Journal of Renewable Energy and Smart Device (JORESD)

ISSN: 3026-216X
Vol. 3, No. 1, May 2025

Implementasi dan Evaluasi Sistem *Keyword Spotting Real-Time* berbasis Google *Speech* API pada Asisten Cerdas Sederhana di Google Colaboratory

Singgih Astaguna T[1] , Muhammad Fauzan Yusuf[2], Izza Etthiyyah RN[3], Dessy Ana Laila Sari[4]

[1,2,3,4]Universitas Negeri Makassar
[(*)]Coresponding Author E-mail: dessynaa@unm.ac.id

ARTICLE HISTORY

Received :
Revised :
Accepted :

ABSTRACT

The development of voice recognition technology, particularly Keyword Spotting (KWS) , offers significant potential for critical applications such as emergency response systems in the healthcare sector. The use of cloud-based Application Programming Interfaces (APIs) like the Google Speech API provides an accessible development pathway for these applications , but its performance in specific contexts, such as for the Indonesian language, requires thorough evaluation. This study aims to implement and evaluate the performance of a real-time KWS system based on the Google Speech API within a simple smart assistant prototype designed for healthcare emergency scenarios. The system was implemented on Google Colaboratory using a Text-based KWS approach. Quantitative testing was conducted with two male participants who uttered five Indonesian target keywords ("Hi", "Tolong", "Aduh", "Oke", "Salam") , with 25 repetitions for each keyword, resulting in a total of 250 test samples. System performance was measured using the Success Rate metric, calculated from the classification of True Positives (TP) and False Negatives (FN). The results show that the system was functionally implemented with a combined detection success rate of 62.8%. Significant performance variation was found both among keywords (highest success rate for 'Tolong' at 72.0%) and between participants (77.6% vs. 48.0%), highlighting speaker dependency. It is concluded that the Google API-based KWS approach is feasible for rapid prototyping, but the current accuracy and performance variability indicate the need for more robust and specific model development before it can be relied upon for critical healthcare applications.

Keywords: Keyword Spotting, Google Speech API, Smart Assistant, Healthcare, Indonesian Language

1. PENDAHULUAN

Pengenalan suara (*automatic speech recognition* / ASR) merupakan teknologi yang membuat mesin untuk mampu mengenali dan mentranskripsikan ucapan manusia ke dala bentuk teks. Dimana teknologi ini telah mengalami perkembangan yang signifikan dalam *deep learning*. Arsitektur yang canggih seperti *Conformer* menunjukkan bagaimana model-model ini mempelajari pola kompleks dari data audio dalam Jumlah yang besar sehingga Menghasilkan peningkatan akurasi yang tinggi [1], [2]. Proses ASR secara umum melibatkan analisis sinyal

audio, ekstraksi fitur vocal yang distingtif, serta pemodelan akustik dan Bahasa yang terintegrasi, menjadikan landasan dalam interpretasi input suara yang akurat.

Potensi teknologi ini menjadi titik revolusi interaksi antar manusia dan mesin, dengan menfasilitasi antarmuka yang lebih alami dan intuitif dan secara khusus telah menjadi tulang punggung dalam pengembangan asisten cerdas (*smart assistant*) yang kini merambah ke berbagai aspek kehidupan [3]. Dalam domain penting seperti kesehatan, asisten cerdas yang diaktifkan melalui pengenalan suara memberikan manfaat unik yaitu dengan kemampuannya berinteraksi tanpa melakukan sentuhan (*hands-free*) menjadi esensial bagi tenaga medis saat melakukan prosedur. Begitu juga dengan pasien, terutama yang memiliki keterbatasan mobilitas yang menyebabkan asisten cerdas memiliki potensi Menghasilkan respon cepat dalam situasi darurat medis [4]. Oleh karena itu, penerapan ASR dalam kesehatan tidak hanya meningkatkan aksesibilitas layanan dan efisiensi pemantauan pasien, tetapi juga secara fundamental berpotensi dalam mengubah paradigma kecepatan dan efektivitas respon medis dalam scenario yang darurat, meskipun dengan tantangan teknis seperti kebisingan lingkungan serta variasi suara akibat stress pengguna tetap perlu diatasi.

