Makalah

Penggunaan Cosine Similarity Function untuk Membuat Rekomendasi Lagu Spotify

Penyusun

Ahmad Faiz Ali Azmi (225150207111072)

Ananda Ravi Kuntadi (225150200111035)

Davin Dalana Fidelio Fredra (225150201111029)

Muhammad Yasin Hakim (2251500200111036)

Salsa Zufar Radinka Akmal (225150207111067)

Prodi Teknik Informatika

Departemen Teknik Informatika

Fakultas Ilmu Komputer

Universitas Brawijaya Tahun Akademik 2022/2023

DAFTAR ISI

DAFTAR ISI **2**
BAB I
LATAR BELAKANG **3**
1.1 Latar Belakang 3
1.2 Rumusan Masalah 4
1.3 Tujuan Pembahasan 4
BAB II
TINJAUAN PUSTAKA **5**
2.1 Sistem Rekomendasi Lagu 5
2.3 Cosine Similarity 6
2.4 Python 7
2.5 Streamlit Library 7
2.6 Matplotlib Library 8
2.7 Plotly Library 8
BAB III
SOURCE CODE **9**
BAB IV
PEMBAHASAN **26**
4.1 Preprocessing Data 26
4.2 Pembahasan File app.py 28
4.3 Pembahasan File run_recommender.py 28
BAB V
SCREENSHOT **31**
5.1 Pemilihan Ciri-Ciri dan Banyak Rekomendasi Lagu 31
5.2 Pencarian lagu 32
5.3 Mendapatkan Rekomendasi Lagu 33
BAB VI
KESIMPULAN **36**
6.1 Kesimpulan 36
6.2 Saran 37
DAFTAR PUSTAKA **38**
LAMPIRAN 39

BAB I
LATAR BELAKANG

1.1 Latar Belakang

Sistem rekomendasi adalah salah satu aplikasi kecerdasan buatan yang bertujuan untuk memberikan saran atau rekomendasi kepada pengguna tentang item atau objek yang mungkin diminati atau dibutuhkan oleh pengguna. Sistem rekomendasi dapat digunakan dalam berbagai bidang, seperti *e-commerce*, media sosial, pendidikan, kesehatan, dan lain-lain. Salah satu bidang yang populer menggunakan sistem rekomendasi adalah musik. Sistem rekomendasi musik dapat membantu pengguna menemukan lagu-lagu yang sesuai dengan selera atau preferensi mereka.

Salah satu metode yang dapat digunakan untuk membuat sistem rekomendasi musik adalah content based *filtering*. *Content based filtering* adalah metode yang merekomendasikan item berdasarkan kesamaan antara fitur atau atribut dari item tersebut dengan fitur atau atribut dari item yang telah dikonsumsi atau dinilai oleh pengguna sebelumnya. Dalam konteks musik, fitur atau atribut yang dapat digunakan untuk merekomendasikan lagu antara lain adalah genre, artis, tempo, ritme, nada, lirik, dan lain-lain. *Content based filtering* memiliki beberapa kelebihan, seperti tidak memerlukan data dari pengguna lain, dapat menangani item baru yang belum dinilai oleh pengguna lain, dan dapat memberikan penjelasan mengapa item tersebut direkomendasikan. Namun, content based *filtering* juga memiliki beberapa kelemahan, seperti sulitnya mengekstrak fitur dari item secara otomatis, adanya kemungkinan overfitting atau terlalu spesifik pada preferensi pengguna, dan kurangnya kemampuan untuk merekomendasikan item yang beragam.

Oleh karena itu, pembuatan model sistem rekomendasi lagu yang menggunakan *content based filtering* merupakan topik yang menarik dan bermanfaat untuk dibahas dalam makalah ini. Model tersebut dapat memberikan saran lagu-lagu yang sesuai dengan selera pengguna berdasarkan fitur-fitur musik yang relevan dan informatif. Model tersebut juga dapat diuji dan dievaluasi dengan menggunakan data riil dari pengguna musik.

1.2 Rumusan Masalah

1. Bagaimana mengimplementasikan *Cosine Similarity Function* dalam sistem rekomendasi lagu?
2. Sejauh mana keakuratan dan ketepatan rekomendasi lagu yang dihasilkan oleh sistem berdasarkan penggunaan *Cosine Similarity Function*?
3. Apakah terdapat batasan atau kendala tertentu dalam penggunaan *Cosine Similarity Function* dalam konteks sistem rekomendasi lagu?

1.3 Tujuan Pembahasan

1. Merancang dan mengimplementasikan sistem rekomendasi lagu berbasis *Cosine Similarity Function*.
2. Menguji dan menganalisis kinerja sistem dalam menghasilkan rekomendasi lagu yang sesuai dengan preferensi pengguna.
3. Mengidentifikasi dan mengatasi batasan atau kendala yang mungkin timbul dalam penerapan *Cosine Similarity Function* di Spotify.

BAB II
TINJAUAN PUSTAKA

2.1 Sistem Rekomendasi Lagu

Sistem rekomendasi merupakan program atau sistem penyaringan informasi yang menjadi solusi dalam masalah kelebihan informasi dengan cara menyaring sebagian informasi penting dari banyaknya informasi yang ada dan bersifat dinamis sesuai dengan preferensi, minat, atau perilaku pengguna terhadap suatu objek. Sistem rekomendasi dirancang untuk memahami dan memprediksi preferensi pengguna berdasarkan perilaku pengguna (Rao, 2019). Sistem rekomendasi diharuskan memiliki kemampuan untuk memprediksi apakah pengguna tertentu akan memilih barang yang berdasarkan preferensi, minat, perilaku pengguna, atau pengguna lainnya. Sistem rekomendasi dapat membantu dalam mengambil keputusan di dalam informasi yang kompleks dan banyak secara obyektif. Terdapat beberapa metode yang dapat digunakan dalam membangun sebuah sistem rekomendasi antara lain *content based filtering*, *collaborative filtering*, *hybrid filtering*, dan lain sebagainya (Isinkaye, Folajimi and Ojokoh, 2015).Terdapat dua metode pendekatan pada sistem rekomendasi tes (Isinkaye, Folajimi dan Ojokoh, 2015):

1. *Content Based Filtering*

   Menggunakan kemiripan antara produk yang akan direkomendasikan dengan produk yang disukai pengguna
2. *Collaborative Filtering*

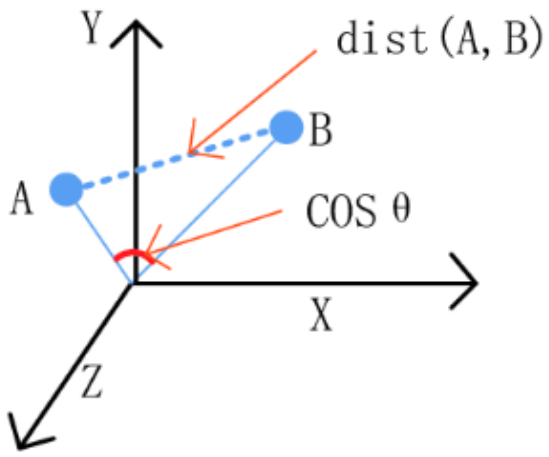
   Menggunakan kemiripan kueri dengan item pengguna dengan pengguna lain.

