Jurnal Komputer dan Teknologi Sains (KOMTEKS)
Vol. 4, No. 2, Juli 2025, hlm. 7-13 e-ISSN: 2964-5247

ANALISA PENCARIAN DAN PENCOCOKAN STRING DALAM APLIKASI BERBASIS PYTHON DENGAN LIBRARY FUZZY WUZZY TERHADAP DATASET CNN DAILY MAIL

Yogiswara Dharma Putra*[1], Yohanes Perdana Putra[2], Putu Satya Saputra*[3]

[1,2]Universitas Udayana
[3]Politeknik Negeri Bali
Email: yogiswaradharmaputra@unud.ac.id[1], yohanesperdanaputra@unud.ac.id[2], satya@pnb.ac.id[3]
*Penulis Korespondensi

(Naskah masuk: 13 Mei 2025, diterima untuk diterbitkan: 26 Juli 2025)

Abstrak

Di era digital ini, pengambilan data yang efisien sangat penting untuk kenyamanan pengguna. Jurnal ini menyajikan implementasi *Text Matching* pada aplikasi pencarian data berbasis Python. Aplikasi ini menggunakan antarmuka pengguna grafis (GUI) dengan tkinter dan mengintegrasikan perpustakaan FuzzyWuzzy dan Natural Language Toolkit (NLTK). Selain itu, teknologi *Speech to Text* (STT) meningkatkan interaksi pengguna. Menghubungkan ke database MySQL menggunakan MySQL *Connector*, aplikasi ini mendukung permintaan pencarian berbasis teks dan suara. FuzzyWuzzy digunakan untuk pencarian berbasis teks, sementara *SpeechRecognition* memfasilitasi konversi suara ke teks. Pencocokan Teks mengukur kesamaan, menyajikan hasil di atas ambang batas yang ditentukan pengguna. Sistem ini beroperasi dengan kumpulan data CNN Daily Mail, menyediakan antarmuka yang ramah pengguna untuk input teks dan suara. Integrasi Pencocokan Teks yang berhasil meningkatkan daya tanggap pencarian dan kemudahan penggunaan. Implementasi ini berfungsi sebagai landasan untuk memajukan fungsionalitas dalam aplikasi pencarian data.

Kata kunci: Sistem Temu Kembali Informasi, *Text Matching*, Python, CNN Daily Mail Dataset

STRING SEARCH AND MATCHING ANALYSIS IN PYTHON BASED APPLICATION WITH FUZZY WUZZY LIBRARY ON CNN DAILY MAIL DATASET

Abstract

In this digital age, efficient data retrieval is crucial for user convenience. This journal presents the implementation of Text Matching in a Python-based data search application. The application utilizes a graphical user interface (GUI) with tkinter and integrates FuzzyWuzzy and Natural Language Toolkit (NLTK) libraries. Additionally, Speech to Text (STT) technology enhances user interaction. Connecting to a MySQL database using MySQL Connector, the application supports both text and voice-based search queries. FuzzyWuzzy is employed for text-based searches, while SpeechRecognition facilitates voice-to-text conversion. Text Matching measures similarity, presenting results above a user-defined threshold. The system operates with the CNN Daily Mail dataset, providing a user-friendly interface for text and voice input. The successful integration of Text Matching enhances search responsiveness and user-friendliness. This implementation serves as a foundation for advancing functionality in data search applications.

Keywords: *Information Retrival, Text Matching, Python, CNN Daily Mail Dataset*

1. PENDAHULUAN

Di era digitalisasi, mencari informasi tentang topik tertentu menjadi lebih mudah dengan menggunakan mesin pencarian seperti Google Scholar atau sistem database terkait artikel ilmiah. Google Scholar adalah sebuah sistem mesin pencari berbasis web yang mengumpulkan teks atau literatur ilmiah dari berbagai disiplin keilmuan (Setiani Rafika dkk., 2017). Mesin pencarian seperti Google Scholar memungkinkan pengguna dapat menemukan jurnal ilmiah dari berbagai bidang keilmuan dan berfungsi sebagai perpustakaan pribadi (DPK Banten, 2024). Pengguna dapat menyimpan jurnal favoritnya dan diakses dari menu Perpustakaanku di sidebar pada Google Scholar. Seorang peneliti dan mahasiswa dapat memperkaya akademiknya dan memperluas pengetahuan dalam berbagai format penerbitan

akademik Google Scholar. Namun karena banyaknya artikel ilmiah, dapat menyulitkan untuk menentukan artikel atau jurnal yang paling sesuai dengan topik ilmiah dan tingkat akurasi yang tinggi (Khairiyah, 2022).

