

LAPORAN AKHIR
PROJECT-BASED LEARNING
MATA KULIAH ALGORITMA DAN PEMROGRAMAN LANJUT
TAHAP 1: RANCANGAN SCRIPT PEMROGRAMAN FUNGSIONAL
DAN/ATAU OOP PADA STUDI KASUS
NATURAL LANGUAGE PROCESSING
KELAS C



**“PENGEMBANGAN SISTEM PEMERIKSAAN EJAAN KATA BAHASA
INDONESIA MENGGUNAKAN ALGORITMA LEVENSHTEIN
DISTANCE”**

DISUSUN OLEH KELOMPOK “III” :

- | | |
|--------------------------------|-------------------------|
| 1. FAJRI USWATUL HANIFAH | (21083010058) - KETUA |
| 2. MUHIMMATUL AROFAH | (21083010055) - ANGGOTA |
| 3. NURMALITA FITRI RAMADANI | (21083010067) - ANGGOTA |
| 4. DIVA KEISHYA KIRANA | (21083010069) - ANGGOTA |
| 5. LYDIA ALMIRA RAHMA NOVANGGA | (21083010119) - ANGGOTA |

DOSEN PENGAMPU:

TRESNA MAULANA FAHRUDIN, S.S.T., MT (20219930501200)
SUGIARTO, S.KOM., M.KOM (198702142021211001)

PROGRAM STUDI SAINS DATA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN”
JAWA TIMUR
2022

KATA PENGANTAR

Dengan menyebut nama Allah Yang Maha Pengasih lagi Maha Penyayang, puji syukur kami panjatkan kehadirat Allah SWT yang telah melimpahkan Rahmat, Hidayah, dan Inayah-Nya sehingga kami dapat menyelesaikan penyusunan laporan akhir project-based learning mata kuliah Algoritma dan Pemrograman Lanjut Tahap 1 dengan judul “Pengembangan Sistem Pemeriksaan Ejaan Kata Dan Imbuhan Bahasa Indonesia Menggunakan Algoritma *Levenshtein Distance*” tepat pada waktunya.

Penyusunan laporan semaksimal mungkin kami upayakan dan didukung bantuan berbagai pihak, kami mengucapkan terima kasih kepada:

1. Bapak Tresna Maulana Fahrudin., S.ST., M.T dan Bapak Sugiarto, S.Kom., M.Kom. selaku dosen mata kuliah Algoritma dan Pemrograman Lanjut yang telah membimbing kami dalam pembuatan laporan ini.
2. Teman-teman yang telah mendukung segala sesuatu untuk pembuatan laporan ini sehingga dapat memperlancar dalam penyusunannya.
3. Semua pihak yang telah membantu kami dalam menyelesaikan laporan ini.

Namun tidak lepas dari semua itu, penulis menyadari sepenuhnya bahwa masih terdapat banyak kekurangan baik dari segi penyusunan bahasa dan aspek lainnya. Oleh karena itu, dengan lapang dada penulis membuka selebar-lebarnya pintu bagi para pembaca yang ingin memberi saran maupun kritik demi memperbaiki laporan ini. Penyusun sangat mengharapkan, semoga dari laporan ini dapat diambil manfaatnya

Surabaya, 11 Mei 2022

Tim Kelompok 3

DAFTAR ISI

KATA PENGANTAR	i
DAFTAR ISI	ii
DAFTAR GAMBAR	iii
DAFTAR TABEL	iv
BAB I : PENDAHULUAN	1
1.1. Latar Belakang	1
1.2. Permasalahan	2
1.3. Tujuan	2
1.4. Manfaat	2
BAB II : TINJAUAN PUSTAKA	3
2.1. Teori Penunjang	3
2.2. Penelitian Terkait	4
BAB III: METODOLOGI PENELITIAN	6
3.1. Studi Pustaka	7
3.2. Pengumpulan Dataset	7
3.3. Proses Pembuatan	7
3.4. Pengujian Model	15
3.5. Kategori Data Uji Coba	15
BAB IV : HASIL DAN PEMBAHASAN	18
4.1. Hasil Penelitian	18
4.2. Pembahasan	33
BAB V : KESIMPULAN	36
DAFTAR PUSTAKA	37
LAMPIRAN	38

DAFTAR GAMBAR

Gambar 1. Operasi Levenshtein Distance	4
Gambar 2. Flowchart Desain Sistem Ejaan Kata	6
Gambar 3. Representasi matriks	7
Gambar 4. Hasil representasi matriks	8
Gambar 5. Inisiasi urutan karakter	9
Gambar 6. Jarak D(1,1)	9
Gambar 7. Jarak D(1,2)	10
Gambar 8. Jarak D(1,3)	10
Gambar 9. Jarak D(1,4)	10
Gambar 10. Jarak D (1, 4)	11
Gambar 11. Jarak D(1, 4)	11
Gambar 12. Jarak D(7, 6)	12
Gambar 13. Script contoh matriks	14
Gambar 14. Script contoh matriks (1)	15
Gambar 15. Potongan script model	18
Gambar 16. Word Explanation Output	38
Gambar 17. Error Output	38
Gambar 18. Detected Word Output	38
Gambar 19. Notice Typo Output	39
Gambar 20. Command Output	39

DAFTAR TABEL

Tabel 1. Jarak Levenshtein Distance dari kata bandung	12
Tabel 2. Jarak Levenshtein Distance dari kata mcet	13
Tabel 3. Input Output Kategori Command	20
Tabel 4. Input Output Kategori Error	21
Tabel 5. Input Output Kategori Notice Typo	22
Tabel 6. Input Output Kategori Word Explanation	26
Tabel 7. Input Output Kategori Detected Word	32

BAB I : PENDAHULUAN

1.1. Latar Belakang

Teks merupakan salah satu media komunikasi yang digunakan dalam kehidupan sehari-hari. Teks atau dokumen memiliki peranan penting dalam bidang pendidikan. (A. I. Fahma et al, 2018) Umumnya siswa dan mahasiswa dituntut untuk membuat berbagai macam teks dan dokumen seperti laporan praktikum, makalah, jurnal, maupun skripsi. Dalam pembuatan teks maupun dokumen, kesalahan penulisan atau *typographical error* seperti ejaan, imbuhan, pemakaian huruf, penulisan kata, istilah serapan, singkatan dan akronim, gabungan kata, penulisan angka, pemakaian tanda baca, bentuk dan pilihan kata, serta struktur kalimat yang kurang tepat kerap terjadi dan dapat membuat arti dari kata yang disampaikan menjadi keliru atau memiliki arti lain.

Terdapat tiga faktor utama yang menyebabkan sering terjadinya *typographical error*, yang pertama yaitu karena belum pahamnya siswa atau mahasiswa dalam pembuatan dokumen dengan bahasa Indonesia yang baku dan sesuai dengan kaidah EYD. Karena sering ditemukan pada tulisan-tulisan siswa atau mahasiswa yang imbuhan katanya, penempatan tanda baca, penggunaan huruf kapital, dan huruf kecil yang kurang tepat dan lain sebagainya. Yang kedua, kurangnya kosakata standar yang dipahami dikarenakan kurangnya intensitas waktu untuk membaca. Dan yang ketiga yaitu kesulitan dalam menyatukan ide siswa atau mahasiswa dan kutipan dari bacaan yang digunakan.

Karena dituntut untuk menulis dan merangkum sesuatu berdasarkan fakta yang diambil dari sumber-sumber terpercaya seperti jurnal, paper, skripsi atau tugas akhir yang sudah ada maupun buku yang berkaitan, tanpa mengurangi kutipan tersebut. Untuk menghindari plagiarisme.

Untuk mendeteksi *typographical error* pada teks dibutuhkan suatu sistem yang disebut *spelling checker*. *Spelling checker* ini mempunyai fungsi yaitu mencari kata yang mengalami kesalahan ejaan berdasarkan kata korpus yang digunakan terhadap sistem serta berfungsi untuk memberikan saran kata yang dilakukan dengan algoritma yang juga digunakan oleh aplikasi. Sementara pengoreksian kata yang terdapat kesalahan ejaan (*Spelling Correction*) adalah sistem yang dibuat untuk mendeteksi kesalahan ejaan dan memperbaikinya.

Untuk membantu siswa maupun mahasiswa dalam mengatasi masalah kesalahan penulisan dokumen tersebut dibuat suatu aplikasi yang dapat melakukan pengecekan kesalahan penulisan atau pengetikan teks (*typographical error*) beserta imbuhan kata berbahasa indonesia menggunakan metode *Levenshtein Distance*. Metode ini digunakan untuk menentukan kandidat kata untuk setiap *typographical error* yang teridentifikasi oleh sistem. Diharapkan sistem ini dapat membantu memberikan gambaran yang cukup baik terhadap koreksi *error* pada dokumen teks berbahasa Indonesia berdasarkan hasil pada sistem tersebut. Dengan adanya sistem ini, diharapkan dapat membantu mahasiswa dalam mengatasi masalah kesalahan penulisan atau pengetikan teks.

1.2. Permasalahan

1. Bagaimana cara membangun sistem pendekripsi kesalahan kata / *spell checker* pada kata berbahasa Indonesia menggunakan Algoritma *Levenshtein Distance*?
2. Bagaimana tingkat akurasi yang didapatkan jika menggunakan Algoritma *Levenshtein Distance* terhadap analisis *spell checker* pada tulisan yang tipo?

1.3. Tujuan

1. Membuat serta mengetahui sistem pendekripsi kesalahan kata / *spell checker* pada kata berbahasa Indonesia menggunakan Algoritma *Levenshtein Distance*.
2. Dapat mengetahui tingkat akurasi *spell checker* dengan menggunakan Algoritma *Levenshtein Distance*.

1.4. Manfaat

Proyek ini dapat memberikan manfaat bagi banyak pihak. Proyek ini dapat membantu meningkatkan kualitas ilmu pengetahuan dari sisi ejaan tulisan berbahasa Indonesia, karena masyarakat/mahasiswa/pengguna dapat menjadi lebih mudah untuk mengoreksi kesalahan yang terdapat pada tulisan mereka secara cepat dan efektif. Pagi penulis proyek ini dapat menjadi sarana pembelajaran terutama dibidang *text mining* terlebih lagi proyek ini dapat melatih pemahaman penulis tentang analisis *spell checker*. Sedangkan manfaat bagi masyarakat yaitu mereka Selain itu mereka juga dapat mendapat ilmu baru tentang analisis *spell checker*.

BAB II : TINJAUAN PUSTAKA

2.1. Teori Penunjang

a. *Text Mining*

Text Mining atau sering disebut Pemrosesan Teks merupakan proses penambangan data yang berupa teks dimana sumber data biasanya didapatkan dari dokumen, dan tujuannya adalah mencari kata-kata yang memberikan penjelasan isi dari dokumen sehingga dapat dilakukan analisis keterkaitan antar dokumen (Baskoro, 2016).

Text mining perlu untuk mempersiapkan data dengan cara mengubah teks menjadi lebih terstruktur dengan beberapa tahap. Salah satu tahapan tersebut adalah *preprocessing* yang meliputi *case folding*, *tokenizing*, *filtering* dan *stemming*. Pada tahapan ini, sistem melakukan seleksi data yang diproses pada setiap dokumen yang ada.

b. *Spelling Checker*

Spelling Checker adalah proses mengidentifikasi/mendeteksi dan menangani *error* dalam suatu kata. Terdapat dua metode utama yang digunakan untuk membangun aplikasi *spelling checker* yaitu identifikasi (*error detection*) dan koreksi (*error correction*). Selain itu, *spelling checker* dibagi menjadi dua tipe yaitu *non-word error spell checker* dan *real-word spell checker*. *Non-word error spell checker* menangani kata-kata salah ejaan yang terbentuk karena kesalahan ketik, sedangkan *real-word error spell checker* mengutamakan menangani kata-kata pengganti kata yang *error* pada kalimat (Soleh dan Purwarianti, 2011).

c. Algoritma *Levenshtein Distance*

Levenshtein-distance atau *edit-distance* adalah algoritma yang ditemukan oleh Vladimir Levenshtein, seorang ilmuwan Rusia, pada tahun 1965. Algoritma ini berguna untuk memeriksa kemiripan dari dua buah string yang umumnya ditemukan pada aplikasi-aplikasi pengecekan suatu ejaan. *Levenshtein Distance* melibatkan operasi penghapusan, penyisipan, dan penukaran. Setiap operasi yang digunakan dalam mengubah satu string ke string lain tersebut bernilai 1 (satu), tetapi apabila tidak dibutuhkan pengubahan maka operasi bernilai 0 (nol).

