MATERI KULIAH: Semantic Similarity & Sentence Embedding

Pertemuan Lanjutan – Natural Language Processing

1. **Pengantar Semantic Similarity**

   Semantic similarity adalah teknik NLP untuk mengukur **seberapa mirip makna** antara dua teks (kata, kalimat, atau dokumen).

   Contoh:

   - ✓ "Saya lapar" ↔ "Saya ingin makan" → mirip secara makna
   - ✓ "Laptop saya rusak" ↔ "Komputer saya bermasalah" → mirip
   - ✓ "Pengiriman cepat" ↔ "Harganya murah" → tidak mirip

   Model semantic similarity tidak hanya melihat kesamaan kata, tetapi **kesamaan konteks makna**.

2. **Mengapa Semantic Similarity Penting?**

   Semantic similarity digunakan dalam banyak aplikasi:

   a. **Rekomendasi dokumen**
      Mencari dokumen yang mirip dengan input pengguna.
   b. **Duplicate detection**
      Mendeteksi komentar/ulasan yang duplikat.
   c. **Search Engine / Semantic Search**
      Pencarian berdasarkan makna, bukan keyword.
   d. **Chatbot & QnA**
      Menemukan jawaban mirip dari database.
   e. **Clustering**
      Mengelompokkan kalimat berdasarkan kesamaan semantik.

3. **Sentence Embedding**

   Sentence embedding adalah teknik untuk mengubah kalimat menjadi vektor numerik yang merepresentasikan makna semantik.

   Contoh:

   - ✓ "Saya suka makan bakso" → [0.12, -0.87, 0.33, ...]
   - ✓ "Bakso adalah makanan favorit saya" → vektor mirip

   Embedding populer:

   1) **TF-IDF Vector** (basis kata)
   2) **Word2Vec / FastText Averaging**
   3) **Doc2Vec**
   4) **Transformer-based (Sentence-BERT, IndoBERT)** → akurasi terbaik

4. **Pendekatan Semantic Similarity**

   a. **Metode Klasik – TF-IDF + Cosine Similarity**

      Kelebihan:

      - ✓ Mudah
      - ✓ Cepat

      Kekurangan:

      - ✓ Hanya melihat kata yang sama
      - ✓ Tidak memahami sinonim atau konteks

b. Word Embedding Averaging

Menggunakan rata-rata Word2Vec atau FastText.

Kelebihan:

- ✓ Menangkap sebagian makna semantik
- ✓ Cocok untuk Bahasa Indonesia (FastText Indo)

Kekurangan:

- ✓ Tidak menangkap makna antar kata dalam kalimat (context-insensitive)

c. Sentence-BERT (SBERT)

- State-of-the-art untuk semantic similarity.
- Menggunakan arsitektur Transformer (BERT) yang sudah dituning untuk membuat embedding kalimat.

  Model Indonesia:

  - ✓ **indobert-base-p1**
  - ✓ **sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2** (multilingual, sangat akurat)
  - ✓ **indonesia-bert-sentence-similarity (HuggingFace)**

5. Contoh Coding Python (TF-IDF → Cosine Similarity)

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

sentences = [
   "saya suka makan bakso",
   "bakso adalah makanan favorit saya",
   "pengiriman barang sangat cepat"
]

vectorizer = TfidfVectorizer()
vectors = vectorizer.fit_transform(sentences)

# hitung similarity antara kalimat 1 dan 2
sim = cosine_similarity(vectors[0], vectors[1])
print("Similarity TF-IDF:", sim[0][0])
```

6. Contoh Coding Python (Sentence Embedding – Sentence-BERT)

```
!pip install sentence-transformers

from sentence_transformers import SentenceTransformer, util

model = SentenceTransformer('sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2')

sent1 = "saya suka makan bakso"
sent2 = "bakso adalah makanan favorit saya"

emb1 = model.encode(sent1, convert_to_tensor=True)
emb2 = model.encode(sent2, convert_to_tensor=True)
```

```
similarity = util.cos_sim(emb1, emb2)
print("SBERT Similarity:", similarity.item())
```

Output contoh:

0.82 (mendekati 1 → mirip)

7. Contoh Aplikasi Nyata

a. Duplicate Review Detection

```
reviews = [
   "pengiriman cepat sekali",
   "produk datang sangat cepat",
   "harganya murah sekali",
   "pengiriman lambat"
]
emb = model.encode(reviews)
sim_matrix = util.cos_sim(emb, emb)
print(sim_matrix)
```

Mahasiswa bisa mengidentifikasi ulasan mana yang mirip.

b. Semantic Search

Pengguna menulis:

"laptop saya error"

Sistem mencari dokumen paling mirip:

```
query = "laptop saya error"

emb_query = model.encode(query)
sims = util.cos_sim(emb_query, emb)

# tampilkan dokumen paling mirip
most_similar = sims.argmax()
print("Paling mirip:", reviews[most_similar])
```

8. Visualisasi Embedding (TSNE)

Optional untuk kelas lanjutan.

```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

X = model.encode(reviews)
X_2d = TSNE(n_components=2).fit_transform(X)

plt.scatter(X_2d[:,0], X_2d[:,1])
for i, txt in enumerate(reviews):
   plt.annotate(txt, (X_2d[i,0], X_2d[i,1]))
plt.title("Sentence Embedding Visualization")
plt.show()
```

9. LATIHAN MAHASISWA

Latihan 1 — Teori

1. Jelaskan perbedaan TF-IDF vector vs Sentence Embedding.
2. Mengapa cosine similarity banyak digunakan untuk mengukur kesamaan?
3. Mengapa SBERT lebih unggul daripada TF-IDF dalam semantic similarity?

Latihan 2 — TF-IDF Similarity

Gunakan 5 kalimat berbeda.

Tugas:

1. Buat TF-IDF
2. Hitung similarity pairwise
3. Tuliskan 2 pasang kalimat yang paling mirip secara semantik
4. Jelaskan mengapa

Latihan 3 — Word2Vec / FastText Averaging

1. Gunakan pretrained FastText Indonesia
2. Buat embedding kalimat dengan rata-rata
3. Hitung cosine similarity
4. Bandingkan dengan TF-IDF

Latihan 4 — SBERT

1. Masukkan 6–10 kalimat
2. Hitung similarity matrix
3. Tampilkan kalimat paling mirip & paling tidak mirip
4. Buat visualisasi 2D (TSNE)
5. Jelaskan clustering yang muncul

10. MINI PROJECT (Tugas Kelompok maksimal 3 org)

Gunakan dataset **50–100 ulasan** dari marketplace.

Tugas:

1. Buat embedding SBERT
2. Hitung similarity matrix
3. Lakukan clustering (KMeans)
4. Visualisasikan hasil cluster
5. Interpretasi cluster (topik apa?)
6. Cari 5 pasang ulasan paling mirip → jelaskan
7. Cari 5 ulasan yang anomali (tidak mirip dengan cluster lain)

Output:

Laporan 3–5 halaman + coding python.

--o0o--