Untuk mengoptimalkan kecepatan respon dan efisiensi komputasi, khususnya dalam scenario yang menuntut deteksi *real-time* dengan sumber daya terbatas seperti perangkat *wearable* atau sistem pemantauan ambien, metode *Keyword Spotting* (KWS) menjadi pilihan yang tepat.

Konsep kerja dari KWS adalah dengan mendeteksi kata atau frasa kunci spesifik langsung dari aliran audio. Penentuan kata atau prasa kunci juga dilakukan dengan spesifik dan jumlahnya terbatas untuk menghindari transkripsi penuh penuh seluruh ucapan yang memberikan dampak komputasi yang mahal. Model KWS modern yang beredar umumnya sudah berbasis *neural network* yang dibuktikan oleh [5]. Penerapan KWS pada sistem cerdas juga sangat beragam. Dalam konteks kesegatan misalnya, KWS difungsikan sebagai pemicu pada sistem peringatan dini otomatis Ketika pasien mengucapkan kata kunci yang mengindikasikan keadaan darurat sehingga mengaktifkan permintaan bantuan kepada perawat atau anggota keluarga, atau bahkan memicu *protocol* darurat yang telah terprogram seperti notifikasi ke layanan medis [6], [7]. Kinerja KWS dalam sistem *real-time* menjadi parameter krusial dalam Aplikasi kesehatan kritis ini. Desain sistem KWS dalam domain ini perlu mempertimbangakn *trade-off* antara sensitifitasi dalam mendeteksi seluruh kejadian dengan kata kunci yang relevan (*false negatives)* dan spesifisitas (*false positives)* karena kasus kesalahan ini mampu memberikan konsekuensi signifikan dalam konteks medis.

Pesatnya pengembangan terhadap teknologi pengenalan suara, baik dalam domain kesehatan didukung dengan adalnya *Application Programming Interfaces* (API) dari penyedia layanan seperti Cloud. *Google Speech Recognition* API menawarkan kapabilitas pengenalan suara yang canggih dalam berbagai Bahasa. Walaupun telah tersedianya dukungan dalam Bahasa Indonesia [8], [9]tantangan dalam pengembangan tetap ada. Karakter linguistic Bahasa Indonesia yang unik, banyaknya dialek regional serta berbagai variabilitas akustik yang tinggi dalam kondisi stress atau *emergency* seperti perubahan intonasi, kecepatan bicara atau kebisingan lainnya mampu mempengaruhi kinerja sistem ASR secara umum.

Penelitian terkait pengembangan asisten cerdas dalam pengenalan suara telah menunjukkan perkembangan yang signifikan dan multidisiplin. Selain focus berkelanjutan dalam peningkatan

