Osadzenia słów i reprezentacje wektorowe

Laboratorium 7

1. Wprowadzenie

Osadzenia słów (ang. word embeddings) to gęste wektory liczbowe reprezentujące znaczenie słowa w taki sposób, że słowa o podobnym znaczeniu znajdują się blisko siebie w przestrzeni wektorowej. Stanowią one historyczny fundament dzisiejszych modeli transformerowych, które również używają warstw Embedding.

Na zajęciach zbudujemy reprezentacje od podstaw — od wektora one-hot, przez rozkład macierzy współwystępowań (SVD/PPMI), aż po klasyczne osadzenia word2vec trenowane za pomocą biblioteki Gensim — wszystko w całości lokalnie na procesorze (CPU), bez potrzeby dużych pretrenowanych modeli ani GPU.

- **Hipoteza dystrybucyjna —** znaczenie słowa wypływa z kontekstu, w którym bywa.
- **Similarity kosinusowe —** miara kąta między wektorami; im bliżej 0°, tym słowa bardziej pokrewne.
- **Algebra wektorów —** relacje typu król − mężczyzna + kobieta ≈ królowa wynikają z regularności kontekstów.

Techniki zastosowane w przykładach: PPMI, SVD oraz PCA

PPMI (Positive Pointwise Mutual Information) to statystyczna miara określająca, ile razy częściej dwa słowa występują razem, niż wynikałoby to z przypadku. Surowe liczby współwystąpień są bowiem zdominowane przez słowa bardzo częste (np. „i", „to"), dlatego zamiast nich używa się PMI; człon „Positive" oznacza, że ujemne (nieinformatywne) wartości przycina się do zera. Otrzymana w ten sposób macierz PPMI opisuje siłę powiązania każdej pary słów.

SVD (rozkład według wartości osobliwych, ang. Singular Value Decomposition) to metoda rozkładu macierzy na iloczyn trzech macierzy ($M = U \cdot \Sigma \cdot V^T$). Pozwala zredukować wielką, rzadką macierz PPMI do niewielkiej liczby gęstych, najważniejszych współrzędnych – w ten sposób uzyskujemy kompaktowe osadzenia słów, którymi dużo łatwiej operować niż całą macierzą współwystępowań.

PCA (ang. Principal Component Analysis, pol. analiza głównych składowych) to technika redukcji wymiaru służąca tu głównie do wizualizacji: rzutuje wysokowymiarowe osadzenia na 2–3 wymiary tak, aby zachować jak najwięcej wariancji. Dzięki temu na płaszczyźnie można zaobserwować, że słowa o podobnym znaczeniu tworzą widoczne skupiska.

2. Przykłady implementacji

Przykład 1: One-hot encoding i podobieństwo kosinusowe

```
# Program 1 - One-hot encoding i podobieństwo kosinusowe
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity

slowa = ['kot','pies','ryba','jablko','biega','lata']
V = len(slowa)
idx = {w:i for i,w in enumerate(slowa)}

def one_hot(w):
    v = np.zeros(V); v[idx[w]] = 1; return v

M = np.array([one_hot(w) for w in slowa])
print(M)

# Podobieństwo kosinusowe wszystkich par
sim = cosine_similarity(M)
print('macierz podobieństwa:'); print(np.round(sim,2))

# Wszystkie różne pary mają podobieństwo 0 -> brak wiedzy o znaczeniu
print('similarity kot-pies:', sim[idx['kot'], idx['pies']])
# Wizualizacja macierzy cieplnej similarności
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(6,5))
im = ax.imshow(sim, cmap='coolwarm', vmin=0, vmax=1)
ax.set_xticks(range(V)); ax.set_yticks(range(V))
ax.set_xticklabels(slowa, rotation=45); ax.set_yticklabels(slowa)
plt.colorbar(im); plt.title('Podobieństwo kosinusowe (one-hot)'); plt.show()
```

Przykład 2: Osadzenia z macierzy współwystępowań (PPMI + SVD)