2.2 Content Based Filtering

Content Based Filtering pada Sistem rekomendasi adalah metode yang mempertimbangkan perilaku dari pengguna dari masa lalu yang kemudian diidentifikasi pola perilakunya untuk merekomendasikan barang yang sesuai dengan pola perilaku tersebut (Reddy et al., 2019). Metode *content based filtering* menganalisis preferensi dari perilaku pengguna dimasa lalu untuk membuat model. Model tersebut akan dicocokkan dengan serangkaian karakteristik atribut dari barang yang akan direkomendasikan. Barang dengan tingkat kecocokan tertinggi akan menjadi rekomendasi untuk pengguna.

2.3 Cosine Similarity

Cosine similarity adalah salah satu metode pengukuran kemiripan antara dua dokumen yang berbeda dengan menghitung *cosinus* sudut yang terbentuk oleh vektor yang merepresentasikan masing-masing dokumen (Fauzi, Arifin and Yuniarti, 2017). Fitur yang ada pada suatu dokumen yang merupakan dimensi membentuk sebuah vektor. Kedua vektor yang terbentuk dari dua objek dapat dicari kemiripannya dengan menghitung jarak antar vektor. Ada beberapa metode untuk menghitung jarak antara dua vektor seperti *euclidean distance* dan *cosine similarity* seperti pada Gambar 1.



Gambar 1 *EuclideanDistancedan CosineSimilaritySumber*: Wang, Chen, & Wu, 2017

Pada gambar 1 simbol A dan B merupakan vektor yang dicari jarak antar keduanya menggunakan *euclidean distance* yang dengan simbol dist(A, B) dan *cosine similarity* dengan simbol COS θ. Simbol Z, X, dan Y merupakan fitur dari objek. Pada umumnya metode *cosine similarity* memang digunakan untuk *data mining*, sistem temu kembali informasi, dan sistem rekomendasi untuk mencari kemiripan antara kedua vektor dokumen. Rumus *cosine similarity* dapat dilihat pada Persamaan 1 dan untuk normalisasi TF-IDF dapat dilihat pada Persamaan 2.

$$\cos_{(q,d)} = \frac{q \times d}{|q| \times |d|} = \frac{\sum_{i=1}^{|v|} q_i d_i}{\sqrt{\sum_{i=1}^{|v|} q_i d_i{}^2 \times \sum_{i=1}^{|v|} q_i d_i{}^2}} \quad (1)$$

Keterangan:

q= bobot TF-IDF pada kueri

d= bobot TF-IDF pada dokumen

Dengan normalisasi pada pembobotan term:

$$\cos_{(q,d)} = q \times d = \sum_{i=1}^{|v|} q_i d_i \quad (2)$$

Keterangan:

q= bobot TF-IDF pada kueri

d= bobot TF-IDF pada dokumen

Hasil yang didapat dari Persamaan 5 dan 2.6 akan berupa nilai dengan rentang[0,1]. Semakin besar nilai yang didapat (mendekati 1) maka semakin kecil sudut yang dihasilkan oleh kedua vektor tersebut yang berarti semakin mirip kedua dokumen yang dibandingkan, sebaliknya semakin kecil nilai yang didapat (mendekati 0) maka semakin besar sudut yang dihasilkan oleh kedua vektor tersebut dan semakin beda kedua dokumen yang dibandingkan, sehingga dapat disimpulkan bahwasanya tingkat kemiripan berbanding lurus dengan nilai *cosinus*.

2.4 Python

Python adalah bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif. Python bisa dibilang bahasa pemrograman dengan tujuan umum yang dikembangkan secara khusus untuk membuat source code mudah dibaca. Python juga memiliki *library* yang lengkap sehingga memungkinkan programmer untuk membuat aplikasi yang mutakhir dengan menggunakan *source code* yang tampak sederhana (Ljubomir Perkovic, 2012).

2.5 Streamlit Library

Streamlit merupakan *open source framework* dengan bahasa pemrograman Python yang memudahkan pengembangan *web-apps* terutama untuk *data science* dan *machine learning* karena terdapat beberapa fitur untuk mempermudah pengembangan *model machine learning*. *Streamlit* menawarkan *visualisasi* melalui antarmuka interaktif untuk

mengkomunikasikan informasi yang relevan untuk user. Proses *deploy* pada *streamlit* cukup mudah melalui platform cloud-sharing *Streamlit* dan dapat diakses di berbagai platform. (Koh, Joly, & Chan, 2021).

2.6 Matplotlib Library

Matplotlib merupakan pustaka *Python cross-platform* untuk membuat grafik 2 dimensi dengan kualitas tinggi. *Matplotlib* dapat digunakan dalam *script Python*, Interpreter *Python* dan *iPython*, server, dan 6 *GUI toolkit*. Dengan *Matplotlib*, membuat plots, histograms, spectra, *bar charts*, *errorchards*, *scatterplots*, akan lebih mudah dilakukan. Dengan adanya kemudahan tersebut, *matplotlib* dapat digunakan untuk mengembangkan aplikasi laporan yang profesional, aplikasi analisis interaktif, aplikasi *dashboard* yang lengkap atau di-embed kan ke dalam *web* atau aplikasi *GUI*(*Grafik User Interface*). (K.Nagesh, D.Nageswara Rao, Song K.Choi, 2015).

2.7 Plotly Library

Plotly adalah library untuk pembuatan plot yang tersedia dalam bahasa pemrograman *Python* dan R. dari segi kompatibilitas pada diagram, *library* ini tidak jauh berbeda dengan *matplotlib*. Plot garis, diagram batang, hingga *heatmaps* merupakan keunggulan dari *Plotly*. Pada *Plotly* secara *default* sudah tersedia beberapa *tools* yang mendukung interaksi pada plot, sebagai contoh pembesaran diagram dan tombol *screenshot* tersedia secara otomatis. Berbeda dengan *matplotlib*, penulisan kode secara manual diperlukan untuk menyimpan hasil diagram dari *Plotly*.