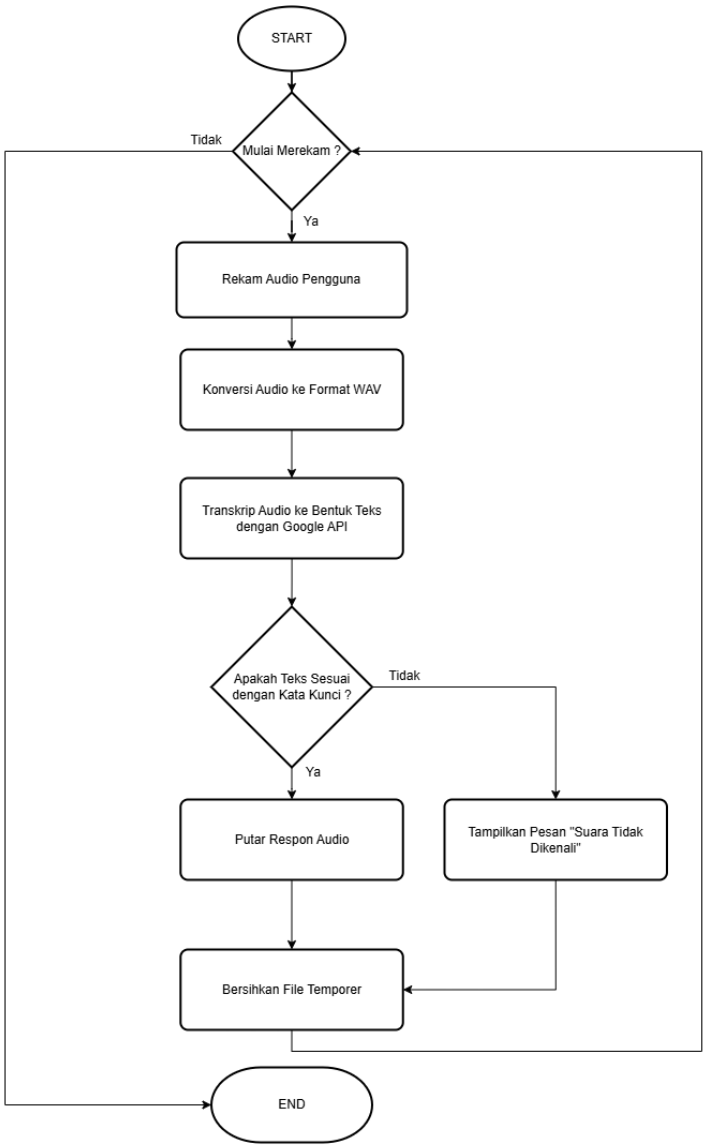
akurasi dan efisiensi model inti ASR dan KWS [1], [5], banyak penelitian yang mengeksplorasi aspek interaksi yang lebih kaya dan cerdas. Ini termasuk dengan kemampuan sistem dalam memberikan rekomendasi yang dipersonalisasi, pemahaman dialog secara mendalam dan bahkan kemampuan menunjukkan empati dalam interaksi [7]. Dalam domain kesehatan, Aplikasi teknologi suara sudah berkembang pesat dan mencakup area seperti sistem pemantauan pasien jarak jauh yang mampu mendeteksi perubahan kondisi pasien seperti pola batuk maupun kesulitan bernafas melalui analisis suara pasif, sistem deteksi jatuh berbasis suara untuk lansia, dukungan interaktif dalam manajemen penyakit kronis serta pengembangan sistem respon keadaan darurat otomatis yang mampu diaktifkan melalui perintah suara sederhana seperti teriakan atau rintihan [5], [8].

Dalam penelitian ini, dilakukan pendekatan berbasis teks (*Text-based* KWS) dimana Google API digunakan untuk mentranskripsi bagian dari audio yang kemudian dilakukan pencocokan teks dengan kata kunci dengan memanfaatkan model ASR skala besar milik Google. Meskipun demikian, dengan kombinasi pendekatan ini dengan *platform* prototipe seperti Google Colaboratory digunakan untuk dihasilkannya sistem yang mudah diakses.

2. Metodologi

2.1 Arsitektur Sistem

Untuk memahami lebih jelas arsitektur sistem yang dibuat dapat dilihat pada Gambar 1 di bawah ini. Sistem dirancang sebagai sebuah *pipeline* Pemrograman sekuensial yang dimulai dari tahap perekaman *audio* hingga tahap pemberian respon suara yang sesuai dengan *library* yang diminta. Proses diawali dengan *audio* dari pengguna ditangkap melalui *browser* di Google Colab dengan fungsi *build-in* yaitu record_audio_colab. *Audio* mentah yang didapatkan kemudian diproses secara otomatis dan dikonversi menjadi format *audio* yang diinginkan dengan *library* pydub [10] untuk memastikan kompatibilitas dengan tahapan selanjutnya.



Gambar 1. Arsitektur Sistem Pengenal Keyword yang Dikembangkan

File audio yang siap kemudian dianalisis oleh modul pengenalan suara dengan memanfaatkan *library* rekognisi suara [11], data yang telah dikirimkan ke *Google Speech Recognition* API [12]ditranskripsikan menjadi teks dalam Bahasa Indonesia (id-ID). Teks hasil transkripsi ini menjadi *input* bagi modul *Keyword Spotting* (KWS) dimana dilakukan normalisasi dan pencocokan dengan kamus kata kunci yang didefinisikan sebelumnya.