Cara yang dapat dilakukan dengan pemilihan artikel ilmiah yang berkaitan dengan topik tertentu adalah pencocokan objek. Pencocokan objek adalah metode pemilihan artikel akademis yang berkaitan dengan topik tertentu. Metode tersebut mengaitkan perhitungan persamaan atau perbedaan suatu objek yang terdapat dalam artikel ilmiah dengan subjek yang akan diteliti. Alat tersebut adalah kata atau frasa yang mewakili sebuah topik tertentu (Barlaug & Gulla, 2021).

Nilai kemiripan dihitung menggunakan sebuah *library* yaitu *library* Fuzzy-Wuzzy. Fuzzy Wuzzy adalah *library* Python yang didalamnya terdapat fungsi untuk melakukan pemodelan pencocokan *string* menggunakan pendekatan fuzzy. *Library* ini menggunakan salah satu metode perhitungan jarak yaitu jarak Levenshtein untuk menghitung perbedaan antara string dalam sebuah string. Pustaka Fuzzy Wuzzy menyediakan beberapa metode perbandingan string, termasuk Fuzzy Wuzzy Ratio, Fuzzy Wuzzy Partial Ratio, Fuzzy Wuzzy Token Sort Ratio, Fuzzy Wuzzy Token Set Ratio, dan Fuzzy Wuzzy W Ratio. Saat digunakan, *library* Fuzzy Wuzzy mengembalikan skor antara 0 dan 100. Skor ini akan menampilkan seberapa mirip atau relevan kedua *string* yang dibandingkan, dan skor yang lebih tinggi menunjukkan semakin miripnya kedua *string* tersebut (GeeksforGeeks, 2024).

Dengan dilakukannya penelitian Analisa Pencarian Dan Pencocokan *String* Dalam Aplikasi Berbasis Python Dengan Library Fuzzy Wuzzy Terhadap Dataset CNN Daily Mail diharapkan dapat membantu menemukan sebuah cara terbaik dalam menyeleksi artikel atau jurnal ilmiah terkait topik tertentu serta dapat meningkatkan efektifitas dan efisiensi dari metode *subject match* pada artikel Fairness in AI. Jurnal ini diharapkan juga dapat memberikan manfaatat bagi peneliti dan praktisi dalam mengembangkan sebuah teknologi AI yang adil dan tidak diskriminatif.

2. KAJIAN PUSTAKA

2.1. Algoritma Pencocokan *String*

String adalah kumpulan suatu karakter atau atribut (angka, huruf, atau atribut lainnya) dan direpresentasikan sebagai sebuah struktur data *array*. *String* dapat berupa kata, frasa, atau kalimat. Algoritma pencarian *string* atau pencocokan *string* merupakan suatu algoritma dalam menemukan semua kemunculan *string* pendek dan panjang, untuk *string* pendek disebut pola dan untuk *string* panjang disebut teks. Hasil pencarian *string* pada suatu dokumen bergantung pada teknik atau metode pencocokan *string* yang digunakan. Untuk dapat menentukan isi suatu dokumen secara tepat sesuai dengan kebutuhan informasi, diperlukan suatu metode pengambilan isi dokumen yang baik.

Approximate String Matching adalah pencocokan antara *string* berdasarkan kesamaan ejaan (jumlah atribut dan penempatan atribut), tingkat *similarity* kemiripan ditentukan pada jauh atau tidaknya perbedaan signifikan dalam penulisan dua *string* yang dibandingkan (Hariyanto, B., 2004). Penggantian *string* dapat berupa satu atribut huruf ke atribut huruf lainnya, menghilangkan satu atribut huruf dari suatu *string*, atau menyisipkan satu huruf ke dalam suatu string. Operasi ini digunakan untuk menghitung jumlah perbedaan untuk memeriksa *string* agar cocok dengan *string* sumber. Banyaknya selisih ditentukan oleh jumlah seluruh perubahan yang dapat terjadi pada setiap operasi. Operasi pergantian dalam mengubah *string* tersebut terdapat tiga tindakan seperti penghapusan, penyisipan dan penukaran (Dirgandhavi, 2019).