Levenshtein Distance antara dua string ditentukan berdasarkan jumlah minimum pengeditan yang diperlukan untuk melakukan transformasi dari satu bentuk string ke bentuk string yang lain. Notasi yang digunakan untuk *Levenshtein Distance* adalah $LD(s,t)$ dengan s yaitu sumber dan t adalah target. Misalnya, jika *source* string (s) adalah “tihun” dan target string (t) adalah “tahun” maka nilai *Levenshtein Distance* adalah 1, dalam hal ini berarti dibutuhkan sebuah operasi yaitu *substitution* untuk mengubah *source* string (s) menjadi sama dengan target string (t).

Operasi dilakukan dengan cara menukar posisi karakter yang berdekatan dan menemukan kata yang sama dalam *dictionary* (Naradhipa et al., 2011). Secara matematis, *Levenshtein Distance* antara dua string, misal string sumber a dan string

target b (panjang $|a|$ dan $|b|$) dengan $\text{lev}_{a,b}(|a|, |b|)$ pada indeks i dan j dimana telah dijelaskan pada persamaan pada gambar berikut:

$$\text{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \min(i,j)=0, \\ \min \begin{cases} \text{lev}_{a,b}(i-1,j)+1 \\ \text{lev}_{a,b}(i,j-1)+1 \\ \text{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{lainnya} \end{cases}$$

Gambar 1. Operasi Levenshtein Distance

d. JupyterLab

Perangkat lunak yang digunakan adalah Jupyterlab. Pengguna dapat menjalankan notebooks, mengedit file teks, menggunakan remote terminal. Di sini, pengguna tidak perlu menangani alat yang tersebar, JupyterLab sudah menempatkan sebagian besar dari instrumen yang diperlukan bersama-sama dan menawarkan lingkungan kerja yang dapat disesuaikan dan fleksibel. JupyterLab berfokus pada komputasi interaktif dan eksplorasi di berbagai bahasa pemrograman, dan membuat banyak fitur tetap ditemukan dalam IDE tradisional. (A Erokhin dan A Zarochentsev, 2020)

e. Python

Python diciptakan oleh Guido van Rossum di Belanda pada tahun 1990 dan namanya diambil dari acara televisi kesukaan Guido Monty Python's Flying Circus. Van Rossum mengembangkan Python sebagai hobi, kemudian Python menjadi bahasa pemrograman yang dipakai secara luas dalam industri dan pendidikan karena sederhana, ringkas, sintaks intuitif dan memiliki pustaka yang luas

Python adalah pemrograman berorientasi objek (OOP). Data dalam Python adalah sebuah objek yang dibuat dari kelas (class). Pemrograman berorientasi objek merupakan alat ampuh untuk mengembangkan perangkat lunak yang dapat digunakan kembali. (Schuerer dan Maufrais, 2010).

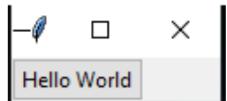
f. Tkinter

Tkinter (Tk Interface) merupakan GUI (*Graphical User Interface*) untuk pembuatan Interface pada Python. Tkinter mendukung kumpulan widget Tk yang mendukung sebagian besar kebutuhan aplikasi. Tkinter sendiri adalah bentuk OOP dari TCL/TK. TCL (Tool Command Language) yang merupakan sebuah bahasa pemrograman dan TK adalah library yang digunakan oleh TCL untuk membuat model berbasis GUI (Hariyanto, 2017).

Untuk melihat output tampilan yang dihasilkan, kita mencoba membuat tampilan “hello world” dengan menggunakan Tkinter.

```
from tkinter import *
from tkinter import ttk
root = Tk()
ttk.Button(root, text="Hello World").grid()
root.mainloop()
```

output:



Representasi

2.2. Penelitian Terkait

a. Typographical error

Kesalahan umum yang sering dilakukan mahasiswa dalam hal menulis adalah *Typo* atau *Tipo*. Sejak awal, mahasiswa akan diberi tugas berupa laporan praktikum, makalah ataupun karya ilmiah pada beberapa mata kuliah untuk membiasakan dan melatih kemampuan menulisnya. Namun dalam hal menulis ini tidak jarang ditemukan mahasiswa cukup sering dianjurkan untuk melakukan banyak revisi dikarenakan ditemukan kesalahan penulisan yang kurang sesuai dengan kaidah Ejaan Yang Disempurnakan (EYD) (Londo, 2020). Hasil dan insight yang didapat dari penelitian ini antara lain :

- 1) Algoritma Jaro-Winkler *distance* hanya bisa melakukan pengecekan kata dengan menggunakan dataset yang telah diunggah dari Kamus Besar Bahasa Indonesia.
- 2) Waktu eksekusi model tergantung dari jumlah dataset dan data yang digunakan, semakin banyak semakin lama waktu eksekusinya.
- 3) Pada implementasi Aplikasi Web akan memperlihatkan kata-kata yang salah dengan diberi tanda warna biru.

b. Mikrotext pada Twitter

Twitter merupakan media sosial berbentuk *microblogging*. *Microblogging* adalah media sosial yang membatasi atau memiliki maksimal karakter untuk ditulis, akibat dari *microblogging* ini tidak sedikit pengguna Twitter melakukan pemendekan atau penyingkatan kata yang dikenal dengan “mikroteks”. Umumnya penyingkatan dilakukan dengan menghilangkan beberapa huruf vokal yang menyebabkan sedikit terjadi keambiguan pada teks. Pada penelitian ini dilakukan normalisasi mikroteks dengan penggunaan *Dictionary-Based* dan metode algoritma *Longest Common Subsequence* (LCS) (Afzalurrahmah, 2019)

Hasil dan insight yang didapat dari penelitian ini antara lain :

- 1) Algoritma *Longest Common Subsequence* dapat menormalisasi data *tweet* dengan hasil pengujian akurasi sebesar 94%, presisi 95%, *recall* 97% dan *f-score* 0,96 dengan pengujian data sebanyak 400 *tweets*.
- 2) Term Frequency yang digunakan efektif membantu algoritma LCS pada pemberian hasil normalisasi menjadi lebih optimal.

c. Penapis Ejaan Otomatis

Sebuah penelitian menemukan kesalahan berulang pada teks berbahasa Indonesia disebabkan oleh ketidaktepatan tata bahasa. Ketidaktepatan yang dimaksud yaitu sebuah kata berimbuhan ataupun kata baku belum sesuai dengan kaidah Pedoman Umum Ejaan Bahasa Indonesia (PUEBI) dan Kamus Besar Bahasa Indonesia (KBBI).

Pada penelitian ini metode yang digunakan adalah dengan pendekatan kuantitatif seperti melakukan observasi dan wawancara untuk kebutuhan pengumpulan data analisis. Hasil dari pendekatan kuantitatif ini juga akan dijadikan perbandingan untuk mengetahui perbedaan pada penggunaan aplikasi U-Tapis.

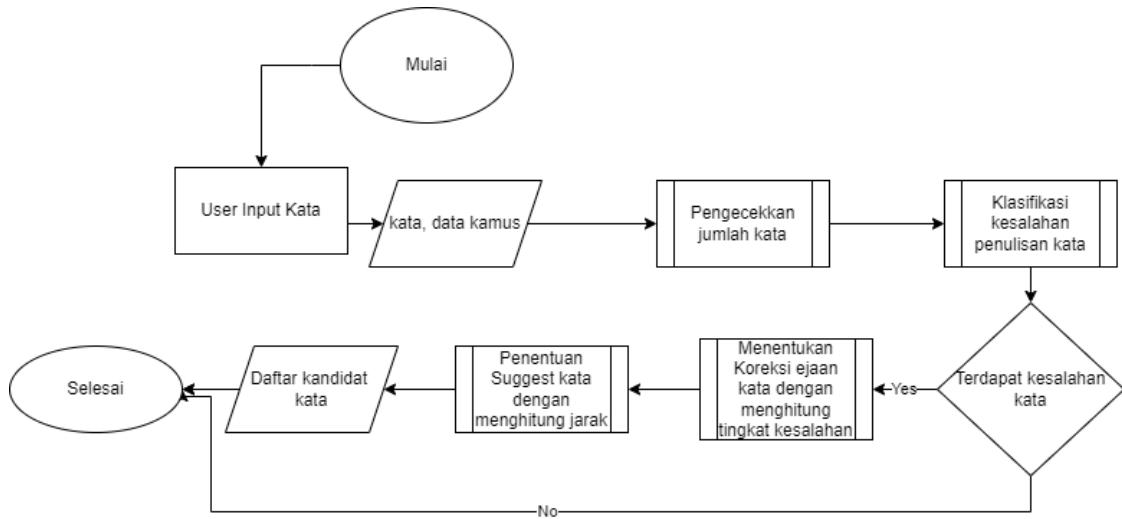
Metode yang digunakan untuk aplikasi U-Tapis sendiri adalah analisis teks yang disebut juga dengan text mining. Analisis ini melibatkan proses analisis statistik, linguistik komputasi, dan kecerdasan buatan (Mediyawati, 2021).

Hasil dan insight yang didapat dari penelitian ini antara lain :

- 1) Tingkat efektivitas penggunaan U-Tapis untuk mengidentifikasi kesalahan adalah sebesar 92,31%.
- 2) Aplikasi U-Tapis juga dapat digunakan untuk pembelajaran peningkatan kompetensi bahasa siswa maupun guru.

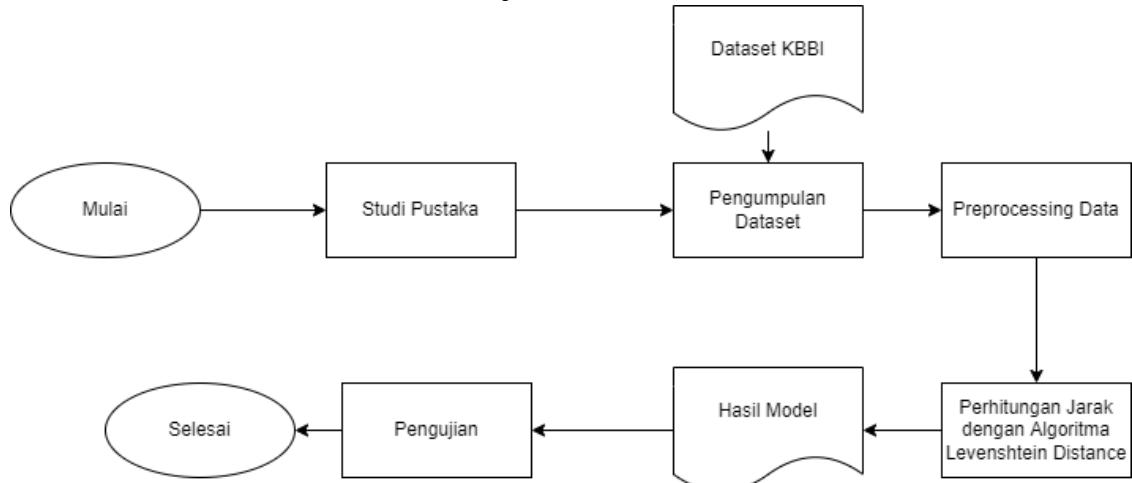
BAB III: METODOLOGI PENELITIAN

3.1. Desain Sistem Secara Umum



Gambar 2. Flowchart Desain Sistem Ejaan Kata

3.2. Arsitektur Sistem dan Prosedur Kerja



3.2.2. Pengumpulan Dataset

Mengambil dataset kata Bahasa Indonesia yang sesuai dengan kata baku menurut Kamus Besar Bahasa Indonesia (KBBI). Dataset bisa di download melalui link dibawa ini :

3.2.3. Import Library dan Module

Mengimpor Library yang dibutuhkan untuk membantu proses pengolahan inputan kalimat. Library yang digunakan pada model spell checker ini antara lain:

```
from nltk.metrics.distance import edit_distance
from time import sleep
import pandas as pd
import re
```

- Pandas

Digunakan untuk membaca data korpus menggunakan fungsi pd.read_csv()

- Pyimage

Untuk memproses gambar yang digunakan

- Tkinter

Digunakan untuk membuat *Graphical User Interface* (GUI) dari model

- nltk.metrics.distance

Untuk mengukur kesamaan string input dengan string pada korpus

3.2.3. Proses Cek Kalimat

1. Input Kalimat

Menginput atau memasukkan kalimat yang akan diperiksa letak kesalahannya.