BAB III
SOURCE CODE

app.py

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import streamlit as st
from io import BytesIO
import plotly.io as pio

from packages.search_song import search_song
from packages.run_recommender import
get_feature_vector, show_similar_songs,
radar_chart

# load data
dat =
pd.read_csv('data/Processed/dat_for_recommend
er.csv')
comparison_dat =
pd.read_csv('data/Raw/comparison_data.csv')

song_features_normalized = ['valence',
'acousticness', 'danceability', 'energy',
```

```
        'instrumentalness', 'liveness',
        'speechiness']
        song_features_not_normalized =
        ['duration_ms', 'key', 'loudness', 'mode',
        'tempo']

        all_features = song_features_normalized +
        song_features_not_normalized + ['popularity']

        # set app layout
        # st.set_page_config(layout="wide")

        # set a good looking font
        st.markdown(
            """
            <style>
            .big-font {
                font-size:20px !important;
            }
            </style>
            """,
            unsafe_allow_html=True,
        )



        def main():
            st.markdown("# Sistem Rekomendasi Lagu
```

```
hanya untukmu!")
    st.markdown("Selamat datang di Sistem
Rekomendasi Lagu! \
                \n Kamu dapat memasukkan lagu
dan mendapatkan rekomendasi berdasarkan
ciri-ciri lagu yang telah kau masukkan. \
                \n Kamu juga dapat
mengkostumisasi rekomendasinya dengan memilih
ciri-ciri yang kamu inginkan. Selamat
Mencari! \
                \n NB: Rekomendasi lagu ini
hanya menelusuri lagu antara 2018 - 2020.
")

    # add selectbox for selecting the
features
    st.sidebar.markdown("### Memilih
Ciri-ciri")
    features = st.sidebar.multiselect('Pilih
ciri-ciri yang kamu inginkan', all_features,
default=all_features)
    # add a slider for selecting the number
of recommendations
    st.sidebar.markdown("### Banyak
rekomendasi lagu yang didapatkan")
    num_recommendations =
st.sidebar.slider('Pilih berapa banyak
rekomendasi yang diinginkan', 10, 50, 10)
```

```
    # add a search box for searching the song
by giving capital letters and year
    st.markdown("### Siap untuk mendapatkan
rekomendasi dari lagu yang kamu masukkan?")
    song_name = st.text_input('Masukkan judul
lagunya', key="search_input")
    # Filter options based on the search
query
    filtered_options = [option for option in
dat['name'] if song_name.upper() in
option.upper()]
    st.write(filtered_options)
    if song_name != '':
        song_name = song_name.upper()
    year = st.text_input('Masukkan tahun dari
lagu tersebut (contoh: 2019). \
                         \nJika kamu tidak
yakin lagunya ada di databasenya atau tidak
yakin dengan tahunnya, \
                         Tolong biarkan tahun
lagunya kosong dan klik tombol dibawah ini
untuk mencari lagu tersebut.')
    if year != '':
        year = int(year)

    # exmaples of song name and year:
    # song_name = 'YOUR HAND IN MINE'
```

```
        # year = 2003

        # add a button for searching the song if
    the user does not know the year
        if st.button('Cari lagu saya'):
            found_flag, found_song =
    search_song(song_name, dat)
            if found_flag:
                st.markdown("Wow, Lagu ini ada di
    dataset:")
                st.markdown(found_song)
            else:
                st.markdown("Maaf, lagu ini tidak
    ada di dataset. Tolong cari lagu yang lain!")

        # add a button for getting
    recommendations
        if st.button('Dapatkan Rekomendasi!'):
            if song_name == '':
                st.markdown("Tolong masukkan nama
    lagunya!")
            elif year == '':
                st.markdown("Tolong masukkan
    tahun lagunya!")
            else:
                # show the most similar songs in
    wordcloud
                fig_cloud =
```

```
        show_similar_songs(song_name, year, dat,
        features, num_recommendations,
        plot_type='wordcloud')
                    st.markdown(f"### Keren! Inilah
        rekomendasinya dari lagu \
                                \n#### {song_name}
        ({year})!")
                    st.pyplot(fig_cloud)

                    # show the most similar songs in
        bar chart
                    fig_bar =
        show_similar_songs(song_name, year, dat,
        features, top_n=10, plot_type='bar')
                    st.markdown("### Lihatlah lebih
        dekat dari 10 lagu rekomendasi untukmu!")
                    st.pyplot(fig_bar)

                    #Menampilkan perbandingan radar
        chart antara lagu yang dimasukkan dengan 5
        lagu teratas
                    fig_radar =
        radar_chart(comparison_dat,
        song_features_normalized)
                    st.markdown("### Gambaran
        Kemiripan Ciri-Ciri lagumu dengan
        Rekomendasinya!")
                    st.plotly_chart(fig_radar)
```

```
if __name__ == "__main__":
    main()
```

run_recommender.py

```
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
import numpy as np
sns.set_palette("Set2")
from wordcloud import WordCloud


def get_feature_vector(song_name, year,
dat, features_list):
    print(dat.head())
    # select dat with the song name and
year
    dat_song = dat.query('name ==
@song_name and year == @year')
    song_repeated = 0
    if len(dat_song) == 0:
        raise Exception('The song does not
exist in the dataset or the year is wrong!
\
                        \n Use search
```

```
        function first if you are not sure.')
            if len(dat_song) > 1:
                song_repeated = dat_song.shape[0]
                print(f'Warning: Multiple
        ({song_repeated}) songs with the same name
        and artist, the first one is selected!')
                dat_song = dat_song.head(1)
            feature_vector =
        dat_song[features_list].values
            return feature_vector, song_repeated

        names = []

        def dot_product(vector_a, vector_b):
            return sum(a * b for a, b in
        zip(vector_a, vector_b))



        def cosine_similarity_2d(array1, array2):

            similarities = []
            array2 = array2[0]  # Extract the
        single row from array2
            print(array2)
            print(len(array1))
            for row in array1:
                dot_product = np.dot(row, array2)
                magnitude_a = np.linalg.norm(row)
```

```
            magnitude_b =
        np.linalg.norm(array2)

            similarity = dot_product /
        (magnitude_a * magnitude_b) if (magnitude_a
        != 0 and magnitude_b != 0) else 0
            similarities.append(similarity)

        return similarities

        # define a function to get the most similar
        songs
        def show_similar_songs(song_name, year,
        dat, features_list, top_n=10,
        plot_type='wordcloud'):
            """
            Fungsi untuk mendapatkan lagu dengan
        tingkat kemiripan tertinggi berdasarkan
        rumus cosine similarity pada semua
        feature/ciri-ciri.
            :param song_name: Nama lagu (all
        letters)
            :param year: Tahun dari lagu tersebut
        [int]
            :param dat: dataset yang dipakai
            :param features_list: List feature yang
        dipakai untuk perhitungan kemiripan
            :param top_n: Banyaknya lagu yang akan
```

```
direkomendasikan
    :param plot_type: Jenis plot yang
dipakai untuk visualisasi (Wordcloud atau
barplot)
    """
    feature_vector, song_repeated =
get_feature_vector(song_name, year, dat,
features_list)
    feature_for_recommendation =
dat[features_list].values

    # menghitung nilai kemiripannya dengna
cosine similarity
    similarities =
cosine_similarity_2d(feature_for_recommenda
tion, feature_vector)
    similarities = np.array(similarities)



    # mengambil index top_n, tidak termasuk
lagu yang diinputkan
    if song_repeated == 0:
        related_song_indices =
similarities.argsort()[-(top_n+1):][::-1][1
:]
    else:
        related_song_indices =
```