2.2. Pencocokan Fuzzy Wuzzy

Secara linguistik fuzzy diartikan secara samar-samar, dengan kata lain algoritma fuzzy merupakan algoritma yang samar. Dalam logika fuzzy, suatu nilai bisa menjadi “true” dan “false” pada saat yang bersamaan. Derajat nilai “true” atau “false” dalam logika fuzzy bergantung pada bobot anggotanya. Pencocokan *string* fuzzy merupakan teknik pencarian *string* yang menggunakan proses pendekatan terhadap pola string yang dicari. Cari string yang sama serta string yang dekat dengan string lain dalam kamus (Dirgandhavi, 2019).

Kunci dari konsep pencarian ini adalah apakah string yang dicari mirip dengan string dalam kamus, meskipun struktur karakternya tidak persis sama. Sebuah fungsi yang disebut *similarity function* digunakan untuk menentukan "kesamaan" ini. Fungsi tersebut digunakan bertugas menentukan string hasil pencarian ketika string hasil yang diharapkan ditemukan.

Fuzzy search dapat mengimbangi kesalahan input umum serta kesalahan yang disebabkan oleh pemindaian dokumen cetak menggunakan pengenalan karakter optik (Taufik dkk. 2017). Program pencocokan fuzzy biasanya menampilkan istilah-istilah yang tidak relevan dan terkait. Dimungkinkan juga untuk mencampur dan mencocokkan beberapa makna dengan hasil yang berulang-ulang, di mana hanya satu yang merupakan makna yang dimaksudkan pengguna. Fuzzy search jauh lebih canggih daripada pencarian string yang tepat ketika digunakan untuk penelitian dan investigasi. Fuzzy search sangat berguna saat mencari bahasa atau istilah asing yang ejaannya benar dan tidak umum diketahui. Pencarian fuzzy juga dapat digunakan untuk mencari individu yang menggunakan informasi lengkap atau sebagian untuk memperoleh informasi yang tepat.

2.3. Algoritma Levenshtein Distance

Algoritma Levenshtein Distance, yang juga dikenal sebagai Edit Distance, dihitung menggunakan matriks untuk menentukan jumlah perbedaan antara dua *string* (Dirgandhavi, 2019).

Algoritma Levenshtein Distance secara luas digunakan di berbagai bidang, seperti mesin pencarian, pengecekan sebuah ejaan, pengenal ucapan, pengucapan dialek bahasa, analisis DNA dan deteksi pemalsuan. Algoritma Levenshtein Distance dapat menghitung jumlah operasi paling sedikit yang diperlukan untuk mentransformasikan atau merubah antara dua string yang meliputi penghapusan, penyisipan, dan penukaran.

Selisih perbedaan antara *string*, yang disebut juga *edit distance* atau jarak Levenshtein, yang diperoleh dengan memeriksa kesesuaian antara *string* sumber dengan *string* target. Jarak Levenshtein antara *string* "s" dan "t" adalah fungsi D yang memetakan (s,t) ke bilangan real nonnegatif. Misalnya, jika diberikan dua *string* s = s(1), s(2), s(3),..., s(m) dan t = t(1), t(2), t(3),..., t(n) dengan panjang |s| = m dan |t| = n, jarak Levenshtein dapat didefinisikan sebagai jumlah operasi minimum yang diperlukan untuk menyamakan kedua *string* tersebut.

