Assignment 7

Angewandte Modellierung 25

Carl Colmant

June 16, 2025

Exercise 1.

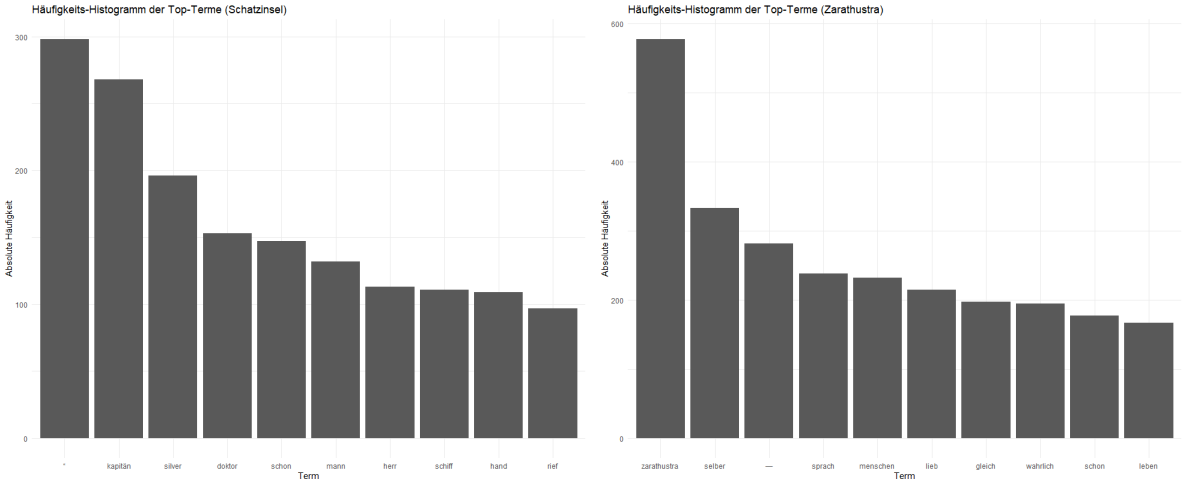
Ich habe mich für die Bücher "Also Sprach Zarathustra" und "Die Schatzinsel" entschieden in Deutscher original Sprache (ein großer Fehler). Ich habe direkt zwei gewählt weil ich später an probleme gestoßen bin und ich so die Möglichkeit habe zu vergleichen ob es an der Sprache liegt oder an den Texten.

2.

Zu erst habe ich die text dateien in R studio geladen. Zun Dann habe ich die Stoppwords raus genommen heir für kann man die sprache der erkannten stopp wörter angeben. Da zu kommt aber noch dass die Texte beide sehr alt sind (Die Schatzinsel ist von 1883 und Also Sprach Zarathustra auch von 1883) das führt dazu dass stopp wörter vor kommen die heute nicht mehr gebräuchlich sind und somit auch nicht in der Stoppwörter liste sind. ("dass", "diess", "'s", "wer", "—", "the", "konnt", "ganz", "mehr", "sagt","gutenberg™","project"). Nun entferne ich befor ich die Stoppwords aus dem Text entferne alle kommata, punkte, gänsefüße usw. und änder den Text in Kleinbuchstaben.

3.

Nun werden noch die Frequenzen der Wörter gezählt und dann kann auhc schon die Wordcloud erstellt werden. Daraus ergeben sich dann folgende Frequency Histogramme und Wordclouds:



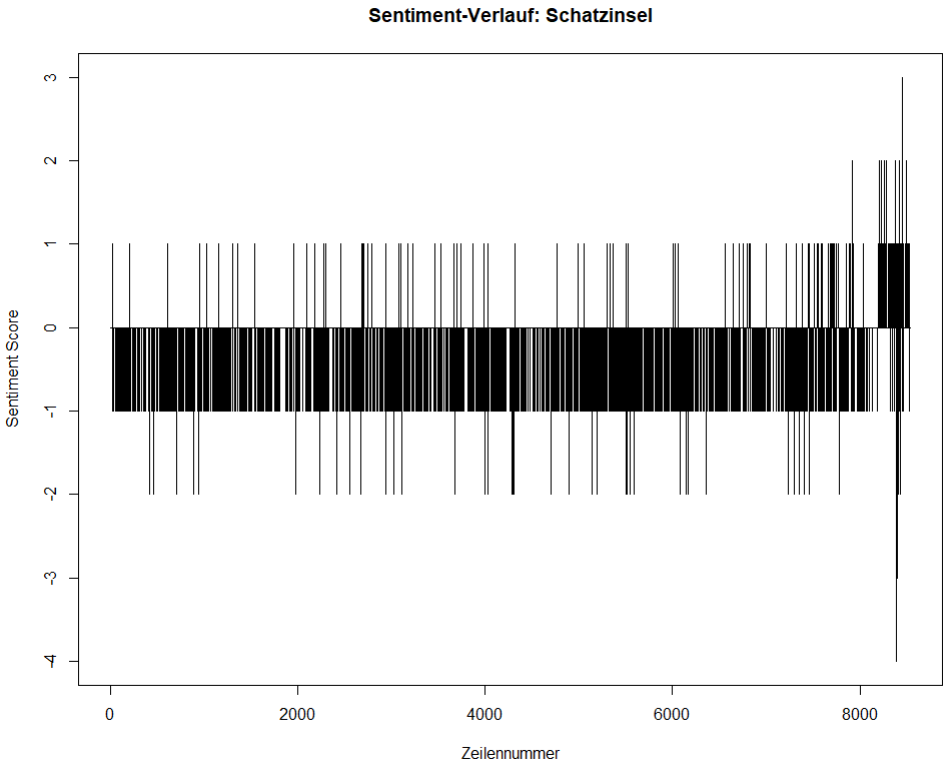
Wie man sieht ist bei der Schatzinsel immer noch ein hochkomma im Text enthalten, das passiert weil das ein extra sonderzeichen ist welches ich nicht gefunden habe. In zarathustra gibt es das gleiche Problem mit dem Wort "–" welches ich nicht gefunden habe.

