BAB IV PEMBAHASAN

Penelitian ini mengimplementasikan *Hybrid* NER dari data jadwal dosen Fakultas Ilmu Komputer (Fasilkom) semester genap 2025/2026 yang diberikan oleh Tata Usaha Fasilkom dan profil dosen dan tenaga kependidikan Fasilkom yang didapatkan dengan cara *scraping* dari website profil Mercu Buana. *Hybrid* NER ini digunakan untuk mengidentifikasi entitas, berupa dosen dan tenaga kependidikan fakultas Fasilkom Universitas Mercu Buana. Lalu penelitian ini juga menerapkan algoritma A* yang dimana didasarkan dari denah Universitas Mercu Buana Meruya yang didapatkan dari gambar denah yang terletak di Gedung Rektorat. Penerapan algoritma A* pada sistem asisten virtual kampus ini berfungsi sebagai navigasi untuk membantu mahasiswa Fasilkom maupun pengunjung yang tidak familiar dengan lingkungan Universitas Mercu Buana Meruya.

4.1 Tampilan Antarmuka Sistem Asisten Virtual Kampus

Berikut ialah tampilan utama asisten virtual kampus yang dijalankan di web browser :

Gambar 4.1 Tampilan utama asisten virtual kampus

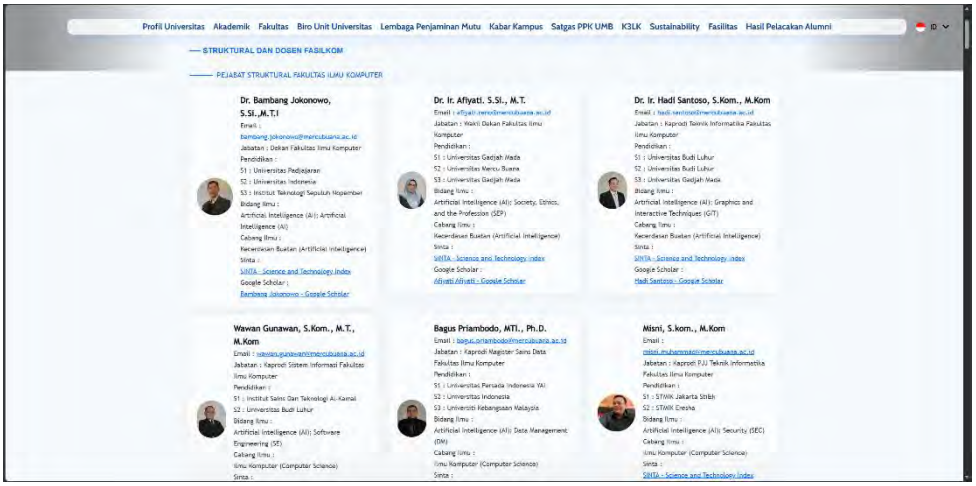
Pada gambar 4.1 menampilkan antarmuka asisten virtual kampus dengan gambar latar belakangnya Universitas Mercu Buana Meruya, lalu ada karakter Live2D yang dibuat menggunakan aplikasi Live2D cubism editor, nama karakter tersebut ialah Sakiko sebagai asisten virtual kampus yang ditugaskan untuk membantu mahasiswa Fasilkom maupun pengunjung yang tidak familiar dengan lingkungan Universitas Mercu Buana Meruya. Dari gambar di atas, ada juga button mikrofon yang digunakan untuk merekam suara. Cara kerjanya, pengguna cukup tekan button mikrofon sekali hingga *button* mikrofon berubah menjadi warna merah. Lalu pengguna mulai berbicara atau menanyakan hal-hal yang ingin ditanyakan. Setelah selesai, pengguna menekan button mikrofon merah tadi sekali lagi, maka *button* mikrofon akan kembali seperti semula dan Sakiko akan menjawab pertanyaan pengguna.

4.2 Implementasi *Hybrid* NER

Pada asisten virtual kampus ini mengimplementasikan *Named Entity Recognition* (NER) yang dirancang untuk mengenali entitas dosen dan tenaga kependidikan Fakultas Ilmu Komputer Universitas Mercu Buana Meruya. Sistem ini dibangun menggunakan pendekatan hybrid yang menggabungkan dua jalur pemrosesan, yaitu : *Fast Lane* berbasis *fuzzy string matching* dan *Slow Lane* berbasis pencarian semantik menggunakan ChromaDB. Arsitektur hybrid ini dirancang untuk memaksimalkan akurasi pengenalan entitas sekaligus menjaga kecepatan respons sistem agar tetap responsif dalam interaksi percakapan *real-time*.

4.2.1 Knowledge Base

NER menggunakan basis pengetahuan (*knowledge base*). Untuk basis pengetahuannya ini peneliti menggunakan data profil dosen dan tenaga kependidikan Fasilkom yang diambil dari website Universitas Mercu Buana.



Gambar 4.2 Halaman profil dosen Fasilkom

Dari gambar 4.2 memperlihatkan data profil dosen Fasilkom yang terdiri dari nama lengkap dosen, jabatan, pendidikan, dan bidang ilmu. Dari halaman profil tersebut, peneliti melakukan *scraping* untuk diambil datanya, lalu diubah menjadi bentuk JSON. Basis pengetahuan ini terdiri dari dua file utama yang dimuat dan digabungkan (*merged*) pada saat inisialisasi sistem.

```
{
  "nama": "Muhaimin Hasanudin, S.T, M.Kom",
  "kategori": "Dosen Teknik Informatika",
  "jabatan": "Dosen Tetap Fakultas Ilmu Komputer",
  "s1": "Sekolah Tinggi Sains Dan Teknologi Indonesia",
  "s2": "Universitas Budi Luhur",
  "s3": null,
  "bidang_ilmu":"Artificial Intelligence (AI)",
  "alias": [
    "Muhaimin Hasanudin",
    "Muhaimin",
    "Hasanudin"
  ]
},
```

