ANSWERS_MUNDART[("positiv", "verbundenheit_naehe")] = [
    "Schön so Momänt!",
    "Fühlt sich guet a, gäll?",
    "Mega schön, so Verbindig!",
    "Das tuet immer guet.",
    "Das sind die beschte Momänt.",
]

DEFAULT_ANSWERS_MUNDART[("positiv", "motivation_vorfreude")] = [
    "Geili Energie! Was packsch als Nächsts a?",
    "Nice – nutz grad dä Drive!",
    "Klingt mega! Uf was freusch di?",
    "Top Stimmig! Was isch dä Plan?",
]

DEFAULT_ANSWERS_MUNDART[("positiv", "kurzreaktion_positiv")] = [
    "Mega!",
    "Nice!",
    "Voll!",
    "Haha geil!",
    "Love it",
]


DEFAULT_BY_LABEL_MUNDART = {
    "negativ": "Das tönt nöd eifach. Wenn du wotsch, luegemer zäme, was dir hälfe chönnt.",
    "positiv": "Schön z ghöre. Danke, dass du das teilsch.",
    "neutral": "Alles klar, merci für dini Nachricht.",
}

def get_default_answer_mundart(label: str, intent: str) -> str:
    key = (str(label), str(intent))

    # 1) Intent-spezifische Defaults (Liste oder String)
```

```
    if key in DEFAULT_ANSWERS_MUNDART:
        val = DEFAULT_ANSWERS_MUNDART[key]
        if isinstance(val, (list, tuple)):
            return random.choice(val)
        return str(val)

    # 2) Fallback: nur nach Label (negativ / neutral / positiv)
    if label in DEFAULT_BY_LABEL_MUNDART:
        return DEFAULT_BY_LABEL_MUNDART[label]

    # 3) Ultimativer Fallback
    return "Alles klar."

# ============================================================
# 3) Dataset-Build-Funktionen (ohne Augmentation)
# ============================================================

def build_base_dataset(
    out_csv: str = DATA_CSV_BASE,
) -> pd.DataFrame:
    """Seed-Basisdatensatz bauen (nur EXAMPLES, keine Augmentation)."""
    rows = []
    for (label, intent), texts in EXAMPLES.items():
        for txt in texts:
            rows.append({
                "text": txt,
                "label": label,
                "intent": intent,
                "is_seed": True,
            })

    base_df = pd.DataFrame(rows)

    # doppelte Kombinationen aus Text/Label/Intent entfernen (zur Sicherheit)
    base_df = base_df.drop_duplicates(
        subset=["text", "label", "intent"]
    ).reset_index(drop=True)

    # Preprocessing für Modell/Features
```

```
    base_df["text_clean"] = base_df["text"].astype(str).apply(preprocess_text_chat)

    base_df.to_csv(out_csv, index=False, encoding="utf-8")
    print(f"Neues Basis-DF gespeichert als: {out_csv}")
    print(base_df.head())
    print("\nAnzahl Beispiele total:", len(base_df))
    print("\nKlassenverteilung (label):")
    print(base_df["label"].value_counts())
    print("\nIntent-Verteilung:")
    print(base_df["intent"].value_counts())
    print("\nAnteil Seeds (is_seed):")
    print(base_df["is_seed"].value_counts())

    return base_df


def build_chatpairs_dataset(
    in_csv: str = DATA_CSV_BASE,
    out_csv: str = DATA_CSV_CHATPAIRS,
) -> pd.DataFrame:
    """Chatpair-Datensatz (Usertext + Standardantwort) bauen und speichern."""
    df = pd.read_csv(in_csv)
    required_cols = {"text", "label", "intent", "text_clean"}
    missing = required_cols - set(df.columns)
    if missing:
        raise ValueError(
            f"Fehlende Spalten in {in_csv}: {missing} – bitte zuerst Basis-Datensatz bauen."
        )

    chatpairs_df = df.copy().rename(columns={
        "text": "user_text",
        "text_clean": "user_text_clean",
    })
    if "is_seed" not in chatpairs_df.columns:
        chatpairs_df["is_seed"] = True

    chatpairs_df["answer_mundart"] = chatpairs_df.apply(
        lambda row: get_default_answer_mundart(row["label"], row["intent"]),
        axis=1,
```

