Lab 2 - Bangun Indeks & Cari

ukuran chunk 1000, overlap 15%, backend: lsa

Langkah 1. Membangun indeks

Tiga PDF diletakkan di folder dokumen/. Setiap file melaporkan jumlah chunk-nya:

File	Jumlah chunk
etika_keamanan_ai.pdf	4
machine_learning_dasar.pdf	4
pengantar_rag.pdf	5
Total	13

```
etika_keamanan_ai.pdf: 4 chunk
machine_learning_dasar.pdf: 4 chunk
pengantar_rag.pdf: 5 chunk
Total 13 chunk | ukuran=1000 overlap=15% | backend=lsa
```

Langkah 2. Lima pertanyaan dan pemeriksaan chunk teratas

Sebuah chunk dianggap relevan bila memuat semua kata kunci jawaban. Tabel menunjukkan 3 chunk teratas.

Siapa yang memperkenalkan RAG dan pada tahun berapa?

Peringkat	Skor	Sumber	Relevan?	Cuplikan
1	0.741	pengantar_rag.pdf #0	ya	Pengantar Retrieval-Augmented Generation (RAG) 1. Pengertian RAG Retrieval-Augmented Generation (RAG) adalah arsitektur yang menggabungkan p...
2	0.736	etika_keamanan_ai.pdf #3	tidak	gkasan penulis. 6. Tata Kelola dan Akuntabilitas Tata kelola yang baik mencakup audit berkala, keterlibatan manusia dalam pengambilan keputu...
3	0.590	etika_keamanan_ai.pdf #2	tidak	um adalah adversarial training, yaitu melatih model dengan contoh serangan. 5. Prompt Injection Prompt injection adalah serangan berupa inst...

Berapa dimensi embedding yang dihasilkan model MiniLM?

Peringkat	Skor	Sumber	Relevan?	Cuplikan
1	0.969	pengantar_rag.pdf #1	ya	tore. Pada tahap penjawaban, pertanyaan pengguna diubah menjadi vektor dengan model embedding yang sama. Sistem lalu mengambil k chunk yang ...
2	0.351	pengantar_rag.pdf #0	tidak	Pengantar Retrieval-Augmented Generation (RAG) 1. Pengertian RAG Retrieval-Augmented Generation (RAG) adalah arsitektur yang menggabungkan p...
3	0.308	pengantar_rag.pdf #3	tidak	dan presisi topik 1500 karakter Konteks luas Skor encer dan boros Dokumen naratif panjang token Sumber: ringkasan penulis. 5. Vector Store V...

Apa itu overfitting dan apa cirinya?

Peringkat	Skor	Sumber	Relevan?	Cuplikan
1	0.939	machine_learning_dasar.pdf #0	ya	Dasar-Dasar Machine Learning 1. Supervised Learning Pada supervised learning, model belajar dari data yang sudah berlabel. Tugas klasifikasi...
2	0.558	machine_learning_dasar.pdf #1	ya	i validasi, misalnya 72 persen. Model seolah menghafal data latih, termasuk derau di dalamnya. Underfitting terjadi ketika model terlalu sed...
3	0.392	machine_learning_dasar.pdf #3	tidak	erlalu besar membuat pelatihan divergen, sedangkan yang terlalu kecil membuat pelatihan sangat lambat. Pada mini-batch gradient descent, gra...

Undang-undang apa yang mengatur pelindungan data pribadi di Indonesia?

Peringkat	Skor	Sumber	Relevan?	Cuplikan
1	0.987	etika_keamanan_ai.pdf #0	ya	Etika dan Keamanan Kecerdasan Buatan 1. Bias pada Data Model kecerdasan buatan mewarisi bias yang ada pada data latihnya. Contoh yang sering...
2	0.199	etika_keamanan_ai.pdf #2	tidak	um adalah adversarial training, yaitu melatih model dengan contoh serangan. 5. Prompt Injection Prompt injection adalah serangan berupa inst...
3	0.170	etika_keamanan_ai.pdf #1	tidak	titas secara permanen, sedangkan pseudonimisasi hanya mengganti identitas dengan kode yang masih dapat ditelusuri kembali. Keduanya menguran...

Bagaimana cara memitigasi prompt injection?