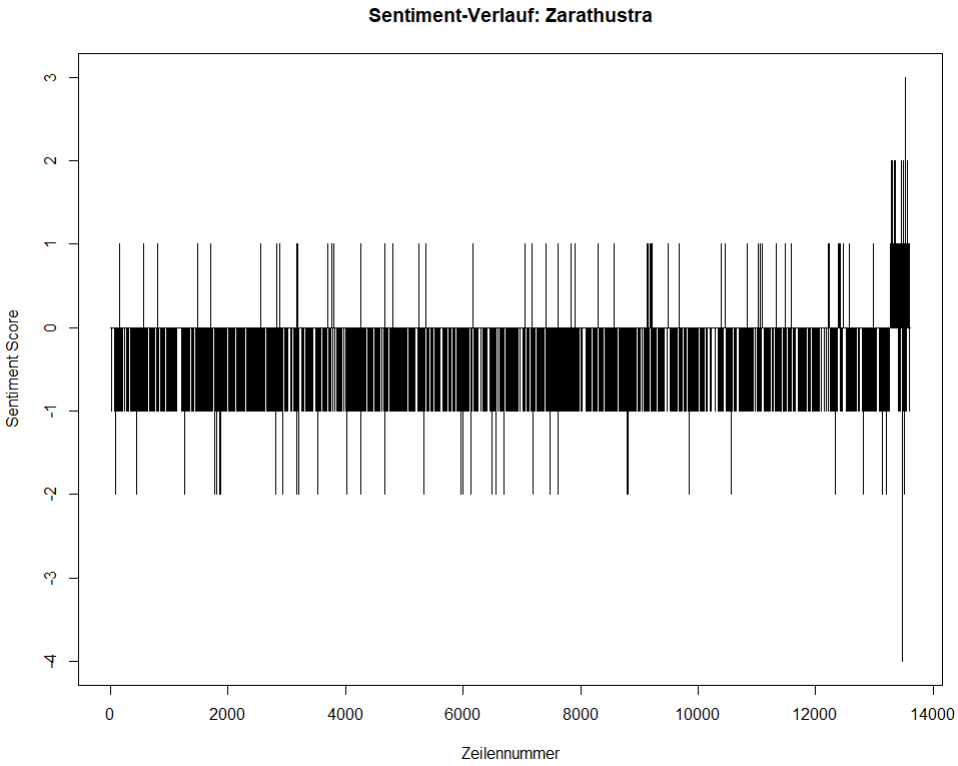

gehen
augenblick
fragt
beim
sicher
kaum
her
erwidert
hawkin
drei
stand
wenig
wasser
stimm
weiß
altenging
mutter
gesehen
sehen
wäre
john
fort
deck
beiden
fast
gut
squir
nie
sah
lag
haus
kopf
unser
kapitän
flint
hielt
gutenberg
smollett
jung
seit
work
jim
dabei
blockhaus
seid
doktor“
schon
leut
wort
macht
recht
zurück
herr“
hand
bord
ehe
alt
gerad
silver
rief
schiff
bald
gesicht
leben
lang
mann
land
herr
you
kam
glaub
werdet
habt
augen
immer
insel
ben
zwei
paar
sagen
hätte
zeit
sofort
ziemlich
with
this
schien
livesay
boot
erst
begann
plötzlich
hispaniola
sach
mußt
gunn
konnten

Wie man sieht beschäftigt sich die Schatzinsel als Roman vor allem mit seinen Charaktären und every day life so zu sagen. Zarathustra hingegen ist ein philosophisches Buch mit philosophischen Themen z.B. Leben und Tod und dem Geist und der Seele.

4.

Nun kommen wir aber an Probleme die sich ab hier durch die Aufgaben ziehen, Sentiment analysis ist sehr komplex und die modelle die es gibt sind oft für die englische Sprache trainiert. daher bekommt man hier:


Sentiment-Verlauf: Schatzinsel
Sentiment Score
3
2
1
0
-1
-2
-3
-4
0
2000
4000
6000
8000
Zeilennummer



Man sieht das sich die Stimmung der Texte kaum unterscheidet und kaum ausschlägt das ist besonders deswegen komisch weil Zarathustra ein mitreißendes Buch was mit vielen Emotionen geschrieben ist.

Exercise 2.

1.

Ich habe die Texte von davor wiederverwendet und also ein 'Genre Mashup' gemacht. Dazu habe ich in python folgende Bibliotheken benutzt:

```
import os
import re
import nltk
from nltk.corpus import stopwords
from nltk.probability import FreqDist
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from collections import Counter
```

2.

Nun müssen die texte für die nächsten Aufgabe vorbereitet werden. Dazu werden stoppwords entfernt.

```
ip install nltk matplotlib wordcloud spacy textblob-de
# python -m spacy download de_core_news_sm
#
# NLTK-Download für Stopwörter (nur einmalig nötig)
try:
    stopwords.words('german')
except LookupError:
    nltk.download('stopwords')

# Helper, um Dateipfade relativ zum Skript zu finden
# HINWEIS: Da __file__ in manchen Umgebungen nicht existiert, wird ein fester Pfad angenommen.
# Passen Sie BASE_DIR bei Bedarf an Ihr Arbeitsverzeichnis an.
BASE_DIR = os.path.dirname(os.path.abspath(__file__)) if '__file__' in locals() else '.'

def load_text(filename):
    """Lädt eine Textdatei aus dem Basisverzeichnis."""
    path = os.path.join(BASE_DIR, filename)
    if not os.path.exists(path):
        raise FileNotFoundError(f"Datei nicht gefunden: {path}
. Stellen Sie sicher, dass die Textdateien im selben Ordner wie das Skript liegen.")
    with open(path, 'r', encoding='utf-8') as f:
        return f.read()

# 1. Texte laden
try:
    t1 = load_text('pg49424.txt')              # "Die Schatzinsel"
    t2 = load_text('thus-spoke-zarathustra-data.txt')  # "Also sprach Zarathustra"
except FileNotFoundError as e:
    print(e)
    exit() # Beendet das Skript, wenn Dateien nicht gefunden werden

# 2. Vorverarbeitung
def preprocess(text):
    """Bereinigt und tokenisiert den Text."""
    text = re.sub(r'<.*?>', ' ', text)       # Entfernt HTML-Tags
    text = re.sub(r'[^\w\s]', ' ', text)    # Entfernt Satzzeichen
    text = re.sub(r'\d+', ' ', text)         # Entfernt Zahlen
    text = text.lower()                      # Konvertiert zu Kleinbuchstaben
    tokens = text.split()
    german_stops = set(stopwords.words('german'))
    return [t for t in tokens if t.isalpha() and t not in german_stops]

print("Verarbeite Texte...")
toks1 = preprocess(t1)
toks2 = preprocess(t2)
print("Texte verarbeitet.")
```

3.

