

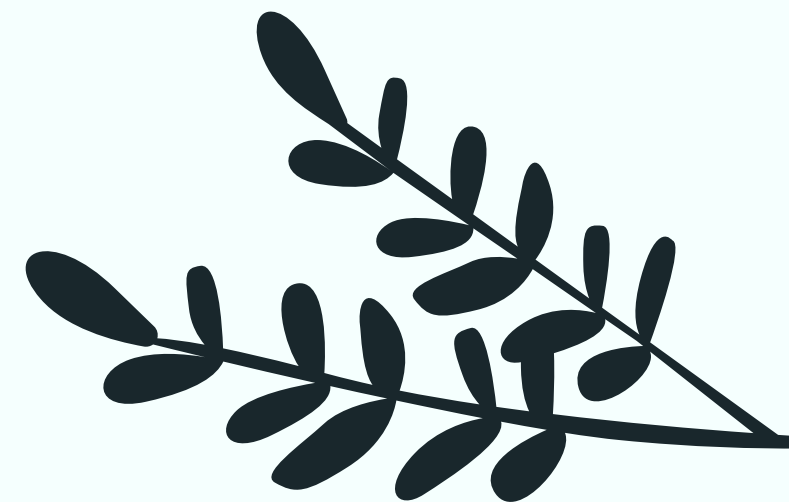
PRESENTASI FINAL PROJECT

SISTEM AUTOCORRECT

ANGGOTA KELOMPOK :

Kartika Diva Asmara Gita 5025211039

Hana Maheswari 5025211182



PENDAHULUAN

Autocorrect adalah suatu sistem yang mengoreksi kesalahan ejaan saat mengetik. Sistem akan mengidentifikasi kata yang salah eja, kemudian digunakan algoritma untuk mengidentifikasi kata yang mungkin dimaksud untuk diedit sesuai dengan hasil tersebut.

Autocorrect menggunakan AI untuk meningkatkan typing experience. Sistem ini menggunakan natural language processing dan algoritma machine learning untuk mengoreksi kesalahan ejaan dan tata bahasa secara otomatis. Sistem autocorrect bergantung pada kumpulan data teks besar dan pola penggunaan bahasa untuk dipelajari dan ditingkatkan dari waktu ke waktu.

NLP

Natural Language Processing adalah bidang ilmu komputer dan kecerdasan buatan yang berkaitan dengan pemahaman, pemrosesan, dan generasi bahasa manusia oleh komputer. NLP melibatkan berbagai teknik dan pendekatan untuk menganalisis, memahami, dan menghasilkan teks atau ucapan dalam bahasa manusia.

Dalam NLP, terdapat berbagai tugas dan aplikasi yang termasuk di dalamnya, seperti pemrosesan teks, analisis sentimen, pemodelan bahasa, pemahaman pertanyaan, terjemahan mesin, pemrosesan ucapan, dan banyak lagi. Pendekatan dan teknik yang digunakan dalam NLP mencakup statistik, pembelajaran mesin, deep learning, pemrosesan bahasa alami, dan berbagai alat dan metode lainnya.



METODE

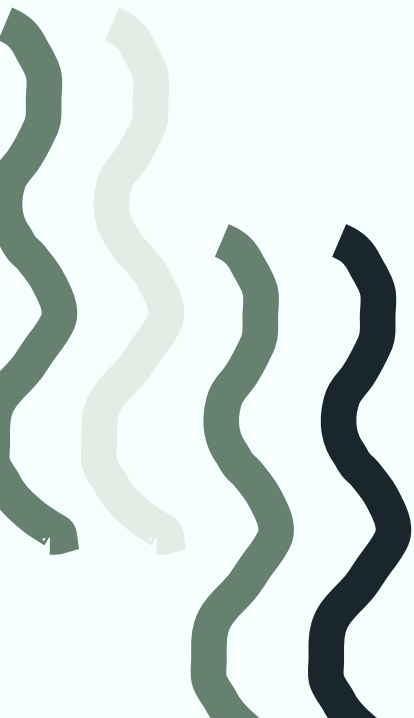
Pendekatan yang digunakan adalah cosine similarity dengan teknik vektorisasi kata. Dalam pendekatan ini, kata-kata direpresentasikan sebagai vektor menggunakan teknik vektorisasi, di mana setiap kata direpresentasikan sebagai vektor yang menggambarkan kemunculannya dalam dokumen atau korpus kata. Kemudian, cosine similarity digunakan untuk mengukur kesamaan antara vektor kata masukan dengan vektor-vektor kata dalam korpus kata. Dengan kombinasi ini, program dapat memberikan rekomendasi kata-kata yang mirip atau serupa dengan kata masukan dalam proses autocorrect.