```
    )
    chatpairs_df["needs_review"] = True

    chatpairs_out = chatpairs_df[[
        "user_text",
        "user_text_clean",
        "label",
        "intent",
        "answer_mundart",
        "needs_review",
        "is_seed",
    ]]

    chatpairs_out.to_csv(out_csv, index=False, encoding="utf-8")
    print(f"\nNeuer Mundart-Chatpair-Datensatz gespeichert als: {out_csv}")
    print(chatpairs_out.head(10))
    print("\nVerteilung label:")
    print(chatpairs_out["label"].value_counts())
    print("\nVerteilung intent:")
    print(chatpairs_out["intent"].value_counts())
    print("\nAnteil Seeds (is_seed):")
    print(chatpairs_out["is_seed"].value_counts())
    return chatpairs_out


if __name__ == "__main__":
    # Nur Datensätze aus den EXAMPLES erstellen (ohne Augmentation)
    build_base_dataset()
    build_chatpairs_dataset()
```

Code für Streamlit

```python
import numpy as np
import pandas as pd
from collections import Counter

import streamlit as st

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.pipeline import Pipeline
from sklearn.metrics import (
    classification_report,
    accuracy_score,
    confusion_matrix,
)
from sklearn.metrics.pairwise import cosine_similarity

from sentence_transformers import SentenceTransformer

import matplotlib.pyplot as plt

from mundartchat_data import (
    RANDOM_STATE,
    DATA_CSV_BASE,
    DATA_CSV_CHATPAIRS,
    LABEL_ORDER,
    TOKEN_PATTERN,
    preprocess_text_chat,
    build_base_dataset,
    build_chatpairs_dataset,
)

# ============================================================
# Globale Config
# ============================================================

SBERT_MODEL_NAME = "sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2"
BATCH_SIZE = 32
```

```python
# ============================================================
# Daten laden / erstellen
# ============================================================

@st.cache_data
def load_datasets():
    """Basis- und Chatpair-Datensätze laden, bei Bedarf neu erstellen."""
    try:
        base_df = pd.read_csv(DATA_CSV_BASE)
    except FileNotFoundError:
        base_df = build_base_dataset()

    if "text_clean" not in base_df.columns:
        base_df["text_clean"] = base_df["text"].astype(str).apply(preprocess_text_chat)

    try:
        resp_df = pd.read_csv(DATA_CSV_CHATPAIRS)
    except FileNotFoundError:
        resp_df = build_chatpairs_dataset()

    return base_df, resp_df


# ============================================================
# N-Gramm Language Model
# ============================================================

def tokenize_for_lm(text: str):
    clean = preprocess_text_chat(text)
    if not clean:
        return []
    return clean.split()


def train_ngram_lm(texts, n_max: int = 3):
    ngram_counts = {n: Counter() for n in range(1, n_max + 1)}
    for t in texts:
        toks = tokenize_for_lm(t)
        if not toks:
            continue
```

```
        toks = ["<s>"] + toks + ["</s>"]
        for n in range(1, n_max + 1):
            if len(toks) < n:
                continue
            for i in range(len(toks) - n + 1):
                ngram = tuple(toks[i:i + n])
                ngram_counts[n][ngram] += 1

    def lm_analyzer(text: str):
        return tokenize_for_lm(text)

    return ngram_counts, lm_analyzer


def _is_good_token(tok: str) -> bool:
    if tok in ("<s>", "</s>"):
        return False
    if tok.startswith("<") and tok.endswith(">"):
        return False
    if len(tok) < 2:
        return False
    return True


def get_top_ngrams(ngram_counts, n: int, topk: int = 20) -> pd.DataFrame:
    if n not in ngram_counts:
        return pd.DataFrame(columns=["ngram", "count", "rel_freq"])

    counter = ngram_counts[n]
    if not counter:
        return pd.DataFrame(columns=["ngram", "count", "rel_freq"])

    total = sum(counter.values())
    rows = []
    for ng, cnt in counter.most_common(topk * 3):
        text = " ".join(ng)

        if "<s>" in ng or "</s>" in ng:
            continue
        if text.strip() in ("<s>", "</s>"):
```