Pada sistem yang dibuat, penginputan kata menggunakan fungsi final() yang didefinisikan seperti pada gambar di bawah

```
def final():
    text_input = inputtxt.get('1.0', 'end-1c')
    data_read = get_data.getData()
```

Gambar 6. Script Input Kalimat

2. Kalkulasi Input

Akan dilakukan kalkulasi untuk memeriksa kata atau kalimat yang diinput dengan menggunakan perhitungan *Levenshtein Distance*. Tujuan dari kalkulasi input adalah untuk memeriksa huruf pada kalimat yang dimasukkan terdapat salah penulisan atau tidak, sehingga ketika ditemukan adanya ketidaksamaan perhitungan pada salah satu matriks, akan diubah sesuai dengan perhitungan. Perhitungan nilai edit *distance* dilakukan dalam mencari nilai yang paling minimum untuk mendapatkan daftar kata pada kamus sesuai *typographical error*. Kemudian ditentukannya rangking kandidat kata yang akan menjadi hasil output pada sistem. Untuk menghitung jarak dengan rumus yang dimasukkan ke dalam fungsi tingkat kesalahan :

$$\text{tingkat_kesalahan} = \text{leven} / \text{panjang_kata} * 100$$

Untuk menghitung jaraknya tanpa menggunakan proses rekursif, digunakan matriks $(n + 1) \times (m + 1)$ di mana n adalah panjang string s1 dan m adalah panjang string s2. Berikut dua string yang akan digunakan sebagai contoh: MAJAN PSGI dengan MAKAN PAGI

Jika kita melihat sekilas, kedua string tersebut sama. Berarti untuk mengubah string MAJAN PSGI menjadi MAKAN PAGI diperlukan 2 operasi, yaitu:

1. Mensubstitusikan J dengan K MAJAN PSGI -> MAKAN PSGI
2. Mensubstitusikan S dengan A MAKAN PSGI -> MAKAN PAGI

	M	A	J	A	N	P	S	G	I		
	0	1	2	3	4	5	6	7	8	9	10
M	1										
A	2										
K	3										
A	4										
N	5										
	6										
P	7										
A	8										
G	9										
I	10										

Representasi matriks

	M	A	J	A	N	P	S	G	I		
	0	1	2	3	4	5	6	7	8	9	10
M	1	0	1	2	3	4	5	6	7	8	9
A	2	1	0	1	2	3	4	5	6	7	8
K	3	2	1	1	2	3	4	5	6	7	8
A	4	3	2	2	1	2	3	4	5	6	7
N	5	4	3	3	2	1	2	3	4	5	6
	6	5	4	4	3	2	1	2	3	4	5
P	7	6	5	5	4	3	2	1	2	3	4
A	8	7	6	6	5	4	3	2	2	3	4
G	9	8	7	7	6	5	4	3	3	2	1
I	10	9	8	8	7	6	5	4	4	1	2

Hasil representasi matriks

Elemen terakhir (yang paling kanan bawah) adalah elemen yang nilainya menyatakan jarak kedua string yang dibandingkan. Elemen terakhir (yang paling kanan bawah) adalah elemen yang nilainya menyatakan jarak kedua string yang dibandingkan.

1. Langkah awal Algoritma *Levenshtein Distance* adalah menghitung jarak string sumber pertama “makan pagi” dengan string target pertama yaitu “majan psgi”. Perhitungan matriks dimulai dengan inisiasi urutan karakter pada masing-masing string.
2. Diketahui bahwa string sumber “makan pagi” memiliki 10 (sepuluh) karakter dan string target “majan psgi” memiliki 10 (sepuluh) karakter. Selanjutnya karakter ke-1 pada masing-masing string dibandingkan dan diketahui bahwa isi karakter ke-1 pada masing-masing string sama, maka nilai matriks yang diberikan sesuai dengan Persamaan (4) yaitu $D(1,1) = D(1 - 1,1 - 1)$, $sj = ti$. Jadi nilai matriks yang diberikan pada $D(1,1) = D(0,0)$ yang bernilai 0. Kemudian nilai matriks pada $D(1,1)$ diisi seperti pada gambar diatas.

3. Selanjutnya perhitungan jarak dilakukan pada karakter ke-1 string sumber dengan karakter ke-2 pada string target dan diketahui bahwa dibutuhkan operasi penyisipan karakter “a” pada string sumber, maka nilai yang diberikan sesuai dengan rumus operasi penyisipan pada Rumus $2D(1,2) = D(1,2 - 1) + 1$. Jadi nilai yang diberikan pada $D(1,2) = D(1,1) + 1$ yang bernilai $D(1,2) = 0 + 1 = 1$. Kemudian nilai matriks pada $D(1,2)$ diisi dengan nilai 1
4. Selanjutnya perhitungan jarak dilakukan pada karakter ke-1 string sumber dengan string target sampai dengan karakter ke-3 dan diketahui bahwa dibutuhkan operasi penyisipan karakter “j” dan “k” pada string sumber, maka nilai yang diberikan sesuai dengan rumus operasi penyisipan pada Rumus $2D(1,3) = D(1,3 - 1) + 1$. Jadi nilai yang diberikan pada $D(1,3) = D(1,2) + 1$ yang bernilai $D(1,3) = 1 + 1 = 2$. Kemudian nilai matriks pada $D(1,3)$ diisi dengan nilai 2
5. Selanjutnya perhitungan jarak dilakukan pada karakter ke-1 string sumber dengan string target sampai dengan karakter ke-4 dan diketahui bahwa dibutuhkan operasi penyisipan karakter “a”, “k”, dan “a” pada string sumber, maka nilai yang diberikan sesuai dengan rumus operasi penyisipan pada Persamaan (2) $D(1,4) = D(1,4 - 1) + 1$. Jadi nilai yang diberikan pada $D(1,4) = D(1,3) + 1$ yang bernilai $D(1,3) = 2 + 1 = 3$. Kemudian nilai matriks pada $D(1,4)$ diisi dengan nilai 3
6. Selanjutnya perhitungan jarak dilakukan pada karakter ke-1 string sumber dengan string target sampai dengan karakter ke-5 dan diketahui bahwa dibutuhkan operasi penyisipan karakter “a”, “k”, “a”, dan “n” pada string sumber, maka nilai yang diberikan sesuai dengan rumus operasi penyisipan pada Persamaan (2) $D(1,5) = D(1,5 - 1) + 1$. Jadi nilai yang diberikan pada $D(1,5) = D(1,4) + 1$ yang bernilai $D(1,5) = 3 + 1 = 4$. Kemudian nilai matriks pada $D(1,5)$ diisi dengan nilai 4
7. Selanjutnya perhitungan jarak dilakukan pada karakter ke-1 string sumber dengan string target sampai dengan karakter ke-6 dan diketahui bahwa dibutuhkan operasi penyisipan karakter “a”, “j”, “a”, “n”, dan “p” pada string sumber, maka nilai yang diberikan sesuai dengan rumus operasi penyisipan pada Persamaan (2) $D(1,6) = D(1,6 - 1) + 1$. Jadi nilai yang diberikan pada $D(1,6) = D(1,5) + 1$ yang bernilai $D(1,6) = 4 + 1 = 5$. Kemudian nilai matriks pada $D(1,6)$ diisi dengan nilai 5
8. Selanjutnya perhitungan jarak dilakukan pada karakter ke-1 string sumber dengan string target sampai dengan karakter ke-6 dan diketahui bahwa dibutuhkan operasi penyisipan karakter “a”, “j”, “a”, “n”, “p”, dan “a” pada string sumber, maka nilai yang diberikan sesuai dengan rumus operasi penyisipan pada Persamaan (2) $D(1,5) = D(1,5 - 1) + 1$. Jadi nilai yang diberikan pada $D(1,5) = D(1,4) + 1$ yang

bernilai $D(1,5) = 5 + 1 = 6$. Kemudian nilai matriks pada $D(1,6)$ diisi dengan nilai 6

9. Selanjutnya perhitungan jarak dilakukan pada karakter ke-1 string sumber dengan string target sampai dengan karakter ke-6 dan diketahui bahwa dibutuhkan operasi penyisipan karakter “a”, “j”, “a”, “n”, “p”, “a”, dan “g” pada string sumber, maka nilai yang diberikan sesuai dengan rumus operasi penyisipan pada Persamaan (2) $D(1,6) = D(1,6 - 1) + 1$. Jadi nilai yang diberikan pada $D(1,6) = D(1,5) + 1$ yang bernilai $D(1,6) = 6 + 1 = 7$. Kemudian nilai matriks pada $D(1,7)$ diisi dengan nilai 7
10. Selanjutnya perhitungan jarak dilakukan pada karakter ke-1 string sumber dengan string target sampai dengan karakter ke-6 dan diketahui bahwa dibutuhkan operasi penyisipan karakter “a”, “j”, “a”, “n”, “p”, “a”, “g” dan “i” pada string sumber, maka nilai yang diberikan sesuai dengan rumus operasi penyisipan pada Persamaan (2) $D(1,7) = D(1,7 - 1) + 1$. Jadi nilai yang diberikan pada $D(1,7) = D(1,6) + 1$ yang bernilai $D(1,7) = 7 + 1 = 8$. Kemudian nilai matriks pada $D(1,6)$ diisi dengan nilai 8
11. Perhitungan jarak *Levenshtein Distance* selanjutnya berjalan sampai semua nilai pada matriks terisi. Jarak *Levenshtein Distance* adalah nilai yang terdapat di bawah-kanan matriks, dan pada kasus string sumber “makan pagi” dan string target “majan psgi” berada di $D(7,6)$. Setelah dilakukan seluruh perhitungan matriks diketahui hasil dari perhitungan jarak antara string sumber “makan pagi” dan string target “majan psgi” adalah 2 (dua) seperti yang digambarkan pada matriks

Adapun matriks 2 (dua) dimensi digunakan dalam perhitungan nilai jarak *Levenshtein Distance*. Isian nilai pada matriks tersebut adalah jumlah operasi penghapusan, penyisipan dan penukaran yang dibutuhkan dalam mengubah string sumber ke string target. Rumus operasi penghapusan, penyisipan, dan penukaran karakter yang digunakan untuk mengisi nilai matriks adalah sebagai berikut :

$$D(s, t) = \min D(s - 1, t) + 1 \text{ (Penghapusan)} \quad (1)$$

$$D(s, t) = \min D(s, t - 1) + 1 \text{ (Penyisipan)} \quad (2)$$

$$D(s, t) = \min D(s - 1, t - 1) + 1, sj \neq ti \text{ (Penukaran)} \quad (3)$$

$$D(s, t) = \min D(s - 1, t - 1), sj = ti \text{ (Tidak ada perubahan)} \quad (4)$$

s = String Sumber

t = String Target

D = Jarak *Levenshtein Distance*

$s(j)$ = Karakter String Sumber ke- j

$t(i)$ = Karakter String Target ke- i

Untuk Pemrosesan input yang dilakukan pada sistem, menggunakan fungsi cek(kata, data, callback) yang di dalamnya mendefinisikan pemrosesan input kata dengan perhitungan *Levenshtein Distance* bawaan dari Python.

```
class cek_kata:
    def cek(kata, data, callback):
        hasil = []
        arr_kata = kata.split()
        for i in range(len(arr_kata)):
            if arr_kata[i] in data:
                hasil.append(arr_kata[i])
            else:
                for x in data:
                    leven = edit_distance(arr_kata[i].lower(), x)
                    kesalahan = callback(leven, len(x))
                    if(kesalahan <= 35):
                        hasil.append(x)
                        break
```

Gambar 6. Script Kalkulasi Cek Input

Setelah dilakukan perhitungan jarak yang mengacu dari beberapa kata yang memungkinkan mirip, selanjutnya dilakukan perhitungan persentase dari kandidat kata yang akan dijadikan Output dari percobaan. Menggunakan fungsi persentase(leven, len_kata_benar) yang mendefinisikan pencarian nilai yang paling minimum dari daftar acuan kata pada korpus. Semakin minim tingkat kesalahannya, semakin akurat kata pembenarannya.

```
class perhitungan:
    def presentase(leven, len_kata_benar):
        tingkat_kesalahan = leven / len_kata_benar * 100
        return round(tingkat_kesalahan)
```

Gambar 7. Script Kalkulasi Persentase Kata

3. Output

Hasil dari perhitungan *Levenshtein Distance* akan ditampilkan menggunakan fungsi “tampil_hasil”.

```
class tampil_hasil:
    def tampilkan(arr_hasil, kata):

        print("===== OUTPUT =====")
        print("")
        print("Kata : ", kata)
        print("Perbaikan : +" ".join(arr_hasil))

        print("")
        print("")
```