```
similarities.argsort()[-(top_n+1+song_repea
ted):][::-1][1+song_repeated:]

    # get the name, artist, and year of the
most similar songs
    similar_songs =
dat.iloc[related_song_indices][['name',
'artists', 'year']]

    names.clear()

    names.append(song_name)

names.extend(similar_songs['name'].head().t
olist())
    #
names.extend(dat.iloc[related_song_indices]
['name'].tolist())

    fig, ax = plt.subplots(figsize=(7, 5))
    if plot_type == 'wordcloud':
        # #Membuat Word cloud dari lagu dan
tahun yang mirip, menggunakan nilai
kemiripan untuk menentukan ukuran wordnya

        similar_songs['name+year'] =
similar_songs['name'] + ' (' +
similar_songs['year'].astype(str) + ')'
```

```
        # Membuat dictionary dari lagu dan
nilai kemiripannya
        song_similarity =
dict(zip(similar_songs['name+year'],
similarities[related_song_indices]))

        # Mengurutkan dictionarynya
berdasarkan nilainya
        song_similarity =
sorted(song_similarity.items(), key=lambda
x: x[1], reverse=True)



        # membuat word cloudnya
        wordcloud = WordCloud(width=1200,
height=600, max_words=50,

background_color='white',
colormap='Set2').generate_from_frequencies(
dict(song_similarity))
        plt.imshow(wordcloud,
interpolation='bilinear')
        plt.axis('off')
        plt.title(f'{top_n} most similar
songs to: {song_name} ({year})',
fontsize=16)
```

```
            plt.tight_layout(pad=0)

        elif plot_type == 'bar':
            similar_songs['artists'] =
similar_songs['artists'].apply(lambda s:
s[2:len(s)-2])
            # # Menggambarkan text dari lagu
dan tahur termirip secara berurut dalam
bentuk bar chart

            similar_songs['name+year'] =
similar_songs['name'] + ' - ' +
similar_songs['artists'] +' (' +
similar_songs['year'].astype(str) + ')'

            # Membuat dictionary dari lagu dan
nilai kemiripannya
            song_similarity =
dict(zip(similar_songs['name+year'],
similarities[related_song_indices]))

            # Mengurutkan dictionarynya
berdasarkan nilainya
            song_similarity =
sorted(song_similarity.items(), key=lambda
x: x[1], reverse=True)

            # Menggambarkan text dari lagu dan
```

```
        tahur termirip secara berurut dalam bentuk
        bar chart

        plt.barh(range(len(song_similarity)),
        [val[1] for val in song_similarity],
                        align='center',
        color=sns.color_palette('pastel',
        len(song_similarity)))

        plt.yticks(range(len(song_similarity)),
        [val[0] for val in song_similarity])
                plt.gca().invert_yaxis()
                plt.title(f'{top_n} most similar
        songs to: {song_name} ({year})',
        fontsize=16)
                min_similarity =
        min(similarities[related_song_indices])
                max_similarity =
        max(similarities[related_song_indices])

                # Menambahkan nama lagu disetiap
        bar chart
                for i, v in enumerate([val[0] for
        val in song_similarity]):
                    plt.text(min_similarity*0.955,
        i, v, color='black', fontsize=8)

                plt.xlabel('Similarity',
```

```
        fontsize=15)
                # plt.ylabel('Song', fontsize=15)
                plt.xlim(min_similarity*0.95,
        max_similarity)

                # menghilangkan frame dan tanda
                plt.box(False)
                plt.tick_params(axis='both',
        which='both', bottom=False, top=False,
        labelbottom=False, left=False, right=False,
        labelleft=False)

            else:
                raise Exception('Plot type must be
        either wordcloud or bar!')

            return fig

        def radar_chart(dat, features_list):
            # Membuat Radar Chart
            fig = go.Figure()
            angles =
        list(dat[features_list].columns)
            angles.append(angles[0])
            layoutdict = dict(
                        radialaxis=dict(
                        visible=True,
                        range=[0, 1]
```

```
                    ))


        for i in range(len(names)):
            subset = dat[dat['name'] ==
    names[i]]
            data = [np.mean(subset[col]) for
    col in subset[features_list].columns]
            data.append(data[0])
            fig.add_trace(go.Scatterpolar(
                r=data,
                theta=angles,
                fill='toself',
                name=names[i]))



        fig.update_layout(
            polar=layoutdict,
            showlegend=True,
            template='plotly_dark'
        )