```
            continue

        rows.append({
            "ngram": text,
            "count": cnt,
            "rel_freq": round(cnt / total, 3),
        })

    return pd.DataFrame(rows[:topk])


def next_word_candidates(prefix, ngram_counts, analyzer, n_max=3, topk=5):
    toks = analyzer(preprocess_text_chat(prefix))
    backoff_level = None

    for n in range(n_max, 0, -1):
        if n == 1:
            total = sum(
                cnt
                for (tok,), cnt in ngram_counts[1].items()
                if _is_good_token(tok)
            )
            if total == 0:
                continue
            candidates = [
                (tok, cnt)
                for (tok,), cnt in ngram_counts[1].most_common()
                if _is_good_token(tok)
            ][:topk]
            backoff_level = 1
            probs = [(w, c / float(total)) for w, c in candidates]
            return probs, backoff_level

        if len(toks) < n - 1:
            continue

        context = tuple(toks[-(n - 1):])
        candidates = []
        for ng, cnt in ngram_counts[n].items():
            if ng[:-1] == context:
```

```
                w = ng[-1]
                if _is_good_token(w):
                    candidates.append((w, cnt))

        if candidates:
            candidates.sort(key=lambda x: x[1], reverse=True)
            candidates = candidates[:topk]
            total_cnt = sum(c for _, c in candidates)
            backoff_level = n
            probs = [(w, c / float(total_cnt)) for w, c in candidates]
            return probs, backoff_level

    return [], None


# ============================================================
# Modelle trainieren
# ============================================================

@st.cache_resource
def train_all_models(base_df: pd.DataFrame, resp_df: pd.DataFrame):
    """Trainiert Klassifikationsmodelle, LM und Retrieval-Komponenten."""
    # Split
    X_tr_clean, X_te_clean, y_train, y_test = train_test_split(
        base_df["text_clean"],
        base_df["label"],
        test_size=0.25,
        random_state=RANDOM_STATE,
        stratify=base_df["label"],
    )

    X_tr_raw = base_df.loc[X_tr_clean.index, "text"]
    X_te_raw = base_df.loc[X_te_clean.index, "text"]

    # BoW / TF-IDF
    bow = Pipeline([
        ("vec", CountVectorizer(token_pattern=TOKEN_PATTERN, min_df=2)),
        ("clf", LogisticRegression(max_iter=1000, random_state=RANDOM_STATE)),
    ]).fit(X_tr_clean, y_train)
```

```
tfidf = Pipeline([
    ("vec", TfidfVectorizer(ngram_range=(1, 2), token_pattern=TOKEN_PATTERN, min_df=2)),
    ("clf", LogisticRegression(max_iter=1000, random_state=RANDOM_STATE)),
]).fit(X_tr_clean, y_train)

# SBERT + LR
sbert_model = SentenceTransformer(SBERT_MODEL_NAME)
X_list_tr = pd.Series(X_tr_raw).astype(str).tolist()
emb_train = sbert_model.encode(
    X_list_tr,
    convert_to_numpy=True,
    batch_size=BATCH_SIZE,
)
sbert_clf = LogisticRegression(
    max_iter=1000,
    random_state=RANDOM_STATE,
).fit(emb_train, y_train)

# Evaluation
X_list_te = pd.Series(X_te_raw).astype(str).tolist()
emb_test = sbert_model.encode(
    X_list_te,
    convert_to_numpy=True,
    batch_size=BATCH_SIZE,
)
y_pred_sbert = sbert_clf.predict(emb_test)

eval_info = {}

# BoW
y_pred_bow = bow.predict(X_te_clean)
report_bow = classification_report(
    y_test, y_pred_bow, digits=3, output_dict=True
)
eval_info["bow"] = {
    "report": report_bow,
    "accuracy": accuracy_score(y_test, y_pred_bow),
}

# TF-IDF
```

```
y_pred_tfidf = tfidf.predict(X_te_clean)
report_tfidf = classification_report(
    y_test, y_pred_tfidf, digits=3, output_dict=True
)
eval_info["tfidf"] = {
    "report": report_tfidf,
    "accuracy": accuracy_score(y_test, y_pred_tfidf),
}