Gambar 4.3 Format JSON untuk dosen_tendik_fasilkom

Dari gambar 4.3 merupakan tampilan format JSON untuk dosen dan tenaga kependidikan. Tiap nama dosen dan tenaga kependidikan diberikan nama alias, karena tidak semua pengguna tahu nama lengkap dosen atau tenaga kependidikan tersebut. Selain itu, asisten virtual kampus ini juga menggunakan basis pengetahuan dari jadwal mengajar dosen Fasilkom Meruya.

```
{
  "nama": "Andi Nugroho, Dr. S.T., M.Kom.",
  "id_dsn": "11631",
  "alias": [
    "andi",
    "andi nugroho",
    "nugroho"
  ],
  "jadwal": [
    {
      "hari": "Senin",
      "jam_mulai": "07:30",
      "jam_selesai": "10:00",
      "mata_kuliah": "Calculus II",
      "kode_mk": "W152501021",
      "sks": 3,
      "kelas": "1A2151IP",
      "ruang": "I-501",
      "jenis": "luring",
      "gedung": "Gedung T - Kelas Internasional",
      "nav_node_id": "loc_T",
      "jumlah_mhs": 16
    },
    {
      "hari": "Kamis",
      "jam_mulai": "07:30",
      "jam_selesai": "10:00",
      "mata_kuliah": "Network Programming",
      "kode_mk": "W152501025",
      "sks": 3,
      "kelas": "1A4154IP",
      "ruang": "I-402",
      "jenis": "luring",
      "gedung": "Gedung T - Kelas Internasional",
      "nav_node_id": "loc_T",
      "jumlah_mhs": 15
    }
  ],
  "mengajar_di_kampus_lain": true
},
```


Gambar 4.4 Format JSON untuk jadwal_dosen_fasilkom

Dari gambar 4.4 merupakan tampilan format JSON untuk jadwal mengajar dosen di kampus Meruya. Isinya mulai dari hari ngajar, jam mulai, jam selesai, mata kuliah, dan ruang kelas. Data ini digunakan untuk membantu menjawab pertanyaan terkait jadwal mengajar dosen.

4.2.2 Jalur Cepat (*Fast Lane*)

Fast Lane bertugas mengenali entitas bernama dari teks masukan pengguna. Dalam prosesnya ini bekerja dengan mencocokkan potongan-potongan kata dari teks terhadap basis data dengan menggunakan teknik *n-gram generation* dan *fuzzy string matching*. Proses *fast lane* ada empat tahap.

Pada tahap pertama, ialah *n-gram generation* yang dimana setelah teks melalui tahap preprocessing, teks tersebut dipecah menjadi seluruh

kemungkinan kombinasi kata berurutan mulai dari satu kata (*1-gram*) sampai lima kata (*5-gram*). Contoh teks dari suara pengguna "Apa bidang ilmu Pak Muhaimin?" yang terdiri dari lima kata, sistem asisten virtual kampus menghasilkan *n-gram* sebegai berikut :

Tabel 4.1 Tabel contoh *n-gram*

n	Jumlah	Contoh *n-gram*
1	5	"apa", "bidang", "ilmu", "pak", "muhaimin"
2	4	"apa bidang", "bidang ilmu", "ilmu pak", "pak muhaimin"
3	3	"apa bidang ilmu", "bidang ilmu pak", "ilmu pak muhaimin"
4	2	"apa bidang ilmu pak", "bidang ilmu pak muhaimin"
5	1	"apa bidang ilmu pak muhaimin"

Pada tabel 4.1 ialah hasil dari seluruh *n-gram* yang kemudian diurutkan dari yang terpanjang ke yang terpendek (*descending*). Hasilnya seperti pada tabel 4.2 dibawah ini :

Tabel 4.2 Pengurutan kata dari yang terpanjang ke yang terpendek

Urutan Coba	*N-gram*	Panjang Karakter
1	"apa bidang ilmu pak muhaimin"	28
2	"bidang ilmu pak muhaimin"	24
3	"apa bidang ilmu pak"	19
4	"ilmu pak muhaimin"	17
5	"bidang ilmu pak"	15
6	"apa bidang ilmu"	15
7	"pak muhaimin"	12
8	"bidang ilmu"	11
9	"apa bidang"	10
10	"ilmu pak"	8
11	"muhaimin"	8
12	"bidang"	6

13	“ilmu”	4
14	“pak”	3
15	“apa”	3

```
STOPWORDS = {
    "di", "ke", "dari", "dan", "atau", "yang", "ini", "itu", "ada", "apa",
    "aku", "saya", "kamu", "dia", "kami", "mereka", "juga", "bisa", "mau",
    "untuk", "dengan", "pada", "tidak", "sudah", "akan", "harus", "baru",
    "lagi", "saja", "biar", "agar", "jika", "bila", "bukan", "belum",
    "masih", "dulu", "nanti", "sini", "sana", "situ", "mana", "kapan",
    "siapa", "kenapa", "mengapa", "dimana", "kemana", "bagaimana", "gimana",
    "apakah", "berapa", "tolong", "mohon", "bisa", "boleh", "dosen",
    "kampus", "kuliah", "mahasiswa", "prodi", "jurusan", "fakultas", "biro",
    "pak", "bu", "ibu", "bapak", "mas", "mba", "mbak", "kak",
    "ya", "dong", "sih", "lah", "kok", "kan", "deh", "nih",
    "oke", "okay", "ok", "baik", "terima", "kasih", "halo", "hai",
    "punya", "pernah", "sama", "tentang", "soal", "tanya", "jawab",
    "gak", "ngga", "enggak", "nggak", "ga", "gk", "udah", "udh",
    "matkul", "mata", "kelas", "nilai", "jadwal", "ujian", "tugas",
    "gedung", "ruang", "ruangan", "lantai", "jam", "hari", "waktu",
    "informasi", "info", "data", "nomor", "alamat", "kontak", "email",
    "biaya", "bayar", "uang", "daftar", "masuk", "lulus", "wisuda",
    "kerja", "magang", "skripsi", "tesis", "disertasi", "penelitian",
    "sekarang", "sedang", "saat",
}

```

Gambar 4.5 Kata-kata yang masuk kedalam *stopword*

Pada tahap kedua, ialah *filter stopword* dan *fuzzy string matching*. Dalam proses *stopword* dilakukan pembuangan kata-kata umum yang tidak merujuk pada entitas, contoh kata-kata tersebut dapat dilihat pada gambar 4.5 yang terdiri dari “dimana”, “siapa”, “apa” dan lain-lain. Contohnya, pada *n-gram* “pak muhaimin”, kata “pak” termasuk *stopword* dan kata “muhaimin” tidak termasuk *stopword*. Oleh karena itu “pak muhaimin” lolos *filter stopword*.