Gambar 8. Script tampil_hasil

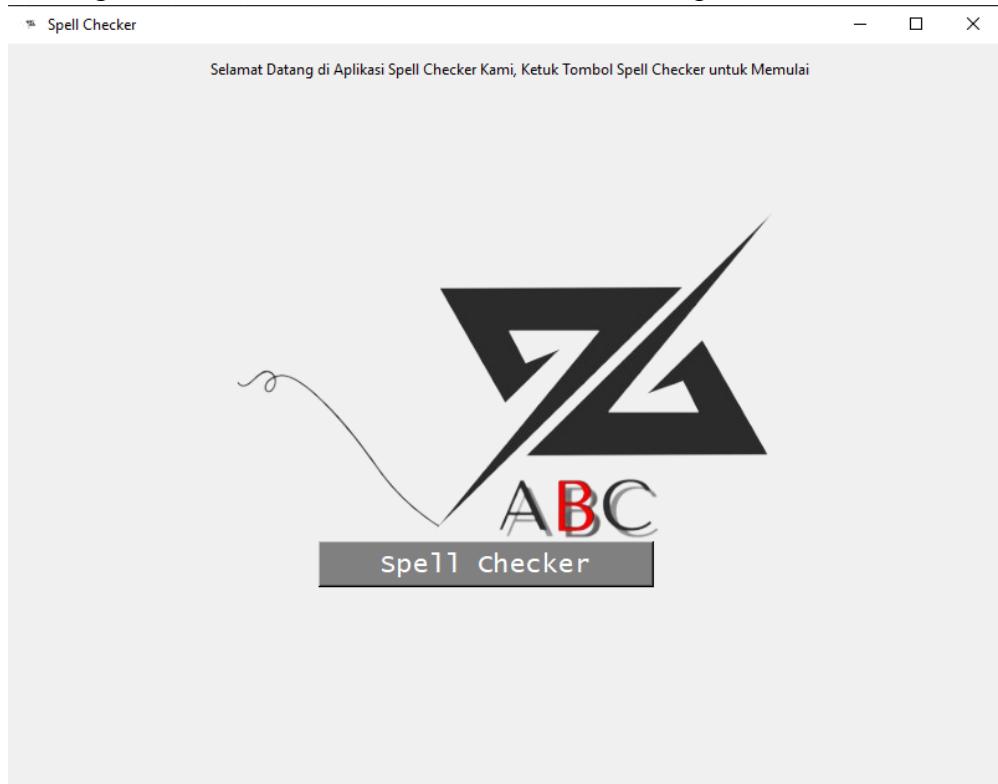
3.2.4. Pengujian Model

Dilakukan pengecekan apakah model berfungsi mendeteksi sebagai *spell checker*. Dengan melakukan uji coba memasukkan data uji testing sebanyak 5, yang memiliki total kata dan ... kesalahan ejaan. nantinya akan dijadikan

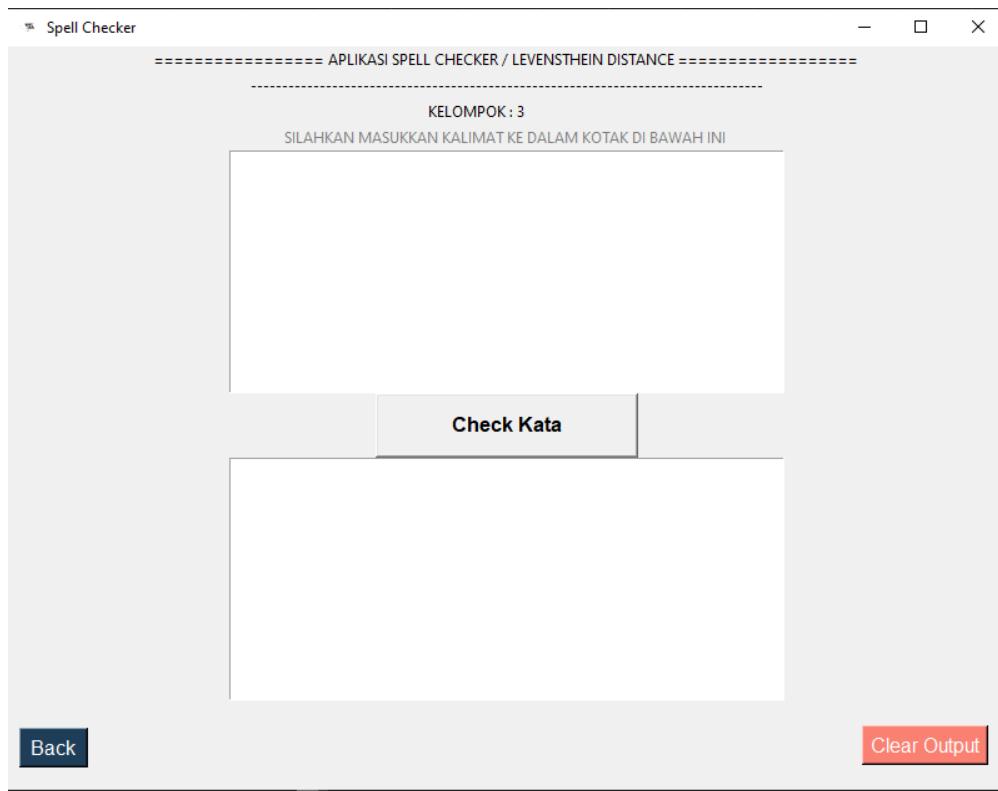
perbandingan dan penilaian terhadap model apakah bekerja dengan baik atau terjadi *error*.

3.2.5. Membuat Model Berbasis GUI Tkinter

Model yang sudah berfungsi dibuat menjadi *Graphical User Interface* (GUI) menggunakan library dari Tkinter yang telah diimporkan. Tujuan dijadikan GUI adalah agar lebih leluasa dan mudah dalam melakukan percobaan dataset.



Gambar(). Homepage GUI Spell Checker



Gambar 4. Spell Checker Page

3.3. Proses Pembuatan

Perhitungan nilai edit *distance* dilakukan dalam mencari nilai yang paling minimum untuk mendapatkan daftar kata pada kamus sesuai *typographical error*. Kemudian ditentukannya rangking kandidat kata yang akan menjadi hasil output pada sistem. Untuk menghitung jarak dengan rumus yang dimasukkan ke dalam fungsi tingkat kesalahan :

$$\text{tingkat_kesalahan} = \text{leven} / \text{panjang_kata} * 100$$

Untuk menghitung jaraknya tanpa menggunakan proses rekursif, digunakan matriks $(n + 1) \times (m + 1)$ di mana n adalah panjang string s1 dan m adalah panjang string s2. Berikut dua string yang akan digunakan sebagai contoh: MAJAN PSGI dengan MAKAN PAGI

Jika kita melihat sekilas, kedua string tersebut sama. Berarti untuk mengubah string MAJAN PSGI menjadi MAKAN PAGI diperlukan 2 operasi, yaitu:

3. Mensubstitusikan J dengan K MAJAN PSGI -> MAKAN PSGI
4. Mensubstitusikan S dengan A MAKAN PSGI -> MAKAN PAGI

	M	A	J	A	N	P	S	G	I		
	0	1	2	3	4	5	6	7	8	9	10
M	1										
A	2										
K	3										
A	4										
N	5										
	6										
P	7										
A	8										
G	9										
I	10										

Representasi matriks

	M	A	J	A	N	P	S	G	I		
	0	1	2	3	4	5	6	7	8	9	10
M	1	0	1	2	3	4	5	6	7	8	9
A	2	1	0	1	2	3	4	5	6	7	8
K	3	2	1	1	2	3	4	5	6	7	8
A	4	3	2	2	1	2	3	4	5	6	7
N	5	4	3	3	2	1	2	3	4	5	6
	6	5	4	4	3	2	1	2	3	4	5
P	7	6	5	5	4	3	2	1	2	3	4
A	8	7	6	6	5	4	3	2	2	3	4
G	9	8	7	7	6	5	4	3	3	2	1
I	10	9	8	8	7	6	5	4	4	1	2

Hasil representasi matriks

RONALDINHO ROLANDO

Jika kita melihat sekilas, kedua string tersebut memiliki jarak 6. Berarti untuk mengubah string RONALDINHO menjadi ROLANDO diperlukan 6 operasi, yaitu:

5. Mensubstitusikan N dengan L RONALDINHO -> ROLALDINHO
6. Mensubstitusikan L dengan N ROLALDINHO -> ROLANDINHO
7. Mensubstitusikan I dengan O ROLANDINHO -> ROLANDONHO
8. Menghapus O ROLANDONHO -> ROLANDONH
9. Menghapus H ROLANDONH -> ROLANDON
10. Menghapus N ROLANDON -> ROLANDO

Dengan menggunakan representasi matriks dapat ditunjukkan tabel berikut:

	R	O	N	A	L	D	I	N	H	O	
R	0	1	2	3	4	5	6	7	8	9	10
O	1										
L	2										
A	3										
N	4										
D	5										
O	6										
O	7										

Gambar 3. Representasi matriks

Pada tabel ini, elemen baris 1 kolom 1 ($M[1,1]$) adalah jumlah operasi yang diperlukan untuk mengubah substring dari kata ROLANDO yang diambil mulai dari karakter awal sebanyak 1 (R) ke substring dari kata RONALDINHO yang diambil mulai dari karakter awal sebanyak 1 (R). Sementara elemen $M[3,5]$ adalah jumlah operasi antara ROL (substring yang diambil mulai dari karakter awal sebanyak 3) dengan RONAL (substring yang diambil mulai dari karakter awal sebanyak 5). Berarti elemen $M[p,q]$ adalah jumlah operasi antara substring kata pertama yang diambil mulai dari awal sebanyak p dengan substring kata kedua yang diambil dari awal sebanyak q. Sehingga dengan peraturan ini matriks dapat diisi, menghasilkan :

	R	O	N	A	L	D	I	N	H	O	
R	0	1	2	3	4	5	6	7	8	9	10
O	1	0	1	2	3	4	5	6	7	8	9
L	2	1	0	1	2	3	4	5	6	7	8
A	3	2	1	1	2	3	4	5	6	7	8
N	4	3	2	2	1	2	3	4	5	6	7
D	5	4	3	3	2	2	3	4	5	6	7
O	6	5	4	4	3	3	2	3	4	5	6
O	7	6	5	5	4	4	3	3	4	5	6

Gambar 4. Hasil representasi matriks

Elemen terakhir (yang paling kanan bawah) adalah elemen yang nilainya menyatakan jarak kedua string yang dibandingkan.

Fungsi Algoritma *Levenshtein Distance* diuji pada kasus tweet berisi “@PemkotBandung bandung mcet” sebagai string sumber dan kata kunci banjir dan kebanjiran yang terdapat pada kategori isu kebanjiran, kata kunci kebakaran yang terdapat pada kategori isu kebakaran, kata kunci kecelakaan dan tabrakan yang terdapat pada kategori isu kecelakaan, kata kunci kemacetan dan macet yang terdapat pada kategori isu kemacetan, kata kunci pencurian dan perampokan yang terdapat pada kategori isu kriminal, kata kunci pelanggaran dan melanggar yang terdapat pada kategori isu pelanggaran, dan kata kunci pengemis yang terdapat pada kategori isu pengemis sebagai string target. Berdasarkan contoh kasus tersebut, setiap kata pada

tweet yaitu “bandung” dan “mcet” dihitung jaraknya dengan seluruh kata kunci. Sedangkan kata “@PemkotBandung” tidak dihitung karena kata tersebut tidak diidentifikasi sebagai isi dari tweet oleh Twitter, melainkan mention. Langkah awal Algoritma *Levenshtein Distance* adalah menghitung jarak string sumber pertama “bandung” dengan string target pertama yaitu “banjir”. Perhitungan matriks dimulai dengan inisiasi urutan karakter pada masing-masing string seperti digambarkan pada Gambar 5.

		String Target					
		b	a	n	j	i	r
String Sumber	0	1	2	3	4	5	6
	b	1					
	a	2					
	n	3					
	d	4					
	u	5					
	n	6					
	g	7					

Gambar 5. Inisiasi urutan karakter

Pada Gambar 5 diketahui bahwa string sumber “bandung” memiliki 7 (tujuh) karakter dan string target “banjir” memiliki 6 (enam) karakter. Selanjutnya karakter ke-1 pada masing-masing string dibandingkan dan diketahui bahwa isi karakter ke-1 pada masing-masing string sama, maka nilai matriks yang diberikan sesuai dengan Persamaan (4) yaitu $D(1,1) = D(1 - 1,1 - 1)$, $s_j = t_i$. Jadi nilai matriks yang diberikan pada $D(1,1) = D(0,0)$ yang bernilai 0. Kemudian nilai matriks pada $D(1,1)$ diisi seperti yang digambarkan pada Gambar 6.