# SBERT
report_sbert = classification_report(
    y_test, y_pred_sbert, digits=3, output_dict=True
)
cm_sbert = confusion_matrix(y_test, y_pred_sbert, labels=LABEL_ORDER)
cm_df = pd.DataFrame(
    cm_sbert,
    index=[f"true_{l}" for l in LABEL_ORDER],
    columns=[f"pred_{l}" for l in LABEL_ORDER],
)
eval_info["sbert"] = {
    "report": report_sbert,
    "accuracy": accuracy_score(y_test, y_pred_sbert),
    "confusion_matrix": cm_df,
}

# N-Gramm LM
ngram_counts, lm_analyzer = train_ngram_lm(base_df["text_clean"], n_max=3)

# Antwort-Retrieval
resp_df = resp_df[
    resp_df["answer_mundart"].astype(str).str.len() > 0
].reset_index(drop=True)

resp_emb = sbert_model.encode(
    resp_df["user_text"].astype(str).tolist(),
    convert_to_numpy=True,
    batch_size=BATCH_SIZE,
)

models = {
```

```
        "bow": bow,
        "tfidf": tfidf,
        "sbert_model": sbert_model,
        "sbert_clf": sbert_clf,
        "ngram_counts": ngram_counts,
        "lm_analyzer": lm_analyzer,
        "resp_df": resp_df,
        "resp_emb": resp_emb,
        "eval_info": eval_info,
    }
    return models


# ============================================================
# Inferenz-Helper (Klassifikation & Retrieval)
# ============================================================

def probs_pipeline(model, texts):
    vec = model.named_steps["vec"]
    clf = model.named_steps["clf"]
    X = vec.transform(texts)
    P = clf.predict_proba(X)
    cls = clf.classes_
    out = []
    for p in P:
        out.append({c: float(p[i]) for i, c in enumerate(cls)})
    return out


def sbert_predict(models, texts):
    sbert_model = models["sbert_model"]
    sbert_clf = models["sbert_clf"]
    X = pd.Series(texts).astype(str).tolist()
    emb = sbert_model.encode(
        X,
        convert_to_numpy=True,
        batch_size=BATCH_SIZE,
    )
    return sbert_clf.predict(emb)
```

```python
def sbert_predict_proba(models, texts):
    sbert_model = models["sbert_model"]
    sbert_clf = models["sbert_clf"]
    X = pd.Series(texts).astype(str).tolist()
    emb = sbert_model.encode(
        X,
        convert_to_numpy=True,
        batch_size=BATCH_SIZE,
    )
    P = sbert_clf.predict_proba(emb)
    cls = sbert_clf.classes_
    out = []
    for p in P:
        out.append({c: float(p[i]) for i, c in enumerate(cls)})
    return out


def classify_text(models, raw_inp: str):
    clean_inp = preprocess_text_chat(raw_inp)

    bow = models["bow"]
    tfidf = models["tfidf"]

    bow_pred = bow.predict([clean_inp])[0]
    tfidf_pred = tfidf.predict([clean_inp])[0]
    sbert_pred = sbert_predict(models, [raw_inp])[0]

    bow_probs = probs_pipeline(bow, [clean_inp])[0]
    tfidf_probs = probs_pipeline(tfidf, [clean_inp])[0]
    sbert_probs = sbert_predict_proba(models, [raw_inp])[0]

    return {
        "bow_pred": bow_pred,
        "tfidf_pred": tfidf_pred,
        "sbert_pred": sbert_pred,
        "bow_probs": bow_probs,
        "tfidf_probs": tfidf_probs,
        "sbert_probs": sbert_probs,
    }
```

```python
def generate_answer(
    models,
    user_text: str,
    predicted_label: str | None = None,
    topk: int = 5,
    min_sim: float = 0.2,
):
    resp_df = models["resp_df"]
    resp_emb = models["resp_emb"]
    sbert_model = models["sbert_model"]

    if len(resp_df) == 0:
        return None, None

    q_emb = sbert_model.encode([user_text], convert_to_numpy=True)
    sims = cosine_similarity(q_emb, resp_emb)[0]

    candidate_idx = np.arange(len(resp_df))
    if predicted_label is not None and "label" in resp_df.columns:
        mask = (resp_df["label"].astype(str) == str(predicted_label))
        if mask.any():
            candidate_idx = np.where(mask)[0]

    if len(candidate_idx) == 0:
        candidate_idx = np.arange(len(resp_df))

    sims_sub = sims[candidate_idx]
    top_local = np.argsort(-sims_sub)[:min(topk, len(sims_sub))]