Ketika kata kunci berhasil terdeteksi, sistem akan mengambil respon teks yang sesuai dari kamus tersebut. Respon kemudian disintesis menjadi *file audio* oleh layanan Google *Text-to-Speech* [13]. Sistem diakhiri dengan pemutaran *audio* respons kepada pengguna melalui

lingkungan Colab. Seluruh proses dilengkapi dengan mekanisme penanganan kesalahan dasar untuk antisipasi kegagalan koneksi atau pengenalan suara, serta prosedur pembersihan file temporer sebagai efisiensi.

2.2 Rancangan Eksperimen

Eksperimen sistem dilakukan untuk mengevaluasi kinerja sistem yang dikembangkan secara sistematis dengan focus utama pada akurasi deteksi kata kunci target dalam kondisi yang terkontrol. Pengujian dilakukan dengan melibatkan dua partisipan pria dewasa dimana keduanya akan mengucapkan serangkaian kata kunci yang telah ditentukan sebelumnya.

Dataset *audio* uji untuk penelitian ini terdiri secara ekslusif dari lima kata kunci tarfet yang ada dalam kamus sistem seperti yang ditunjukkan pada Tabel 1.

Tabel 1. Kamus Perintah dan Respon Sistem yang Dikembangkan

Kata Kunci Target	Respon yang Diharapkan
“Hi”	“Halo, Saya adalah Asisten Anda”
“Tolong”	“Ada yang Bisa Dibantu”
“Aduh”	“Apakah Anda Butuh Pertolongan”
“Oke”	“Baiklah”
“Salam”	“Waalaikumsalam”

Setiap partisipan diminta untuk mengucapkan setiap kata kunci tersebut sebanyak 25 kali, sehingga Menghasilkan total 250 sampel *audio* uji (2 partisipan x 5 kata kunci x 25 repetisi) yang akan dianalisis. Seluruh sesi Pengujian dilaksanakan di dalam ruangan yang sunyi untuk meminimalisir interfensi dari kebisingan latar. Prosedur Pengujian dilakukan secara interaktif dengan setiap sampelnya direkam dan diproses dalam satu Pemrograman melalui Google Colab.

2.3 Metrik Evaluasi Kinerja Sistem

Untuk mengukur kinerja sistem secara kuantitatif, Evaluasi performa dilakukan dengan didasarkan pada metrik-metrik standar yang umum digunakan dalam tugas klasifikasi dan deteksi [14], [15]. Kinerja sistem dalam mendeteksi kata kunci target dievaluasi dengan mengklasifikasikan setiap hasil uji ke dalam salah satu dari empat kategori yang diadaptasi untuk scenario Pengujian sistem. Mengingat pengujian yang dilakukan hanya berfokus pada kata kunci target, maka akan diterapkan beberapa parameter yaitu.

a) *True Positive* (TP)
   TP merupakan kasus dimana kata kunci target yang diucapkan oleh partisipan berhasil dikenali dengan benar oleh sistem, yang kemudian dihasilkan respon sistem yang sesuai.
b) *False Negative* (FN)
   FP merupakan kasus dimana kata kunci target diucapkan namun sistem gagal memberikan respon yang benar. Kegagalan ini dapat terjadi apabila sistem tidak mampu mengenali ucapan sama sekali atau akibat hasil transkripsi yang tidak cocok dengan kata kunci manapun.

c) *False Positive* (FP)
Dalam konteks penelitian ini, FP didefinisikan sebagai kasus dimana sebuah kata kunci target diucapakan akan tetapi sistem keliru mentraskripsikannya sebagai kata kunci target lainnya yang masih valid dan memberikan respon yang salah.

d) *True Negative* (TN)
Kategori ini tidak dapat diterapkan dalam rancangan sistem yang diterapkan karena tidak adanya penggunaan data negative atau kata kunci non-target dalam pengujian.