Nun soll die Vocabular der beiden Texte verglichen werden das habe ich mit folgendem Code gemacht.

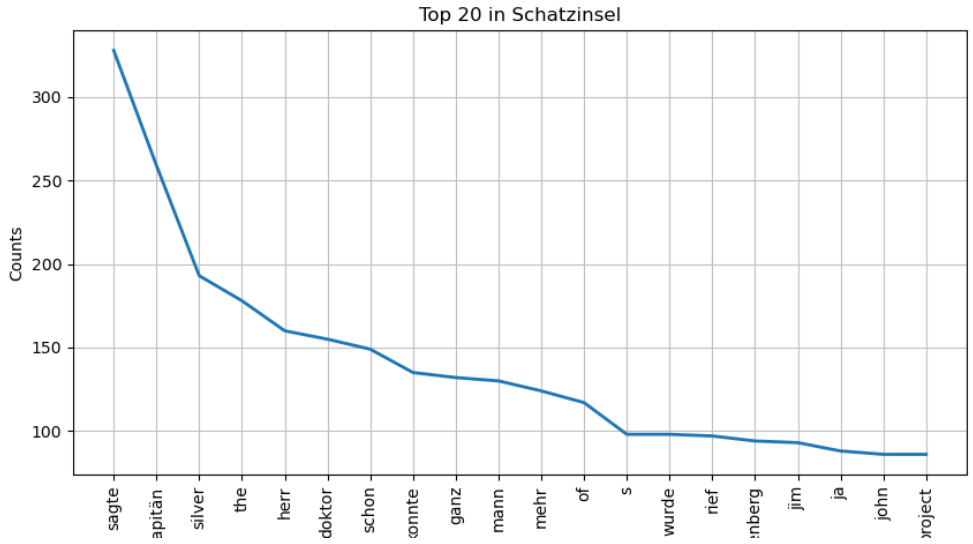
```
# 3. Worthäufigkeitsverteilungen
fd1 = FreqDist(toks1)
fd2 = FreqDist(toks2)

# Vergleichende Einblicke
shared = set(fd1) & set(fd2)
unique1 = set(fd1) - set(fd2)
unique2 = set(fd2) - set(fd1)

# ---------------------------------------------------------
# Shared Vocabulary und Unique Lexicons ausgeben
# ---------------------------------------------------------
print(f"\n--- Shared Vocabulary ({len(shared)} Wörter) ---")
print(', '.join(sorted(shared)[:100]))  # limitiert auf die ersten 100 Wörter

print(f"\n--- Unique to Schatzinsel ({len(unique1)} Wörter) ---")
print(', '.join(sorted(unique1)[:100]))

print(f"\n--- Unique to Zarathustra ({len(unique2)} Wörter) ---")
print(', '.join(sorted(unique2)[:100]))

# Anzeige der Top-20 Begriffe
def show_top(fd, label, N=20):
    print(f"\nTop {N} Wörter in {label}:")
    for w, c in fd.most_common(N):
        print(f"{w}: {c}")
    print()

show_top(fd1, 'Schatzinsel')
show_top(fd2, 'Also sprach Zarathustra')

# Balkendiagramme für Frequenzen
for fd, title in [(fd1, 'Schatzinsel'), (fd2, 'Zarathustra')]:
    plt.figure(figsize=(10, 5))
    fd.plot(20, title=f'Top 20 in {title}')
    plt.show()

# WordClouds für visuellen Kontrast
```