        return fig
```

search_song.py

```
def search_song(song_name, dat):
    dat_song = dat.query('name == @song_name')
    if dat_song.shape[0] == 0:
        found_flag = False
        found_song = None
        # raise Exception('The song does not
exist in the dataset!')
    else:
        found_flag = True
        found_song = dat_song[['name', 'artists',
'release_date']].to_numpy()
         # print(f"Great! This song is in the
dataset: \n {dat_song[['name', 'artists',
'release_date']].to_numpy()}")
    return found_flag, found_song
```

BAB IV
PEMBAHASAN

4.1 Preprocessing Data

1. Data Spotify

Data yang diambil adalah data berupa karakteristik dari sebuah lagu. Ciri-ciri tersebut dibagi menjadi dua, *metadata* dan *audio feature*. Metadata terdiri dari nama lagu, nama artis, tahun rilis, genre lagu, dan lain-lain. Sedangkan *audio feature* adalah ciri-ciri dari audio dari lagu tersebut.

Data tersebut didapatkan dari dataset di kaggle.

2. Karakteristik yang dipakai

Karakteristik yang dipakai bisa dibagi menjadi tiga:

a. Karakteristik info. Dengan rincian:
   i. nama lagu
   ii. nama artis
   iii. id
   iv. tanggal rilis
   v. tahun
   vi. Kepopuleran

   Tetapi karakteristik yang diambil untuk penilaian *cosine similarity* hanyalah karakteristik kepopuleran.

b. Karakteristik ternormalisasi

   Karakteristik ini adalah karakteristik yang memiliki ukuran ternormalisasi. Karakteristik itu dengan rincian:
   i. *valence*: Menggambarkan tingkat positif atau negatifnya emosi dalam sebuah lagu.
   ii. *acoustic ness*: Menunjukkan seberapa banyak unsur-unsur akustik ada dalam rekaman musik, dibandingkan dengan unsur elektronik.
   iii. *danceability*: Menilai sejauh mana lagu cocok untuk menari berdasarkan ritme, kecepatan, dan regangan musiknya.

iv. *energy*: Mengukur intensitas dan kekuatan keseluruhan dari sebuah lagu.

v. *instrumentales*: Menunjukkan seberapa besar kehadiran vokal dalam sebuah lagu. Semakin tinggi nilai *instrumental ness*, semakin sedikit atau tidak adanya vokal.

vi. *liveness*: Menggambarkan sejauh mana kesan live performance atau rekaman langsung terdengar dalam sebuah lagu.

vii. *speechiness*: Menilai seberapa banyak unsur pidato atau rap yang ada dalam sebuah lagu, dibandingkan dengan nyanyian atau instrumen musik.

c. Karakteristik belum ternomalisasi

Karakteristik ini adalah karakteristik yang tidak ada ukuran yang teratur. Karakteristik itu dengan rincian:

a. *duration ms*: Durasi atau panjang lagu dalam milidetik (ms), menunjukkan lama waktu lagu berlangsung.

b. *key*: Menunjukkan berapa banyak kunci musik (misalnya, C major, D minor) yang dipakai dalam lagu.

c. *loudness*: Tingkat atau intensitas relatif dari suara dalam lagu

d. mode: Menunjukkan apakah lagu tersebut dalam mode mayor (biasanya lebih ceria) atau minor (biasanya lebih gelap)