Berdasarkan klasifikasi hasil uji coba dalam kategori TP, FN dan FP beberapa metrik kinerja turunan dapat dihitung untuk Menghasilkan Gambaran terkait akurasi sistem yang dibuat [14]. Metrik pertama yaitu Tingkat keberhasilan (*Success Rate*) yang secara langsung mengukur proporsi antara kata kunci yang diucapkan dengan benar dengan hasil yang berhasil dideteksi oleh sistem dari keseluruhan sampel uji yang dimiliki. Metrik ini mampu memberikan Gambaran umum tentang keandalan sistem dalam tugas utamanya mengenali kata kunci yang diberikan pengguna.

$$Sucess\ Rate = \frac{Jumlah\ TP}{Total\ Sampel\ Uji} \times 100\%$$

Selanjutnya untuk analisis yang lebih mendalam, digunakan juga metrik presisi (*Precision*) dan *Recall* (*Sensitivity*). Metrik presisi digunakan untuk mengukur Tingkat akurasi dari deteksi positif yang dilakukan oleh sistem yaitu dari semua kata kunci yang berhasil dideteksi dan direspon sistem, berapa fraksi yang benar-benar tepat [14], [15].

$$Precision = \frac{TP}{TP + FP}$$

Disisi lain, *Recall* dugunakan untuk mengukur kelengkapan atau kemampuan sistem dalam menemukan semua sampel yang relevan atau bisa disebutkan dari semua kata kunci yang seharusnya dideteksi, berapa fraksi yang berhasil ditemukan oleh sistem [1],[2].

$$Rcall = \frac{TP}{TP + FN}$$

Akibat seringnya terjadi *trade-off* antara presisi dan *recall*, F1-*score* dihitung untuk mencari rata-rata harmonic dari kedua metrik tersebut. F1-*score* mampu memberikan satu nilai tunggal yang menyeimbangkan kemampuan sistem untuk tidak membuat kesalahan deteksi sekaligus tidak melewatkan deteksi yang seharusnya dilakukan menjadi satu metrik Evaluasi yang kuat [15].

$$F1 - score = 2\ \times \frac{Presisi \times Recall}{Presisi + Recall}$$

Dengan memanfaatkan metrik-metrik ini, sistem dapat dianalisis kinerjanya secara objektif dan terstruktur.

3. HASIL DAN PEMBAHASAN

3.1 Implementasi Sistem Asisten Cerdas Sederhana

Berdasarkan dengan metodologi yang diuraikan sebelumnya, sistem pengenalan suara berbasis *keyword spotting* telah berhasil diimplementasikan menggunakan Bahasa Pemrograman Python dalam lingkungan Google Colaboratory. Program yang dihasilkan mampu menjalankan alur kinerjanya secara *end-to-end* dimulai dari perekaman *audio* partisipan,