    best_local_idx = top_local[0]
    best_idx = candidate_idx[best_local_idx]
    best_sim = float(sims_sub[best_local_idx])
    best_answer = resp_df.iloc[best_idx]["answer_mundart"]

    if best_sim < min_sim:
        return None, best_sim

    return best_answer, best_sim
```

```python
def debug_neighbors(
    models,
    raw_inp: str,
    topn: int = 5,
    filter_by_label: bool = True,
):
    resp_df = models["resp_df"]
    resp_emb = models["resp_emb"]
    sbert_model = models["sbert_model"]

    sbert_label = sbert_predict(models, [raw_inp])[0]

    q_emb = sbert_model.encode([raw_inp], convert_to_numpy=True)
    sims = cosine_similarity(q_emb, resp_emb)[0]

    candidate_idx = np.arange(len(resp_df))
    if filter_by_label and "label" in resp_df.columns:
        mask = (resp_df["label"].astype(str) == str(sbert_label))
        if mask.any():
            candidate_idx = np.where(mask)[0]

    if len(candidate_idx) == 0:
        return sbert_label, []

    sims_sub = sims[candidate_idx]
    order = np.argsort(-sims_sub)[:min(topn, len(sims_sub))]

    neighbors = []
    for local_idx in order:
        idx = candidate_idx[local_idx]
        row = resp_df.iloc[idx]
        sim = sims_sub[local_idx]

        neighbors.append({
            "similarity": float(sim),
            "user_text": str(row["user_text"]),
            "answer_mundart": str(row["answer_mundart"]),
            "label": row.get("label", "?"),
```

```
            "intent": row.get("intent", "?"),
            "is_seed": bool(row.get("is_seed", False)),
        })

    return sbert_label, neighbors


# ============================================================
# Token-/Zipf-Analyse
# ============================================================

def get_token_freqs(texts):
    freqs = Counter()
    for t in texts:
        clean = preprocess_text_chat(str(t))
        if not clean:
            continue
        for tok in clean.split():
            freqs[tok] += 1
    return freqs


def plot_zipf_with_fit(
    df: pd.DataFrame,
    text_col: str = "text_clean",
    min_freq: int = 1,
    fit_range: tuple[int, int] | None = (5, 100),
    title_suffix: str = "",
):
    texts = df[text_col].astype(str)
    freqs = get_token_freqs(texts)

    counts = np.array(sorted(
        [c for c in freqs.values() if c >= min_freq],
        reverse=True
    ))
    ranks = np.arange(1, len(counts) + 1)

    log_ranks = np.log(ranks)
    log_counts = np.log(counts)
```

```
hapax_mask = counts == 1
non_hapax_mask = ~hapax_mask

fig, ax = plt.subplots(figsize=(7, 5))

if non_hapax_mask.any():
    ax.loglog(
        ranks[non_hapax_mask],
        counts[non_hapax_mask],
        "o",
        markersize=4,
        alpha=0.7,
        label="Tokens (freq > 1)",
    )

if hapax_mask.any():
    ax.loglog(
        ranks[hapax_mask],
        counts[hapax_mask],
        "o",
        markersize=4,
        alpha=0.7,
        color="red",
        linestyle="none",
        label="Hapax (freq = 1)",
    )

if fit_range is not None:
    r_start, r_end = fit_range
    mask = (ranks >= r_start) & (ranks <= r_end)
else:
    mask = np.ones_like(ranks, dtype=bool)

X = log_ranks[mask].reshape(-1, 1)
y = log_counts[mask]

reg = LinearRegression().fit(X, y)
slope = reg.coef_[0]
```

```
    y_fit = reg.predict(log_ranks.reshape(-1, 1))
    ax.loglog(
        ranks,
        np.exp(y_fit),
        "--",
        label=f"Fit: slope={slope:.2f}",
    )

    zipf_slope = -1.0
    zipf_intercept = np.log(counts[0]) - zipf_slope * np.log(ranks[0])
    zipf_line = np.exp(zipf_intercept + zipf_slope * log_ranks)
    ax.loglog(
        ranks,
        zipf_line,
        ":",
        label="Referenz: slope = -1",
    )

    ax.set_xlabel("Rang")
    ax.set_ylabel("Häufigkeit")
    title = "Zipf-Plot mit Regressionsgeraden"
    if title_suffix:
        title += f" – {title_suffix}"
    ax.set_title(title)
    ax.legend()
    fig.tight_layout()

    alpha = -slope
    return fig, alpha, slope


# ============================================================
# Streamlit UI
# ============================================================

def main():
    st.set_page_config(
        page_title="Mundart-Chat Demo",
        page_icon="💬",
        layout="wide",
```