Gambar 6. Jarak $D(1,1)$

Selanjutnya perhitungan jarak dilakukan pada karakter ke-1 string sumber dengan karakter ke-2 pada string target dan diketahui bahwa dibutuhkan operasi penyisipan karakter “a” pada string sumber, maka nilai yang diberikan sesuai dengan rumus operasi penyisipan pada Rumus $2D(1,2) = D(1,2 - 1) + 1$. Jadi nilai yang diberikan pada $D(1,2) = D(1,1) + 1$ yang bernilai $D(1,2) = 0 + 1 = 1$. Kemudian nilai matriks pada $D(1,2)$ diisi dengan nilai 1 seperti yang digambarkan pada Gambar 7.

		String Target					
		b	a	n	j	i	r
String Sumber	0	1	2	3	4	5	6
	b	1	0	1			
	a	2					
	n	3					
	d	4					
	u	5					
	n	6					
	g	7					

Gambar 7. Jarak $D(1,2)$

Selanjutnya perhitungan jarak dilakukan pada karakter ke-1 string sumber dengan string target sampai dengan karakter ke-3 dan diketahui bahwa dibutuhkan operasi penyisipan karakter “a” dan “n” pada string sumber, maka nilai yang diberikan sesuai dengan rumus operasi penyisipan pada Rumus $2D(1,3) = D(1,3 - 1) + 1$. Jadi nilai yang diberikan pada $D(1,3) = D(1,2) + 1$ yang bernilai $D(1,3) = 1 + 1 = 2$. Kemudian nilai matriks pada $D(1,3)$ diisi dengan nilai 2 seperti yang digambarkan pada Gambar 8.

		String Target						
		b	a	n	j	i	r	
String Sumber	b	0	1	2	3	4	5	6
	a	1	0	1	2			
	n		2					
	d			3				
	u				4			
	n					5		
	g						6	
							7	

Gambar 8. Jarak $D(1,3)$

Selanjutnya perhitungan jarak dilakukan pada karakter ke-1 string sumber dengan string target sampai dengan karakter ke-4 dan diketahui bahwa dibutuhkan operasi penyisipan karakter “a”, “n”, dan “j” pada string sumber, maka nilai yang diberikan sesuai dengan rumus operasi penyisipan pada Persamaan (2) $D(1,4) = D(1,4 - 1) + 1$. Jadi nilai yang diberikan pada $D(1,4) = D(1,3) + 1$ yang bernilai $D(1,3) = 2 + 1 = 3$. Kemudian nilai matriks pada $D(1,4)$ diisi dengan nilai 3 seperti yang digambarkan pada Gambar 9.

		String Target						
		b	a	n	j	i	r	
String Sumber	b	0	1	2	3	4	5	6
	a	1	0	1	2	3		
	n		2					
	d			3				
	u				4			
	n					5		
	g						6	
							7	

Gambar 9. Jarak $D(1,4)$

Selanjutnya perhitungan jarak dilakukan pada karakter ke-1 string sumber dengan string target sampai dengan karakter ke-5 dan diketahui bahwa dibutuhkan operasi penyisipan karakter “a”, “n”, “j”, dan “i” pada string sumber, maka nilai yang diberikan sesuai dengan rumus operasi penyisipan pada Persamaan (2) $D(1,5) = D(1,5 - 1) + 1$. Jadi nilai yang diberikan pada $D(1,5) = D(1,4) + 1$ yang bernilai $D(1,5) = 3 + 1 = 4$. Kemudian nilai matriks pada $D(1,5)$ diisi dengan nilai 4 seperti yang digambarkan pada Gambar 10.

		String Target					
		b	a	n	j	i	r
String Sumber	0	1	2	3	4	5	6
	b	1	0	1	2	3	4
	a	2					
	n	3					
	d	4					
	u	5					
	n	6					
	g	7					

Gambar 10. Jarak $D(1, 4)$

Selanjutnya perhitungan jarak dilakukan pada karakter ke-1 string sumber dengan string target sampai dengan karakter ke-6 dan diketahui bahwa dibutuhkan operasi penyisipan karakter “a”, “n”, “j”, “i”, dan “r” pada string sumber, maka nilai yang diberikan sesuai dengan rumus operasi penyisipan pada Persamaan (2) $D(1,6) = D(1,6 - 1) + 1$. Jadi nilai yang diberikan pada $D(1,6) = D(1,5) + 1$ yang bernilai $D(1,6) = 4 + 1 = 5$. Kemudian nilai matriks pada $D(1,6)$ diisi dengan nilai 5 seperti yang digambarkan pada Gambar 11.

		String Target					
		b	a	n	j	i	r
String Sumber	0	1	2	3	4	5	6
	b	1	0	1	2	3	4
	a	2					
	n	3					
	d	4					
	u	5					
	n	6					
	g	7					

Gambar 11. Jarak $D(1, 4)$

Perhitungan jarak *Levenshtein Distance* selanjutnya berjalan sampai semua nilai pada matriks terisi. Jarak *Levenshtein Distance* adalah nilai yang terdapat di bawah-kanan matriks, dan pada kasus string sumber “bandung” dan string target “banjir” berada di $D(7,6)$. Setelah dilakukan seluruh perhitungan matriks diketahui hasil dari perhitungan jarak antara string sumber “bandung” dan string target “banjir” adalah 4 (empat) seperti yang digambarkan pada matriks Gambar 12.

		String Target					
		b	a	n	j	i	r
String Sumber	0	1	2	3	4	5	6
	b	1	0	1	2	3	4
	a	2	1	0	1	2	3
	n	3	2	1	0	1	2
	d	4	3	2	1	1	2
	u	5	4	3	2	2	3
	n	6	5	4	3	3	3
	g	7	6	5	4	4	4

Gambar 12. Jarak $D(7, 6)$

Karena batasan masalah pada penelitian ini adalah jarak *Levenshtein Distance* yang digunakan dalam mengubah kata pada tweet menjadi kata kunci bernilai 0 (nol) sampai 3 (tiga), maka kata “bandung” tidak diubah menjadi “banjir”. Selanjutnya perhitungan dilakukan pada kata kunci kedua sampai dengan kata kunci terakhir sebagai string target, sehingga diketahui masing-masing jarak *Levenshtein Distance*-nya seperti pada Tabel 1.

String Sumber	Kategori	String Target	Jarak
bandung	Kebanjiran	banjir	4
		kebanjiran	7
	Kebakaran	kebakaran	7
		kecelakaan	9
	Kemacetan	tabrakan	7
		kemacetan	8
	Kriminal	macet	6
		pencurian	7
	Pelanggaran	perampokan	9
		pelanggaran	9
		melanggar	7
	Pengemis	pengemis	7

Tabel 1. Jarak Levenshtein Distance dari kata bandung

Berdasarkan Tabel 1 diketahui bahwa jarak *Levenshtein Distance* antara string sumber “bandung” dengan seluruh string target tidak ada yang memiliki jarak 0 (nol) sampai 3 (tiga), maka string sumber “bandung” diabaikan dan perhitungan jarak *Levenshtein Distance* berlanjut pada string sumber kedua, yaitu “mcet”. Perhitungan jarak *Levenshtein Distance* antara string sumber “mcet” dengan seluruh string target sebagai kata kunci pada kategori isu dapat diketahui pada Tabel 2.

String Sumber	Kategori	String Target	Jarak
---------------	----------	---------------	-------

mcet	Kebanjiran	banjir	6
		kebanjiran	10
	Kebakaran	kebakaran	9
	Kecelakaan	kecelakaan	8
		tabrakan	8
	Kemacetan	kemacetan	5
		macet	1
	Kriminal	pencurian	8
		perampokan	9
	Pelanggaran	pelanggaran	11
		melanggar	8
	Pengemis	pengemis	7

Tabel 2. Jarak Levenshtein Distance dari kata mcet

Berdasarkan Tabel 2 diketahui bahwa jarak *Levenshtein Distance* antara string sumber “mcet” dengan string target “macet” memiliki nilai 1 (satu), maka kemudian string sumber “mcet” diubah menjadi string target “macet” dan tweet “@PemkotBandung bandung mcet” dimasukkan ke dalam kategori “kemacetan”.

```

function levDis (s1 : string, s2 : string) : integer
kamus
    i, j, cost : integer
    m : array [0 s1.length, 0 .. s2.length] of integer
algoritma
    for i ← 0 to s1.length do
        for ← 0 to s2.length do
            if i = 0 then
                m [ij] ← j {perbandingan dengan kosong}
            else if j = 0 then
                m [ij] ← i {perbandingan dengan kosong}
            else {implementasi pemrograman dinamis}
                if s1[i] =s2[i] then
                    cost ← 0
                else
                    cost ← 1

                m[i,j] = minimum (
                    m[i-1, j-1] + cost, {substitusi}
                    m[i-1, j] + 1, {penghapusan}
                    m[i, j-1] + 1, {penambahan}
                )

    return m[s1.length, s2.length]

```

Gambar 13. Script contoh matriks

```

function levDis (s1 : string, s2 : string) : integer
kamus
    i, cost : integer
    mBefore : array [0 .. s1.length] of integer
    mCurrent : array [0 .. s1.length] of integer
algoritma
    for i ← 0 to s1.length do
        {inisialisasi baris awal dengan nilai jarak
        perbandingan dengan kosong}
        mBefore[i] ← i

    for i ← 0 to s1.length do
        for j ← 1 to s2.length do
            if i = 0 then {perbandingan dengan kosong}
                mCurrent [i] ← j
            else
                if s1[i] = s2[j] then
                    cost ← 0
                else
                    cost ← 1

            mCurrent[i] = minimum (
                mBefore[i-1] + cost, {substitusi}
                mCurrent[i-1] + 1, {penghapusan}
                mBefore[i] + 1, {penambahan}
            )

    mBefore ← mCurrent

return mCurrent[s1.length]

```

Gambar 14. Script contoh matriks (1)

BAB IV : HASIL DAN PEMBAHASAN

4.1. Hasil Penelitian

4.1.1. Script Kode Program

Berikut ini ada lah potongan potongan script kode program untuk pembuatan model spell checker kalimat menggunakan python :

```
class get_data:  
    def getData():  
        read = pd.read_csv("./data/data2revisi.csv")  
        return read['a'].values.tolist()
```

Gambar 15. Potongan script model

Pada *class* *get_data* terdapat method *getData* atau function yang berfungsi untuk melakukan pemanggilan atau meinport data courpus menggunakan library pandas dari python

```
class cek_kata:  
    def cek(kata, data, callback):  
        hasil = []  
        arr_kata = kata.split()  
        for i in range(len(arr_kata)):  
            if arr_kata[i] in data:  
                hasil.append(arr_kata[i])  
            else:  
                for x in data:  
                    leven = edit_distance(arr_kata[i].lower(), x)  
                    kesalahan = callback(leven, len(x))  
                    if(kesalahan <= 35):  
                        hasil.append(x)  
                        break  
  
        return hasil
```

Gambar 15. Potongan script model

Pada *class* *cek kata* terdapat method *cek* atau function yang berfungsi untuk melakukan perhitungan kata dengan menggunakan algoritma levenshtein distance

```

class perhitungan:
    def presentase(leven, len_kata_benar):
        tingkat_kesalahan = leven / len_kata_benar * 100
        return round(tingkat_kesalahan)

```

Gambar 15. Potongan script model

Pada *class* cek_perhitungan terdapat method presentase atau function yang berfungsi untuk melakukan perhitungan tingkat kesalahan saat telah diketahui jarak parameter antar katanya

```

class tampil_hasil:
    def tampilkan(arr_hasil, kata):

        print("===== OUTPUT =====")
        print("")
        print("Kata : ", kata)
        print("Perbaikan : "+" ".join(arr_hasil))

        print("")
        print("")

```

Gambar 15. Potongan script model

Pada *class* tampil_hasil terdapat method tampilkan atau function yang berfungsi untuk melakukan penampilan output dan pemanggilan fungsi cek_kata sebelumnya. class ini hanya dipakai saat script masih berupa .ipynb. jika script sudah menjadi UI maka, script ini tidak diperlukan

4.1.2. Script untuk User Interface Tkinter:

Pertama dilakukan penambahan library dan module tkinter

```

pip install pyimage
from tkinter import *
from tkinter import ttk
from tkinter.font import Font
from PIL import Image, ImageTk

