ISSN: 2614-6754 (print)
ISSN: 2614-3097(online)

Halaman 51185-51194
Volume 8 Nomor 3 Tahun 2024

Penggunaan Bahasa Indonesia dan Pengenalan Pola pada Mesin Pendeteksi Program *Auto-Correction LodiSearch*

Lodi Galang Putra Sugianto[1], Althaf Hilmi Haidar[2], Nabil Haidar Zaki Hermawan[3], Taqiyya Indartono[4], Rafli Muhammad Pradana[5], Endang Sholihatin[6]
[1,2,3,4,5] Informatika, UPN "Veteran" Jawa Timur
[6] Bahasa Indonesia, UPN "Veteran" Jawa Timur
e-mail: 24081010193@student.upnjatim.ac.id[1], 24081010150@student.upnjatim.ac.id[2], 24081010065@student.upnjatim.ac.id[3], 24081010247@student.upnjatim.ac.id[4], 24081010148@student.upnjatim.ac.id[5], endang.sholihatin.ak@upnjatim.ac.id[6]

Abstrak

Kesalahan dalam pengetikan dapat merubah kata baku menjadi kata tidak baku jika ejaan tidak sesuai. Hal tersebut dapat diatasi dengan menggunakan fitur *auto-correction* yang sudah ada di beberapa aplikasi *Word processing,* seperti Microsoft Word, Google Docs dan LibreOffice Writer. Namun, pada aplikasi di atas masih memerlukan pengecekan dari *user* karena aplikasi *Word processing* menggunakan metode pencocokan *string*, sehingga aplikasi akan memunculkan beberapa rekomendasi kata yang harus *user* cek secara manual. Penelitian ini mengembangkan sebuah inovasi baru dalam bentuk situs web bernama *"LodiSearch"* yang dirancang untuk mendeteksi penggunaan kata-kata tidak baku dan menggantinya dengan kata-kata yang benar sesuai aturan bahasa. Web ini dirancang untuk memudahkan pengguna untuk memeriksa kata tidak baku menjadi tidak baku dengan otomatis. Adapun tujuan dari penelitian ini adalah untuk 1) mengetahui cara kerja *auto-correction* program pendeteksi *LodiSearch* dan 2) mengetahui seberapa efektif penggunaan *auto-correction* dalam mengkoreksi penulisan Bahasa Indonesia yang baik dan benar. Jenis penelitian yang kami gunakan adalah metode kuantitatif dengan data yang didapat melalui survei secara *online* dalam bentuk *Google Form* yang telah diisi oleh 60 kuesioner. Hasil penelitian ini menunjukkan bahwa 1) cara kerja *auto-correction* pada program pendeteksi *LodiSearch* dimulai dengan memasukan dokumen yang memiliki format .docx atau .pdf menggunakan library Python, seperti *Flask, docx, csv, os, dan PyPDF2* untuk membaca isi dari dokumen dan load_correction_dict (file_path) yang berperan untuk memeriksa kumpulan kata baku yang ada di dalam file CSV. Dokumen tersebut diperiksa dengan mencari kata baku maupun yang tidak baku. Jika kata tersebut ada dalam baku_words, maka kata tersebut dianggap sebagai kata baku. Namun jika tidak, maka akan dicari bentuk bakunya dalam correction_dict, lalu menjalankan program dalam mode debug jika *script* dijalankan secara langsung. 2) Berdasarkan survei terhadap 60 responden, pengguna menyatakan bahwa sebanyak 87% responden menganggap bahwa *LodiSearch* dinilai cukup efektif dalam menemukan kata yang tidak baku dalam Dokumen yang dimasukkan dan melakukan perbaikan kata tidak baku dengan cukup baik, serta menunjukkan potensi yang positif untuk membantu pengguna dalam mengetahui ketepatan bahasa.

Kata Kunci: *Kesalahan Pengetikan, Kata Baku, Tidak Baku, Otomatis*

Abstract

Mistakes in typing can turn standard words into non-standard words if the spelling is not appropriate. This can be overcome by using the auto-correction feature that already exists in several Word processing applications, such as Microsoft Word, Google Docs and LibreOffice Writer. However, the above applications still require checking from the user because Word processing applications use the string matching method, so the application will bring up several word recommendations that the user must check manually. This research develops a new innovation in the form of a website called "*LodiSearch*" which is designed to detect the use of nonstandard words and replace them with correct words according to language rules. This website