$$D(s,t) = d(s_1,t_1) + d(s_2,t_2) + \cdots + d(s_l,t_l)$$

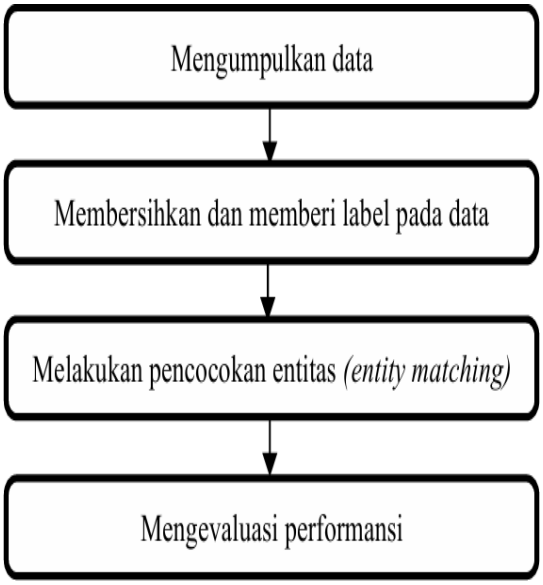
$$D(s,t) = \sum_{i=1}^{l} d(s_i,t_i) \quad (1)$$

$$dimana\ s_i, t_i \in V\ untuk\ i = 1,2,3,\ldots,l$$

$$d(s_i,t_i) = 0\ jika\ s_i = t_i\ dan\ d(s_i,t_i) = 1\ jika\ s_i \neq t_i$$

3. METODE PENELITIAN

Metode penelitian terdapat empat tahap yang akan dilakukan yaitu, pengumpulan data, pencocokan entitas, dan evaluasi kinerja.



Gambar 1. Alur Penelitian

Tahap pertama merupakan pencarian dan pengolahan data yang diambil dari website Kagle. Data tersebut berisi kumpulan ribuan artikel maupun berita yang diterbitkan oleh CNN Daily Mail. Dataset tersebut terdiri dari beberapa atribut, seperti nama penulis, judul artikel, isi artikel.

Pada tahap pencocokan entitas, peneliti menggunakan metode fuzzy-wuzzy sebagai metode untuk mengukur kemiripan dua string pada proses pencocokan entitas. Fuzzy Wuzzy merupakan suatu metode dengan teknik pencocokan *string* fuzzy dalam melakukan pencocokkan objek atau entitas dari dua data sumber yang berbeda. Pencocokan entitas dapat dilakukan dengan cara membandingkan antara judul dengan judul, penulis dengan penulis dan *keyword* dengan *keyword*. Metode Fuzzy Wuzzy Token Set Ratio digunakan karena metode tersebut dapat mengatasi sebuah perbedaan urutan token atau atribut dalam sebuah *string*.

Penelitian ini mengevaluasi kinerja dari beberapa metrik yaitu akurasi, presisi, recall dan F1-score. Akurasi menilai seberapa akurat sistem mendeteksi objek atau entias yang sama dalam dua database, presisi menilai jumlah objek atau entitas yang diidentifikasi oleh sistem benar-benar identik dengan objek di database data lainnya. Recall menilai berapa banyak objek yang diidentifikasi oleh sistem yang sebenarnya ada di sumber data lain. F1-Score merupakan kombinasi keakuratan dan kelengkapan data yang memberikan gambaran holistik tentang kinerja sistem. Rumus untuk menghitung setiap metrik dapat dilihat pada persamaan di bawah ini (Ardan dkk., 2023).

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \quad (2)$$

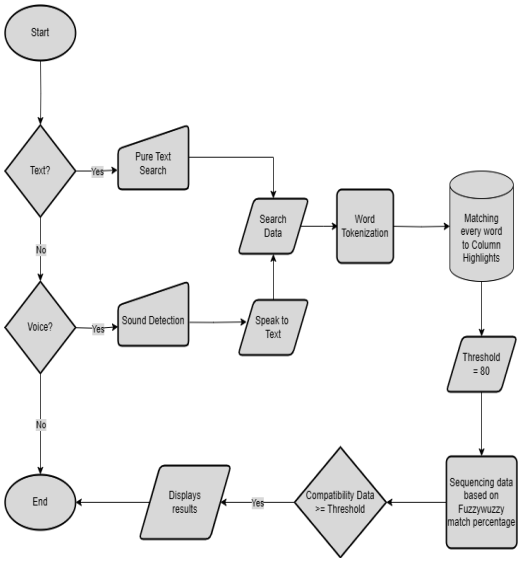
$$Precision = \frac{TP}{TP+FP} \quad (3)$$

$$Recall = \frac{TP}{TP+FN} \quad (4)$$

$$F1\ Score = 2 \times \frac{Recall \times Precision}{Recall+Precision} \quad (5)$$

Evaluasi kinerja pada penelitian ini digunakan untuk membandingkan kinerja pencocokan objek dengan mempertimbangkan bobot yang berbeda dan membandingkan penggunaan *blocking* dan *non-blocking*. Tujuan dari evaluasi ini adalah untuk menemukan suatu objek memiliki bobot terbaik yang dapat memberikan hasil terbaik dalam menentukan tingkat kemiripan antar artikel terkait.