```

Gambar 15. Potongan script model

Kemudian dilakukan penambahan fungsi fungsi baru untuk pemrograman gui tkinter

```

def raise_frame(frame):
    frame.tkraise()
root = Tk()
root.geometry("800x600")
root.title(" Spell Checker ")
root.iconbitmap("iconspellcheck.ico")
root.configure(background="#f5f0e1")

lucida = Font(
    family = "Lucida Console",
    size = 16,
    weight = "normal",
)

# FRAME HOMEPAGE
homepage = Frame(root)
homepage.place(x=0, y=0, width=800, height=600)
# FRAME SpellChecker
spellcheckerpage = Frame(root)
spellcheckerpage.place(x=0, y=0, width=800, height=600)

# LOGO di HOMEPAGE
logo = PhotoImage(file="iconspellcheck.png")
labellogo = Label(homepage, image = logo)
labeltext = Label(homepage, text = "Selamat Datang di Aplikasi Spell
Checker Kami, Ketuk Tombol Spell Checker untuk Memulai")
labeltext.place(x=160, y=10)
labellogo.pack()

# Tombol untuk ke spell checker
tospellchecker = Button(homepage, text="Spell Checker", width="20",
font=lucida, bg="gray", fg="white", command=lambda:
raise_frame(spellcheckerpage))
tospellchecker.place(x=250 , y = 400)

def final():
    text_input = inputtext.get('1.0', 'end-1c')
    data_read = get_data.getData()

    hasil = cek_kata.cek(text_input, data_read, perhitungan.presentase)

```

```

        output.insert(END, hasil)

def clear():
    output.delete('1.0', END)

l = Label(spellcheckerpage, text = "===== APLIKASI
SPELL      CHECKER      /      LEVENSTHEIN      DISTANCE
=====")
g       =       Label(spellcheckerpage,       text       =
"-----")
k = Label(spellcheckerpage, text = "               KELOMPOK : 3
")
h = Label(spellcheckerpage, text = "SILAHKAN MASUKKAN KALIMAT
KE DALAM KOTAK DI BAWAH INI ",fg="gray", )

inputtext = Text(spellcheckerpage, height = 12, width = 55)

output = Text(spellcheckerpage, height = 12, width = 55)

input_n = Entry(spellcheckerpage, width = 5)

display = Button(spellcheckerpage, height = 2,
                 width = 20,
                 text ="Check Kata", command = nyoba, font = "aerial 12 bold")

delete_button = Button(spellcheckerpage, text = "Clear Output", command =
clear, font = "#1e3d59", bg="salmon", fg="white")
l.pack()
g.pack()
k.pack()
h.pack()
inputtext.pack()
display.pack()
output.pack()
delete_button.pack(side = RIGHT, padx = 15, pady = 20)

# BUTTON BACK TO HOMEPAGE
btn_back    =   Button(spellcheckerpage,   text="Back",   width="5",
font="Arial,(12)", \

```

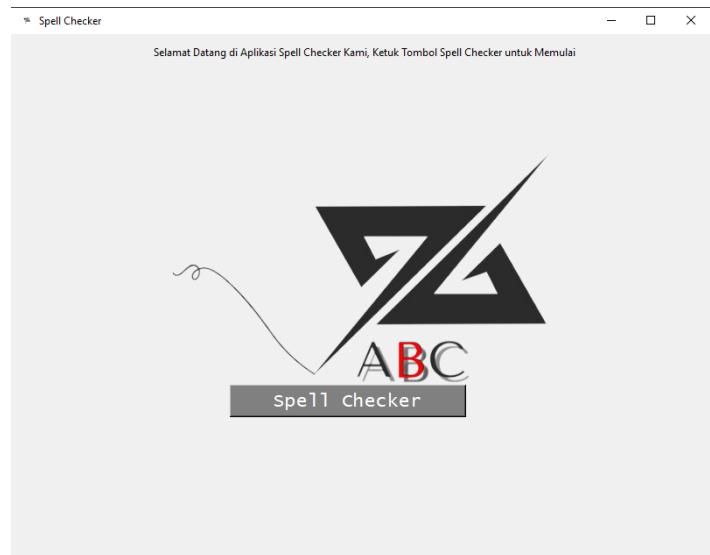
```

        command=lambda: raise_frame(homepage), bg="#1e3d59",
fg="white")
btn_back.place(x=10, y=550)

root.mainloop()

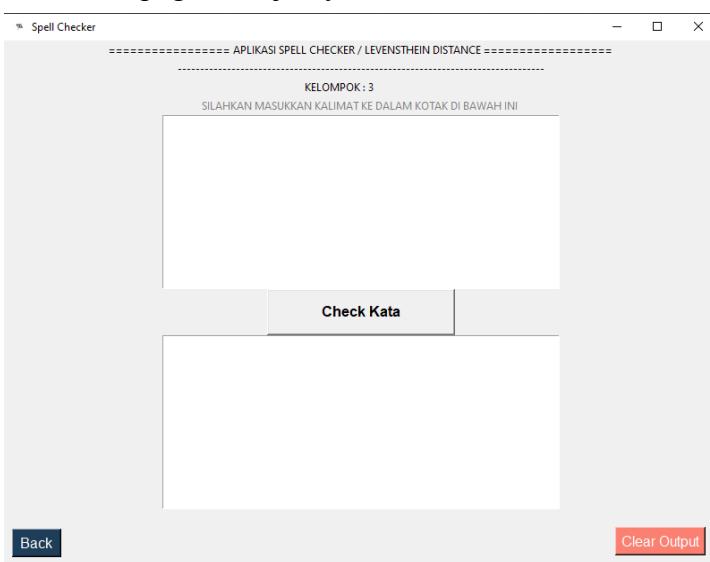
```

Berikut tampilan dari User Interface (UI)



Gambar(). Homepage Spell Checker

Ini merupakan tampilan homepage dari UI spellchecker. Di dalam homepage terdapat logo dan button yang berfungsi mengarahkan pengguna untuk ke page selanjutnya



Gambar(). Spell Checker Main Page

Sedangkan gambar diatas merupakan tampilan spellcheckerpage dari UI spellchecker. Di dalam spellcheckerpage terdapat kotak yang berfungsi sebagai input dan output an, kemudian ada button check kata

yang berfungsi untuk memulai memeriksa kalimat. Selain itu juga terdapat button clear output dan button back untuk kembali ke homepage.

4.1.3. Mengubah Kode Pemrograman Menjadi .EXE

Setelah kode pemrograman kami telah berbasis UI maka perlu dijadikannya menjadi file .exe(aplikasi) juga. hal itu berfungsi untuk pengguna lainnya dapat memakai nya dengan mudah tanpa harus membuka lewat jupyterlab. Berikut ini langkah langkahnya :

1. Pertama pastikan file code pemrograman telah menjadi .py
 2. Setelah itu buka CMD (Command Prompt Anaconda) yang ada di komputer
 3. Lalu masukan/ketik pip install auto-py-to-exe. Dan tunggu sampai proses download berhasil

```
[2] Select C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19044.1766]
(c) Microsoft Corporation. All rights reserved.

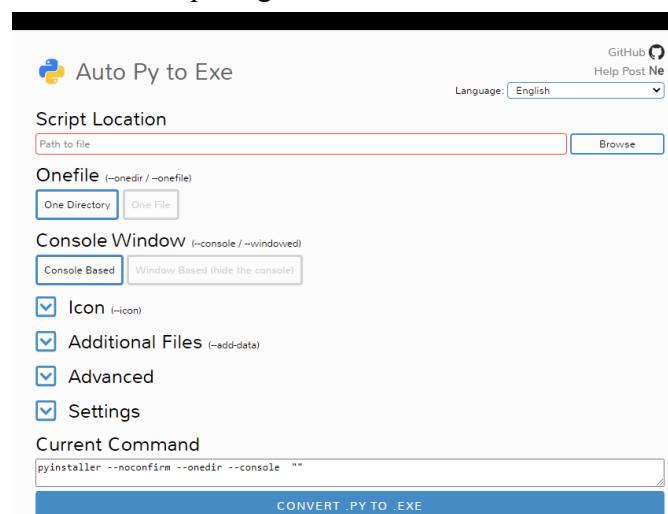
(base) C:\Users\hp\anaconda\lib\site-packages\operable\_py-to-exe
operable\_py-to-exe is not recognized as an internal or external command,
operable program or batch file.

(base) C:\Users\hp\pop\ install auto\_py\_to\_exe
Requirement already satisfied: auto\_py\_to\_exe in c:\users\hp\anaconda\lib\site\_packages (2.20.1)
Requirement already satisfied: Eel==0.14.0 in c:\users\hp\anaconda\lib\site\_packages (from auto\_py\_to\_exe) (0.14.0)
Requirement already satisfied: pyinstaller<4.0,>=3.6 in c:\users\hp\anaconda\lib\site\_packages (from auto\_py\_to\_exe) (3.6.1)
Requirement already satisfied: pygments<3.0,>=2.6 in c:\users\hp\anaconda\lib\site\_packages (from auto\_py\_to\_exe) (2.6.1)
Requirement already satisfied: bottle in c:\users\hp\anaconda\lib\site\_packages (from Eel==0.14.0->auto\_py\_to\_exe) (0.6.1)
Requirement already satisfied: bottle-websocket in c:\users\hp\anaconda\lib\site\_packages (from Eel==0.14.0->auto\_py\_to\_exe) (0.12.21)
Requirement already satisfied: pygments<3.0,>=2.6 in c:\users\hp\anaconda\lib\site\_packages (from bottle==0.6.1->auto\_py\_to\_exe) (0.8.2.9)
Requirement already satisfied: pyparsing<3.0,>=2.4 in c:\users\hp\anaconda\lib\site\_packages (from bottle==0.6.1->auto\_py\_to\_exe) (3.8.4)
Requirement already satisfied: pygments<3.0,>=2.6 in c:\users\hp\anaconda\lib\site\_packages (from bottle-websocket==0.12.21->auto\_py\_to\_exe) (0.8.2.9)
Requirement already satisfied: setuputils<1.0,>=0.6 in c:\users\hp\anaconda\lib\site\_packages (from pyinstaller==4.6->auto\_py\_to\_exe) (0.1.2)
Requirement already satisfied: pyfile==2022.8.1 in c:\users\hp\anaconda\lib\site\_packages (from pyinstaller==4.6->auto\_py\_to\_exe) (2022.5.30)
Requirement already satisfied: pyinstaller\_hooks\_contrib==2021.4 in c:\users\hp\anaconda\lib\site\_packages (from pyinstaller==4.6->auto\_py\_to\_exe) (2022.7)
Requirement already satisfied: aligner<1.0,>=0.1 in c:\users\hp\anaconda\lib\site\_packages (from pyinstaller==4.6->auto\_py\_to\_exe) (0.1.2)
Requirement already satisfied: bottle<0.15,>=0.14 in c:\users\hp\anaconda\lib\site\_packages (from pyinstaller==4.6->auto\_py\_to\_exe) (0.14.0)
Requirement already satisfied: bottle-websocket<1.0,>=0.14 in c:\users\hp\anaconda\lib\site\_packages (from bottle-websocket==0.12.21->auto\_py\_to\_exe) (0.10.1)
Requirement already satisfied: gevent<1.5,>=1.4 in c:\users\hp\anaconda\lib\site\_packages (from gevent==gevent\_websocket->bottle-websocket==0.14.0->auto\_py\_to\_exe) (21.12.0)
Requirement already satisfied: zope.event<1.0,>=0.14 in c:\users\hp\anaconda\lib\site\_packages (from gevent==gevent\_websocket->bottle-websocket==0.14.0->auto\_py\_to\_exe) (4.5)
Requirement already satisfied: pyzmq<2.0,>=1.1 in c:\users\hp\anaconda\lib\site\_packages (from gevent==gevent\_websocket->bottle-websocket==0.14.0->auto\_py\_to\_exe) (5.4.0)
Requirement already satisfied: greenlet<1.0,>=1.1.0 in c:\users\hp\anaconda\lib\site\_packages (from gevent==gevent\_websocket->bottle-websocket==0.14.0->auto\_py\_to\_exe) (1.1.1)
Requirement already satisfied: cffi==1.12.2 in c:\users\hp\anaconda\lib\site\_packages (from gevent==gevent\_websocket->bottle-websocket==0.14.0->auto\_py\_to\_exe) (1.15.0)
Requirement already satisfied: pycparser in c:\users\hp\anaconda\lib\site\_packages (from cffi==1.12.2->gevent==gevent\_websocket->bottle-websocket==0.14.0->auto\_py\_to\_exe) (2.21)

(base) C:\Users\hp\pop\
```

Gambar 6. Command pip install

- Setelah proses download/install selesai, buka py installer dengan memasukkan/mengetik auto-py-to-exe kemudian py installer akan otomatis terbuka seperti gambar dibawah ini