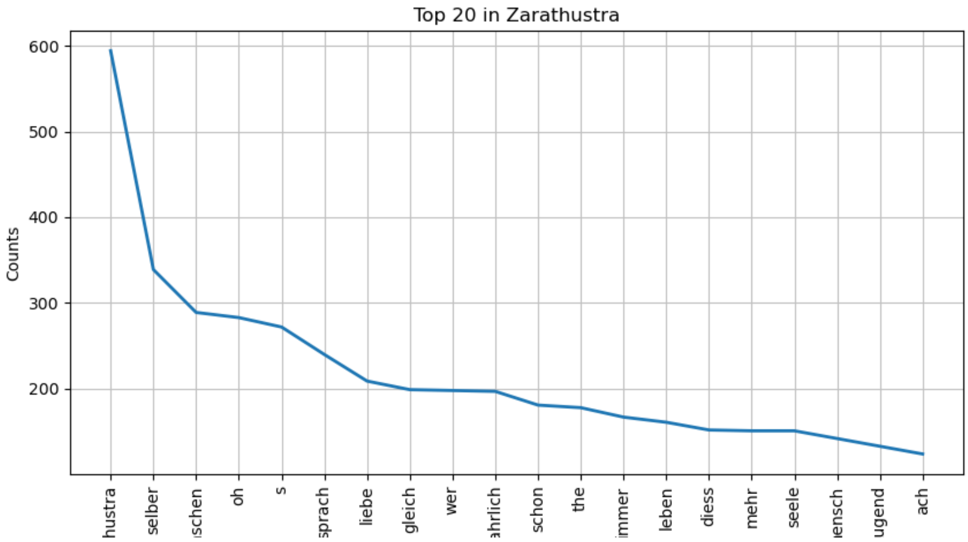
```
def wc_plot(fd, title):
    wc = WordCloud(width=800, height=400, background_color='White', collocations=False)
    wc.generate_from_frequencies(fd)
    plt.figure(figsize=(12, 6))
    plt.imshow(wc, interpolation='bilinear')
    plt.axis('off')
    plt.title(title)
    plt.show()

wc_plot(fd1, 'WordCloud – Schatzinsel')
wc_plot(fd2, 'WordCloud – Zarathustra')

# Häufigkeitsvergleich für ausgewählte Begriffe
def face_off(words):
    print("\nHäufigkeitsvergleich:")
    for w in words:
        print(f"{w}: Schatzinsel={fd1[w]}, Zarathustra={fd2[w]}")

    x = range(len(words))
    w1_counts = [fd1[w] for w in words]
    w2_counts = [fd2[w] for w in words]

    plt.figure(figsize=(8, 4))
    plt.bar([i - 0.2 for i in x], w1_counts, width=0.4, label='Schatzinsel')
    plt.bar([i + 0.2 for i in x], w2_counts, width=0.4, label='Zarathustra')
    plt.xticks(x, words)
    plt.ylabel('Häufigkeit')
    plt.title('Frequency Face-Off')
    plt.legend()
    plt.show()

selected_words = ['leben', 'tod', 'welt', 'mensch', 'gott', 'see']
face_off(selected_words)
```

Hieraus entsteht dann die Ausgabe:
— Shared Vocabulary (3104 Wörter) — a, ab, abend, abenteuer, aberglauben, abermals, abhang, abide, abkunft, abnehmen, about, abscheuliches, abschied, abwärts, abzeichnete, accept, accepted, accepting, access, accessed, accessible, accordance, ach, achseln, acht, achten, achtete, active, actual, addition, additional, additions, address, addresses, adern, adler, affe, affen, against, aged, agent, agree, agreed, agreement, ah, ahnt, all, alledem, allein, allow, almost, alone, already, alt, alte, altem, alten, alter, alteration, alternate, altes, amen, anbeginn, anblick, and, anfang, angenehm, angenehme, angenehmen, angenehmer, angst, anhöhe, anker, annehmen, antwort, antworten, antwortete, antworteten, any, anyone, anything, anywhere, anzuhören, apfel, appear, appearing, appears, applicable, apply, approach, arbeit, arbeiten, archive, are, arg, arise, arm, arme, armen, armer
— Unique to Schatzinsel (6063 Wörter) — aal, aalglatt, abbrechend, abdrücke, abe, abendbrot, abenden, abendwind, abenteuerlichen, abenteuern, abergläubischen, abfahrenden, abfallen, abfallenden, abfeuern, abflauen, abfuhr, abgebrochen, abgebrochenen, abgebröckelten, abgefangen, abgefeuert, abgehalten, abgehauenen, abgehärmten, abgekühlt, abgelegt, abgelenkt, abgemacht, abgeplattet, abgesandte, abgeschnitten, abgeschrägte, abgeschwindelt, abgesetzt, abgesperrten, abgesplitterter, abgestattet, abgetragenen, abgetrieben,

