Übung 05
Wortspiel mit `Glove` in **Python**
INFI-IS
5xHWII

November 27, 2025

Abgabetermin:	lt. Moodle
Übungsleiter:	Albert Greinöcker

CCA – COMPETENCE CENTRE
HTL Anichstraße

Ziel der Übung:

- Umgang mit `gensim` (GENerate SIMilar)
- Finden von ähnlichen Wörtern
- Wie sieht die Aufbereitung von Texten für KI aus
- Umgang mit Emdeddings
- Kennenlernen von `SpaCy` und `nltk`

1 Vorbereitung

Installation

- `pip install gensim`
-

Folgendes Skript aus unserem `github`-Projekt `machine_learning_examples` könnte dabei hilfreich sein:

- `ex_15_gensim__glove_words2vec_embeddings.py`: Ein Beispiel wie man ähnliche Wörter findet und auch aus mehreren Wörtern die findet die nicht dazu gehören
- Laden eines fertig trainierten Word-Embedding-Modell:

```
import gensim.downloader as api
model = api.load("glove-wiki-gigaword-100")
```

2 4-Wort-Spiel (Odd-One-Out)

Entwickle ein Programm, das auf Basis von Wortvektoren (wir verwenden z.B. `GloVe`) ein kleines Spiel generiert. Das Spiel soll vier Wörter ausgeben, von denen drei semantisch ähnlich sind und ein Wort deutlich aus der Reihe fällt.

Der/die Benutzer*in soll erraten, welches Wort das "Odd One Out" ist. Anschließend soll das Programm die Wörter nach ihrer Vektorähnlichkeit sortieren.

Folgende Schritte sind dabei zu machen:

1. Laden des Word-Embedding-Modells (siehe oben)
2. Auswahl eines zufälligen Basiswortes aus dem Vokabular des Modells. Idealerweise sollte es ein wenig länger sein)
3. Ermitteln von drei Wörtern, die dem Basiswort besonders ähnlich sind (über `model.most_similar()`).
4. Erzeugen eines vierten Wortes, das möglichst unähnlich zum Basiswort ist (z. B. Cosine Similarity ¡ 0.1). Checke es auch nochmal mit `model.doesnt_match(words)`
5. Mischen der vier Wörter, damit nicht sofort erkennbar ist, welches nicht passt.
6. Gib die vier Wörter dem/der Spieler*in aus und lasse es auswählen. Danach checken ob das falsche Wort richtig ausgewählt wurde

Eine besondere Challenge ist die Wahl eines guten Basiswortes. Darauf könnte man achten:

- Auf die Wortlänge achten
- Wörter wählen, für die `most_similar()` stabile Ergebnisse liefern.
- Haptwörter eignen sich besser, diese kann man mit `Spacy` oder `nltk` herausfinden

```
import spacy
nlp = spacy.load("en_core_web_sm")

def is_noun(word):
  doc = nlp(word)
  return doc[0].pos_ == "NOUN" or doc[0].pos_ == "PROPN"
```

```
from nltk.corpus import wordnet as wn

def is_noun(word):
  return any(ss.pos() == 'n' for ss in wn.synsets(word))
```

Hier eine Beispielausgabe für das Basiswort `king`:

```
Wörter: ['queen', 'duke', 'prince', 'banana']

Deine Wahl: banana
Korrekt! 'banana' passt nicht zu den anderen.

Sortierung nach Ähnlichkeit:
```

```
queen   → 0.75
prince  → 0.62
duke    → 0.55
banana  → 0.02
```

Wie immer ist jede Erweiterung bzw. andere Spielidee mit ähnlichem Hintergrund gut denkbar.