e. tempo: Kecepatan atau tempo dari lagu, diukur dalam jumlah ketukan per menit (beats per minute/BPM), menunjukkan seberapa cepat lagu tersebut dimainkan.

3. Data Scaling dan Kapitalisasi Nama Lagu

Selanjutnya dilakukan *data scaling* pada datasetnya supaya tidak terjadi kecondongan terhadap salah satu karakteristik menggunakan *standard scaler*.

Lalu yang terakhir adalah merubah nama lagu pada dataset menjadi kapital supaya tidak ada kesusahan dalam mencari lagu.

4.2 Pembahasan File app.py

Pada file ini, *di import* beberapa *package* dan *library* yang dipakai, yaitu pandas, *streamlit*, dan beberapa package dari file library buatan, yaitu dari file search_song.py dan run_recommender.py.

Dengan menggunakan *library streamlit*, kami membuat tampilan *GUI* yang lebih menarik. Sebelum dimasukkan nama lagu, pada *GUI* diberikan fitur untuk memilih ciri-ciri yang ingin dijadikan penilaian persamaan dan berapa banyak rekomendasi lagu yang akan diberikan. Pada *GUI* tersebut terdapat *text field* untuk memasukkan nama lagu dan tahun lagu. Jika tidak yakin akan tahun lagu yang dicari, maka bisa didapatkan dengan mengklik tombol "Cari Lagu saya" sehingga bisa didapatkan tahun lagu tersebut menggunakan fungsi dari file search_song.py.

Setelah itu terdapat tombol "Dapatkan Rekomendasi!" untuk mendapatkan lagu-lagu yang direkomendasikan. Cara untuk mendapatkan rekomendasi adalah dengan menjalankan fungsi dari file run_recommender.py yaitu fungsi show_similar_song. Lalu setelahnya ditampilkan lagu yang direkomendasikan dalam bentuk bar plot dan *word cloud*. Selain itu terdapat *radar chart* yang menampilkan seberapa kemiripan yang dimiliki oleh lagu yang diinputkan dengan 5 lagu teratas yang dinilai mirip.

4.3 Pembahasan File run_recommender.py

1. fungsi get_feature_vector

   Fungsi ini berguna untuk mencari nilai karakteristik dari lagu yang dicari rekomendasinya lalu dimasukkan ke dalam variabel feature_vector. Selain itu, Ini berguna untuk menandai index dari nama lagu tersebut supaya tidak masuk ke dalam rekomendasi.

2. fungsi dot_product

   Fungsi untuk melakukan perkalian *dot* antar dua matriks

3. fungsi cosine_similarity

Fungsi ini berguna untuk mencari nilai *cosine similarity* antara lagu yang diinputkan dengan setiap lagu di dataset. Nilai kemiripan tersebut akan dimasukkan ke dalam array bernama *similarities*.

Parameter array1 adalah dataset lagu yang akan dibandingkan. Parameter array2 adalah nilai *feature* dari lagu yang dimasukkan.

4. fungsi show_similar_songs

Fungsi ini adalah fungsi utama untuk mendapatkan lagu dengan tingkat kemiripan berdasarkan rumus *cosine similarity* yang sudah dijabarkan pada fungsi *cosine_similarity*.

Pertama Ia akan mengekstrak nilai karakteristik dari lagu yang dimasukkan ke dalam variabel feature_vector. Lalu dengan feature_for_recommendation dan feature_vector, akan dicari nilai *cosine similarity* dari setiap perbandingan lagu yang dimasukkan dengan semua lagu dalam dataset, lalu dimasukkan ke variabel similarities.