Pada tahap ketiga, ialah *fuzzy string matching*. Setiap *n-gram* yang lolos *filter stopword*, dilakukan pencocokan seluruh alias dengan *knowledge base* dengan menggunakan *fuzzy string matching*. Implementasi ini menggunakan *library RapidFuzz* dengan *fuzz.ratio* yang menghitung skor kemiripan dalam rentang 0 (tidak cocok) sampai 100 (cocok). Sistem menerapkan ambang batas (*threshold*) dinamis yang disesuaikan dengan panjang karakter *n-gram*, sebagaimana ditunjukkan pada tabel 4.3

Tabel 4.3 Tresshold berdasarkan panjang *n-gram*

Panjang Teks	Threshold Minimum	Alasan
≤ 3 karakter	100%	Teks sangat pendek rawan false positive
4-5 karakter	95%	Hampir exact match
6-8 karakter	88%	Toleransi typo minimal
> 8 karakter	80%	Threshold standar untuk nama lengkap

Ambang batas (*tresshold*) ini dirancang agar sistem dapat menoleransi kesalahan transkripsi (*typo*) dari *Speech-to-Text* pada nama-nama panjang, namun tetap ketat pada kata-kata pendek untuk menghindari kesalahan pencocokan. Salah satu contohnya, pada *n-gram* ”pak muhaimin” yang terdiri dari 12 karakter dicocokan dengan alias ”muhaimin” dari *knowledge base* yang terdiri dari 8 karakter dengan menggunakan rumus :

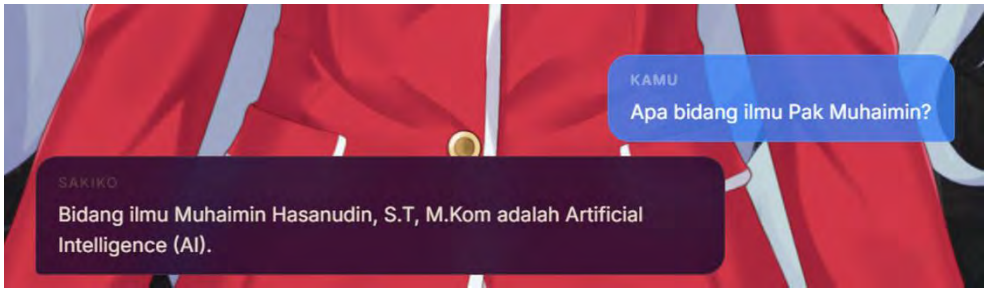
$$Ratio = \frac{(2 \times 8)}{(12 + 8)} \times 100 = 80\%$$

Dari hasil diatas memiliki kecocokan 80%, maka *n-gram* ”pak muhaimin” cocok dengan alias ”muhaimin”.

Pada tahap keempat, ialah *overlap prevention* dimana suatu *n-gram* yang berhasil dicocokkan dengan entitas, rentang posisi karakter yang tercakup oleh *n-gram* tersebut dicatat sebagai area yang telah digunakan. Setiap *n-gram* berikutnya yang posisi karakternya tumpang tindih dengan area yang telah digunakan akan otomatis dilewati. Mekanisme ini memastikan bahwa satu kata dalam teks hanya dapat menjadi bagian dari satu entitas yang terdeteksi.

Pada tahap kelima, ialah hasil pengenalan entitas dimana setelah n-*gram* ”pak muhaimin” cocok dengan alias ”muhaimin”. Dapat dilihat pada

gambar 4.6, maka asisten virtual kampus memberikan jawaban kepada pengguna sesuai dengan data yang diminta.



Gambar 4.6 Contoh pertanyaan yang masuk kedalam proses *fast lane*

4.2.3 Jalur Lambat (*Slow Lane*)

Proses *slow lane* dilakukan ketika *fast lane* tidak menemukan kecocokan entitas apapun pada *knowledge base*. Jadi, semua data JSON yang berisi profil dosen dan tenaga kependidikan, dan jadwal mengajar dosen disimpan ke dalam ChromaDB. Seluruh data tersebut diubah menjadi vektor numerik. Vektor ini mempresentasikan makna semantik dari data-data yang dikumpulkan tersebut. Teks dari pertanyaan pengguna yang sebelumnya gagal diproses *fast lane*, sekarang teks tersebut diubah menjadi vektor. Lalu pertanyaan pengguna yang sudah berubah menjadi vektor dibandingkan dengan data yang tersimpan di ChromaDB.

Gambar 4.7 Contoh pertanyaan yang masuk kedalam *slow lane*

Pada gambar 4.7 misalnya pertanyaan pengguna, “dosen yang mengajar science data siapa”, maka ChromaDB akan membandingkan vektor pertanyaan pengguna dengan vektor seluruh entitas. Entitas dosen yang mengandung informasi “science data” akan memiliki kemiripan yang

tinggi. Setelah itu asisten virtual kampus menjawab pertanyaan pengguna sesuai dengan konteksnya.

4.2.4 Klasifikasi *Intent*

Sebelum asisten virtual kampus memberikan jawaban kepada pengguna. Teks yang melalui proses *fast lane* maupun *slow lane* akan dilakukan klasifikasi *intent* dahulu untuk mengetahui konteks dari pertanyaan pengguna, agar asisten virtual dapat memberikan jawaban yang sesuai dengan konteks. Dalam proses *fast lane*, setelah menerima data entitas, sistem asisten virtual kampus akan memindai teks keseluruhan pengguna untuk menemukan kata kunci yang terkait dengan kategori informasi.

```
FACTUAL_PATTERNS = {
    "jabatan": ["jabatan", "posisi", "menjabat", "pegang jabatan", "jadi apa"],
    "bidang_ilmu": [
        "bidang ilmu", "bidang keahlian", "keahlian", "spesialisasi",
        "fokus penelitian", "riset", "penelitian", "bidang",
    ],
    "s1": ["s1", "sarjana", "s1 dari", "lulusan s1", "s1 nya"],
    "s2": ["s2", "magister", "s2 dari", "lulusan s2", "s2 nya", "master"],
    "s3": ["s3", "doktor", "s3 dari", "lulusan s3", "s3 nya", "doktoral"],
    "gelar": ["gelar", "gelar akademik", "titel"],
    "siapa": ["siapa", "siapa itu", "siapa yang"],
}
```

Gambar 4.8 *Factual patterns*

Misalnya, pada contoh *fast lane* sebelumnya pada kalimat pengguna “apa bidang ilmu pak muhaimin”, dapat dilihat pada gambar 4.8 sistem asisten virtual kampus memindai teks tersebut, dan ketemu kata kunci “bidang ilmu”, dan kata tersebut masuk kedalam kategori informasi ”bidang_ilmu”. Setelah itu sistem asisten virtual kampus memberikan jawaban kepada pengguna sesuai konteks dengan menjawab “Bidang ilmu Muhaimin Hasanudin, S.T, M.Kom adalah Artificial Intelligence (AI).”. Tidak hanya kategori informasi bidang ilmu saja, tapi ada juga jabatan, s1, s2, s3, gelar, dan siapa. Kategori informasi “siapa” ini untuk mengatasi pertanyaan terkait profil lengkap dosen atau tenaga pendidik.