Gambar () . auto-py-to-exe

5. Kemudian masukkan file script yang telah berbentuk .py tadi kedalam Script Location



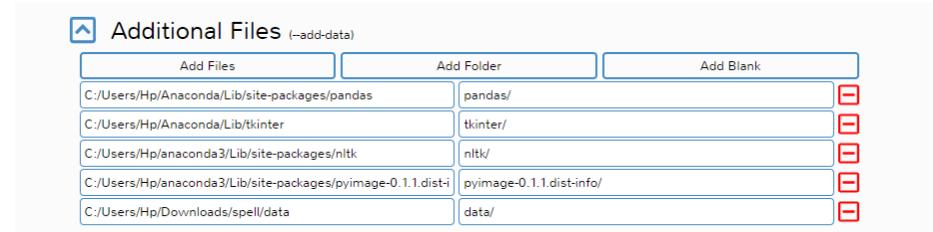
Gambar(). Script Berupa py

6. Lalu pilih One File dan Console Based untuk window-nya.



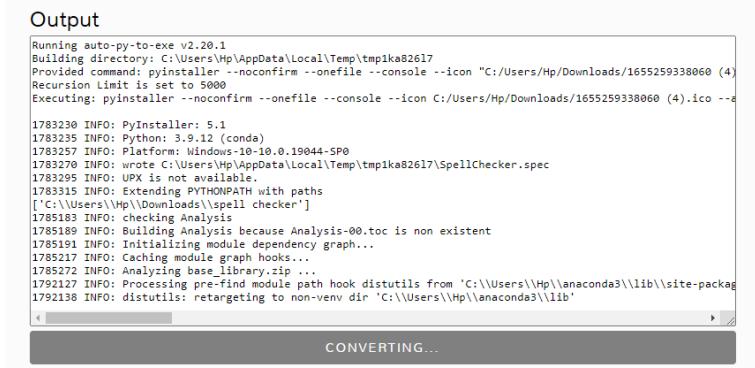
Gambar(). One File dan Console Based

7. Setelah itu masukkan file library ke Additional File. Masukkan semua file library yang dibutuhkan. Library dimasukkan semua agar nanti tidak crash saat membuka exe.



Gambar(). Insert Library

8. Setelah itu klik convert , maka akan terjadi process convert to exe seperti berikut ini.



Gambar(). Process Coverting

9. Setelah proses convert to exe selesai, tekan open output folder untuk membuka file exe nya.

- 10.Kemudian otomatis anda akan diarahkan ke file output location yang berupa file application seperti gambar di bawah ini.

11. Setelah dibuka, tampilannya akan sama persis seperti tampilan User Interface yang sudah dibuat sebelumnya.

4.1.4. Hasil Pengujian Data Testing

Pengujian dilakukan dengan memasukkan data uji testing sebanyak 5 teks data. Setelah model melalui beberapa kali pengujian didapatkan model yang dapat menghasilkan input sesuai dengan tujuan pembuatan. Berikut merupakan hasil dari model pengujian yang disajikan menjadi 2 tabel :

a. Tabel Pengujian Data Testing

No Data Testing	Jumlah Kata	Kata yang typo atau ejaan salah	Jumlah kata typo atau ejaan salah	Jumlah kata yang dapat dikoreksi dengan tepat	Jumlah kata yang tidak dikoreksi dengan tepat	Akurasi
1.	76	<ul style="list-style-type: none">● siatu● pertgi● menancing● biss● memkannya● subgai● setelahitu● iksn● nerubah● wsnita● yabg● berlenan● taoi● engan● menceriakan● asl● kepsda● tidsk● inginmengula mi	19	17	2	90%

2.	80	<ul style="list-style-type: none"> ● bssis ● wajiib ● ksmu ● kuasii ● badis ● ads ● siseem ● seldin ● komyuter ● untyk ● dalah ● dasatnya 	12	11	1	92%
3.	52	<ul style="list-style-type: none"> ● membua ● dikeanl ● karfna ● fufgsi ● sangatg ● baisk ● sakaligis ● perdlatan ● pe buatan ● meripakan ● factoo 	11	9	2	82%
4.	76	<ul style="list-style-type: none"> ● dima ksud ● adalsh ● statii ● mioip ● srkali ● malanan ● oloh ● itus ● tekt ● csra ● mudsh ● cukupt ● meleakkan 	13	11	2	85%

5.	79	<ul style="list-style-type: none"> • pasiyn • pemerrnths • kasud • hinga • oramg • penabahan • itj • seban • lwbih • infpmssi • sastuan • pensngsnan • wrtwan • yandg 	14	12	2	86%
----	----	---	----	----	---	-----

- b. Tabel Kata yang Tidak Muncul atau Tidak Terdapat Saran Kata yang Tepat

Data Testing 1	Data Testing 2	Data Testing 3	Data Testing 4	Data Testing 5
setelahitu	scientist	de	window	www.covid.go.id
	query	Tk()	statik	
	technical		mudah	
	skills			

4.2. Pembahasan

4.2.1 Sistem Pemeriksaan Ejaan Kalimat

Dengan menggunakan metode perhitungan *Levenshtein Distance* untuk memeriksa sebuah kalimat yang dimasukkan maka didapatkan output kata yang telah dikoreksi oleh sistem. Hasil yang didapat berdasarkan tabel ... data uji coba adalah :

1. Pada percobaan data testing 1 terdapat jumlah total 76 kata dengan 19 kata ejaan salah. Pada pengujian ini didapati 17 kata dapat dikoreksi sistem dengan tepat dan 2 kata lainnya tidak dapat dikoreksi sistem dengan tepat.
2. Pada percobaan data testing 1 terdapat jumlah total 76 kata dengan 19 kata ejaan salah. Pada pengujian ini didapati 17 kata dapat dikoreksi sistem dengan tepat dan 2 kata lainnya tidak dapat dikoreksi sistem dengan tepat
3. Pada percobaan data testing 1 terdapat jumlah total 76 kata dengan 19 kata ejaan salah. Pada pengujian ini didapati 17 kata dapat dikoreksi sistem dengan tepat dan 2 kata lainnya tidak dapat dikoreksi sistem dengan tepat
4. Pada percobaan data testing 1 terdapat jumlah total 76 kata dengan 19 kata ejaan salah. Pada pengujian ini didapati 17 kata dapat dikoreksi sistem dengan tepat dan 2 kata lainnya tidak dapat dikoreksi sistem dengan tepat
5. Pada percobaan data testing 1 terdapat jumlah total 76 kata dengan 19 kata ejaan salah. Pada pengujian ini didapati 17 kata dapat dikoreksi sistem dengan tepat dan 2 kata lainnya tidak dapat dikoreksi sistem dengan tepat

Dalam tabel... terdapat beberapa kesalahan ejaan kata yang tidak dapat diperbaiki atau tidak muncul koreksi katanya, maka sistem tidak menampilkan output untuk kata tersebut. Kata yang tidak bisa dikoreksi itu kebanyakan adalah kata yang berbahasa inggris, hal itu karena sistem kami hanya memeriksa kata yang berbahasa indonesia saja. Di sistem ini juga yang tidak dapat mengoreksi dua kata yang tidak memiliki spasi.

4.2.2 Keakuratan

$$\begin{aligned} \text{Keakuratan (persen)} &= 100 - | \text{galat} | \\ \text{galat} &= (\text{selisih nilai}) / (\text{nilai aktual}) \times 100 \end{aligned}$$

Dari kelima pengujian data testing yang telah dilakukan , Data testing 1 memiliki akurasi rata-rata keakuratan 90%. Data testing 2 memiliki akurasi rata-rata keakuratan 92%. Data testing 3 memiliki akurasi rata-rata keakuratan 82%. Data testing 4 memiliki akurasi rata-rata keakuratan 85%. Dan Data testing 5 memiliki akurasi rata-rata keakuratan 86%. Sehingga dari rata-rata tersebut menghasilkan rata-rata keakuratan 88%.

4.2.3 Kelebihan

Terdapat beberapa kelebihan dari sistem ini diantaranya :

1. Dapat membantu dalam mendeteksi ejaan kata atau kalimat yang salah dan dapat mengeluarkan output berupa kata atau kalimat yang baku dan benar
2. Menggunakan *Graphical Interface User* (GUI) untuk mengakses atau dilakukan uji coba
3. Model berupa file .exe sehingga mempermudah pengaksesan file jika ingin dilakukan uji coba di device lain.

4.2.4 Kekurangan

Terdapat beberapa kekurangan dari sistem ini diantaranya :

1. Sistem yang kami punya masih belum memiliki dataset yang lengkap yang menyebabkan kesalahan dalam mendeteksi kesalahan kata
2. Tidak mendeteksi kata asing, output dari hal tersebut yaitu akan dihilangkan kata asing yang di input
3. Tidak mendeteksi ketika ada 2 kata yang tidak memiliki spasi antara satu sama lain, output dari hal tersebut adalah tidak akan menampilkan perbaikan dari kata tersebut dan dihilangkan

BAB V : KESIMPULAN

5.1. Kesimpulan

Dilakukan beberapa kali uji coba model dan menguji dengan beberapa data testing, didapatkan model sistem yang bisa mendekripsi kalimat yang memiliki kesalahan penulisan. Sistem memproses inputan dengan menggunakan perhitungan dari algoritma *Levenshtein Distance*. Sistem melakukan perhitungan Algoritma tersebut dari fungsi bawaan yang ada di Python dengan mencari kata dengan panjang matriks terdekat dari inputan kemudian dari banyak kata yang memungkinkan menjadi kandidat output akan dihitung persentase tingkat kesalahannya, untuk meminimalisir dan menentukan output mana yang sekiranya tepat.

Pada uji coba sebelumnya, sistem yang dibuat hanya bisa mendekripsi inputan berupa 1 kata dan tidak akan bekerja jika memasukkan sebuah kalimat. Setelah dilakukan pengembangan script, maka didapatkan sistem yang bisa mendekripsi setiap *typo* setiap kata pada inputan kalimat, dengan memanfaatkan fungsi split yang akan memisah tiap kata dan dilakukan proses cek kata tersebut.

Setelah menguji dengan beberapa data testing berupa kalimat, didapatkan bahwa sistem dapat mendekripsi atau memproses kata yang mempunyai kesalahan dengan rata-rata akurasi sebesar 88%. Angka rata-rata ini didapat dari perhitungan pada bab 4.2.2 Keakuratan, dan juga angka dengan hasil dari faktor lain yaitu data korpus.

5.2. Saran

Berdasarkan hasil analisis yang dilakukan, peneliti menyarankan untuk :

1. Sistem spell checker ini akan lebih baik jika memiliki data keyword/kosakata yang lengkap, sehingga kata yang disarankan bisa mendekati yang diharapkan oleh pengguna.
2. Optimasi kata perbaikan yang diberikan sistem dapat ditingkatkan dengan mengimplementasikan metode-metode yang lebih efektif dan efisien.
3. Sebaiknya menambahkan atau menggunakan preprocessing kalimat atau kata yang lebih lengkap terlebih dahulu agar sistem menjadi lebih baik dalam pendekripsi keakuratan pencarian kata yang ingin di cari,