Lalu setelahnya adalah mengambil index top_n teratas dari variabel *similarities* yang telah diurutkan.

Lalu dari index tersebut diambil nama lagu, nama artis, dan tahun lagu dari dataset dan dimasukkan ke vaiabel similar_songs.

Selanjutnya adalah pembuatan *wordcloud* dan *barplot* untuk visualisasi lagu yang dijadikan rekomendasi. Pembuatan *wordcloud* dan *barplot* menggunakan *matplotlib*.

5. fungsi radar_chart

Fungsi ini bertujuan untuk membuat *radar chart* menggunakan library Plotly dalam bahasa pemrograman *Python*.

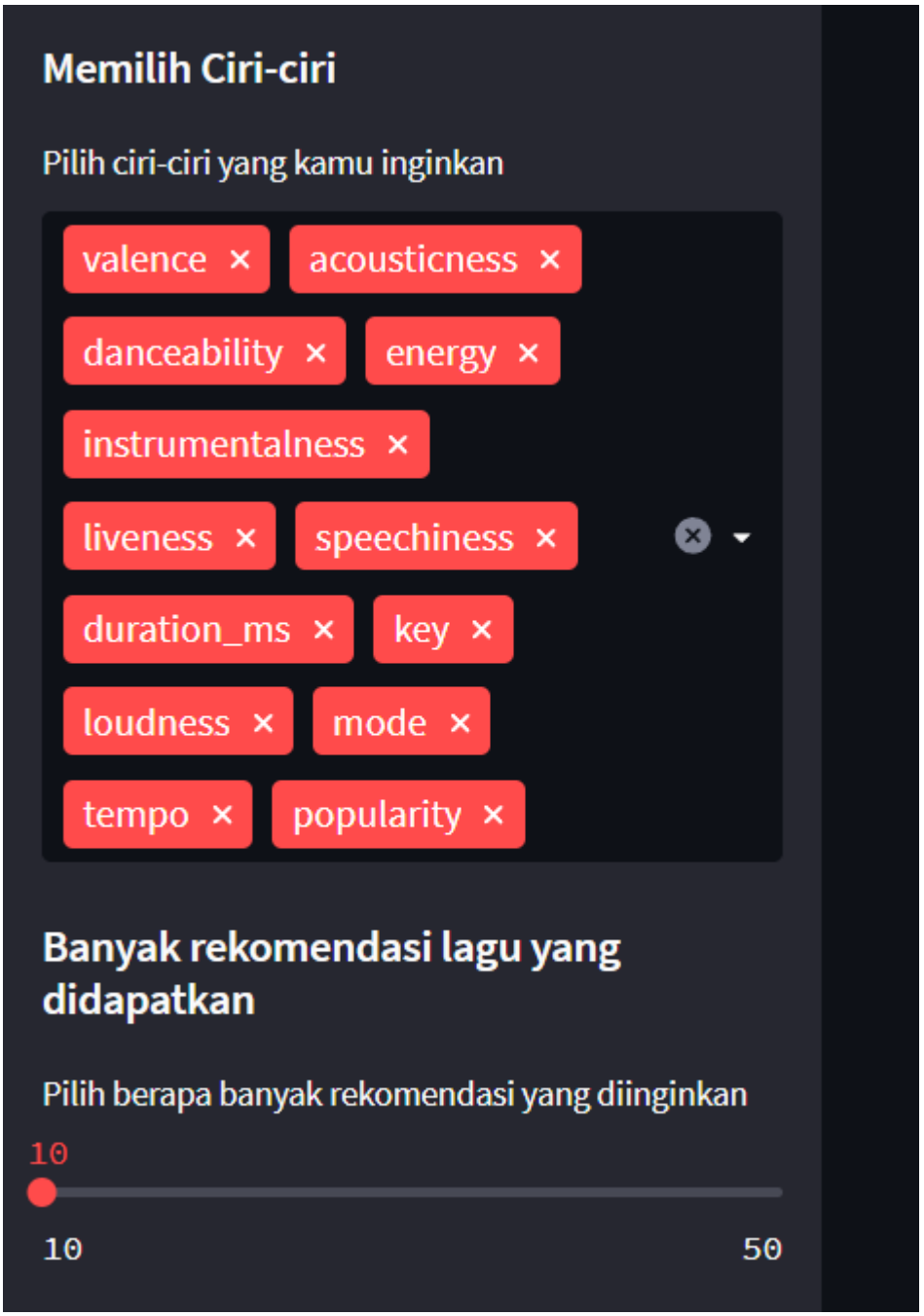
Pertama-tama, objek kosong untuk menampung plot dibuat dengan bantuan `go.Figure()`. Sudut-sudut untuk *radar chart* ditentukan dari kolom-kolom yang dipilih dalam DataFrame `dat` menggunakan `features_list`.

Setelah itu, sudut pertama ditambahkan kembali ke daftar sudut untuk menutup lingkaran radar. Selanjutnya, dilakukan iterasi melalui setiap entitas dalam `names`. Pada setiap iterasi, subset dari data diambil berdasarkan entitas yang sesuai dengan nilai 'name'. Nilai rata-rata dari setiap fitur dalam `features_list` pada subset tersebut dihitung dan disimpan dalam list `data`. Nilai pertama kembali ditambahkan ke dalam list `data` untuk menutup lingkaran radar.

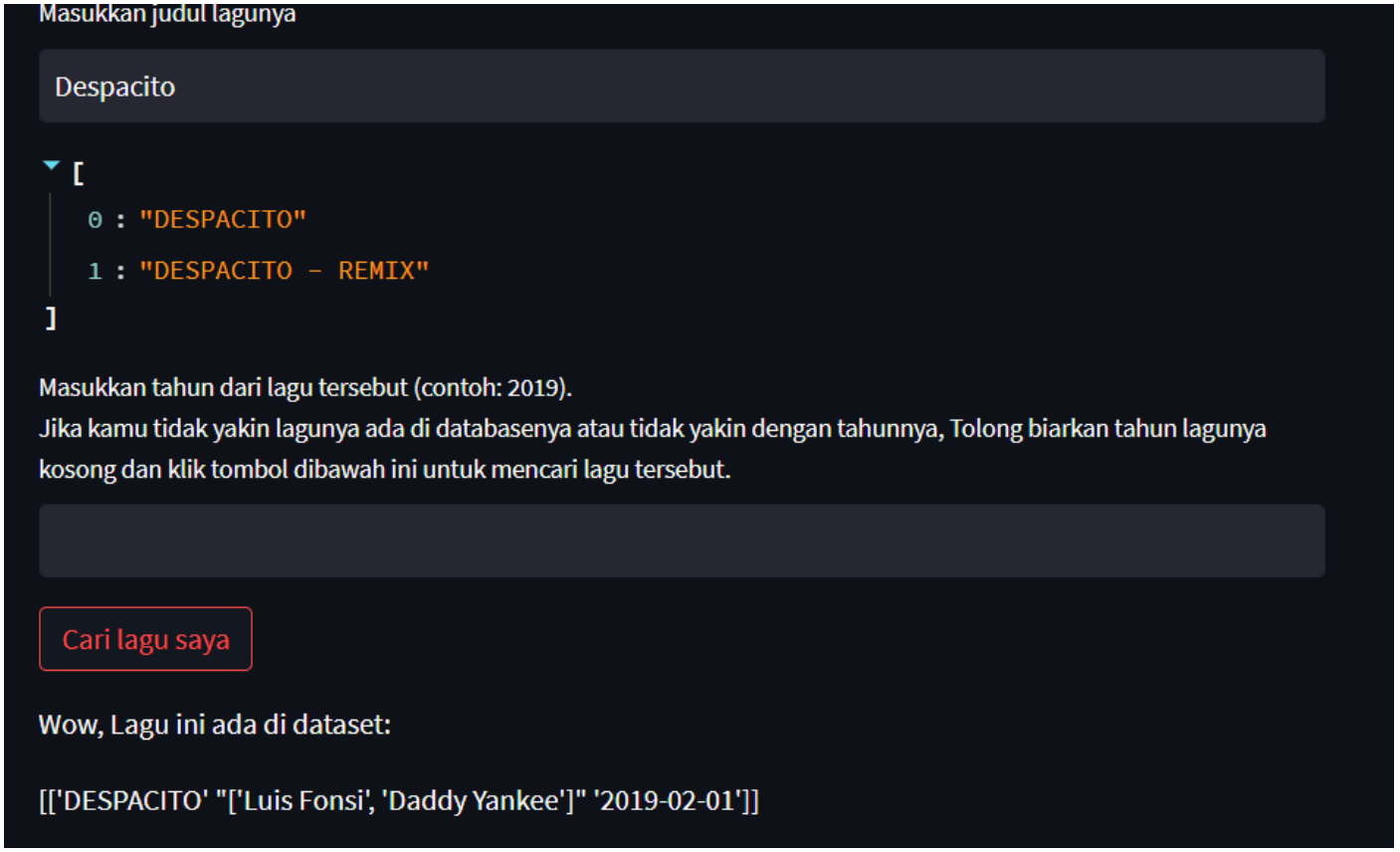
Radar chart kemudian dibuat dengan menambahkan jejak atau trace untuk setiap entitas menggunakan `go.Scatterpolar`. Jejak ini mewakili nilai-nilai rata-rata dari fitur-fitur yang dipilih untuk setiap entitas dalam `names`, dan chart tersebut memvisualisasikan perbedaan relatif antara entitas-entitas tersebut berdasarkan fitur-fitur yang dipilih.

BAB V
SCREENSHOT

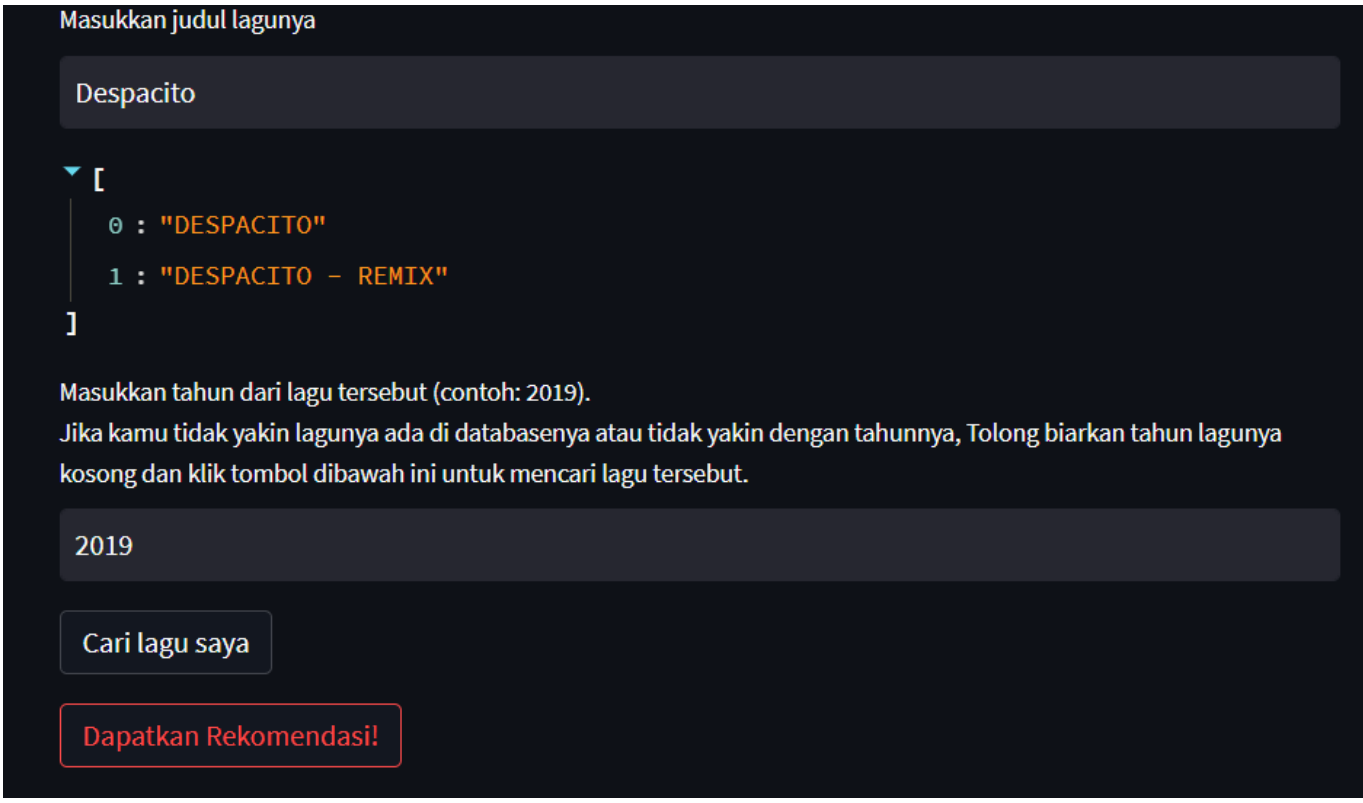
5.1 Pemilihan Ciri-Ciri dan Banyak Rekomendasi Lagu



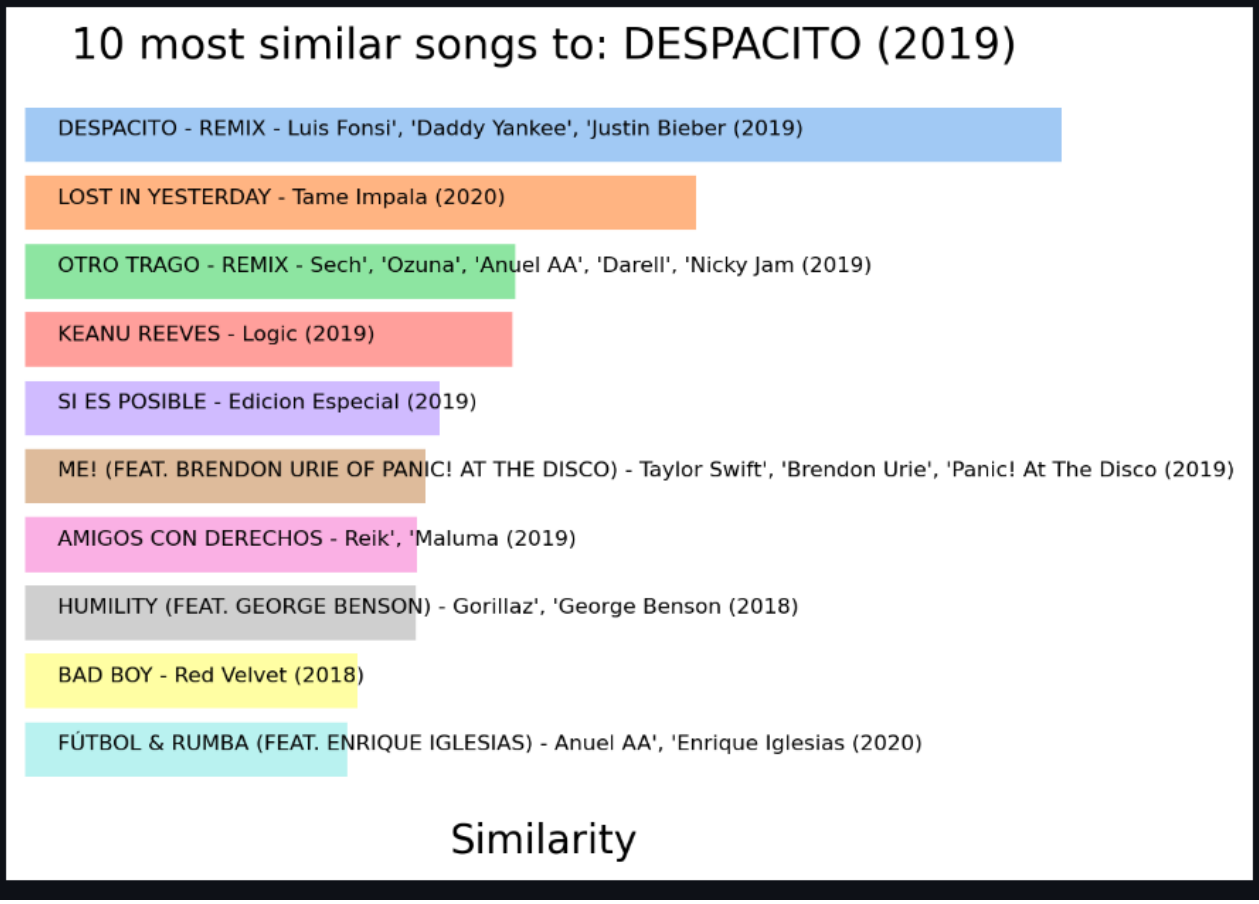
5.2 Pencarian lagu



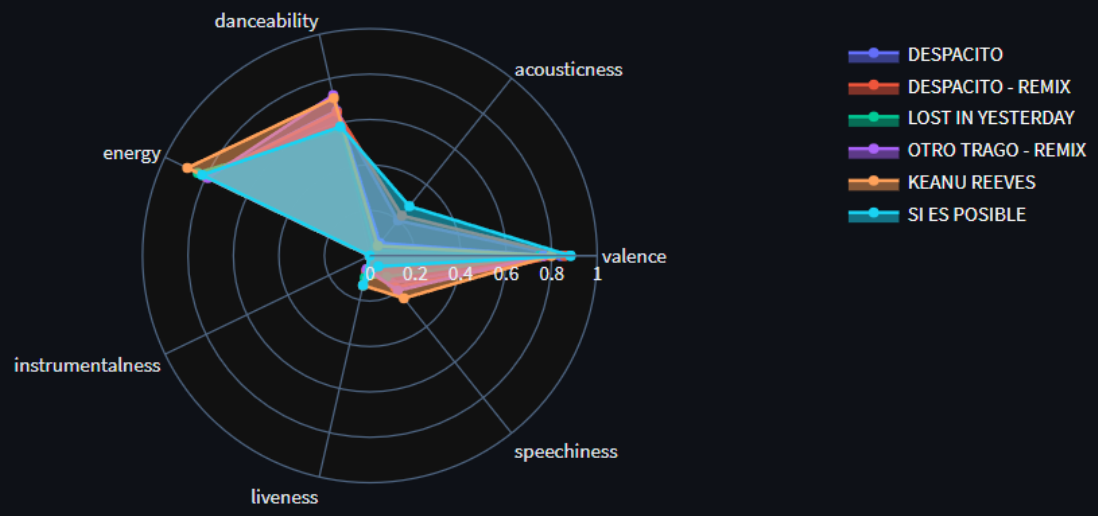
5.3 Mendapatkan Rekomendasi Lagu



Lihatlah lebih dekat dari 10 lagu rekomendasi untukmu!



Gambaran Kemiripan Ciri-Ciri lagumu dengan Rekomendasinya!



BAB VI
KESIMPULAN

6.1 Kesimpulan

Dalam makalah ini, telah dibahas implementasi *Cosine Similarity Function* dalam pembuatan sistem rekomendasi lagu berbasis *Content Based Filtering*. Metode ini mempertimbangkan kemiripan antara karakteristik lagu yang disukai pengguna dengan lagu-lagu lain dalam *dataset*.

Cosine Similarity Function digunakan untuk mengukur kemiripan antara dua vektor karakteristik lagu. Nilai *cosine similarity* berkisar antara 0 dan 1, di mana nilai yang lebih tinggi menunjukkan kemiripan yang lebih besar antara dua lagu.

Sistem rekomendasi yang dikembangkan menggunakan Python, dengan memanfaatkan beberapa library seperti Streamlit untuk membuat antarmuka pengguna yang interaktif, Matplotlib untuk visualisasi data, dan *Plotly* untuk membuat *radar chart*. Preprocessing data dilakukan untuk menskalakan data dan normalisasi beberapa fitur lagu.