is designed to make it easier for users to check words that are not standardized automatically. The objectives of this study are to 1) find out how the auto-correction of the *LodiSearch* detection program works and 2) find out how effective the use of auto-correction is in correcting good and correct Indonesian writing. The type of research we use is a quantitative method with data obtained through an online survey in the form of a Google Form that has been filled out by 60 questionnaires. The results of this study show that 1) how auto-correction works in the *LodiSearch* detection program starts with entering a document that has a .docx or .pdf format using Python libraries, such as Flask, docx, csv, os, and PyPDF2 to read the contents of the document and load_correction_dict (file_path) whose role is to check the set of standard words in the CSV file. The document is checked by looking for both standard and nonstandard words. If the word is in standard words, then the word is considered a standard word. But if not, it will look for the standard form in correction_dict, then run the program in debug mode if the script is run directly. 2) Based on a survey of 60 respondents, users stated that as many as 87% of respondents considered that *LodiSearch* was considered quite effective in finding nonstandard words in the entered Document and correcting nonstandard words quite well, and showed positive potential to help users in knowing language accuracy.

Keywords: *Typing Errors, Standard Words, Nonstandard Words, Automatic*

PENDAHULUAN

Dalam era digital yang berkembang pesat, teknologi telah mengubah berbagai aspek kehidupan manusia, termasuk di bidang pendidikan (Purba & Saragih, 2023). Banyak penutur Bahasa Indonesia masih mengalami kesulitan membedakan antara bahasa baku dan tidak baku, yang dapat berdampak pada kualitas tulisan dan komunikasi mereka. Beberapa faktor mempengaruhi penggunaan bahasa, seperti siapa yang berbicara, siapa yang mendengarkan, kondisi, situasi, serta ruang dan waktu (Devianty, 2021).

Untuk mengatasi masalah ini, pengguna Bahasa Indonesia memerlukan alat yang dapat mendeteksi dan mengoreksi penggunaan kata tidak baku secara efisien. *LodiSearch* memberikan solusi inovatif untuk memenuhi kebutuhan tersebut. *LodiSearch* adalah mesin pencari bahasa yang canggih yang secara otomatis mengidentifikasi kata-kata yang tidak sesuai dengan aturan dan mengubahnya menjadi kata-kata yang sesuai dengan bahasa Indonesia baku.

Pengembangan *LodiSearch* didasarkan pada beberapa penelitian sebelumnya di bidang *Natural Language Processing* (NLP) dan teknologi deteksi kesalahan tata bahasa. Sistem ini mengintegrasikan KBBI dengan pencocokan pola dan algoritma yang dipelajari mesin untuk mengidentifikasi kata-kata tidak baku dan menyarankan kata-kata baku pengganti yang sesuai.

Penerapan *LodiSearch* diharapkan dapat memberikan dampak signifikan terhadap peningkatan kualitas penulisan Bahasa Indonesia, baik dalam konteks akademik, jurnalistik, maupun komunikasi formal lainnya. Selain itu, *LodiSearch* juga dapat menjadi alat pembelajaran yang efektif bagi penutur Bahasa Indonesia, khususnya siswa sekolah menengah dan mahasiswa, untuk meningkatkan pemahaman mereka tentang penggunaan bahasa standar yang benar.

Tujuan dari penelitian ini yakni mengetahui cara kerja *auto-correction* mesin pendeteksi *LodiSearch* dan mengetahui seberapa efektif penggunaan program *auto-correction LodiSearch* dalam mengoreksi penulisan Bahasa Indonesia yang baik dan benar. Penelitian ini diharapkan dapat memberikan dampak positif pada pengembangan teknologi pemrosesan bahasa alami untuk bahasa Indonesia serta mendorong penggunaan bahasa Indonesia yang lebih efektif dan akurat di berbagai sektor.

Untuk mendukung permasalahan yang ada dalam pembahasan, peneliti berusaha untuk mencari berbagai literatur dan penelitian terdahulu (*prior research*) yang masih relevan terhadap masalah yang menjadi objek penelitian saat ini.Berdasarkan hasil eksplorasi terhadap penelitian-penelitian terdahulu, peneliti menemukan beberapa penelitian terdahulu yang relevan dengan penelitian ini. Meskipun terdapat keterkaitan pembahasan, penelitian ini masih sangat berbeda dengan penelitian terdahulu. Adapun beberapa penelitian terdahulu tersebut yaitu: Algoritma *String Matching Brute Force* dan *Knuth-Morris-Pratt* sebagai *Search Engine* Berbasis Web pada Kamus Istilah Jaringan Komputer. Oleh Adli dan Miftahul. Penelitian

ini menerapkan dua konsep algoritma, yaitu algoritma *String Matching Brute Force* dan *Knuth-Morris-Pratt* sebagai basis aplikasi program yang kami buat yang dapat dijadikan acuan algoritma mana yang paling cepat dan tepat dalam pencarian informasi, untuk itu perlu adanya sebuah penerapan algoritma *string matching*. Nantinya, akan ditampilkan berdasarkan sebuah atau beberapa kata yang akan dicari dan menentukan kata yang paling mendekati benar sebagai pembenarannya.