Metodologi yang diterapkan dalam pengembangan aplikasi pencarian data ini menganut pendekatan yang sistematis dan terstruktur. Seperti yang diilustrasikan pada Gambar 2, aplikasi pencocokan teks menggunakan database SQLite sebagai penyimpanan data di *backend*.



Gambar 2. Cara Kerja Aplikasi

Tahap pertama adalah mengumpulkan data berupa teks atau suara. Jika masukan berasal dari suara, gunakan pustaka Speech_Recognition untuk mengubahnya menjadi teks. Perlu diketahui, perangkat harus terhubung dengan koneksi internet. Selanjutnya, tokenisasi *input* per kata menggunakan perpustakaan NLTK melalui kode: kata kunci_tokens = kata_tokenize (key_word.lower()). Misalnya, Anda dapat memasukkan "*Coronation Street star tops*", lalu setelah tokenisasi menjadi "[*'coronation', 'street', 'star', 'tops'*]". Kemudian buat koneksi ke database MySQL menggunakan perpustakaan Mysql.connector. Jika keterhubungan terjalin, pencocokan dilakukan berdasarkan kolom *highlight*. Pencocokan dilakukan dengan memasukkan token kata ke nilai seluruh baris dalam tabel artikel. Misalnya penulis menginputkan kata "*Coronasion Strit star top poll*" yang menjadi 5 token kata. Kemudian dilakukan pencarian kata yang dimulai dari "*Coronasion*" untuk setiap nilai pada kolom *highlight*. Lanjutkan pencarian kata "*Strit*", hingga kata token terakhir adalah "*poll*".

Jika hasil pencarian sudah ditemukan maka akan dilakukan proses perhitungan rata-rata nilai pencarian. Perhitungan nilai rata-rata kecocokan dicari dengan menggunakan Fuzzy wuzzy library. Pada kasus sebelumnya ditemukan kata "*Coronasion*" di database mendekati kata "*Coronation*" sehingga persentase kecocokannya sekitar 90%. Begitu seterusnya perhitungan dilakukan sampai token terakhir dan ditemukan rata-rata seluruh token (ada lima token). Kemudian diberikan batasan pada hasil pencarian yang ditampilkan yaitu harus berada di atas ambang batas (80%). Selanjutnya persentase pencocokannya juga diurutkan dari yang tertinggi hingga terendah dengan menggunakan Fuzzy wuzzy library. Pada gambar di bawah, muncul 3 hasil yang mendekati, dengan rentang persentase 80% hingga 90%.

4. HASIL DAN PEMBAHASAN

4.1. Grafis Tampilan User

Pustaka tkinter digunakan untuk membuat GUI yang estetis dan intuitif. Cuplikan kode didedikasikan untuk mendefinisikan dan menempatkan elemen seperti label, kolom entri, tombol, dan kotak teks untuk menampilkan hasil pencarian. Implementasinya memastikan antarmuka yang kohesif dan ramah pengguna yang selaras dengan prinsip desain yang diuraikan dalam metodologi.

```
# Simple Kode Implementasi
Tampilan user menggunakan GUI
import tkinter as tk
from tkinter import ttk

class DataSearchGUI(tk.Tk): def
init (self):
super(). init ()
self.title("Data Search
Application")
self.geometry("400x200")
# GUI element definitions and
placements
self.label_search =
ttk.Label(self, text="Enter
search query:")
self.label_search.pack(pady=10)
```

```
self.entry_query = ttk.Entry
(self)
self.entry_query.pack(pady=10)
self.button_search = ttk.Button
(self, text="Search",
command=self.perform_search)
self.button_search.pack(pady=10)
self.text_result = tk.Text(self,
wrap="none", height=10,
width=40)
self.text_result.pack(pady=10)
def perform_search(self):
# Code for initiating the search
based on user input pass
if  name == " main ":
data_search_app = DataSearchGUI
()
data_search_app.mainloop()
```