Budujemy mały korpus, liczemy macierz współwystępowań, przekształcamy ją do postaci PPMI i redukujemy wymiar metodą SVD — otrzymujemy gęste osadzenia w pełni „z ręki”.

```
# Program 2 - Własne osadzenia słów (PPMI + SVD)
import numpy as np

corpus = [
    'kot siedzi na plotku i patrzy na psa'.split(),
    'pies biega za kotem po ogrodzie'.split(),
    'jablko spada z drzewa a kot patrzy'.split(),
    'ryba plywa w wodzie a pies szczeka'.split(),
]
words = sorted({w for s in corpus for w in s})
idx = {w:i for i,w in enumerate(words)}
```

```
V = len(words)
print('słownik:', words)
WINDOW = 2
cooc = np.zeros((V,V))
for s in corpus:
    for pos, t in enumerate(s):
        for dpos in range(-WINDOW, WINDOW+1):
            q = pos + dpos
            if 0 <= q < len(s) and q != pos:
                cooc[idx[t], idx[s[q]]] += 1
print('Macierz współwystępowań (fragment):')
print(cooc[:6,:6])
# PPMI (Positive Pointwise Mutual Information)
tot   = cooc.sum()
joint = cooc / tot
row   = cooc.sum(1, keepdims=True) / tot
col   = cooc.sum(0, keepdims=True) / tot
pmi   = np.log2((joint + 1e-9) / (row * col + 1e-9))
ppmi  = np.maximum(pmi, 0)

# Redukcja wymiaru SVD -> gęste osadzenia
U, S, Vt = np.linalg.svd(ppmi, full_matrices=False)
k = 2
emb = U[:, :k] * np.sqrt(S[:k])
for w in words:
    print(f'{w:12s}', np.round(emb[idx[w]], 3))
# Rzutowanie i wykres
import matplotlib.pyplot as plt
xs, ys = emb[:,0], emb[:,1]
plt.figure(figsize=(6,5))
plt.scatter(xs, ys, s=120)
# etykiety (bez polskich znaków ze względu na czcionkę konsoli)
short = {'jablko':'jablko','plotku':'plotek','ogrodzie':'ogrod'}
for w in words:
    lab = w
    x, y = xs[idx[w]], ys[idx[w]]
    plt.annotate(lab, (x,y), xytext=(5,5), textcoords='offset points')
plt.axhline(0,color='grey',lw=.5); plt.axvline(0,color='grey',lw=.5)
plt.title('Osadzenia PPMI+SVD'); plt.show()
```

Przykład 3: Trening Word2Vec z Gensim na własnym korpusie

```
#!pip install gensim
from gensim.models import Word2Vec

sentences = [
    ['python','jest','jezykiem','programowania'],
    ['java','to','inny','jezyk','programowania'],
    ['kot','jest','zwierzetem','domowym'],
    ['pies','jest','rowniez','zwierzetem'],
    ['javascript','sluzy','do','programowania','w','przegladarce'],
]
```

```
model = Word2Vec(sentences, vector_size=50, window=2,
                 min_count=1, sg=1, epochs=200, seed=42)
print('rozmiar słownika:', len(model.wv.key_to_index))
w = model.wv
print('wektor słowa python:'); print(w['python'])
print()
print("najbardziej podobne do 'jezyk':")
for word, score in w.most_similar('jezyk'):
    print(f'   {word:20s} {score:.3f}')
# Analogie (gdyby istniała relacja)
try:
    r = w.most_similar(positive=['pies','kot'], negative=[], topn=3)
    print('kot ~ pies -> ', [x[0] for x in r])
except Exception as e:
    print('za mało danych na analogię:', e)
```