Kata Baku dalam Bahasa Indonesia

Bahasa baku merupakan salah satu bentuk ragam bahasa yang tercermin dari penggunaan kaidah yang benar meliputi ejaan, lafal, struktur, dan pemakaiannya (Rochmawati & Kusumaningrum, 2015). Dalam penggunaan bahasa baku sendiri, terdapat kaidah standar yang perlu untuk dipatuhi. Kaidah standar tersebut ialah pedoman yang mengatur penulisan dan pengucapan kata dalam bahasa indonesia, seperti PEUBI, tata bahasa baku, dan kasus umum.

Kata Tidak Baku dalam Bahasa Indonesia

Kata tidak baku adalah kata yang digunakan tidak sesuai dengan kaidah bahasa yang sudah ditentukan atau Kamus Besar Bahasa Indonesia (KBBI) (Rasuli, 2019). Ketidak bakuan suatu kata tidak hanya ditimbulkan karena kesalahan dalam penulisannya saja, tetapi juga karena kesalahan dalam pengucapan. Hal tersebut terjadi karena penggunaan kata-kata yang tidak baku sering muncul dalam percakapan kita sehari-hari.

Algoritma

Dalam ilmu komputer istilah algoritma digunakan untuk menggambarkan metode pemecahan masalah yang efektif, terbatas, deterministik, dan cocok untuk sebuah implementasi dalam program komputer (Nababan & Jannah, 2019). Algoritma yang digunakan dalam penelitian ini adalah algoritma *string matching*, yaitu algoritma *naive string matching* dan *exact string matching*. Hal ini karena, menurut banyak peneliti, komputer mengharapkan pencocokan *string* untuk mencari pola dalam teks. Algoritma yang dirancang untuk melakukan pencocokan *substring* pada *string* besar disebut algoritma *string matching.* (Nababan & Jannah, 2019).

Algoritma *String Matching*

Algoritma *String Matching* merupakan algoritma pencarian sebuah pola pada sebuah teks. Terdapat *string* yang disebut dengan *text*, yang merupakan bagian dari *string* yang di-input akan dicocokkan ke dalam *string text.* Algoritma *String Matching* digunakan untuk menemukan satu atau lebih *string* yang disebut dengan *pattern* atau pola, yang mana ditujukan untuk melakukan pencocokan *sub string* pada *string.* (Nababan & Jannah, 2019). *Auto-correct*

Menurut jurnal Algoritma *Jaro-Winkler Distance*: Fitur *Autocorrect* dan *Spelling Suggestion* pada Penulisan Naskah Bahasa Indonesia di BMS TV (2018), *Auto-correct* adalah suatu fitur yang dapat memeriksa dan memperbaiki kesalahan penulisan kata secara otomatis. Menurut Gueddah, dkk (2016), pemeriksaan ejaan terdiri dari perbandingan antara kata yang salah dengan daftar kata pada basis data dan menyarankan kata-kata yang mirip dengan kata yang salah dengan menghitung kemiripan jarak antara kata-kata tersebut.

LodiSearch

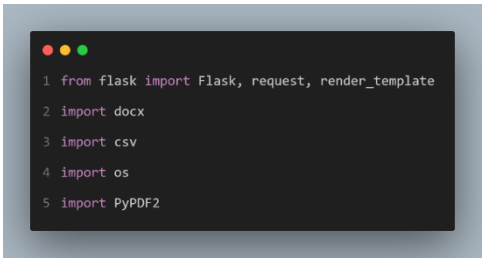
LodiSearch adalah program pendeteksi kata untuk melakukan *auto-correction* pada suatu dokumen secara otomatis berdasarkan KBBI dengan mendeteksi kata atau kalimat dalam suatu dokumen yang tidak baku dan tidak sesuai dengan kaidah kebahasaan Indonesia dan memperbaiki kata atau kalimat tersebut menjadi baku dan sesuai kaidah kebahasaan Indonesia.