Kode Program 1. Graphics Tampilan User

4.2. Konektivitas Basis Data

Konektor MySQL digunakan untuk membuat koneksi ke database yang mendasarinya. Parameter koneksi, termasuk host, pengguna, kata sandi, dan nama basis data, dikonfigurasi untuk memastikan akses yang aman dan lancar ke kumpulan data CNN Daily Mail.

```
# Contoh kode program untuk koneksi basis data import mysql.connector

# Establishing a connection to the MySQL database db_connection = mysql.connector.connect(
host="localhost",
user="username",
password="password",
database="cnn dailv mail"
```

Kode Program 2. Koneksi Database

4.3. Pencocokan Kata dengan Fuzzy Wuzzy

Pustaka FuzzyWuzzy terintegrasi untuk memfasilitasi Pencocokan Teks. Fungsi rasio parsial digunakan untuk membandingkan kesamaan antara kueri yang dimasukkan pengguna dan entri kumpulan data. Penerapannya memastikan pendekatan yang fleksibel dan adaptif untuk menangani variasi dalam kueri penelusuran.

```
# Example code snippet for Text Matching us from fuzzywuzzy import fuzz

# Example data and user input dataset_entry = "Example dataset entry" user_query = "Exmple datset enty"
# Calculating similarity using partial ratio
similarity_score = fuzz.partial_ratio(user_query.lower(), dataset_entry.lower())
print(f"Similarity Score: {similarity_score}")
```

Kode Program 3. Pencocokan Kata

4.4. Pemrosesan Bahasa Alami (NLP) dengan NLTK

NLTK digunakan untuk tokenisasi kata dari kueri pengguna, sehingga meningkatkan pemahaman aplikasi tentang maksud pengguna. Implementasinya memastikan penggabungan fungsi NLTK untuk meningkatkan presisi pencarian.

```
# Example code snippet for Natural Language Processing (NLP) with NLTK from nltk.tokenize import word_tokenize

# Example user query
user_query = "Natural Language Processing with NLTK"

# Tokenizing user query
tokenized_query = word_tokenize(user_query.lower())
print(f"Tokenized Query: {tokenized_query}")
```

Kode Program 4. Pemrosesan Bahasa Alami (NLP) dengan NLTK

4.5. Integrasi Ucapan ke Teks

Pustaka Speech Recognition terintegrasi untuk mengaktifkan fungsionalitas *Speech to Text* (STT). Pengguna memiliki opsi untuk memasukkan permintaan pencarian melalui perintah lisan, meningkatkan aksesibilitas dan mendiversifikasi mode interaksi.

```
# Example code snippet for Speech to Text (STT) integration using SpeechRecognition import speech_recognition as sr
def perform_speech_to_text():
recognizer = sr.Recognizer()

with sr.Microphone() as source:
print("Please speak your search query...")
recognizer.adjust_for_ambient_noise(source)
audio = recognizer.listen(source, timeout=5)

try:
text_query = recognizer.recognize_google (audio, language="en-US") print (f"Speech to Text Result: {text_query}")
except sr.UnknownValueError:

print ("Sorry, I could not understand the speech.")

# Call the function to perform Speech to Text
perform_speech_to_text()
```

Kode Program 5. Integrasi Ucapan ke Teks

4.6. Pencarian Berbasis Teks

Mengilustrasikan fungsionalitas aplikasi, perhatikan implementasi berikut. Misalkan pengguna tertarik untuk menemukan artikel terkait "kecerdasan buatan" di kumpulan data CNN Daily Mail. Pengguna dapat memasukkan kueri ini menggunakan perintah berbasis teks dan suara tradisional.

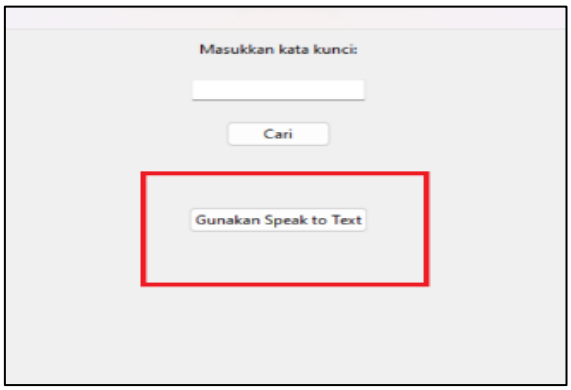
Pengguna memasukkan "kecerdasan buatan" ke dalam kolom input teks aplikasi. Aplikasi ini menggunakan Pencocokan Teks dengan FuzzyWuzzy untuk membandingkan kueri dengan entri dalam kumpulan data.