DAFTAR PUSTAKA

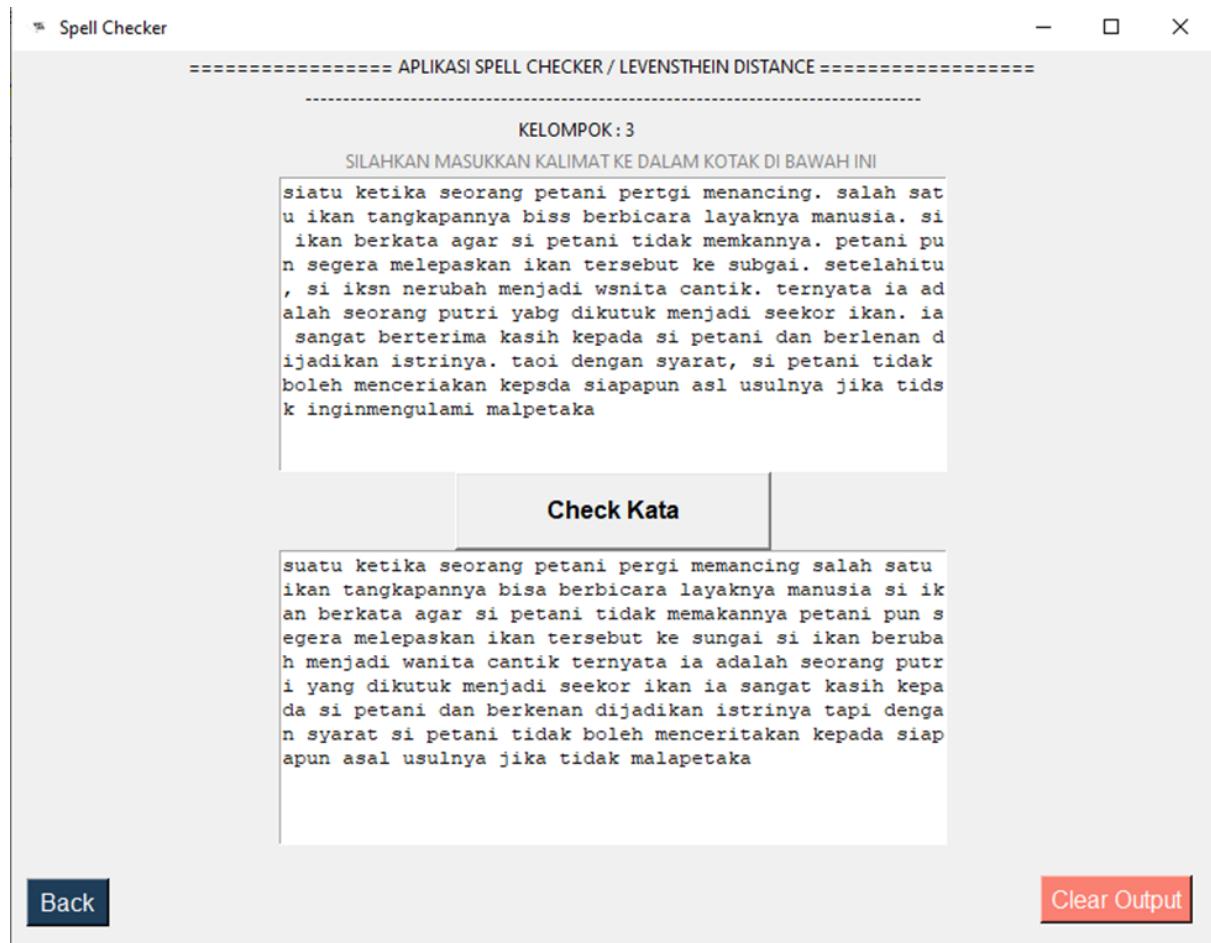
- Londo, G. L. Y., Purnomo WP, Y. S., & Maslim, M. (2020). Pembangunan Aplikasi Identifikasi Kesalahan Ketik Dokumen Berbahasa Indonesia Menggunakan Algoritma Jaro-Winkler Distance. *Jurnal Informatika Juita*, 8(1), 19-27.
https://e-jurnal.uajy.ac.id/21699/1/Juita_Pembangunan%20Aplikasi.pdf
- Mediyawati, N., Young, J. C., & Nusantara, S. B. (2021). U-TAPIS: AUTOMATIC SPELLING FILTER AS AN EFFORT TO IMPROVE INDONESIAN LANGUAGE COMPETENCIES OF JOURNALISTIC STUDENTS. *Jurnal Cakrawala Pendidikan*, 40(2).
<https://journal.uny.ac.id/index.php/cp/article/view/34546>
- Gunawan, D., & Rahmat, R. F. (2019). Normalisasi Mikroteks Berbentuk Kontekstual Berbahasa Indonesia pada Twitter Menggunakan Dictionary-Based dan Algoritma Longest Common Subsequence (LCS).
<http://repository.usu.ac.id/handle/123456789/16147>
- Widiatry, W., Sari, N. N. K., Pranatawijaya, V. H., & Putra, P. B. A. A. (2019). Penerapan Algoritma *Levenshtein Distance* Untuk Pencarian Pada Sistem Informasi Perpustakaan Fakultas Kedokteran Universitas Palangka Raya. *Jurnal SAINTEKOM*, 9(1), 66-82.
<https://ojs.stmikplk.ac.id/index.php/saintekom/article/view/75>
- Fahma, A. I., Cholissodin, I., & Perdana, R. S. (2018). Identifikasi kesalahan penulisan kata (*Typographical Error*) pada dokumen error berbahasa Indonesia menggunakan metode N-gram dan *Levenshtein Distance*. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN*, 2548, 964X.
<https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/690/273>
- Auer, E., & Landers, R. (2019). Creating reproducible and interactive analyses with JupyterLab and Binder. In 34th Annual Conference of the Society for Industrial and Organizational Psychology. <https://mybinder.org> (3 January 2022).
<https://iopscience.iop.org/article/10.1088/1742-6596/1525/1/012062/meta>
- Hariyanto, B. (2017). *Eksplorasi Antarmuka Grafis Pemakain Tkinter Pada Lingkungan Bahasa Python*. Intititutional Repositories & Scientific Journals.
<http://repository.unpas.ac.id/id/eprint/28589>

LAMPIRAN

No Data Testing	Kalimat Awal	Kalimat yang Telah diuba
1.	<p>suatu ketika seorang petani pergi memancing. salah satu ikan tangkapannya bisa berbicara layaknya manusia. si ikan berkata agar si petani tidak memakannya. petani pun segera melepaskan ikan tersebut ke sungai. setelah itu, si ikan berubah menjadi wanita cantik. ternyata ia adalah seorang putri yang dikutuk menjadi seekor ikan. ia sangat berterima kasih kepada si petani dan berkangan dijadikan istrinya. tapi dengan syarat, si petani tidak boleh menceritakan kepada siapapun asal usulnya jika tidak ingin mengalami malapetaka</p>	<p>siatu ketika seorang petani pertgi menancing. salah satu ikan tangkapannya biss berbicara layaknya manusia. si ikan berkata agar si petani tidak memkannya. petani pun segera melepaskan ikan tersebut ke subgai. setelahitu, si iksn nerubah menjadi wsnita cantik. ternyata ia adalah seorang putri yabg dikutuk menjadi seekor ikan. ia sangat berterima kasih kepada si petani dan berlenan dijadikan istrinya. taoi dengan syarat, si petani tidak boleh menceriakan kepsda siapapun asl usulnya jika tidsk inginmengulami malpetaka</p>
2.	<p>mysql dan basis data merupakan technical skills yang wajib kamu kuasai untuk berkarir sebagai ahli data khususnya data scientist. basis data mengatur data menjadi satu atau lebih tabel data yang ada tipe datanya saling terkait, seldin itu mysql memiliki fungsi untuk mengimplementasikan database relasi dalam sistem penyimpanan komputer, kelola pengguna, memungkinkan untuk akses jaringan dan memfasilitasi pengujian integrasi basis data. salah satu kiat sukses untuk bisa mahir mysql adalah dengan belajar query mysql dan dasar dasarnya terlebih dahulu.</p>	<p>mysql dan bsis data merupakan technical skills yang wajib ksmu kuasii untuk berkarir sebagai ahli data khususnya data scientist. badis data mengatur data menjadi satu atau lebih tabel data yang ads tipe data nya saling terkait, seldin itu mysql memiliki fungsi untuk mengimplementasikan database relasi dalam siseem penyimpanan komputer, kelola pengguna, memungkinkan untyk akses jaringan dan memfasilitasi pengujian integrasi basis data. dalah satu kiat sukses untuk bisa mahir mysql adalah dengan belajar query mysql dan dasar dasatnya telebih dahulu.</p>
3.	<p>tkinter adalah module python yang memungkinkan kita untuk membuat versi gui dari code yang</p>	<p>tkinter adalah module python yang memungkinkan kita untuk membua versi gui dari code yang</p>

	<p>kita miliki. tkinter dikenal karena memiliki fungsi yang sangat baik sekaligus mudah dalam pembuatan gui nya, dan hal itu sudah saya buktikan sendiri. tkinter merupakan modul python untuk peralatan gui tk(), dan merupakan gui de facto nya python.</p>	<p>kita miliki. tkinter dikeanl karfnia memiliki fufgsi yang sangatg baisk sekaligis mudah dalam pe buatan gui nya, dan hal itu sudah saya bukikan sendiri. tkinter meripakan modul python untuk peralatan gui tk(), dan merupakan gui de factoo nya python.</p>
4.	<p>label, yang dimaksud dengan kata label adalah tulisan statis yang melekat pada window kita. maksudnya adalah teks dengan fungsi label tidak dapat diubah dan di klik saat dijalankan. mirip sekali seperti label pada kemasan makanan, kita hanya bisa membacanya dan tidak dapat mengutak - atiknya, oleh karena itu maka disebut tekt statik. nah, cara membuat label pada tkinter sangat mudah, kita cukup memanggil fungsi label. di dalam fungsi label, kita dapat meletakkan beberapa parameter</p>	<p>label, yang dima ksud dengan kata label adalah tulisan statii yang melekat pada window kita. maksudnya adalah teks dengan fungsi label tidak dapat dirubah dan di klik saat di jalankan. mioip srkali seperti label pada kemasan malanan, kita hanya bisa membacanya dan tidak dapat mengutak - atiknya, olah karena itus maka disebut tekt statik. nah, csra membuat label pada tkinter samgat mudsh, kita cukupt memanggil fungsi label. didalam fungsi label, kita dapat meleakkan beberapa parameter</p>
5.	<p>jumlah pasien yang terjangkit virus corona atau covid-19 di indonesia masih terus bertambah. berdasarkan data pemerintah kasus yang masuk hingga minggu ini, tercatat ada penambahan kasus covid-19 sebanyak seribu lebih orang dalam sehari terakhir. penambahan pasien itu menjadi sebab kasus covid-19 di indonesia kini mencapai lebih dari 6 juta kasus. Informasi ini diungkap satuan tugas penanganan covid-19 dalam data yang disampaikan</p>	<p>jumlah pasiyn yang terjangkit virus corona atau covid-19 di indonesia masih terus bertambah. berdasarkan data pemrrinth kasud yang masuk hinga minggu ini, tercatat ada penambahan kasus covid-19 sebanyak seribu lebih oramg dalam sehari terakhir. penabahan pasien itj menjadi seban kasus covid-19 di indonesia kini mencapai lwbih dari 6 juta kasus. Infprmssi ini diungkap sastuan tugas pensngsnan covid-19 dalam data yang</p>

	<p>kepada wartawan pada rabu sore. data ini juga bisa diakses melalui website www.covid.go.id yang diperbarui setiap sore.</p>	<p>disampaikan kepada wrtawan pada rabu sore. data ini juga bisa diakses melalui website www.covid.go.id yandg diperbarui setiap sore.</p>
--	--	--



Spell Checker

===== APLIKASI SPELL CHECKER / LEVENSTHEIN DISTANCE =====

KELOMPOK:3

SILAHKAN MASUKKAN KALIMAT KE DALAM KOTAK DI BAWAH INI

mysql dan bsis data merupakan technical skills yang wa
jiib ksmu kuasii untuk berkarir sebagai ahli data khusu
snya data scientist. badis data mengatur data menjadi s
atu atau lebih tabel data yang ads tipe data nya saling
terkait, seldin itu mysql memiliki fungsi untuk mengim
plementasikan database relasi dalam siseem penyimpanan
komputer, kelola pengguna, memungkinkan untyk akses jar
ingan dan memfasilitasi pengujian integrasi basis data.
dalah satu kiat sukses untuk bisa mahir mysql adalah d
engan belajar query mysql dan dasar dasatnya telebih da
hulu.]

Check Kata

mysql dan basis data merupakan yang wajib kamu kuasai u
ntuk berkarir sebagai ahli data khususnya data basis da
ta mengatur data menjadi satu atau lebih tabel data yan
g ada tipe data nya saling terkait selain itu mysql mem
iliki fungsi untuk mengimplementasikan relasi dalam sis
tem penyimpanan komputer kelola pengguna memungkinkan u
ntuk akses jaringan dan memfasilitasi pengujian integra
si basis data kalah satu kiat sukses untuk bisa mahir m
ysql adalah dengan belajar mysql dan dasar dasarnya ter
lebih dahulu

Back Clear Output

Spell Checker

===== APLIKASI SPELL CHECKER / LEVENSTHEIN DISTANCE =====

KELOMPOK:3

SILAHKAN MASUKKAN KALIMAT KE DALAM KOTAK DI BAWAH INI

tkinter adalah module python yang memungkinkan kita unt
uk membuat versi gui dari code yang kita miliki. tkinter
dikenal karena memiliki fufgsi yang sangat baik sekali
gus mudah dalam pe buatan gui nya, dan hal itu sudah say
a buktikan sendiri. tkinter merupakan modul python un
tuk perdilatan gui tk(), dan merupakan gui de facto nya
python.

Check Kata

tkinter adalah module python yang memungkinkan kita unt
uk membuat versi gui dari code yang kita miliki tkinter
dikenal karena memiliki fungsi yang sangat baik sekali
gus mudah dalam pembuatan gui nya dan hal itu sudah say
a buktikan sendiri tkinter merupakan modul python untuk
pembuatan gui dan merupakan gui faktor nya python

Back Clear Output