METODE

Penelitian ini menggunakan metode kuantitatif yang dilakukan secara daring melalui *Google Form*. Teknik pengambilan data yang kami gunakan ialah pengisian kuesioner melalui platform digital yang telah diisi oleh beberapa mahasiswa dan guru di Surabaya, serta analisis terhadap kode pemrograman. Sumber data yang digunakan mencakup data primer dari kuesioner serta analisis kode pemrograman, serta data sekunder berupa referensi literatur terkait. Waktu penelitian direncanakan mulai dari akhir September sampai dengan bulan Desember.

HASIL DAN PEMBAHASAN

1. Cara Kerja *LodiSearch*
   a. Import Library



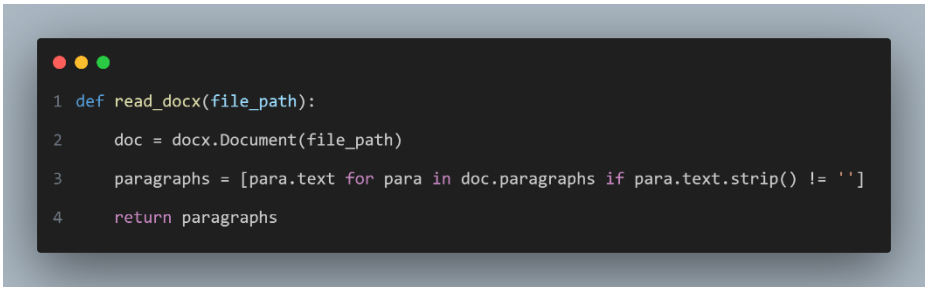
Pada kode yang telah kami gunakan kami menggunakan library Python, seperti *Flask, docx, csv, os, dan PyPDF2.*

- Flask sendiri digunakan untuk pengembangan web di Python. Modul Flask, request, dan render_template diimpor untuk membuat aplikasi web, menangani permintaan, dan merender template HTML.
- *docx* memungkinkan untuk membaca/menulis file yang berupa bentuk dokumen word.
- *csv* (Comma-Separated Values), memungkinkan pembacaan dan penulisan data CSV.
- *os*, modul pustaka standar untuk berinteraksi dengan sistem operasi, berguna untuk operasi file dan direktori.
- *PyPDF2* memungkinkan pembacaan/penulisan file PDF

   b. Inisialisasi Aplikasi Flask

Pada bagian baris kode ini digunakan untuk menginisialisasi aplikasi Flask, dengan variabel app sebagai instance dari aplikasi tersebut. Variabel ini akan digunakan untuk mendefinisikan rute dan menjalankan server web.

   c. Fungsi read_docx (file_path)



Fungsi read_docx (file_path) sendiri untuk membaca file .docx dari file_path yang telah ditentukan yang dimana prosesnya mengambil semua paragraf yang ada di file tersebut dan mengabaikan paragraf kosong.

d. Fungsi read_pdf (file_path)

```
def read_pdf(file_path):
    pdf_text = []
    try:
        with open(file_path, 'rb') as f:
            reader = PyPDF2.PdfReader(f)
            for page_num in range(len(reader.pages)):
                page = reader.pages[page_num]
                pdf_text.append(page.extract_text())
        return pdf_text
    except Exception as e:
        return []
```

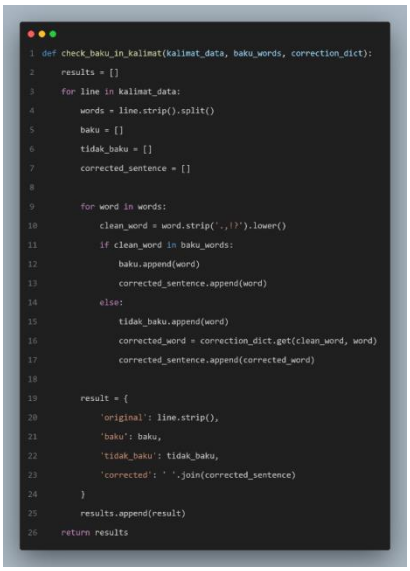
Fungsi read_pdf tidak jauh berbeda dari read_docx, yaitu membaca file pdf dari file_path yang telah ditentukan dan mengekstrak teksnya dari setiap halaman. Jika terjadi error, maka fungsi akan mengembalikannya ke list yang kosong.

e. Fungsi load_correction_dict (file_path)

```
def load_correction_dict(file_path):
    correction_dict = {}
    try:
        with open(file_path, mode='r', encoding='utf-8') as csvfile:
            reader = csv.reader(csvfile)
            for row in reader:
                if len(row) == 2:
                    tidak_baku, baku = row
                    correction_dict[tidak_baku.strip().lower()] = baku.strip()
    except FileNotFoundError:
        return {}
    return correction_dict
```