Gambar 3. Tampilan Aplikasi Pencarian

Hasil ditampilkan di kotak teks, diurutkan berdasarkan persentase kesamaan. Pengguna dapat dengan cepat mengidentifikasi artikel yang relevan berdasarkan informasi yang disajikan.

4.6. Pencarian Menggunakan Suara

Pengguna mengklik tombol "Gunakan Bicara ke Teks". Aplikasi ini mengaktifkan fungsionalitas Ucapan ke Teks, memungkinkan pengguna untuk mengungkapkan pertanyaan secara verbal. Kueri lisan dikonversi menjadi teks menggunakan pustaka SpeechRecognition. Aplikasi melanjutkan dengan Pencocokan Teks dan menampilkan hasilnya serupa dengan pencarian berbasis teks.



Gambar 4. Tampilan Aplikasi Menggunakan Suara

Pengimplementasian di atas menunjukkan keserbagunaan aplikasi, mengakomodasi metode pencarian konvensional dan inovatif untuk memenuhi preferensi pengguna.

4.7. Pengujian Aplikasi

Pengujian aplikasi pencarian data dilakukan dengan mencari persentase kecocokan kata kunci yang dimasukkan pengguna dengan data yang ada di database menggunakan algoritma fuzzy *matching*. Aplikasi akan diuji dengan menggunakan 7 kata acak sebagai kata kunci untuk mengukur kinerja sistem pencarian. Pengujian kinerja mencakup evaluasi empat metrik utama: akurasi (*accuracy*), presisi (*precision*), *recall*, dan *F1 score*. Selain itu, waktu pengujian juga dicatat untuk menilai kecepatan pencarian aplikasi.

Tabel 1. Pengujian Sistem

Kata Pencarian	Kata di Database	Persentase Kecocokan
Coronation Strit	Coronation Street	88%
Kim Kardashin	Kim Kardashian	92%
Liverpol	Liverpool	88%
Nikolas Anelka	Nicolas Anelka	93%
Indonesia	Indonesia	100%
The F18 crashed	The F18 crashed	100%
Hollywod actress	Hollywod actress	94%

Tabel 2. Evaluasi Kinerja Sistem

Indikator	Hasil
Akurasi (accuracy)	1.0
Presisi (precision)	1.0
Recall	1.0
F1 Score	1.0
Waktu Pengujian	15.861 detik per soal

Tabel 1 dan Tabel 2 di atas merupakan hasil pengujian pada aplikasi pencarian kata berbasis metode pencarian Fuzzy Wuzzy dan algoritma Levenshtein Distance. Berdasarkan hasil pengujian menunjukkan bahwa aplikasi pencarian data bekerja dengan sangat efisien dan akurat dalam menemukan data yang relevan berdasarkan kata kunci yang diberikan. Semua metrik kinerja berada pada tingkat maksimal (1.0), yang menunjukkan bahwa sistem pencarian berhasil menangkap semua data yang diinginkan tanpa kesalahan, dan waktu yang diperlukan untuk setiap pencarian adalah cukup baik mengingat kompleksitas proses pencarian yang dilakukan.

5. SIMPULAN

Integrasi *Text Matching* dengan FuzzyWuzzy dan teknologi STT pada aplikasi pencarian data berbasis Python menunjukkan kinerja yang sangat baik dengan akurasi, presisi, *recall*, dan *F1 score* sebesar 1.0. Aplikasi ini selalu menghasilkan pencarian yang tepat dan relevan, serta efisien dengan waktu pengujian rata-rata 15.861 detik per pencarian. Implementasi ini meningkatkan daya tanggap pencarian dan kemudahan penggunaan, serta berfungsi sebagai landasan untuk memajukan fungsionalitas aplikasi pencarian data di masa depan.