pemrosesan dan pengiriman data *audio* hingga deteksi kata kunci spesifik serta pembangkitan respon suara yang sesuai dengan *library* yang didesain. Sistem juga terbukti fungsional dalam mengenali kata kunci yang sudah dijelaskan pada Tabel 1 sebelumnya.

```
from IPython.display import Javascript, display, Audio
from base64 import b64decode
from google.colab import output
from io import BytesIO
from pydub import AudioSegment
import speech_recognition as sr
from gtts import gTTS
import os
import time

# --- FUNGSI UNTUK MEREKAM AUDIO DARI BROWSER COLAB ---
def record_audio_colab(filename='recorded_audio.wav', duration=3):
    """
    Merekam audio dari mikrofon browser Google Colab.
    Args:
        filename (str): Nama file untuk menyimpan audio WAV.
        duration (int): Durasi rekaman dalam detik.
    Returns:
        str: Nama file audio WAV yang berhasil direkam, atau None jika gagal.
    """
    JS_CODE = """
    async function recordAudio(duration) {
        const stream = await navigator.mediaDevices.getUserMedia({ audio: true });
        const mediaRecorder = new MediaRecorder(stream);
        const audioChunks = [];

        mediaRecorder.addEventListener('dataavailable', event => {
            audioChunks.push(event.data);
        });

        const promise = new Promise(resolve => {
            mediaRecorder.addEventListener('stop', () => {
                const audioBlob = new Blob(audioChunks, { type: 'audio/webm' });
                const reader = new FileReader();
                reader.onloadend = () => resolve(reader.result.split(',')[1]);
                reader.readAsDataURL(audioBlob);
            });
        });
```

```
        mediaRecorder.start();
        setTimeout(() => mediaRecorder.stop(), duration * 1000); // Stop after 'duration' seconds

        return promise;
    }
    """
    print(f"Mulai merekam selama {duration} detik...")
    display(Javascript(JS_CODE)) # Jalankan kode JS untuk merekam
    audio_base64 = output.eval_js(f'recordAudio({duration})')
    audio_bytes = b64decode(audio_base64)

    webm_filename = "temp_audio.webm"
    with open(webm_filename, 'wb') as f:
        f.write(audio_bytes)

    # Konversi WEBM ke WAV (16kHz, 16-bit)
    try:
        audio = AudioSegment.from_file(webm_filename, format="webm")
        # Pastikan format WAV dengan spesifikasi yang diinginkan
            audio.export(filename,  format="wav",  parameters=["-acodec",  "pcm_s16le",  "-ar",
"16000", "-ac", "1"]) # -ac 1 untuk mono
        print(f"Rekaman disimpan sebagai {filename} (WAV, 16kHz, 16-bit, Mono)")
        os.remove(webm_filename) # Hapus file webm sementara
        return filename
    except Exception as e:
        print(f"Error converting audio: {e}")
         print("Pastikan ffmpeg terinstal (pydub seharusnya menginstal binari yang diperlukan
secara otomatis).")
        if os.path.exists(webm_filename):
            os.remove(webm_filename)
        return None

# --- FUNGSI UNTUK MEMUTAR AUDIO (RESPONS ASISTEN) DI COLAB ---
def play_audio_colab(file_path):
    """Memutar file audio di lingkungan Google Colab."""
    display(Audio(file_path, autoplay=True))

# --- INISIALISASI RECOGNIZER DAN KAMUS PERINTAH/RESPONS ---
r = sr.Recognizer()

# Tabel Perintah dan Respons Asisten Suara
# Sesuai dengan yang kita bahas:
perintah_dan_respons = {
```

```
    "hai": "halo, saya asisten anda",
    "tolong": "ada yang bisa dibantu",
    "aduh": "apakah anda butuh pertolongan",
    "oke": "baiklah",
    "salam" : "waalaikumsalam"
}

# --- FUNGSI UTAMA DETEKSI PERINTAH DAN RESPONS ---
def run_assistant():
    print("\n--- Asisten Suara Interaktif Dimulai ---")
    print("Perintah yang dikenali: 'hai', 'tolong', 'aduh', 'oke', 'salam' ")
    print("Tekan 'Enter' untuk mulai merekam perintah atau ketik 'exit' untuk mengakhiri.")

    while True:
        _ = input("\nSiap (tekan Enter untuk rekam atau ketik 'exit'): ").strip().lower() # Menggunakan _ karena nilai input tidak langsung dipakai

        if _ == "exit":
            print("Asisten dihentikan. Sampai jumpa!")
            break

        print("\nSilakan ucapkan perintah Anda (misal: hai, tolong, aduh, oke, salam) dalam 3 detik...")
        recorded_file = record_audio_colab(duration=3) # Rekam 3 detik

        if recorded_file and os.path.exists(recorded_file):
            try:
                with sr.AudioFile(recorded_file) as source:
                    # Menyesuaikan kebisingan tidak relevan untuk file, tapi lebih untuk stream langsung.
                    # Namun, SpeechRecognition cukup baik membaca dari file.
                    audio_data = r.record(source)

                # Gunakan Google Speech Recognition API untuk mengenali suara
                print("Mengenali suara...")
                # Penting: Pastikan bahasa Indonesia (id-ID) untuk akurasi yang lebih baik
                text = r.recognize_google(audio_data, language="id-ID")
                print(f"Anda mengatakan: \"{text}\"")

                recognized_command = text.lower()
                # Cek apakah perintah yang dikenali ada di daftar yang kita inginkan
                if recognized_command in perintah_dan_respons:
                    respons_text = perintah_dan_respons[recognized_command]
```

```
                print(f"Asisten merespons: \"{respons_text}\"")

                # Buat file audio respons menggunakan gTTS
                tts = gTTS(text=respons_text, lang='id')
                respons_filename = "asisten_response.mp3"
                tts.save(respons_filename)

                # Putar audio respons di Colab
                play_audio_colab(respons_filename)
                time.sleep(1) # Beri jeda singkat setelah respons selesai diputar
                os.remove(respons_filename) # Hapus file respons setelah diputar

            else:
                print("Perintah tidak dikenal. Coba ucapkan 'hai', 'tolong', 'aduh', 'oke', 'salam.")

        except sr.UnknownValueError:
            print("Maaf, tidak dapat mengenali suara Anda. Mohon coba lagi dengan jelas.")
        except sr.RequestError as e:
             print(f"Terjadi kesalahan koneksi ke layanan pengenalan suara Google; Pastikan
Anda memiliki koneksi internet: {e}")
        except Exception as e:
            print(f"Terjadi kesalahan tak terduga: {e}")
        finally:
            # Pastikan file rekaman sementara dihapus
            if os.path.exists(recorded_file):
                os.remove(recorded_file)
    else:
        print("Gagal merekam audio atau file tidak ditemukan.")

# --- JALANKAN APLIKASI ---
# Panggil fungsi ini untuk memulai asisten
if __name__ == "__main__":
    run_assistant()
```