Kode ini berfungsi untuk memuat kamus kata tidak baku (tidak_baku) dan kata baku (baku) dari file CSV yang telah ada. Hal ini membaca setiap baris dalam file CSV dan menyimpan pasangan kata tidak baku dan baku ke dalam dictionary (correction_dict).

f. Fungsi check_baku_in_kalimat (kalimat_data, baku_words, correction_dict)



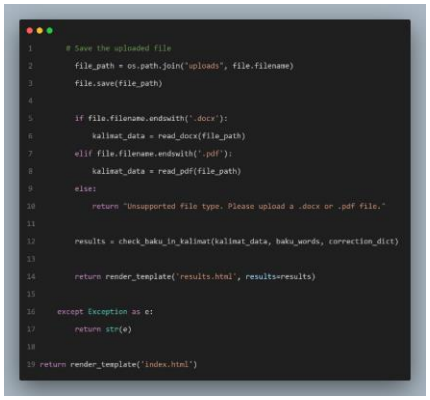
```
def check_baku_in_kalimat(kalimat_data, baku_words, correction_dict):
    results = []
    for line in kalimat_data:
        words = line.strip().split()
        baku = []
        tidak_baku = []
        corrected_sentence = []

        for word in words:
            clean_word = word.strip('.,!?').lower()
            if clean_word in baku_words:
                baku.append(word)
                corrected_sentence.append(word)
            else:
                tidak_baku.append(word)
                corrected_word = correction_dict.get(clean_word, word)
                corrected_sentence.append(corrected_word)

        result = {
            'original': line.strip(),
            'baku': baku,
            'tidak_baku': tidak_baku,
            'corrected': ' '.join(corrected_sentence)
        }
        results.append(result)
    return results
```

Fungsi kode ini adalah untuk memeriksa pada setiap bariskalimat_data (data teks dari file yang telah diunggah) untuk mengidentifikasikan kata baku dan tidak baku dari file yang diunggah. Setiap kata dalam satu baris diproses seperti berikut:

- Jika kata tersebut tercantum dalam kamus baku, maka kata tersebut dianggap sebagai kata yang tepat.
- Jika kata tersebut tidak ada dalam baku_words, maka kata tersebut akan dicari bentuk bakunya dalam correction_dict. Fungsi ini mengembalikan list results yang di mana setiap hasil berisikan kalimat asli, daftar kata baku, daftar kata tidak baku, dan kalimat yang sudah dikoreksi.

g. Rute Flask ke Halaman Utama

```python
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        if 'file' not in request.files:
            return "No file part"
        file = request.files['file']
        if file.filename == '':
            return "No selected file"
        baku_words = set()
        try:
            with open("teks.txt", "r", encoding='utf-8') as f1:
                baku_words = set(f1.read().split())

            correction_dict = load_correction_dict("daftar_baku_tidak_baku.csv")
```

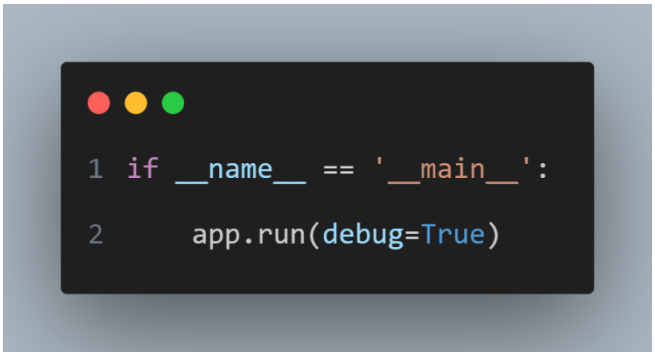


```python
        # Save the uploaded file
        file_path = os.path.join("uploads", file.filename)
        file.save(file_path)

        if file.filename.endswith('.docx'):
            kalimat_data = read_docx(file_path)
        elif file.filename.endswith('.pdf'):
            kalimat_data = read_pdf(file_path)
        else:
            return "Unsupported file type. Please upload a .docx or .pdf file."

        results = check_baku_in_kalimat(kalimat_data, baku_words, correction_dict)

        return render_template('results.html', results=results)

    except Exception as e:
        return str(e)

return render_template('index.html')
```

Pada bagian kode permintaan POST digunakan untuk:

- Memeriksa apakah file sudah diunggah dan memiliki nama.
- Membaca kata-kata baku dari file teks (teks.txt).
- Memuat correction_dict dari file CSV (daftar_baku_tidak_baku.csv).
- Menyimpan file yang diunggah ke folder uploads.
- Berdasarkan jenis file (.docx atau .pdf), membaca isi file menggunakan read_docx atau read_pdf.
- Memproses isi file dengan check_baku_in_kalimat dan menampilkan hasilnya di template results.html.
- Sedangkan untuk permintaan GET, berfungsi menampilkan template index.html

h. Menjalankan Aplikasi Flask



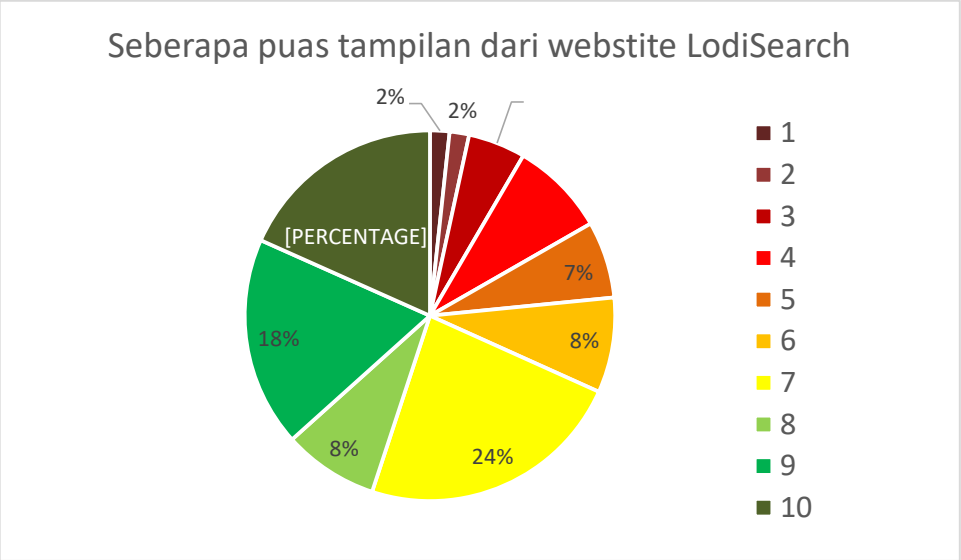
```python
if __name__ == '__main__':
    app.run(debug=True)
```

Baris kode ini untuk menjalankan aplikasi Flask dalam mode debug jika *script* dijalankan secara langsung

2. Efektifitas LodiSearch

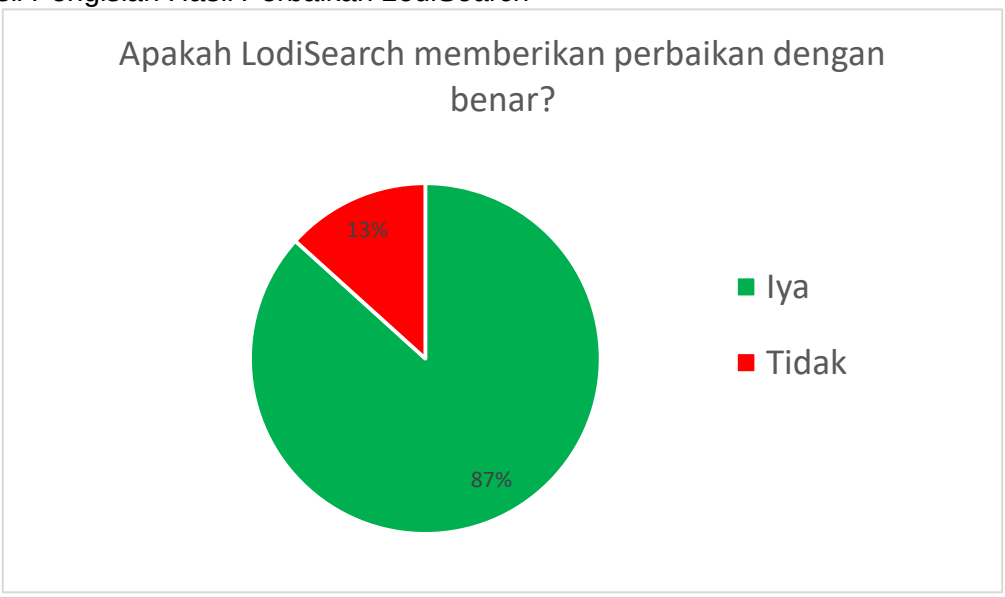
Berikut merupakan hasil survei kami mengenai aplikasi kami, *LodiSearch*