6. DAFTAR PUSTAKA

ARDAN, I.S., SULASTRI, M.J. and RAKHMAWATI, N.A., 2023. *Analisis performansi entity matching dengan FuzzyWuzzy pada artikel Fairness AI*. Jurnal Teknoinfo, 17(2), Article

DOI: 10.33365/jti.v17i2.2711. Available at: https://ejurnal.teknokrat.ac.id/index.php/teknoinfo/article/view/2711.

BARLAUG, N. and GULLA, J.A., 2021. *Neural networks for entity matching: A survey*. ACM Transactions on Knowledge Discovery from Data, 15(3), Article 52, pp.1–37. Available at: https://doi.org/10.1145/3442200.

DIRGANDHAVI, N.A., 2019. *Algoritma fuzzy string matching untuk pencocokan string: Penerapan program dinamis*. [Makalah] IF2211 Strategi Algoritma, Semester II Tahun 2018/2019, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung. Available at: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/Makalah/Makalah-Stima-2019-122.pdf.

DPK BANTEN, 2024. *Google sebagai sumber informasi untuk menulis di era disrupsi Covid-19 | Dinas Perpustakaan dan Kearsipan Provinsi Banten*. [online] Available at: https://dpk.bantenprov.go.id/Layanan/topic/265 [Accessed 2 Jun. 2024].

FITRIAH, S.N., ROSITA, W. and ROHMANIYAH, 2022. *Pemanfaatan sistem klasifikasi dan shelving bahan pustaka yang efektif dalam memenuhi kebutuhan temu kembali informasi di UPT Perpustakaan Politeknik Negeri Sriwijaya*. *El-Pustaka: Jurnal Ilmu Perpustakaan dan Informasi Islam*, 3(2), pp.83–106. Available at: https://ejournal.radenintan.ac.id/index.php/elpustaka/article/view/13949.

GEEKSFORGEEKS, 2024. *FuzzyWuzzy Python library - GeeksforGeeks*. [online] Available at: https://www.geeksforgeeks.org/fuzzywuzzy-python-library/ [Accessed 2 Jun. 2024].

HARIYANTO, B., 2004. *Sistem manajemen basis data*. Bandung: Informatika.

KHAIRIYAH, W. and MARLINI, 2023. *Pemanfaatan Google Scholar dalam pemenuhan kebutuhan informasi penelitian mahasiswa Prodi Perpustakaan dan Ilmu Informasi Universitas Negeri Padang*. Nautical: Jurnal Ilmiah Multidisiplin, 1(12), pp. 1058–1071.

MAULANA, T.I. and ABDILLAH, A.R., 2023. *Pemanfaatan sistem temu kembali informasi dalam pencarian dokumen menggunakan Vector Space Model*. *SINTESIA: Jurnal Sistem dan Teknologi Informasi Indonesia*, 1(2), Article 39365. Available at: https://journal.unj.ac.id/unj/index.php/SINTESIA/article/view/39365.

NANDA, S., 2019. *Perkembangan trend terbaru dalam temu kembali informasi bagi mahasiswa pascasarjana UIN Sunan Kalijaga Yogyakarta*. *Shaut Al-Maktabah: Jurnal Perpustakaan, Arsip dan Dokumentasi*, 11(2), Article 251. Available at: https://rjfahuinib.org/index.php/shaut/article/view/251.

RAFIKA, A., PUTRI, H. and WIDIARTI, F., 2017. *Analisis mesin pencarian Google Scholar sebagai sumber baru untuk kutipan*. Journal Cerita: Creative Education of Research in Information Technology and Artificial Informatics, 3(2), pp.193–205. Available at: https://doi.org/10.33050/cerita.v3i2.657

TAUFIK, I., AISHIA, I.D. and JUMADI, 2017. *Implementasi Fuzzy Search untuk pendeteksi kata asing pada dokumen Microsoft Word*. *Jurnal Teknik Informatika*, 10(1), pp.1–8. Available at: https://media.neliti.com/media/publications/133562-ID-implementasi-fuzzy-search-untuk-pendetek.pdf.

ZAIN, N.Z. and RIDWAN, M.M., 2023. *Analisis konten YouTube sebagai sarana sistem temu kembali informasi*. *Maktabatun: Jurnal Perpustakaan dan Informasi*, 3(1), pp.23–27.