Korpus Kata

1. Sumber daya NLP: Terdapat pustaka dan sumber daya NLP yang dapat digunakan untuk mengakses kamus bahasa. Misalnya, NLTK (Natural Language Toolkit) dalam bahasa Python menyediakan kamus bahasa Inggris yang dapat diakses dan digunakan dalam program (`nltk.corpus`).
2. Data korpus teks: Kamus korpus dapat dibangun dengan mengumpulkan teks-teks dari berbagai sumber, kemudian mengindeks dan memproses kata-kata yang ditemukan. Ada juga kamus korpus yang telah dibangun dan tersedia secara publik, seperti WordNet dan Corpus of Contemporary American English (COCA).
3. Pembentukan kamus bahasa sendiri secara terprogram seperti File Teks/CSV.



The background features a light mint green oval shape in the center. Surrounding this oval are several thick, wavy lines in dark navy blue, forest green, and pale sage green. In the top right and bottom left corners, there are dark navy blue silhouettes of leafy branches.

ALGORITMA


```
import tkinter as tk
from tkinter import scrolledtext
import pandas as pd
import nltk
from nltk import word_tokenize
from nltk.corpus import stopwords
from nltk.corpus import words
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from scipy.sparse import vstack
from nltk.metrics.distance import edit_distance
```

Langkah 1 : Mengimpor modul dan pustaka

- Modul tkinter dan scrolledtext digunakan untuk membuat interface (GUI).
- Pustaka pandas digunakan untuk manipulasi dan analisis data.
- Pustaka nltk (Natural Language Toolkit) digunakan untuk pemrosesan bahasa alami, seperti tokenisasi kata, pembersihan teks, dan penghitungan jarak Levenshtein.
- Pustaka sklearn digunakan untuk fitur ekstraksi dan penghitungan kesamaan kosinus.
- Pustaka scipy digunakan untuk operasi matriks sparse.

```
print("Reading dictionary ... ")
dictionary_input = ""
with open("new_words.txt", "r") as file:
    dictionary_input = file.read()

dictionary_input = dictionary_input.lower()
words_dict = word_tokenize(dictionary_input)
words_dict = list(set(words_dict))
print("Length of dictionary =", len(words_dict))
print("Dictionary ready!!!")
```

Langkah 2 : Membaca kamus file.txt

- Membuka file "new_words.txt" menggunakan fungsi **open()** dalam mode baca ("r").
- Membaca isi file menggunakan metode **read()** dan menyimpannya dalam variabel **dictionary_input**.
- Mengubah semua huruf dalam **dictionary_input** menjadi huruf kecil menggunakan metode **lower()**.
- Menggunakan fungsi **word_tokenize()** dari **nltk** untuk memecah **dictionary_input** menjadi token kata dan menyimpannya dalam variabel **words_dict**.
- Mengubah **words_dict** menjadi himpunan unik menggunakan fungsi **set()**.
- Mengonversi kembali **words_dict** ke dalam daftar menggunakan fungsi **list()**.


```
# Preprocess dictionary
stop_words = set(stopwords.words('english'))

words_dict = [
    word for word in words_dict if word not in
    stop_words and word.isalpha()]


# Build word corpus
english_words = set(words.words())
word_corpus = list(english_words.union
    (set(words_dict)))
```

Langkah 3 : Melakukan preprocessing pada kamus

- Mengimpor kamus kata umum (stopwords) dalam bahasa Inggris menggunakan **stopwords.words('english')** dari nltk.
- Menggunakan list comprehension, memfilter **words_dict** dengan menghilangkan kata-kata yang ada dalam stopwords (and, the, dsb) dan hanya mempertahankan kata-kata yang terdiri dari huruf saja (**word.isalpha()**).
- Hasilnya disimpan kembali dalam variabel **words_dict**.