Tabel 4.4 Kategori *Intent*

Intent	Pemicu	Jalur
ambiguous	Entitas ambigu (lebih dari satu kandidat)	*Fast Lane*
schedule	*Keyword* jadwal + entitas dosen/tendik	*Fast Lane*
factual	*Keyword* factual + entitas apapun *keyword* lokasi	*Fast Lane*
schedule_location	*Keyword* lokasi + entitas dosen/tendik	*Fast Lane*
complex	Entitas ditemukan tapi tidak ada pola yang cocok	*Slow Lane*

Pada tabel 4.4 tidak hanya factual intent saja, namun ada berbagai macam intent untuk mengatasi berbagai macam konteks pertanyaan dari pengguna. Kelebihan utama dari klasifikasi *intent* ini adalah kemampuannya untuk menghasilkan jawaban secara langsung tanpa LLM. Jadi jawaban yang diberikan pengguna merupakan jawaban *template*.

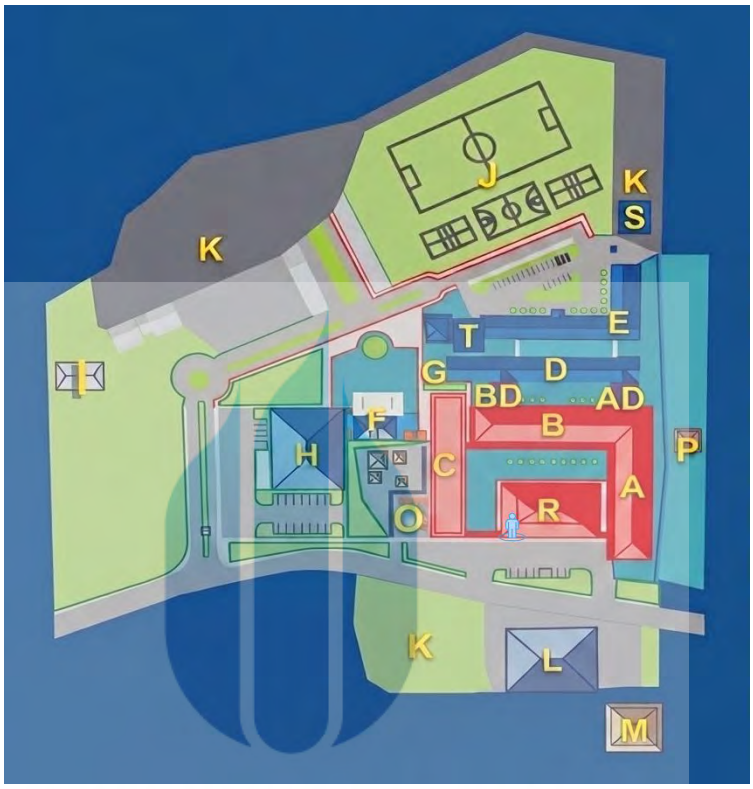
```python
# ── DOSEN / TENDIK templates ──────────────────────────────
if label in ("DOSEN", "TENDIK"):
    if intent_key == "jabatan":
        jabatan = data.get("jabatan", None)
        if jabatan:
            return f"{value} menjabat sebagai {jabatan}."

    elif intent_key == "siapa":
        jabatan = data.get("jabatan", "")
        kategori = data.get("kategori", "")
        parts = [f"{value}"]
        if jabatan:
            parts.append(f"menjabat sebagai {jabatan}")
        if kategori:
            parts.append(f"dengan kategori {kategori}")
        return " ".join(parts) + "."

    elif intent_key == "bidang_ilmu":
        bidang = data.get("bidang_ilmu", None)
        if bidang:
            return f"Bidang ilmu {value} adalah {bidang}."
        return f"Maaf, informasi bidang ilmu {value} belum tersedia."

    elif intent_key == "s1":
        s1 = data.get("s1", None)
        if s1:
            return f"{value} menempuh pendidikan S1 di {s1}."
        return f"Maaf, informasi S1 {value} belum tersedia."

    elif intent_key == "s2":
        s2 = data.get("s2", None)
        if s2:
            return f"{value} menempuh pendidikan S2 di {s2}."
        return f"Maaf, informasi S2 {value} belum tersedia."

    elif intent_key == "s3":
        s3 = data.get("s3", None)
        if s3:
            return f"{value} menempuh pendidikan S3 di {s3}."
        return f"Maaf, informasi S3 {value} belum tersedia."

    elif intent_key == "gelar":
        return f"Nama lengkap dengan gelar: {value}."
```

Gambar 4.9 Denah Universitas Mercu Buana Meruya

Pada gambar 4.9 ialah salah satu contoh jawaban *template* jika pertanyaan pengguna terkait dengan jabatan, gelar, riwayat pendidikan, dan lainnya. Jadi, data diambil dari *knowledge base* langsung dimasukkan ke *value* jawaban *template* tersebut, dan jawaban tersebut yang diberikan kepada pengguna.

4.3 Implementasi Algoritma A* Untuk Navigasi Kampus

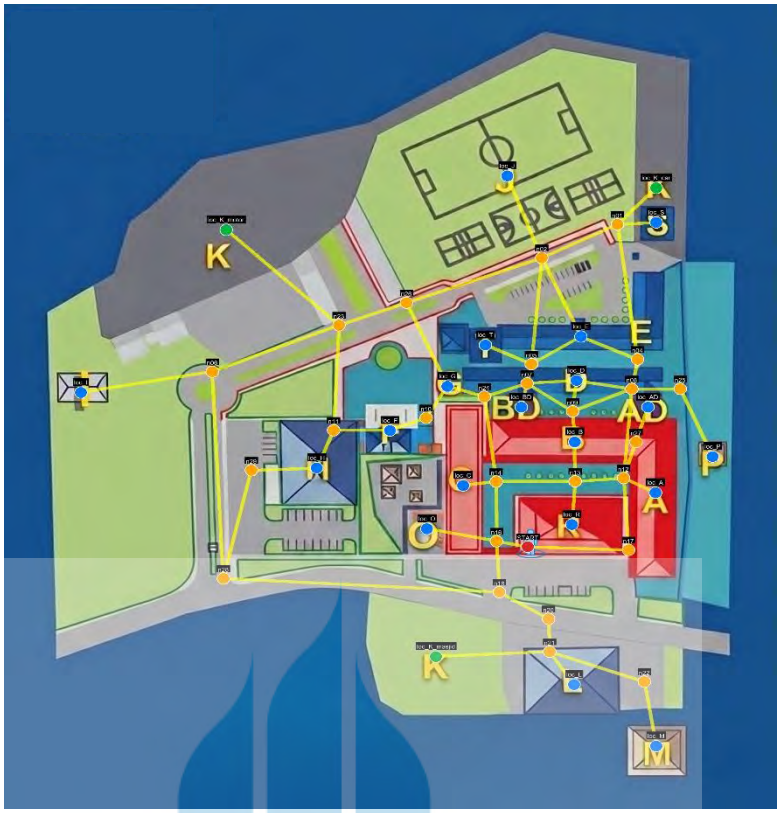
Pada navigasi Universitas Mercu Buana Meruya ini menggunakan algoritma A* untuk melakukan pencarian rute terpendek. Untuk denahnya, peneliti menggunakan gambar denah yang didapat di gedung Rektorat. Berikut gambar denah Universitas Mercu Buana Meruya dapat dilihat pada gambar 4.10.