abgezapft, abgezogen, abhalten, abhanges, abhing, abhänge, abklang, ablaufen, ablehnt, ablehnte, ablud, abmurksen, abnahm, abnahmen, abprallen, abraham, abrechnen, abrechnungsbuch, abrede, abreisen, abrutschte, absatz, abscheuliche, abscheulichen, abscheulicher, abschießen, abschließend, abschloß, abschluß, abschneiden, absetzen, absicht, absichtlich, absitzen, absolut, abspringen, abstehen, absteigen, abstimmen, abstoßendere, abständen, abtrift, abtrocknete, abwarten, abwechselnd, abwechslung, abwechslungsreichsten, abweisungen, abwenden, abwesenheit, abzufangen, abzufeuern, abzuhalten, abzukühlen, abzuschießen, abzuschneiden, abzuwenden, achse, achsel, achselhöhle

— Unique to Zarathustra (6677 Wörter) — aas, aasvögel, abbild, abende, abendliche, abendmahl, abendroth, abendröthen, abends, abendwärts, aberwitzige, abführen, abgeflossen, abgegangen, abgegeben, abgehellter, abgekehrt, abgeknabbert, abgelaufnen, abgemerkt, abgenagt, abgeneidet, abgeraubt, abgerichtet, abgesagt, abgeschirrt, abgeschirrtem, abgethan, abgetödtet, abgewandten, abgezeichnete, abglanz, abgrund, abgrunde, abgrundwärts, abgründe, abgründen, abgründlich, abgründlichen, abgründlicher, abgründliches, abgründlichsten, abgünstig, abhelfen, abhold, abkehr, abkehrte, ablaufe, ableiten, ablerne, ablernen, ablernst, ablernte, abraum, absagen, absagte, abschaffen, abschaum, abschiede, abschieds, abschätzen, abseits, abspeisen, absterbende, absterbenden, abthun, abthätet, abtrennst, abtrünnigen, abwendet, abwägen, abzubitten, abzubrechen, abzählte, abzöge, achtbarkeiten, achtet, achtsam, adel, adelt, adlerhaft, adlern, adlers, afrikanisch, after, ahnungsvollen, allda, alleingehens, alleinsein, alleinseins, allerhöchsten, allerlei, allerliebsten, allermeisten, allerzierlichsten, allesammt, allezeit, allgemach, allgenügsame, allgenügsamen





Top 20 in Zarathustra
Counts
600
500
400
300
200
hustra
selber
ischen
oh
s
sprach
liebe
gleich
wer
ahrlich
schon
the
immer
leben
diess
mehr
seele
ensch
ugend
ach

WordCloud – Schatzinsel
sagte
kapitän
silver
the
doktor
herr
konnte
mehr
schon
ganz
mann
wurde
of
rief
john
jim
schiff
gutenberg
project
sagen
squire
hätte

4.