Peringkat	Skor	Sumber	Relevan?	Cuplikan
1	0.925	etika_keamanan_ai.pdf #1	tidak	titas secara permanen, sedangkan pseudonimisasi hanya mengganti identitas dengan kode yang masih dapat ditelusuri kembali. Keduanya menguran...
2	0.901	etika_keamanan_ai.pdf #2	ya	um adalah adversarial training, yaitu melatih model dengan contoh serangan. 5. Prompt Injection Prompt injection adalah serangan berupa inst...
3	0.291	etika_keamanan_ai.pdf #0	tidak	Etika dan Keamanan Kecerdasan Buatan 1. Bias pada Data Model kecerdasan buatan mewarisi bias yang ada pada data latihnya. Contoh yang sering...

Ringkasan: chunk peringkat 1 relevan pada 4/5 pertanyaan; ada chunk relevan di 3 teratas pada 5/5.

Langkah 3. Pertanyaan berparafrase

Dua pertanyaan memakai kata yang berbeda dari teks dokumen. Dibandingkan: pencarian semantik (lsa) vs TF-IDF.

Pertanyaan	Catatan	lsa: top1 / top3	TF-IDF: top1 / top3
Bagaimana mencegah model bahasa mengarang jawaban yang keliru?	Teks memakai kata 'halusinasi'; pertanyaan memakai 'mengarang jawaban yang keliru'.	ya / ya	ya / ya
Kapan sebuah model dianggap terlalu menghafal data latihan?	Teks memakai 'overfitting' dan 'data latih'; pertanyaan memakai 'menghafal' dan 'data latihan'.	ya / ya	ya / ya

Chunk teratas untuk "Bagaimana mencegah model bahasa mengarang jawaban yang keliru?": pengantar_rag.pdf #4 (skor 0.816)

```
i adalah MRR dan Recall@k. Disarankan menyiapkan minimal 15 sampai 30 pertanyaan uji lengkap dengan kata
    kunci jawaban. Bandingkan beberapa pengaturan chunk pada pertanyaan yang sama agar perbedaannya adil.
     7. Halusinasi dan Sitasi Halusinasi adalah keluaran model yang terdengar meyakinkan tetapi tidak
    didukung fakta. RAG mengurangi halusinasi, tetapi tidak menghilangkannya sepenuhnya, terutama bi...
```

Chunk teratas untuk "Kapan sebuah model dianggap terlalu menghafal data latihan?": machine_learning_dasar.pdf #1 (skor 0.663)

```
i validasi, misalnya 72 persen. Model seolah menghafal data latih, termasuk derau di dalamnya.
    Underfitting terjadi ketika model terlalu sederhana sehingga akurasi latih maupun validasi sama-sama
    rendah. Cara mengatasi overfitting antara lain regularisasi, early stopping, dropout pada jaringan
    saraf, dan menambah jumlah data. 4. Cross-Validation Cross-validation menilai model secara lebih
    stabil d...
```

Kesimpulan

Pada 5 pertanyaan langsung, chunk peringkat 1 relevan pada 4/5 pertanyaan dan salah satu dari 3 chunk teratas relevan pada 5/5 pertanyaan. Pada 2 pertanyaan parafrase, backend 'lsa' menemukan chunk relevan di peringkat 1 pada 2/2 dan di 3 teratas pada 2/2, sedangkan pembanding TF-IDF (kecocokan bentuk kata) 2/2 dan 2/2. Pertanyaan yang chunk peringkat 1-nya belum relevan: "Bagaimana cara memitigasi prompt injection?". Penyebab yang paling mungkin: kalimat jawaban terbelah di batas chunk sehingga kata kuncinya terpisah, sementara chunk di sebelahnya memuat topik yang mirip. Pada dua parafrase ini, semantik dan TF-IDF memberi hasil yang sama, jadi tes ini belum bisa membedakan keduanya: pertanyaan masih berbagi akar kata dengan teks (model, jawaban, data latih) dan TF-IDF n-gram karakter menangkap variasi bentuk kata. Catatan: backend LSA hanya semantik ringan (hubungan antar kata dipelajari dari korpus yang kecil), sehingga parafrase yang benar-benar tidak berbagi kata akan sulit. Untuk hasil lebih kuat, pasang sentence-transformers; kode ini otomatis memakainya.