Gambar 4. 10 Denah Universitas Mercu Buana Meruya

4.3.1 Representasi Graf Kampus

Setelah didapatkan denah Universitas Mercu Buana, peneliti menambahkan node dan edge pada denah tersebut untuk mengimplementasikan algoritma A*.



Gambar 4.11 Denah dengan *node* dan *edge*

Pada gambar 4.11 *node* dilambangkan dengan titik biru, dan hijau, persimpangan dilambangkan dengan titik orange dan untuk jalurnya digambarkan dengan garis kuning. Pada penelitian ini dilakukan penambahan persimpangan nodes pada persimpangan jalan untuk memastikan bahwa rute yang dihasilkan oleh algoritma A* mengikuti jalur pejalan kaki yang tersedia. Untuk jalur (*edge*) memiliki bobot berupa jarak dalam satuan meter yang diukur menggunakan fitur *measure distance* dari Google Maps. Sedangkan untuk node dalam graf dikategorikan ke dalam empat jenis :

Tabel 4.5 Kategori Node dalam Graf Kampus

Tipe Node	Jumlah	Keterangan
Titik awal	1	Titik awal posisi Sakiko
Persimpangan	26	Persimpangan jalan kampus
Bangunan	19	Pintu masuk gedung/fasilitas
Tempat Parkir	3	Area parkir kendaraan

Setiap node memiliki atribut koordinat pixel (x, y) yang merujuk pada posisi aktual pada denah kampus berukuran 2154 x 2475 piksel. Koordinat ini digunakan sebagai dasar perhitungan fungsi heuristik Euclidean distance pada algoritma A*. Setiap *edge* bersifat bidirectional dan memiliki atribut *distance* dalam satuan meter.

4.3.2 Implementasi Algoritma A*

Pada implementasi algoritma A*, setelah text dilakukan preprocessing selanjutnya akan di cek keyword navigasi di *NAV_KEYWORDS* setelah itu cocokan lokasi yang dituju oleh pengguna dengan file mapping.json untuk menemukan lokasinya, jika sudah ketemu

maka dilakukan pencarian jalur, dari titik awal, yaitu tempat asisten virtual kampus berada ke lokasi tujuan pengguna.

Gambar 4.12 Jalur dari Asisten Virtual Kampus menuju Masjid

Perhatikan gambar 4.12, contoh kasus pengguna bertanya “Dimana lokasi masjid manarul amal ?”. Dilakukan preprocessing dahulu dengan melakukan lowercase dan menghapus simbol-simbol, dan menjadi “dimana lokasi masjid”. Setelah itu di cek keyword navigasi : “dimana” dan ternyata ada di *NAV_KEYWORDS*, jika ada lanjut ke pencocokan “masjid manarul amal” di mapping.json dan ditemukanlah node_idnya loc_L. Lalu dicari jalurnya dari titik awal yaitu asisten virtual kampus berada menuju ke loc_L. Denahnya ditampilkan dilayar dalam bentuk canvas, lalu ditimpa titik merah yaitu titik awal, dan titik hijau yaitu lokasi tujuan pengguna, dan

dibuatlah jalurnya. Lalu karakter menjawab : “Kamu cukup ikuti jalur denah dilayar. Jarak yang ditempuh ialah 156,6 meter”.

4.3.3 Integrasi Navigasi dengan Sistem Asisten Virtual Kampus

Jika ada pengguna datang pada hari Selasa, pukul 09.00 dan bertanya : “Dimana saya bisa menemui Pak Abdi Wahab sekarang ?”. Disinilah NER dan algoritma A* saling bekerja sama. Pertama setelah text di preprocessing lowercase dan dihapus simbolnya, lalu dilakukan navigasi cek melalui *cek_keyword*, ternyata ada “dimana” ada di *NAV_KEYWORDS*. Tapi karena tidak ada lokasi yang spesifik ditanya oleh pengguna, maka gagal. Selanjutnya ke proses NER fast lane, entitas dosen Abdi Wahab ditemukan dan sesuai dengan *fuzzy matching* dengan skor 100%. Lalu dilakukanlah intent detection untuk mengetahui makna dari kalimat pengguna, karena ada kata “dimana” dan “sekarang” ada di kamus *LOCATIONS_KEYWORDS*, maka dipanggil schedule_location untuk mengecek data jadwal dosen dan sesuaikan dengan waktu di komputer sekarang. Karena waktunya menunjukkan pukul 09.00 pagi hari Selasa, dan ternyata Pak Abdi Wahab sedang mengajar di ruang D-204. Maka karakter Sakiko akan menjawab pertanyaan pengguna : "Pak Abdi Wahab saat ini sedang mengajar mata kuliah Animasi Komputer Dan Pemodelan 3D di ruang D-204, Gedung D (Lab).” Jalur navigasi yang dihasilkan dapat dilihat pada gambar 4.13, di gambar tersebut, dari titik awal, menuju ke titik D, yaitu Gedung D.

Gambar 4.13 Jalur dari Asisten Virtual Kampus menuju ruang D-204

4.4 Alur Proses Suara Dan Karakter

Interaksi yang natural antara pengguna dan asisten virtual bisa dicapai jika semuanya berjalan dengan lancar. Dari pengguna mulai berbicara sampai asisten virtual kampus menjawab pertanyaan pengguna. Rangkaian komponen ini dirancang untuk bekerja secara berkesinambungan agar sistem dapat memberikan respons yang komprehensif layaknya manusia sesungguhnya, baik dalam bentuk suara yang berkarakter maupun variasi ekspresi visual.