3.2 Hasil Pengujian Kinerja Deteksi Kata Kunci

Pengujian kinerja sistem dilakukan sesuai dengan rancangan yang ditetapkan, yaitu dengan melakukan uji pada 5 kata kunci target yang diucapkan masing-masing sebanyak 25 kali pada setiap partisipan (total ada 2 partisipan). Data hasil pengujian menunjukkan kemampuan sistem dalam mengenali kata kunci target dengan benar. Hasil dari setiap percobaan diklasifikasikan sebagai “Suara Jelas” yang dianggap sebagai TP dan “Suara Tidak Jelas” sebagai FN. Tabel 2

di bawah menyajikan rincian hasil uji dan metrik Tingkat keberhasilan untuk skema Partisipan 1, Partisipan 2 dan hasil gabungannya.

Tabel 2. Rincian Hasil PEngujian Kinerja per Partisipan dan Gabungan

Partisipan	Kata Kunci Target	Jumlah Percobaan	Suara Jelas (Tp)	Suara Tidak Jelas (FN)	Tingkat Keberhasilan (%)
Partisipan 1 (P1)	"Hai"	25	16	9	64.0
	"Tolong"	25	23	2	92.0
	"Aduh"	25	19	6	76.0
	"Oke"	25	21	5	84.0
	"Salam"	25	18	7	72.0
	Sub Total P1	125	97	28	77.6
Partisipan 2 (P2)	"Hai"	25	12	13	48.0
	"Tolong"	25	13	12	52.0
	"Aduh"	25	12	13	48.0
	"Oke"	25	13	12	52.0
	"Salam"	25	10	15	40.0
	Sub Total P2	125	60	65	48.0
Gabungan	"Hai"	50	28	22	56.0
	"Tolong"	50	36	14	72.0
	"Aduh"	50	31	19	62.0
	"Oke"	50	34	16	68.0
	"Salam"	50	28	22	56.0
	Total	250	157	93	62.8

Dari Tabel 2, terlihat adanya variasi performa yang signifikan antara kedua partisipan. Partisipan 1 menunjukkan kinerja yang relative tinggi dengan Tingkat keberhasilan keseluruhan 77.6%. Sebaliknya, Partisipan 2 menunjukkan kinerja yang lebih rendah dengan keberhasilan keseluruhan hanya sebesar 48.0%. Analisis gabungan dari kedua partisipan menunjukkan bahwa kata kunci "Tolong" memiliki Tingkat keberhasilan tertinggi yaitu 72.0% diikuti oleh "Oke" sebesar 68.0% dengan nilai terendah pada kata kunci target "Hai" dan "Salam" sebesar 56.0%.

Perbedaan Tingkat keberhasilan antar kata kunci didapatkan akibat pengaruh karakteristik fonetik dari setiap kata sehingga struktur akustik menjadi lebih sulit maupun mudah untuk dibedakan oleh model yang digunakan. Selain struktur akustik kata kunci, ketergantungan pada penutur juga memberikan pengaruh seperti tinggi-rendah nada (*pitch*), kecepatan bicara, intonasi, akses atau bahkan jarak atau sudut mikrofon terhadap kinerja Google *Speech* API.