Lampiran: kode lab2_indeks_cari.py

```
"""Lab 2 - Bangun indeks & cari (25 menit).

1. Letakkan 2-3 PDF di folder dokumen/, jalankan indeks, pastikan tiap file melaporkan jumlah chunk.
2. Ajukan 5 pertanyaan dengan query(); periksa apakah chunk teratas benar-benar relevan.
3. Ajukan 2 pertanyaan berparafrase; apakah pencarian semantik tetap menemukannya?
"""
import io
import os
import sys
from contextlib import redirect_stdout

sys.path.insert(0, os.path.dirname(os.path.abspath(__file__)))
from laporan_utils import Laporan
from rag_utils import bangun_indeks, cocok, query

BASE = os.path.dirname(os.path.abspath(__file__))
UKURAN, OVERLAP = 1000, 0.15

# kw = kata kunci yang HARUS ada di chunk teratas agar dianggap relevan
PERTANYAAN = [
    {"q": "Siapa yang memperkenalkan RAG dan pada tahun berapa?", "kw": ["Lewis", "2020"]},
    {"q": "Berapa dimensi embedding yang dihasilkan model MiniLM?", "kw": ["384"]},
    {"q": "Apa itu overfitting dan apa cirinya?", "kw": ["overfitting", "data latih"]},
    {"q": "Undang-undang apa yang mengatur pelindungan data pribadi di Indonesia?", "kw": ["27 Tahun
    2022"]},
    {"q": "Bagaimana cara memitigasi prompt injection?", "kw": ["memisahkan instruksi"]},
]
PARAFRASE = [
    {"q": "Bagaimana mencegah model bahasa mengarang jawaban yang keliru?", "kw": ["halusinasi"],
     "catatan": "Teks memakai kata 'halusinasi'; pertanyaan memakai 'mengarang jawaban yang keliru'."},
    {"q": "Kapan sebuah model dianggap terlalu menghafal data latihan?", "kw": ["overfitting"],
     "catatan": "Teks memakai 'overfitting' dan 'data latih'; pertanyaan memakai 'menghafal' dan 'data
    latihan'."},
]


def ringkas(t, n=150):
    t = " ".join(t.split())
    return t if len(t) <= n else t[:n] + "..."


def jalankan(indeks, daftar, k=3):
    hasil = []
    for s in daftar:
        atas = query(indeks, s["q"], k)
        rel = [cocok(a["teks"], s["kw"]) for a in atas]
        hasil.append((s, atas, rel))
    return hasil


def main():
    # ---- Langkah 1
    print("=== Langkah 1: membangun indeks ===")
    buf = io.StringIO()
    with redirect_stdout(buf):
        idx = bangun_indeks(ukuran=UKURAN, overlap=OVERLAP, backend="auto")
    log_indeks = buf.getvalue()
    print(log_indeks)
    idx_leksikal = bangun_indeks(ukuran=UKURAN, overlap=OVERLAP, backend="tfidf", verbose=False)
    per_file = {}
    for m in idx.meta:
        per_file[m["file"]] = per_file.get(m["file"], 0) + 1

    # ---- Langkah 2
    print("=== Langkah 2: 5 pertanyaan ===")
    h2 = jalankan(idx, PERTANYAAN)
    for s, atas, rel in h2:
        print(f"\nQ: {s['q']}  (kata kunci: {s['kw']})")
        for r, (a, ok) in enumerate(zip(atas, rel), 1):
            print(f"  #{r} skor={a['skor']:.3f} {a['file']}#{a['no']} relevan={ok} |
    {ringkas(a['teks'])}")
    n_top1 = sum(rel[0] for _, _, rel in h2)
    n_top3 = sum(any(rel) for _, _, rel in h2)

    # ---- Langkah 3
    print("\n=== Langkah 3: parafrase ===")
    sem = jalankan(idx, PARAFRASE)
    lek = jalankan(idx_leksikal, PARAFRASE)
    for (s, a1, r1), (_, a2, r2) in zip(sem, lek):
        print(f"\nQ: {s['q']}")
```

```
        print(f"  {idx.backend}: top1 relevan={r1[0]} top3 relevan={any(r1)} |
    {a1[0]['file']}#{a1[0]['no']}")
        print(f"  tfidf : top1 relevan={r2[0]} top3 relevan={any(r2)} | {a2[0]['file']}#{a2[0]['no']}")

    # ---- Kesimpulan dari angka
    sem_top3 = sum(any(r) for _, _, r in sem)
    lek_top3 = sum(any(r) for _, _, r in lek)
    sem_top1 = sum(r[0] for _, _, r in sem)
    lek_top1 = sum(r[0] for _, _, r in lek)
    kes = (f"Pada 5 pertanyaan langsung, chunk peringkat 1 relevan pada {n_top1}/5 pertanyaan dan salah
    satu dari "
           f"3 chunk teratas relevan pada {n_top3}/5 pertanyaan. Pada 2 pertanyaan parafrase, backend "