4.4.1 Arsitektur Sistem *End-to-End*

Sistem asisten virtual kampus terdiri dari sepuluh tahap pemrosesan yang bekerja secara sekuensial mulai dari penerimaan suara pengguna hingga pengiriman respons audio dan visual ke antarmuka *frontend*. Arsitektur ini diimplementasikan sebagai Flask server yang menangani request melalui API dan mengirimkan respons secara streaming menggunakan protokol *Server Sent Events* (SSE).

Tabel 4.6 Tahapan Sistem End-to-End Asisten Virtual Kampus

No	Tahap	Teknologi
1	*Audio Conversion*	ffmpeg (WAV 16kHz/16-bit/Mono)
2	*Speech-to-Text*	Gemma 4 E2B Multimodal (GGUF Q4_K_M)
3	*Text Preprocessing*	*Unicode* NFC + Normalisasi
4	*Navigation Check*	A* Pathfinding
5	*Hybrid NER (Fast Lane)*	rapidfuzz *fuzzy matching*
6	*Hybrid NER (Slow Lane)*	ChromaDB + ONNX MiniLM
7	*Intent Detection*	*Keyword classification*
8	*LLM Inference*	Gemma 4 via llama.cpp
9	*Text-to-Speech*	Edge-TTS (id-ID-GadisNeural)
10	*Voice Conversion*	RVC v2 (sakiko.pth, CUDA)

4.4.2 *Large Language Model* (LLM)

Model Gemma 4 E2B yang dijalankan pada llama-server memiliki dua peran dalam sistem asisten virtual. Peran pertama adalah sebagai *Speech-to-Text* (STT) untuk mengubah suara pengguna menjadi teks. Model ini dimuat bersama file multimodal projection (mmproj-F16.gguf) sehingga mampu menerima *input* audio. Audio yang direkam oleh pengguna melalui browser dikonversi terlebih dahulu ke format WAV (16kHz, 16-bit, Mono) menggunakan ffmpeg, kemudian diencode ke format

base64 dan dikirim ke llama-server. Hasil transkripsi berupa teks yang kemudian diteruskan ke tahap preprocessing dan NER.

Untuk pertanyaan yang memerlukan pemrosesan bahasa alami tingkat lanjut (*intent complex*), sistem menggunakan model Gemma 4 E2B dalam format GGUF yang dioptimasi menggunakan kuantisasi Q4_K_M. Model dijalankan secara lokal menggunakan llama.cpp yang dijalankan di GPU. Sistem prompt karakter Sakiko didefinisikan dengan aturan-aturan spesifik: menjawab dalam Bahasa Indonesia, membatasi respons maksimal 3 kalimat, menghindari emoji dan format markdown, serta hanya membahas topik seputar kampus dan akademik. Prompt yang dikirimkan ke model diperkaya (*augmented*) dengan beberapa konteks tambahan: informasi waktu dan tanggal saat ini dalam format Indonesia, data cuaca terkini untuk area Meruya Selatan Jakarta Barat, serta entity context dari sistem NER yang berisi informasi detail tentang entitas kampus yang teridentifikasi dalam pertanyaan pengguna. Respons model di-*stream* secara token-per-token dan dipecah menjadi kalimat-kalimat menggunakan sentence splitter berbasis regex yang memperhitungkan singkatan gelar akademik.

Gambar 4.14 Contoh pertanyaan kompleks

Pada gambar 4.14 diatas, saat pengguna menanyakan pertanyaan kompleks, asisten virtual kampus berhasil memberikan jawaban yang informatif berdasarkan *knowledge base* yang digunakan, tanpa mengarang informasi dari luar data.

4.4.3 *Text-to-Speech* (TTS) dan *Voice Conversion* (RVC)

Dalam proses sintesis suara pada asisten virtual kampus dilakukan dalam dua tahap. Tahap pertama menggunakan Edge-TTS dari Microsoft dengan *voice* model id-ID-GadisNeural untuk menghasilkan suara dasar dalam Bahasa Indonesia. EdgeTTS dipilih karena menghasilkan suara natural yang baik untuk Bahasa Indonesia dan dapat berjalan secara gratis. Hasil Edge-TTS dalam format MP3 dikonversi ke WAV 44100Hz menggunakan ffmpeg sebagai persiapan untuk tahap selanjutnya. Tahap kedua menggunakan *Retrieval-based Voice Conversion* (RVC) v2 untuk mengonversi suara Edge-TTS menjadi suara karakter Sakiko. Model RVC yang digunakan adalah sakiko.pth yang dilatih secara khusus dengan dataset suara karakter.

4.4.4 Kemandirian Komputasi Lokal

Salah satu tujuan utama penelitian ini adalah membangun asisten virtual yang mampu berjalan pada perangkat keras lokal dengan spesifikasi menengah, sehingga meminimalkan ketergantungan terhadap layanan komputasi awan (*cloud*). Seluruh proses komputasi utama pada sistem ini dijalankan secara lokal pada satu perangkat, meliputi :

Tabel 4.7 Daftar teknologi yang berjalan di lokal

Komponen	Teknologi	Lokasi Pemrosesan
Transkripsi suara (STT)	Gemma 4 E2B melalui llama-server	Lokal (GPU)
NER	RapidFuzz + ChromaDB	Lokal (CPU)
Klasifikasi *Intent*	Python	Lokal (CPU)
Pencarian rute (A*)	Python	Lokal (CPU)
LLM	Gemma 4 E2B melalui llama-server	Lokal (GPU)
Konversi suara karakter (RVC)	RVC v2 + HuBERT	Lokal (GPU)
Animasi karakter (Live2D)	PIXI Live2D Display	Lokal (Browser)

Satu-satunya komponen yang memerlukan koneksi internet adalah modul Edge-TTS untuk sintesis suara dasar dan layanan ini bersifat ringan. Keseluruhan sistem dijalankan melalui satu file script (start.bat) yang mengatur urutan inisialisasi secara otomatis : llama-server dijalankan terlebih dahulu dengan jeda 8 detik untuk memuat model ke memori GPU, kemudian server Flask dijalankan pada port 5000, dan terakhir browser dibuka secara otomatis menuju halaman antarmuka asisten virtual. Dengan arsitektur ini, sistem tidak memerlukan biaya berlangganan layanan API berbayar seperti OpenAI atau Google Cloud, serta dapat beroperasi di lingkungan yang memiliki keterbatasan akses internet. Hal ini menjadikan sistem lebih mandiri, hemat biaya, dan sesuai untuk diterapkan pada lingkungan kampus yang membutuhkan solusi efisien.