```
    )

    st.title("Mundart-Chat Demo")
    st.caption("Sentiment, Next-Word, Antwort-Retrieval für Schweizerdeutsch-Chat")

    # ---- Daten & Modelle EINMAL laden/trainieren ----
    base_df, resp_df = load_datasets()
    with st.spinner("Modelle werden geladen / trainiert ..."):
        models = train_all_models(base_df, resp_df)
    eval_info = models["eval_info"]

    # ---------------- Sidebar: Daten & Modelle ----------------
    with st.sidebar:
        st.header("📊 Modelle & Datengrundlage")
        st.write(f"💬 Anzahl Chatnachrichten: {len(resp_df)}")

        with st.expander("😊 Sentiment (3) & Intents (18)"):
            st.markdown(
                """
**😔 negativ**
- 😵 Stress & Überforderung (50)
- ⚡ Konflikte & Spannungen (50)
- 😟 Selbstzweifel & Unsicherheit (50)
- 🥀 Traurigkeit & Einsamkeit (50)
- 🩺 Gesundheit & Sorgen (50)
- 💭 Kurznachricht (negativ) (50)

**😐 neutral**
- 💬 Smalltalk / Allgemeines (50)
- 🗓️ Organisation & Abmachungen (50)
- ❓ Info-Fragen & Erklärungen (50)
- 🎮 Hobbys & Interessen (50)
- 💻 Tech-Support & Sachprobleme (50)
- 🔹 Kurznachricht (neutral) (50)

**😊 positiv**
- 🙏 Dankbarkeit (50)
```

```
- 😄 Freude & Gute Laune (50)
- 🏆 Erfolg & Stolz (50)
- 🤝 Verbundenheit & Nähe (50)
- 🚀 Motivation & Vorfreude (50)
- ✨ Kurznachricht (positiv) (50)
                """
            )

        with st.expander("🧠 Standardantworten (Defaults)"):
            st.markdown(
                """
- Jede Nachricht erhält automatisch eine passende Antwort
  basierend auf **Sentiment** (negativ/neutral/positiv)
  und **Intent** (18 Kategorien).
- Pro Intent gibt es mehrere Varianten inkl. **Kurznachrichten**.
- Falls keine Intent-spezifische Antwort existiert, greift ein
  allgemeiner Fallback je Sentiment.
                """
            )

        with st.expander("💻 Verwendete Modelle"):
            st.markdown(
                """
**1. Klassifikation (Sentiment & Intents)**
- Bag-of-Words + Logistic Regression
- TF-IDF (1–2-Gramme) + Logistic Regression
- SBERT-Embeddings + Logistic Regression

**2. Sprachmodell (Next-Word)**
- Einfaches N-Gramm-Modell (1–3-Gramme, Backoff)

**3. Antwort-Retrieval**
- SBERT-Embeddings + Kosinus-Ähnlichkeit
- Sucht die ähnlichsten Trainingsbeispiele und deren Antworten
                """
            )

        #1️⃣ Modell-Performance
        with st.expander("🔍 Modell-Performance (Testset)", expanded=False):
```

```
    for name, info in eval_info.items():
        st.subheader(name.upper())
        st.metric("Accuracy", f"{info['accuracy']:.3f}")

        report_dict = info["report"]
        df_report = pd.DataFrame(report_dict).T
        if "accuracy" in df_report.index:
            df_report = df_report.drop(index="accuracy")
        df_report = df_report[["precision", "recall", "f1-score", "support"]]
        st.dataframe(df_report, use_container_width=True)

        if "confusion_matrix" in info:
            st.caption(f"Confusion Matrix ({name.upper()})")
            cm_df = info["confusion_matrix"]
            cm = cm_df.to_numpy()
            fig, ax = plt.subplots()
            ax.imshow(cm, cmap="Blues")
            ax.set_xticks(np.arange(len(LABEL_ORDER)))
            ax.set_yticks(np.arange(len(LABEL_ORDER)))
            ax.set_xticklabels(LABEL_ORDER)
            ax.set_yticklabels(LABEL_ORDER)
            ax.set_xlabel("Predicted label")
            ax.set_ylabel("True label")
            for i in range(cm.shape[0]):
                for j in range(cm.shape[1]):
                    ax.text(j, i, int(cm[i, j]), ha="center", va="center")
            fig.tight_layout()
            st.pyplot(fig)