a. Hasil Tampilan *LodiSearch*



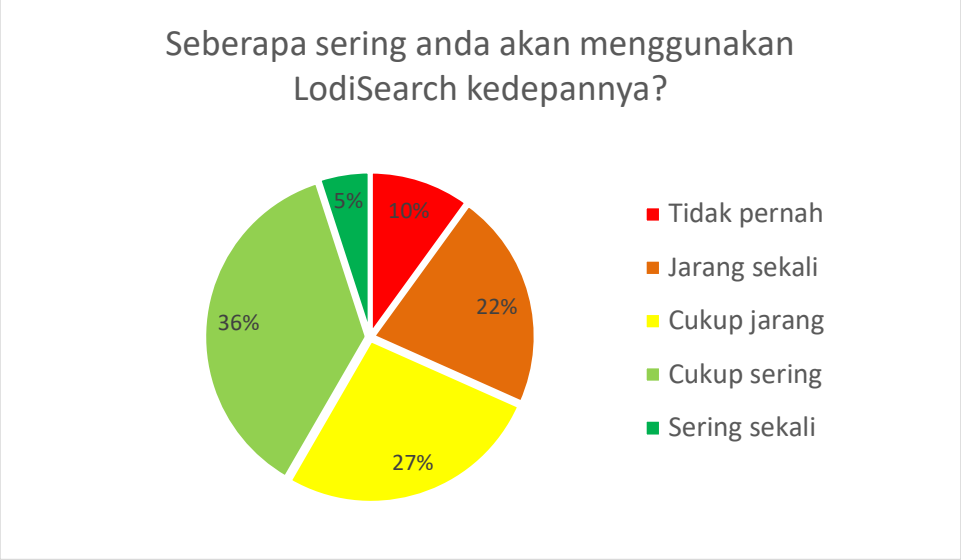
Berdasarkan hasil survei dari 60 responden, mayoritas menjawab bahwa "*LodiSearch"* memiliki tampilan yang cukup mudah untuk dipahami dengan persentase sebanyak 24% atau sebanyak 14 responden memilih skor 7 dari kepuasan tampilan *website*.

b. Hasil Pengisian Hasil Perbaikan *LodiSearch*



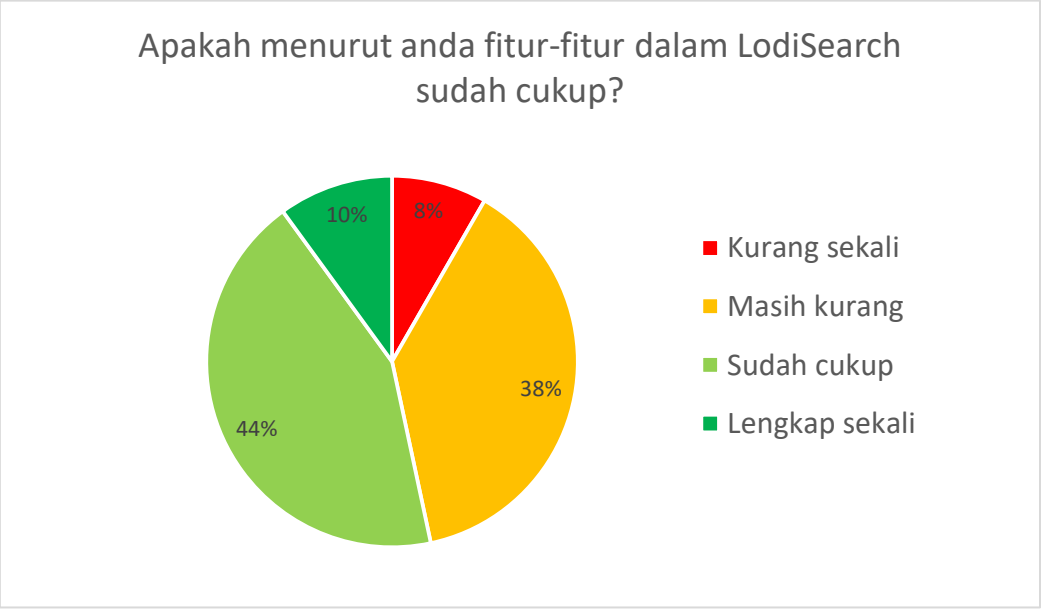
Menurut data dari 60 responden yang telah mengisi kuesioner, sebanyak 52 responden atau 87% responden merasa bahwa mesin pencari "*LodiSearch*" telah memberikan perbaikan yang benar terhadap kata tidak baku yang ada.

c. Hasil Pengisian Seberapa Berguna *LodiSearch*



Berdasarkan hasil survei yang diisi oleh 60 peserta, sebanyak 26% atau sebanyak 10 responden menilai skor 10 dan diikuti nilai skor 8 sebanyak 23% atau 8 responden pada penilaian seberapa berguna *LodiSearch* untuk keperluan mereka di masa yang akan dating.