Arsitektur sekuensial ini dipilih secara sadar untuk mengakomodasi keterbatasan VRAM sebesar 4GB. Pada konfigurasi model Gemma 4 E2B (melalui *llama-server*) dimuat secara persisten di memori GPU dengan parameter -ngl 10 yang hanya memindahkan sebagian layer ke GPU. Sementara itu, modul RVC v2 diinisialisasi secara *lazy* dan baru memuat bobot model ke GPU saat pertama kali dipanggil. Dengan menjalankan tahap STT/LLM dan RVC secara bergantian, sistem menghindari kondisi *Out of Memory* (OOM) yang akan terjadi jika kedua model besar dimuat secara paralel pada GPU dengan kapasitas terbatas.

4.5 Pengujian Sistem

Pengujian sistem asisten virtual kampus ini menggunakan dua metode, yang pertama sistem diuji dengan metode *black box,* pengujian dengan *black box* dan yang kedua melalui evaluasi NER.

4.5.1 Pengujian *Black Box*

Pengujian dengan metode *black box* adalah metode pengujian sistem yang berfokus pada fungsionalitas dan antarmuka sistem dari sudut pandang pengguna, tanpa melihat struktur internal, logika algoritma atau kode program. Pada pengujian sistem asisten virtual kampus ini, pengguna memberikan 20 pertanyaan yang berbeda-beda. Berikut tabel pengujian :

Tabel 4.8 Daftar pertanyaan pengujian *black box*

No	Input Pengujian	Output Diharapkan	Output Aktual	Hasil
1.	Apa jabatan Pak Bambang Jokonowo?	Sistem menyebutkan jabatan Dr. Bambang Jokonowo dengan benar	Sistem menyebutkan jabatan Dr. Bambang Jokonowo dengan benar	Sesuai
2.	Bidang ilmu Pak Muhaimin apa?	Sistem menyebutkan bidang ilmu Muhaimin Hasanudin dengan benar.	Sistem menyebutkan bidang ilmu Muhaimin Hasanudin dengan benar.	Sesuai
3.	S2 Pak Muhaimin di mana?	Sistem menyebutkan riwayat pendidikan S2 Muhaimin Hasanudin dengan benar.	Sistem menyebutkan riwayat pendidikan S2 Muhaimin Hasanudin dengan benar.	Sesuai
4.	Gelar Pak Wawan Gunawan apa?	Sistem menyebutkan gelar akademik Wawan Gunawan dengan benar.	Sistem menyebutkan gelar akademik Wawan Gunawan dengan benar.	Sesuai
5.	Siapa itu Pak Hadi Santoso?	Sistem memberikan informasi profil singkat mengenai Dr. Ir. Hadi Santoso.	Sistem memberikan informasi profil singkat mengenai Dr. Ir. Hadi Santoso.	Sesuai
6.	Jadwal mengajar Pak Andi Nugroho?	Sistem membacakan daftar jadwal mengajar Andi Nugroho secara lengkap.	Sistem membacakan daftar jadwal mengajar Andi Nugroho secara lengkap.	Sesuai
7.	Pak Muhaimin ngajar apa saja?	Sistem membacakan daftar mata kuliah yang diajarkan oleh	Sistem membacakan daftar mata kuliah yang diajarkan oleh	Sesuai

		Muhaimin Hasanudin.	Muhaimin Hasanudin.	
8.	Jadwal kuliah Pak Misni?	Sistem membacakan jadwal perkuliahan yang diampu oleh Misni.	Sistem membacakan jadwal perkuliahan yang diampu oleh Misni.	Sesuai
9.	Pak Muhaimin sekarang dimana?	Sistem menginformasikan posisi ruangan Muhaimin Hasanudin saat ini berdasarkan jadwal, dan menawarkan panduan rute navigasi.	Sistem menginformasikan posisi ruangan Muhaimin Hasanudin saat ini berdasarkan jadwal, dan menawarkan panduan rute navigasi.	Sesuai
10.	Saya ingin menemui Pak Rushendra	Sistem memberikan informasi lokasi Ir. Rushendra saat ini dan menawarkan panduan navigasi menuju lokasi tersebut.	Sistem memberikan informasi lokasi Ir. Rushendra saat ini dan menawarkan panduan navigasi menuju lokasi tersebut.	Sesuai
11.	Apa saja jadwal kuliah Pak Wawan?	Sistem meminta klarifikasi dengan bertanya balik kepada pengguna untuk memastikan nama spesifik dari kandidat yang ada.	Sistem meminta klarifikasi dengan bertanya balik kepada pengguna untuk memastikan nama spesifik dari kandidat yang ada.	Sesuai
12.	Siapa Ibu Saruni?	Sistem berhasil mengenali nama parsial dan memberikan profil mengenai Saruni Dwiasnati.	Sistem berhasil mengenali nama parsial dan memberikan profil mengenai Saruni Dwiasnati.	Sesuai
13	Ceritakan tentang Pak Muhaimin	Sistem membentuk jawaban deskriptif yang natural mengenai profil Muhaimin Hasanudin.	Sistem membentuk jawaban deskriptif yang natural mengenai profil Muhaimin Hasanudin.	Sesuai

14	Apa kabar Sakiko?	Sistem merespons sapaan dengan natural tanpa salah mendeteksi nama entitas di dalam kalimat.	Sistem merespons sapaan dengan natural tanpa salah mendeteksi nama entitas di dalam kalimat.	Sesuai
15.	Siapa Pak Yatna Supriyatna?	Sistem memberikan informasi profil mengenai staf tenaga kependidikan Yatna Supriyatna.	Sistem memberikan informasi profil mengenai staf tenaga kependidikan Yatna Supriyatna.	Sesuai
16.	Jabatan Bu Umniy Salamah apa?	Sistem menyebutkan posisi atau jabatan tenaga kependidikan Umniy Salamah dengan benar.	Sistem menyebutkan posisi atau jabatan tenaga kependidikan Umniy Salamah dengan benar.	Sesuai
17.	Siapa Pak Bambangg Jokonowo?	Sistem berhasil menoleransi kesalahan ejaan (*typo*) pada teks dan tetap menyebutkan profil Dr. Bambang Jokonowo	Sistem berhasil menoleransi kesalahan ejaan (*typo*) pada teks dan tetap menyebutkan profil Dr. Bambang Jokonowo	Sesuai
18.	Dimana lokasi masjid?	Sistem menemukan node tujuan dan menampilkan rute grafis menuju Masjid Manarul Amal	Sistem menemukan node tujuan dan menampilkan rute grafis menuju Masjid Manarul Amal	Sesuai
19.	Tunjukkan jalan ke kantin.	Sistem menemukan node tujuan dan menampilkan rute navigasi menuju area Kantin.	Sistem menemukan node tujuan dan menampilkan rute navigasi menuju area Kantin.	Sesuai
20.	Apa saja keahlian Pak Ilham ?	Sistem mengenali sebagian nama dan menyebutkan keahlian dari Ilham Nugraha	Sistem mengenali sebagian nama dan menyebutkan keahlian dari Ilham Nugraha	Sesuai

4.5.2 Evaluasi NER

Evaluasi performa NER dilakukan menggunakan dataset yang terdiri dari 1.000 entri pertanyaan. Evaluasi dilakukan dengan dua skenario pembagian data (*split*): 80:20 dan 70:30. Metrik yang diukur meliputi precision, recall, dan F1-score.