#2️⃣ Label-Verteilung
with st.expander("📈 Label-Verteilung", expanded=False):
    label_counts = base_df["label"].value_counts().reindex(LABEL_ORDER, fill_value=0)
    st.bar_chart(label_counts)
    st.write(label_counts)

#3️⃣ Textlängen
with st.expander("📏 Textlängen (Tokens)", expanded=False):
    lengths = base_df["text_clean"].astype(str).apply(
        lambda t: len(t.split()) if t.strip() else 0
```

```
    )
    st.write(f"Ø Länge: {lengths.mean():.1f} Tokens")
    st.write(f"Median: {lengths.median():.0f} Tokens")
    st.write(f"Max: {lengths.max():.0f} Tokens")
    st.bar_chart(lengths.value_counts().sort_index())

#④ Token-Statistik
with st.expander("🔤 Token-Statistik", expanded=False):
    unigram_counter = models["ngram_counts"][1]
    total_types = len(unigram_counter)
    total_tokens = sum(unigram_counter.values())
    hapax = [
        tok for (tok,), cnt in unigram_counter.items()
        if cnt == 1 and _is_good_token(tok)
    ]
    st.metric("Token-Typen (Vokab)", total_types)
    st.metric("Token-Instanzen (laufende Wörter)", total_tokens)
    st.metric("Hapax-Typen", len(hapax))
    st.metric("Hapax-Anteil", f"{len(hapax) / total_types:.2f}")
    st.write("Beispiele (Hapax):")
    st.write(", ".join(hapax[:10]))

#⑤ Zipf-Analyse
with st.expander("📐 Zipf-Analyse (Token-Verteilung)", expanded=False):
    try:
        fig_zipf, alpha, slope = plot_zipf_with_fit(
            base_df,
            text_col="text_clean",
            min_freq=1,
            fit_range=(5, 100),
            title_suffix="Basisdaten",
        )
        st.pyplot(fig_zipf)
        st.caption(f"Zipf-Exponent α ≈ {alpha:.3f}, Steigung ≈ {slope:.3f}")
    except Exception as e:
        st.warning(f"Zipf-Plot konnte nicht berechnet werden: {e}")

#⑥ N-Gramm-Statistik
with st.expander("🧩 N-Gramm-Statistik (LM)", expanded=False):
```

```
        ngram_counts = models["ngram_counts"]
        st.subheader("Unigramme (1-Gramme)")
        df_uni = get_top_ngrams(ngram_counts, n=1, topk=20)
        st.dataframe(df_uni, use_container_width=True)
        st.subheader("Bigramme (2-Gramme)")
        df_bi = get_top_ngrams(ngram_counts, n=2, topk=20)
        st.dataframe(df_bi, use_container_width=True)

    #7️⃣ Projekt-PDF
    with st.expander("📄 Projektpräsentation (PDF)", expanded=False):
        pdf_path = "Schlusspräsentation.pdf"
        try:
            with open(pdf_path, "rb") as f:
                pdf_bytes = f.read()
            st.download_button(
                label="📥 Präsentation als PDF herunterladen",
                data=pdf_bytes,
                file_name="Schlusspräsentation.pdf",
                mime="application/pdf",
            )
        except FileNotFoundError:
            st.warning(f"PDF nicht gefunden unter: {pdf_path}")

# ---------------- Eingabe ----------------
user_text = st.text_area(
    "Mundart-Nachricht eingeben",
    height=120,
    placeholder="z.B. «ich ha kei bock meh uf dä stress»",
)

# ---------------- Tabs ----------------
tab1, tab2, tab3, tab4 = st.tabs([
    "Sentiment-Klassifikation",
    "Next-Word Vorschlag",
    "Antwortvorschlag",
    "Debug Nachbarn",
])