Przykład 4: Podobieństwa, analogie i wizualizacja PCA

```
# Program 4 - Algebra wektorów i wizualizacja
# Zwiększamy korpus, aby uzyskać regularne konteksty
korpus = []
temats = {
 'zwierze': ['kot','pies','mysz','krolik','kon','owca'],
 'programowanie': ['python','java','csharp','ruby','php'],
 'pojazd': ['samochod','autobus','rower','motocykl','motor'],
}
for kat, slowa in temats.items():
    for a in slowa:
        for b in slowa:
            if a!=b:
                korpus.append([f'{a} jest jak {b}', 'nalezy do', kat])
sentences = [zd.split() for zd in [' '.join(k).replace("jak"+k,"").strip() for
k in []]]
print('Korpus automatycznie budowany w zadaniu — tu pokazujemy pipeline.')
import numpy as np
# Dla demonstracji trenujemy model na powtarzanym korpusie tematycznym
repeats = 40
sents = []
for kat, slowa in temats.items():
    for a in slowa:
        for _ in range(repeats):
            sents.append([a,           'jest',           kat,           'jak',
slowa[(slowa.index(a)+1)%len(slowa)]])
from gensim.models import Word2Vec
model   =   Word2Vec(sents,   vector_size=30,   window=2,   min_count=1,   sg=1,
epochs=100, seed=1)
w = model.wv

# Podobieństwo kosinusowe wybranych par
from sklearn.metrics.pairwise import cosine_similarity
pary = [('kot','pies'), ('kot','python'), ('python','java'), ('rower','motor')]
for a,b in pary:
    s = cosine_similarity([w[a]],[w[b]])[0][0]
```

```
    print(f'{a:10s}-{b:10s} sim={s:.3f}')
# Wizualizacja PCA przestrzeni osadzeń
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

words_all = sum(temats.values(), [])
X = np.array([w[v] for v in words_all])
pca = PCA(n_components=2).fit_transform(X)
colors = []
palette = {'zwierze':'tab:red','programowanie':'tab:blue','pojazd':'tab:green'}
kat_of = {}
for kat, sl in temats.items():
    for s in sl: kat_of[s]=kat
cols = [palette[kat_of[v]] for v in words_all]

plt.figure(figsize=(8,6))
plt.scatter(pca[:,0], pca[:,1], c=cols, s=130)
for i,v in enumerate(words_all):
    plt.annotate(v,(pca[i,0],pca[i,1]),xytext=(4,4),textcoords='offset points')
plt.title('PCA osadzeń Word2Vec (grupy semantyczne)'); plt.show()
```

3. Zadania do samodzielnego rozwiązania

Zadanie 7.1: Reprezentacja one-hot i jej ograniczenia

Napisz funkcję, która dla dowolnego słownika (co najmniej 20 słów) tworzy macierz one-hot i oblicza podobieństwo kosinusowe wszystkich par słów. Zaprezentuj wyniki jako macierz cieplną (matplotlib/seaborn heatmap). Udowodnij doświadczalnie, że wszystkie różne pary mają podobieństwo 0. W jaki sposób ta reprezentacja ogranicza możliwości modeli?

Zadanie 7.2: Własne osadzenia z macierzy PPMI+SVD

Na większym korpusie (min. 200–300 zdań — użyj tekstu z Wikipedii pobranego biblioteką wikipedia, rozdziały książki albo wygenerowanego korpusu) zbuduj macierz współwystępowań, oblicz PPMI, a następnie zredukuj wymiar SVD do 2 i narysuj wykres punktowy z etykietami słów. Oceń, czy grupy semantyczne (np. zwierzęta vs języki programowania) układają się w skupiska.

Zadanie 7.3: Trening i ewaluacja Word2Vec

Przygotuj korpus minimum 500 zdań (język angielski lub polski). Wytrenuj dwa modele word2vec: sg=0 (CBOW) oraz sg=1 (Skip-gram). Dla każdego modelu:

- wyświetl po 5 słów najbardziej podobnych do wybranych 5 słów kluczowych,
- rozwiąż 10 testów analogii (np. king−man+woman, Paris−France+Poland) i policz skuteczność,
- porównaj wektory osadzeń obu podejść w tej samej projekcji PCA.

Zadanie 7.4: Porównanie metody reprezentacji

Dla tego samego zestawu 10 par słów (pokrewnych, niezwiązanych, antonimów) oblicz podobieństwo kosinusowe w trzech reprezentacjach: one-hot, PPMI+SVD oraz Word2Vec. Zaprezentuj tabelę porównawczą i wykres. Która metoda najlepiej oddaje

relacje semantyczne, a która radzi sobie najgorzej i dlaczego? Sformułuj wnioski dotyczące wpływu doboru okna kontekstu i wymiaru.

4. Przydatne zasoby i biblioteki

NumPy	Operacje na wektorach i macierzach	pip install numpy
scikit-learn	Podobieństwo kosinusowe, PCA	pip install scikit-learn
Gensim	Trenowanie Word2Vec/FastText	pip install gensim
matplotlib / seaborn	Wykresy i macierze cieplne	pip install matplotlib seaborn
wikipedia	Pobieranie tekstów do korpusów	pip install wikipedia

5. Wymagania techniczne

- Środowisko: Python 3.10+, Jupyter Notebook/Lab (tak jak poprzednie laboratoria).
- Wszystkie przykłady działają w całości na CPU — nie jest potrzebny GPU.
- Gensim może wymagać kompilatora C przy instalacji — w systemie Windows użyj oficjalnych wheelów: pip install gensim.
- Korpusy do zadań można budować z dowolnych publicznych tekstów (Wikipedia, Project Gutenberg).