Tabel 4.9 Hasil evaluasi NER dengan split 80:20

	Precision	Recall	F1-Score	Support
Dosen	91.61%	89.31%	90.45%	159
Tendik	100.00%	80.95%	89.47%	21
Accuracy			82.38%	180
Macro Avg	95.81%	85.13%	89.96%	180
Weight Avg	92.59%	88.33%	90.33%	180

Berdasarkan tabel 4.9 yang menyajikan hasil evaluasi pada skenario pengambilan sampel acak 80/20, sistem diuji menggunakan 200 pertanyaan yang terdiri dari 159 pertanyaan terkait entitas Dosen dan 21 pertanyaan terkait entitas Tenaga Kependidikan (Tendik). Hasil pengujian menunjukkan bahwa sistem mampu mencapai nilai Weight F1-Score sebesar 89,78%. Angka ini merepresentasikan keseimbangan performa sistem secara keseluruhan dengan tetap memperhitungkan proporsi data yang tidak seimbang (imbalanced). Menariknya, meskipun data Tendik jauh lebih sedikit, metrik *Recall* pada label Tendik 90,48% justru lebih tinggi dibandingkan label Dosen 87,42%. Hal ini mengindikasikan bahwa mekanisme *fuzzy matching* pada sistem sangat tangguh dan jarang sekali melewatkan atau gagal mengenali nama-nama staf tenaga kependidikan yang diucapkan oleh pengguna.

Tabel 4.10 Evaluasi NER dengan split 70:30

	Precision	Recall	F1-Score	Support
Dosen	91.74%	90.98%	91.36%	244
Tendik	100.00%	82.14%	90.20%	28
Accuracy			83.90%	272

Macro Avg	95.87%	86.56%	90.78%	272
Weight Avg	92.59%	90.07%	91.24%	272

Pada pengujian dengan skenario pengambilan sampel acak 70:30 pada tabel 4.10, jumlah data uji diperbesar menjadi 300 pertanyaan yang mencakup 244 entitas Dosen dan 28 entitas Tendik. Hasil evaluasi pada skenario ini menunjukkan peningkatan performa yang tipis dengan *Weight F1-Score* mencapai 91,24%. Perbandingan hasil antara kedua skenario ini (90,33% berbanding 91,24%) membuktikan bahwa performa algoritma pengenalan entitas (*Hybrid NER*) berjalan sangat konsisten dan stabil. Konsistensi ini menegaskan bahwa tingginya akurasi sistem bukanlah suatu kebetulan yang bergantung pada komposisi pertanyaan tertentu, melainkan hasil dari mekanisme pencocokan *dynamic threshold* yang secara efektif menangani berbagai variasi input suara.

Perbedaan antara *precision* tendik yang sempurna (100%) dengan *recall*-nya yang lebih rendah (82,14%) menunjukkan bahwa apabila sistem berhasil mengenali suatu entitas sebagai tendik, pengenalan tersebut selalu benar (*zero false positive*). Namun, sistem cenderung lebih sering gagal menemukan entitas tendik (*false negative*) dibandingkan dosen. Hal ini disebabkan oleh tiga faktor utama. Pertama, beberapa nama tendik memiliki kesamaan dengan nama dosen, seperti "Rudi Hartono" yang terdapat pada kedua kategori, sehingga sistem menandai hasil sebagai ambigu dan tidak dapat langsung mengenali entitas tersebut. Kedua, staf tendik umumnya memiliki jumlah alias yang lebih sedikit serta nama yang lebih pendek (misalnya "Kholilah", "Febi", "Dhayuria"), yang menyebabkan mekanisme *dynamic threshold* menuntut kecocokan yang sangat tinggi dan rentan gagal ketika terjadi sedikit variasi transkripsi dari modul *Speech-to-Text*. Ketiga, proporsi data tendik yang jauh lebih kecil (12 orang berbanding 85 Dosen) menyebabkan setiap satu kesalahan pengenalan berdampak signifikan terhadap penurunan nilai *recall* secara keseluruhan

4.5.3 Kesimpulan Pengujian

Berdasarkan hasil pengujian yang telah dilakukan menggunakan dua metode pengujian, yaitu pengujian fungsional (*black-box testing*) dan evaluasi kinerja modul pengenalan entitas (NER), dapat disimpulkan bahwa sistem asisten virtual kampus Sakiko telah berjalan sesuai dengan rancangan yang ditetapkan.

Pengujian fungsional (*black-box testing*) yang dilakukan terhadap 20 kasus uji dari berbagai skenario percakapan menunjukkan tingkat keberhasilan sebesar 100%, di mana seluruh kasus uji menghasilkan respons yang sesuai dengan harapan. Sistem mampu menangani berbagai jenis pertanyaan, mulai dari pertanyaan faktual mengenai profil dosen, jadwal mengajar, lokasi dosen secara *real-time*, pertanyaan ambigu yang memerlukan klarifikasi, pertanyaan kompleks yang diteruskan ke model bahasa, hingga navigasi kampus menggunakan algoritma A*. Sistem juga terbukti mampu menoleransi kesalahan transkripsi suara dan mengenali nama dari sebagian kata saja melalui mekanisme *fuzzy matching*.

Evaluasi kinerja modul NER yang dilakukan menggunakan dataset 1.000 pertanyaan pada dua skenario pengambilan sampel acak menunjukkan hasil yang konsisten. Pada skenario 80/20, sistem mencapai *Weight F1-Score* sebesar 90.33%, sedangkan pada skenario 70/30, sistem mencapai 91.24%. Konsistensi nilai di antara kedua skenario ini membuktikan bahwa performa sistem tidak bergantung pada komposisi pertanyaan tertentu, melainkan stabil di berbagai variasi data uji.