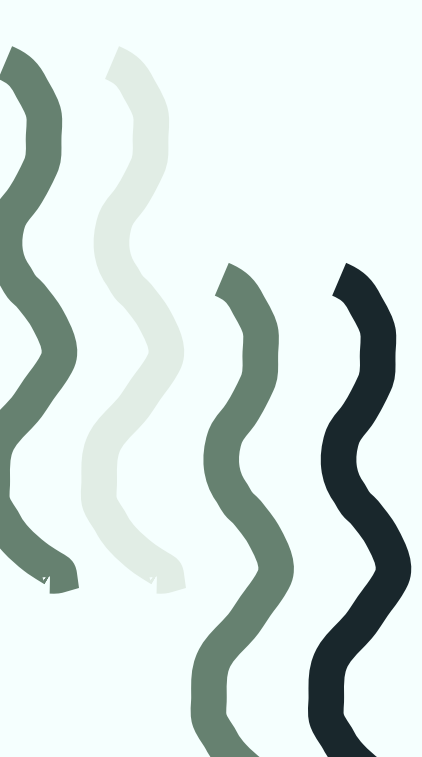

Langkah 4 : Membangun korpus kata


- Mengimpor kamus kata bahasa Inggris menggunakan **words.words()** dari nltk.
- Menggabungkan kedua kamus dengan operasi union (**|**).
- Mengubah hasil penggabungan menjadi daftar kata menggunakan fungsi **list()**. Hasilnya disimpan dalam variabel **word_corpus**.



```
# Vectorize words
vectorizer = CountVectorizer().fit(word_corpus)
word_vectors = vectorizer.transform(word_corpus)
```


Langkah 5 : Vektorisasi kata


- Menggunakan **CountVectorizer** dari **sklearn.feature_extraction.text** untuk membangun representasi vektor kata dalam korpus kata.
 - Menghasilkan matriks vektor kata yang merepresentasikan kemunculan kata-kata dalam korpus kata.
- 
- 



```
def autocorrect(word):  
    if word in words_dict:  
        return "No error!!"  
    else:  
        # Calculate Levenshtein distance  
        lev_distances = [(w, edit_distance(word, w)) for w in words_dict]  
        lev_distances = sorted(lev_distances, key=lambda x: x[1])  
  
        # Get words with low Levenshtein distance  
        possible_words = [w[0] for w in lev_distances if w[1] ≤ 2]
```


Langkah 6 : Autocorrect

- Menerima kata masukan sebagai argumen.
 - Jika kata masukan ada dalam kamus kata, mengembalikan pesan **"No error!!"** yang menunjukkan tidak ada kesalahan.
 - Jika kata masukan tidak ada dalam kamus kata:
 - a. Menghitung jarak Levenshtein antara kata masukan dan kata-kata dalam kamus kata.
 - b. Mengambil kata-kata dengan jarak Levenshtein terendah (dalam jarak 2) sebagai kata-kata yang mungkin benar.
- 



```
if len(possible_words) == 0:
    # Vectorize input word
    input_vector = vectorizer.transform([word])
    # Calculate cosine similarity
    similarities = cosine_similarity(input_vector, word_vectors)[0]
    similar_words = [(word_corpus[i], similarities[i]) for i in range(len(word_corpus))]
```

Langkah 6 : Autocorrect

- Jika tidak ada kata-kata yang ditemukan dengan jarak Levenshtein terendah, gunakan metode cosine similarity.
 - Vektorisasi kata masukan menggunakan **vectorizer** dan mengubah kata masukan (word) menjadi vektor menggunakan **vectorizer.transform()**. Hasilnya disimpan dalam variabel **input_vector**.
 - Menghitung cosine similarity menggunakan fungsi **cosine_similarity()** dari **sklearn.metrics.pairwise** untuk menghitung cosine similarity antara vektor kata masukan (input_vector) dan matriks vektor kata (word_vectors). Hasilnya disimpan dalam variabel similarities.
- 