# --- Tab 1: Klassifikation ---
```

```
with tab1:
    if st.button("Klassifizieren", key="btn_classify"):
        if not user_text.strip():
            st.warning("Bitte oben zuerst eine Nachricht eingeben.")
        else:
            with st.spinner("Klassifiziere ..."):
                cls = classify_text(models, user_text)

            col1, col2, col3 = st.columns(3)
            col1.metric("BoW", cls["bow_pred"])
            col2.metric("TF-IDF", cls["tfidf_pred"])
            col3.metric("SBERT", cls["sbert_pred"])

            st.subheader("Wahrscheinlichkeiten")
            probs_df = pd.DataFrame([
                {**{"modell": "BoW"}, **cls["bow_probs"]},
                {**{"modell": "TF-IDF"}, **cls["tfidf_probs"]},
                {**{"modell": "SBERT"}, **cls["sbert_probs"]},
            ])
            st.dataframe(probs_df, use_container_width=True)

# --- Tab 2: Next-Word ---
with tab2:
    if st.button("Next-Word Vorschläge berechnen", key="btn_nextword"):
        if not user_text.strip():
            st.warning("Bitte oben zuerst eine Nachricht eingeben.")
        else:
            with st.spinner("Berechne Next-Word-Vorschläge ..."):
                cands, backoff = next_word_candidates(
                    user_text,
                    models["ngram_counts"],
                    models["lm_analyzer"],
                    n_max=3,
                    topk=5,
                )
            if not cands:
                st.warning("Keine brauchbaren Vorschläge gefunden.")
            else:
                if backoff == 3:
                    st.info("N-Gramm-Level: 3-Gramm (voller Kontext benutzt)")
```

```
                elif backoff == 2:
                    st.info("N-Gramm-Level: 2-Gramm (Backoff – nur letztes Wort)")
                elif backoff == 1:
                    st.info("N-Gramm-Level: 1-Gramm (Unigram-Fallback)")
                else:
                    st.info("N-Gramm-Level: unbekannt / kein Treffer")

                rows = []
                for w, p in cands:
                    rows.append({
                        "Token": w,
                        "p (relativ)": round(p, 3),
                        "Vorschlag": (user_text + " " + w).strip(),
                    })
                st.table(pd.DataFrame(rows))

# --- Tab 3: Antwortvorschlag ---
with tab3:
    if st.button("Antwort generieren", key="btn_answer"):
        if not user_text.strip():
            st.warning("Bitte oben zuerst eine Nachricht eingeben.")
        else:
            with st.spinner("Klassifiziere & suche passende Antwort ..."):
                cls = classify_text(models, user_text)
                sbert_label = cls["sbert_pred"]
                answer, sim = generate_answer(
                    models,
                    user_text,
                    predicted_label=sbert_label,
                    topk=5,
                    min_sim=0.2,
                )
            st.write(f"**SBERT-Label:** {sbert_label}")
            if answer is None:
                st.warning(
                    f"Keine passende Antwort im Datensatz gefunden "
                    f"(beste Ähnlichkeit: {sim:.2f})."
                )
            else:
                st.subheader("Antwortvorschlag (Mundart)")
```

```
                    st.success(answer)
                    st.caption(f"Ähnlichkeit zu Trainingsbeispielen: {sim:.2f}")

    # --- Tab 4: Debug Nachbarn ---
    with tab4:
        topn = st.slider("Anzahl Nachbarn", min_value=3, max_value=15, value=5)
        if st.button("Ähnlichste Beispiele anzeigen", key="btn_debug"):
            if not user_text.strip():
                st.warning("Bitte oben zuerst eine Nachricht eingeben.")
            else:
                with st.spinner("Suche ähnliche Beispiele ..."):
                    sbert_label, neighbors = debug_neighbors(
                        models,
                        user_text,
                        topn=topn,
                        filter_by_label=True,
                    )
                st.write(f"**SBERT-Label:** {sbert_label}")
                if not neighbors:
                    st.warning("Keine passenden Nachbarn gefunden.")
                else:
                    df_neighbors = pd.DataFrame(neighbors)
                    df_neighbors = df_neighbors.drop(columns=["is_seed"], errors="ignore")
                    st.dataframe(df_neighbors, use_container_width=True)


if __name__ == "__main__":
    main()
```