4. KESIMPULAN DAN SARAN

Penelitian ini berhasil merancang, mengimplementasikan dan mengevaluasi suatu sistem pengenalan suara berbasis *keyword spotting* untuk Aplikasi asisten cerdas sederhana. Sistem yang dibangun di lingkungan Google Colaboratory terbukti fungsional dalam menjalankan alur kerjanya.

Hasil pengujian kuantitatif menunjukkan bahwa sistem memiliki Tingkat keberhasilan gabungan sebesar 62.8% dalam mendeteksi lima kata kunci Bahasa Indonesia yang telah ditentukan sebelumnya. Terdapat variasi kinerja yang signifikan antar kata kunci dimana kata "Tolong" (72.0%), menjadi paling anda dikenali sedangkan "Hai" dan "Salam" paling rendah (56.0%). Lebih lanjut lagi, ditemukan perbedaan performa yang drastic antar partisipan yang menyoroti ketergantungan sistem yang dikembangkan dengan karakteristik penuturan individu.

REFERENSI

[1] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan, "Speech Recognition Using Deep Neural Networks: A Systematic Review," *IEEE Access*, vol. 7, pp. 19143–19165, 2019, doi: 10.1109/ACCESS.2019.2896880.

[2] H. Wijaya, "Teknologi Pengenalan Suara tentang Metode, Bahasa dan Tantangan: Systematic Literature Review," *bit-Tech,* vol. 7, no. 2, pp. 533–544, Dec. 2024, doi: 10.32877/bt.v7i2.1888.

[3] I. López-Espejo, Z.-H. Tan, J. Hansen, and J. Jensen, "Deep Spoken Keyword Spotting: An Overview," Nov. 2021, [Online]. Available: http://arxiv.org/abs/2111.10592

[4] A. Gulati *et al.*, "Conformer: Convolution-augmented Transformer for Speech Recognition," May 2020, [Online]. Available: http://arxiv.org/abs/2005.08100

[5] A. Ermolina and V. Tiberius, "Voice-Controlled Intelligent Personal Assistants in Health Care: International Delphi Study," *J Med Internet Res*, vol. 23, no. 4, p. e25312, Apr. 2021, doi: 10.2196/25312.

[6] R. M. Alsina-Pagès, J. Navarro, F. Alías, and M. Hervás, "homeSound: Real-Time Audio Event Detection Based on High Performance Computing for Behaviour and Surveillance Remote Monitoring.," *Sensors (Basel)*, vol. 17, no. 4, Apr. 2017, doi: 10.3390/s17040854.

[7] S. Pandya and H. Ghayvat, "Ambient acoustic event assistive framework for identification, detection, and recognition of unknown acoustic events of a residence," *Advanced Engineering Informatics*, vol. 47, p. 101238, Jan. 2021, doi: 10.1016/j.aei.2020.101238.

[8] A. Jarin, A. Santosa, M. T. Uliniansyah, L. R. Aini, E. Nurfadhilah, and G. Gunarso, "Automatic speech recognition for Indonesian medical dictation in cloud environment," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 13, no. 2, p. 1762, Jun. 2024, doi: 10.11591/ijai.v13.i2.pp1762-1772.

[9] H. Fakhrurroja, C. Machbub, A. S. Prihatmanto, and A. Purwarianti, "Multimodal Interaction System for Home Appliances Control," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 14, no. 15, p. 44, Sep. 2020, doi: 10.3991/ijim.v14i15.13563.

[10] D. M. W. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation," Oct. 2020, [Online]. Available: http://arxiv.org/abs/2010.16061

[11] A. Uberi, "SpeechRecognition," 2017. [Online]. Available: https://pypi.org/project/SpeechRecognition/

[12] J. Robert, "pydub," 2017. [Online]. Available: https://github.com/jiaaro/pydub

[13] P. Bédard, "P. Bédard," 2020. [Online]. Available: https://pypi.org/project/gTTS/

[14] T. Saito and M. Rehmsmeier, "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets," *PLoS One*, vol. 10, no. 3, p. e0118432, Mar. 2015, doi: 10.1371/journal.pone.0118432.

[15] T. Fawcett, “An introduction to ROC analysis,” *Pattern Recognit Lett*, vol. 27, no. 8, pp. 861–874, Jun. 2006, doi: 10.1016/j.patrec.2005.10.010.