           f"'{idx.backend}' menemukan chunk relevan di peringkat 1 pada {sem_top1}/2 dan di 3 teratas
    pada "
           f"{sem_top3}/2, sedangkan pembanding TF-IDF (kecocokan bentuk kata) {lek_top1}/2 dan
    {lek_top3}/2. ")
    gagal_top1 = [s["q"] for s, atas, rel in h2 if not rel[0]]
    if gagal_top1:
        kes += ("Pertanyaan yang chunk peringkat 1-nya belum relevan: " + "; ".join(f'"{q}"' for q in
    gagal_top1) +
                ". Penyebab yang paling mungkin: kalimat jawaban terbelah di batas chunk sehingga kata
    kuncinya "
                "terpisah, sementara chunk di sebelahnya memuat topik yang mirip. ")
    if sem_top1 == lek_top1 and sem_top3 == lek_top3:
        kes += ("Pada dua parafrase ini, semantik dan TF-IDF memberi hasil yang sama, jadi tes ini belum
    bisa "
                "membedakan keduanya: pertanyaan masih berbagi akar kata dengan teks (model, jawaban,
    data latih) "
                "dan TF-IDF n-gram karakter menangkap variasi bentuk kata. ")
    if idx.backend == "lsa":
        kes += ("Catatan: backend LSA hanya semantik ringan (hubungan antar kata dipelajari dari korpus
    yang kecil), "
                "sehingga parafrase yang benar-benar tidak berbagi kata akan sulit. Untuk hasil lebih
    kuat, pasang "
                "sentence-transformers; kode ini otomatis memakainya.")
    print("\nKesimpulan:", kes)

    # ---- Laporan PDF
    L = Laporan(os.path.join(BASE, "Lab2_Indeks_dan_Pencarian.pdf"), "Lab 2 - Bangun Indeks & Cari",
                f"ukuran chunk {UKURAN}, overlap {int(OVERLAP * 100)}%, backend: {idx.backend}")
    L.h1("Langkah 1. Membangun indeks")
    L.p("Tiga PDF diletakkan di folder dokumen/. Setiap file melaporkan jumlah chunk-nya:")
    L.tabel([["File", "Jumlah chunk"]] + [[f, n] for f, n in per_file.items()] + [["Total",
    len(idx.chunks)]],
            [9 * 28.35, 4 * 28.35])
    L.kode(log_indeks.rstrip())
    L.h1("Langkah 2. Lima pertanyaan dan pemeriksaan chunk teratas")
    L.p("Sebuah chunk dianggap relevan bila memuat semua kata kunci jawaban. Tabel menunjukkan 3 chunk
    teratas.")
    for s, atas, rel in h2:
        L.h2(s["q"])
        L.tabel([["Peringkat", "Skor", "Sumber", "Relevan?", "Cuplikan"]] +
                [[r, f"{a['skor']:.3f}", f"{a['file']} #{a['no']}", "ya" if ok else "tidak",
    ringkas(a["teks"], 140)]
                 for r, (a, ok) in enumerate(zip(atas, rel), 1)],
                [1.6 * 28.35, 1.3 * 28.35, 3.8 * 28.35, 1.6 * 28.35, 8.7 * 28.35])
    L.p(f"Ringkasan: chunk peringkat 1 relevan pada {n_top1}/5 pertanyaan; ada chunk relevan di 3 teratas
     pada {n_top3}/5.")
    L.h1("Langkah 3. Pertanyaan berparafrase")
    L.p("Dua pertanyaan memakai kata yang berbeda dari teks dokumen. Dibandingkan: pencarian semantik "
        f"({idx.backend}) vs TF-IDF.")
    baris = [["Pertanyaan", "Catatan", f"{idx.backend}: top1 / top3", "TF-IDF: top1 / top3"]]
    for (s, a1, r1), (_, a2, r2) in zip(sem, lek):
        baris.append([s["q"], s["catatan"], f"{'ya' if r1[0] else 'tidak'} / {'ya' if any(r1) else
    'tidak'}",
                      f"{'ya' if r2[0] else 'tidak'} / {'ya' if any(r2) else 'tidak'}"])
    L.tabel(baris, [5 * 28.35, 6 * 28.35, 3 * 28.35, 3 * 28.35])
    for (s, a1, r1), _ in zip(sem, lek):
        L.p(f"Chunk teratas untuk \"{s['q']}\": {a1[0]['file']} #{a1[0]['no']} (skor
    {a1[0]['skor']:.3f})")
        L.kode(ringkas(a1[0]["teks"], 400))
    L.h1("Kesimpulan")
    L.p(kes)
    L.halaman_baru()
    L.h1("Lampiran: kode lab2_indeks_cari.py")
    L.kode(open(__file__, encoding="utf-8").read())
    L.simpan()
    print("\nLaporan: Lab2_Indeks_dan_Pencarian.pdf")


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