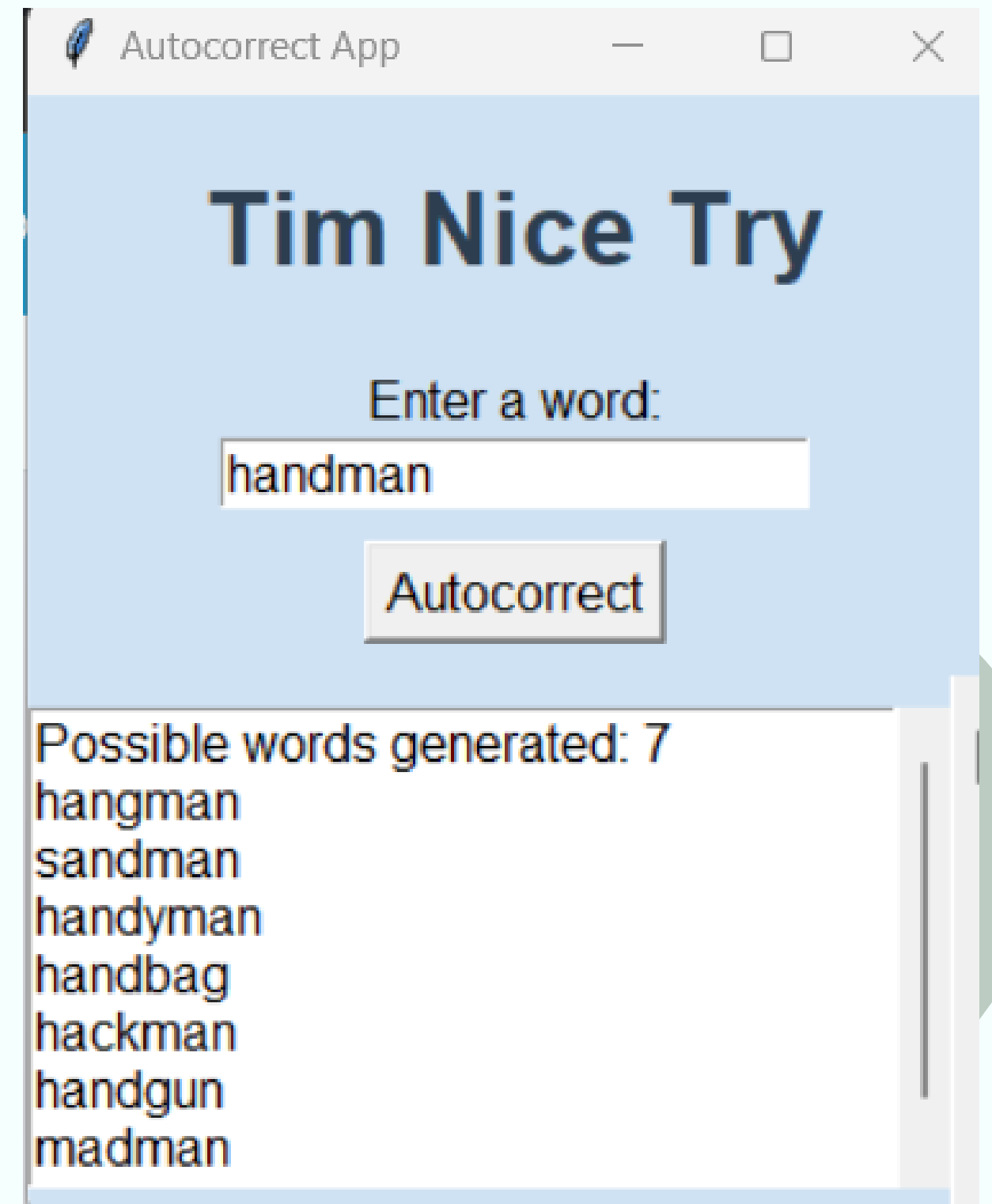
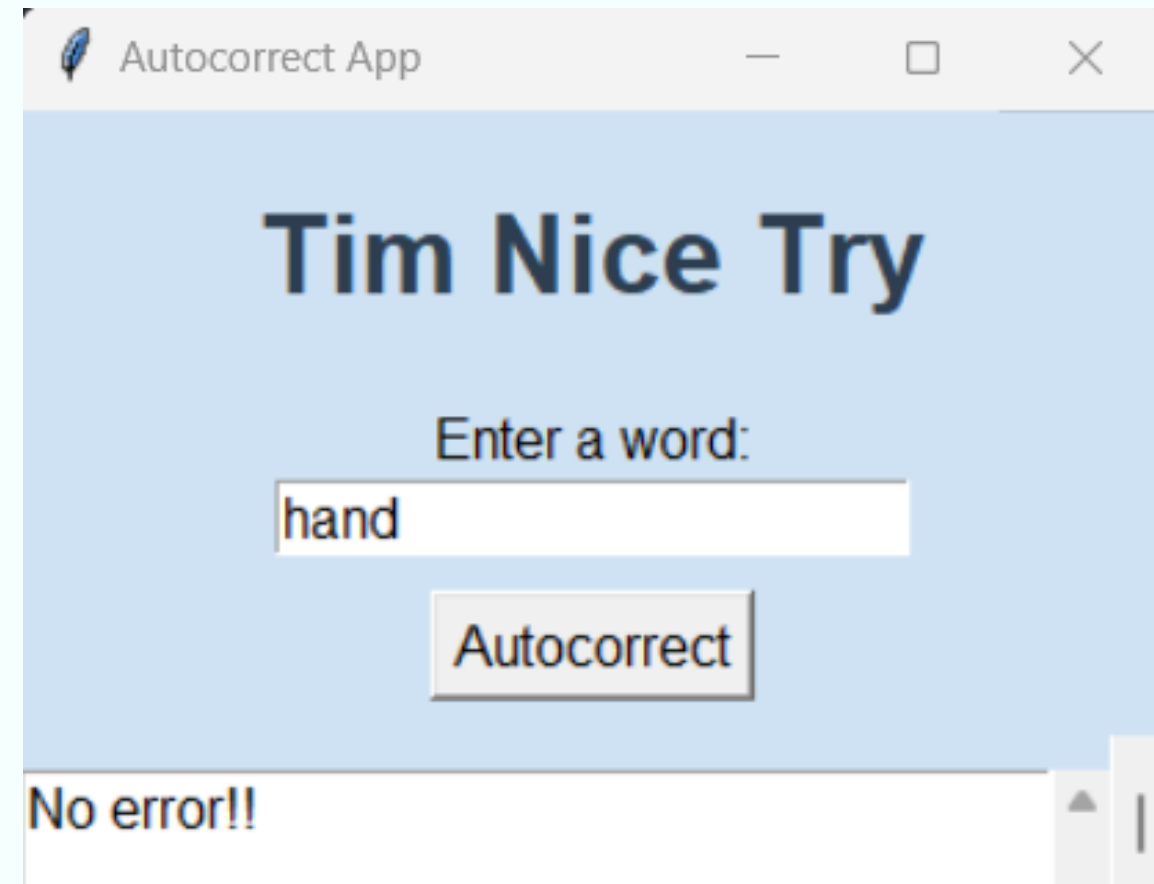
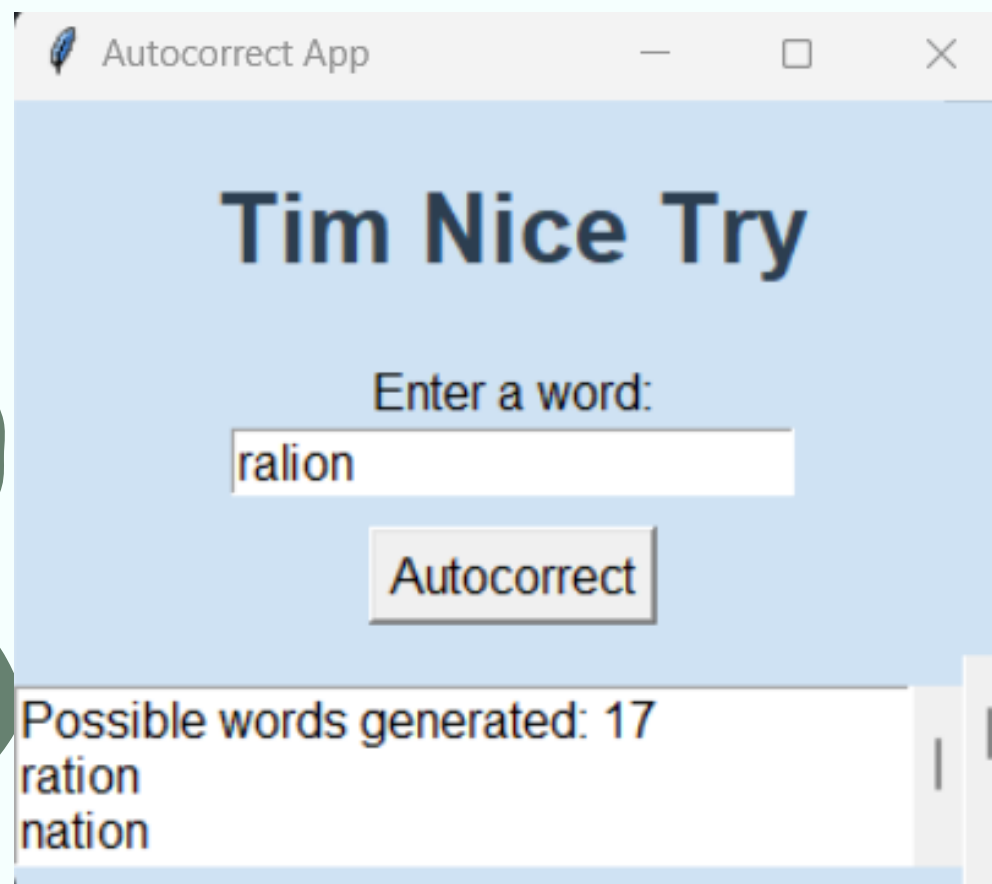
```
# Sort by similarity and filter possible words
similar_words = sorted(
    similar_words, key=lambda x: x[1], reverse=True)
possible_words = [w[0] for w in similar_words[:10]] # Get top 10 similar words
result = f"Possible words generated: {len(possible_words)}\n"
for w in possible_words:
    result += w + "\n"
return result
```