In diesem Teil sollten die Texte genauer auf stimmung, brnutzte Wortarten und Objekte wie z.B. Personen. Die Sentiment analysis hat leider nicht funktioniert weil einerseits der text auf Deutsch ist und sich conda geweigert hat bibliotheken zu finden die das ermöglicht hätten wie z.B. textblob_de. Dennoch habe ich keine mühen gescheut die POS und NER möglichst gut durchzuführen für die NER habe ich sogar ein transformer modell von huggingface benutzt. Hier zuerst mein Code:

```
# ======================================================================
# 4. Erweiterte Untersuchung
# ======================================================================

# 4b. POS Tagging & NER mit spaCy
try:
    import spacy
    try:
        # Lade das deutsche spaCy-Modell
        nlp = spacy.load('de_core_news_sm')
    except OSError:
        print("\nWARNUNG: SpaCy-Modell 'de_core_news_sm' nicht gefunden.")
        print("Bitte führen Sie im Terminal aus: python -m spacy download de_core_news_sm")
        nlp = None

    if nlp:
        print("\nFühre POS-Tagging und NER durch (kann einen Moment dauern)...")

        def pos_analysis(text, label):
            # Wir analysieren auch hier nur einen Teil des Textes zur Performance-Steigerung
            doc = nlp(text[:200000])

            # Zähle die Part-of-Speech Tags
            pos_counts = doc.count_by(spacy.attrs.POS)
            pos_top = sorted([(nlp.vocab[pos].text, count) for pos, count in pos_counts.items()], key
=lambda x: -x[1])[:5]

            print(f"\nAnalyse für: {label}")
            print(f"Top 5 POS-Tags: {pos_top}")

        pos_analysis(t1, 'Schatzinsel')
        pos_analysis(t2, 'Also sprach Zarathustra')
    else:
        print("\nPOS-Tagging und NER übersprungen, da das spaCy-Modell fehlt.")

except ImportError:
    print("\nWARNUNG: spaCy ist nicht installiert. POS-Tagging und NER werden übersprungen.")
    print("Bitte führen Sie aus: pip install spacy")
```

```python
import os
from collections import defaultdict
from transformers import AutoTokenizer, AutoModelForTokenClassification, pipeline

# ----------------------------------------------------------
# 4d. NER mit HuggingFace XLM-RoBERTa-Large für German CoNLL-03
# ----------------------------------------------------------

# 1. Modell-Name anpassen:
MODEL_NAME = "FacebookAI/xlm-roberta-large-finetuned-conll03-german"

# 2. Tokenizer und Modell laden
tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
model     = AutoModelForTokenClassification.from_pretrained(MODEL_NAME)

# 3. NER-Pipeline mit Aggregation
ner_pipeline = pipeline(
    "ner",
    model=model,
    tokenizer=tokenizer,
    aggregation_strategy="simple"  # kombiniert Token-Chunks zu vollständigen Entities
)

def ner_hf(text, label, max_len=5000):
    """
    Extrahiere Named Entities mit dem XLM-RoBERTa-Modell.
    Gibt ein Dict: {ENTITY_LABEL: set(Vorkommen)}
    """
    snippet = text[:max_len]  # auf den Anfang kürzen, um RAM & Zeit zu sparen
    raw_results = ner_pipeline(snippet)

    ents = defaultdict(set)
    for ent in raw_results:
        ents[ent["entity_group"]].add(ent["word"])

    # Ausgabe der Top-Entitäten je Typ
    print(f"\n--- HuggingFace NER ({label}) ---")
    for etype, words in ents.items():
        sample = sorted(words)[:20]
        print(f"{etype} ({len(words)}): {sample}")
    return ents

# 4. Aufruf für deine Texte
ents_xlm_1 = ner_hf(t1, "Schatzinsel")
ents_xlm_2 = ner_hf(t2, "Zarathustra")
```

Das Problem selbst mit nem Transformer Modell ist das ergebnis eher unterdurchschnittlich:
POS Ausgabe:
Analyse für: Also sprach Zarathustra
Top 5 POS-Tags: [('NOUN', 6860), ('PUNCT', 6005), ('PRON', 4537), ('DET', 4479), ('VERB', 3783)]
Analyse für: Schatzinsel
Top 5 POS-Tags: [('PUNCT', 6998), ('NOUN', 5710), ('PRON', 4459), ('VERB', 4207), ('DET', 4178)]

Die NER kann nur wenig identifiezieren:
— HuggingFace NER (Schatzinsel) —
PER (7): ['Ballantyne', 'Cooper', 'Kingston', 'Livesay', 'R. L. St.', 'S. L. O.', 'Trelawney']
MISC (1): ['amerikanische']
LOC (2): ['Admiral Benbow', 'Zum Admiral Benbow']‘
— HuggingFace NER (Zarathustra) —
PER (2): ['Friedrich Wilhelm Nietzsche', 'Zarathustra']
MISC (3): ['Amen-Lied', 'Ja-', 'Lied']
Man kann aber erkennen, dass Die Schatzinsel als Roman mehr wert auf das klare vorstellen der Charaktären legt während Also Sprach Zarathustra nicht so direkt die wichtigkeit der Charaktären festlegt. Beispiels weise wird ganz am anfang eine Figur vorgestellt Der Heilige der wharscheinlich Religiöse führer darstellen soll, dieser ist aber ausser für die vorbereitung der Massage des Buches kommplett unwichtig. Im gegensatz ist der Seiltänzer der etwas später getötet wird ein sehr wichtiger Teil der Argumentation für die übermenschen. Diese beiden Charaktäre kann man aber nur sehr schwer überhaupt automatisch finden da sie weder Namen haben noch besonders häufig vorkommen.

Exercise 3.

Ich hatte leider weder Zeit noch besonders viel erfollg mit Altair AI studio gerade mit Deutschen Texten hab ich das nicht hingekriegt.

Github

Wie immer sind alle meine benutzten Dateien auf meinem Github zu finden.