d. Hasil Pengisian Mengenai Fitur *LodiSearch*



Menurut hasil survei yang diisi sebanyak 60 responden, sebanyak 44% atau 26 responden menilai bahwa fitur dalam *LodiSearch* dirasa sudah cukup, namun sebanyak 38% atau 23 responden menganggap bahwa bahwa fitur dalam *LodiSearch* masih dianggap kurang.

e. Hasil Pengisian Mengenai Penggunaan *LodiSearch* Kedepannya

Hasil survei yang kami lakukan dengan responden sebanyak 60 peserta, disimpulkan bahwa sebanyak 36,7% atau sebanyak 22 responden menganggap bahwa mereka akan menggunakan *LodiSearch* cukup sering untuk kedepannya. Akan tetapi, sebanyak 26,7% atau sebanyak 16 responden menganggap bahwa mereka cukup jarang menggunakan *LodiSearch* untuk kedepannya.

f. Hasil Pengisian Rekomendasi *LodiSearch*

Dari 60 responden yang kami dapat, sebanyak 52 responden atau sebanyak 87% responden menilai bahwa mereka akan merekomendasikan *LodiSearch* kepada orang lain.

SIMPULAN

Berdasarkan hasil penelitian ini, dapat disimpulkan bahwa Hasil penelitian ini menunjukkan bahwa 1) cara kerja *auto-correction* pada program pendeteksi *LodiSearch* dimulai dengan memasukan dokumen yang memiliki format .docx atau .pdf menggunakan library Python, seperti *Flask, docx, csv, os, dan PyPDF2* untuk membaca isi dari dokumen dan load_correction_dict (file_path) yang berperan untuk memeriksa kumpulan kata baku yang ada di dalam file CSV. Dokumen tersebut diperiksa dengan mencari kata baku maupun yang tidak baku. Jika kata

tersebut ada dalam baku_words, maka kata tersebut dianggap sebagai kata baku. Namun jika tidak, maka akan dicari bentuk bakunya dalam correction_dict, lalu menjalankan program dalam mode debug jika *script* dijalankan secara langsung. 2) Berdasarkan survei terhadap 60 responden, pengguna menyatakan bahwa sebanyak 87% responden menganggap bahwa *LodiSearch* dinilai cukup efektif dalam menemukan kata yang tidak baku dalam Dokumen yang dimasukkan dan melakukan perbaikan kata tidak baku dengan cukup baik, serta menunjukkan potensi yang positif untuk membantu pengguna dalam mengetahui ketepatan bahasa. Oleh karena itu, *LodiSearch* adalah alat yang sangat berguna untuk membantu memperbaiki penulisan Bahasa Indonesia agar lebih baik dan benar.

DAFTAR PUSTAKA

Devianty, R. (2021). Penggunaan Kata Baku dan Tidak Baku dalam Bahasa Indonesia. *Jurnal Pendidikan Bahasa Indonesia*, *1*(2), 121–132.

Nababan, A. A., & Jannah, M. (2019). ALGORITMA STRING MATCHING BRUTE FORCE DAN KNUTH-MORRIS-PRATT SEBAGAI SEARCH ENGINE BERBASIS WEB PADA KAMUS ISTILAH JARINGAN KOMPUTER. *Jurnal Mantik Penusa*, *3*(Desember), 87–94.

Prasetyo, A., Baihaqi, W. M., & Had, I. S. (2018). Algoritma Jaro-Winkler Distance: Fitur Autocorrect dan Spelling Suggestion pada Penulisan Naskah Bahasa Indonesia di BMS TV. *Jurnal Teknologi Informasi Dan Ilmu Komputer*, *5*(4), 435–444. https://doi.org/10.25126/jtiik.201854780

Purba, A., & Saragih, A. (2023). Peran Teknologi dalam Transformasi Pendidikan Bahasa Indonesia di Era Digital. *All Fields of Science Journal Liaison Academia and Sosiety*, *3*(3), 43.

Rasuli. (2019). MENINGKATKAN HASIL BELAJAR BAHASA INDONESIA MATERI KATA BAKU DAN TIDAK BAKU DENGAN MEDIA FLASH CARD PADA SISWA KELAS VI SDN. *Jurnal Ilmu Sosail Dan Pendidikan*, *3*.

Rochmawati, Y., & Kusumaningrum, R. (2015). Studi Perbandingan Algoritma Pencarian String dalam Metode Approximate String Matching untuk Identifikasi Kesalahan Kesalahan Pengetikan Teks.