Langkah 6 : Autocorrect

- Setelah mendapatkan daftar kata-kata serupa (`similar_words`), langkah selanjutnya adalah mengurutkan kata-kata tersebut dengan fungsi **`sorted()`**.
- Argumen **`key=lambda x: x[1]`** digunakan untuk menentukan kunci pengurutan berdasarkan nilai similarity (**`x[1]`**), yang merupakan elemen kedua dari setiap pasangan kata dan similarity dalam **`similar_words`**.
- Argumen **`reverse=True`** digunakan untuk mengurutkan secara menurun dan mengambil 10 kata teratas yang memiliki nilai similarity tertinggi dengan slicing **`similar_words[:10]`**.
- Menghitung jumlah kata-kata yang mungkin dihasilkan (`len(possible_words)`) dan menyimpannya dalam string `result`.

OUTPUT

```
[nltk_data] Downloading package stopwords to C:\Users\Kartika
[nltk_data]   Diva\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package words to C:\Users\Kartika
[nltk_data]   Diva\AppData\Roaming\nltk_data...
[nltk_data]   Package words is already up-to-date!
Reading dictionary...
Length of dictionary = 21321
Dictionary ready!!!
```



The background features a light mint green oval shape in the center. Surrounding this oval are several horizontal wavy lines in three colors: dark navy blue, a muted sage green, and a very light mint green. In the top right and bottom left corners, there are dark navy blue leafy branch motifs.

TERIMA KASIH

Referensi :

- [**https://www.geeksforgeeks.org/python-measure-similarity-between-two-sentences-using-cosine-similarity/**](https://www.geeksforgeeks.org/python-measure-similarity-between-two-sentences-using-cosine-similarity/)
- [**https://predictivehacks.com/how-to-build-an-autocorrect-in-python/**](https://predictivehacks.com/how-to-build-an-autocorrect-in-python/)
- [**https://github.com/ShalakaPawar/Autocorrect_Algorithm/tree/main**](https://github.com/ShalakaPawar/Autocorrect_Algorithm/tree/main)
(Memodifikasi kembali sumber Github yang belum menerapkan AI dengan menggunakan pendekatan Cosine Similarity dan teknik vektorisasi kata)