Pengujian kinerja sistem dilakukan dengan memberikan input lagu, dan sistem menghasilkan rekomendasi lagu berdasarkan kemiripan karakteristik. Evaluasi kinerja sistem dilakukan dengan menganalisis sejauh mana rekomendasi lagu sesuai dengan preferensi pengguna.

Keterbatasan yang mungkin muncul dalam penggunaan *Cosine Similarity Function* adalah terkait dengan representasi vektor karakteristik lagu. Jika karakteristik yang digunakan tidak mencakup aspek-aspek yang relevan dengan preferensi pengguna, hasil rekomendasi mungkin tidak akurat. Selain itu, sistem ini bersifat content-based, yang berarti rekomendasi hanya didasarkan pada karakteristik lagu yang disukai pengguna sebelumnya, tanpa mempertimbangkan preferensi pengguna lainnya.

Pengembangan lebih lanjut dapat dilakukan dengan mengintegrasikan metode *Content Based Filtering* dengan *metode Collaborative Filtering* untuk meningkatkan akurasi rekomendasi. Selain itu, penambahan fitur-fitur baru atau penggunaan metode *deep learning* dalam pemrosesan karakteristik lagu dapat menjadi langkah-langkah untuk meningkatkan keakuratan sistem rekomendasi.

Dengan menggunakan metode ini, diharapkan sistem rekomendasi lagu dapat memberikan pengalaman yang lebih personal dan sesuai dengan preferensi pengguna, serta dapat menjadi dasar untuk pengembangan sistem rekomendasi yang lebih kompleks di masa depan.

6.2 Saran

Meningkatkan kualitas rekomendasi, dapat dilakukan penelitian lebih lanjut dalam mengidentifikasi dan menggunakan fitur-fitur audio yang lebih kompleks dan mendalam. Peningkatan representasi lagu akan membantu meningkatkan keakuratan sistem rekomendasi.

Menggabungkan metode *Content-Based Filtering* dengan metode *Collaborative Filtering* dapat mengatasi beberapa keterbatasan *Cosine Similarity*. Pendekatan kombinatif dapat memberikan rekomendasi yang lebih holistik dengan mempertimbangkan preferensi pengguna dan interaksi dengan pengguna lain.

Mengeksplorasi algoritma *machine learning* seperti *k-Nearest Neighbors* atau algoritma clustering dapat menjadi alternatif untuk meningkatkan kemampuan sistem rekomendasi dalam menangani kompleksitas preferensi pengguna.

Melakukan evaluasi sistem rekomendasi secara berkala dengan melibatkan pengguna untuk mendapatkan umpan balik langsung. Ini membantu memahami sejauh mana kepuasan pengguna terpenuhi dan memungkinkan penyesuaian berkelanjutan.

Dengan mengambil langkah-langkah ini, diharapkan sistem rekomendasi lagu dapat terus ditingkatkan dalam hal keakuratan dan ketepatan, memberikan pengalaman yang lebih memuaskan bagi pengguna.

DAFTAR PUSTAKA

Isinkaye, F. O., Folajimi, Y. O. and Ojokoh, B. A. (2015) 'Recommendation systems: Principles, methods and evaluation', Egyptian Informatics Journal. Ministry of Higher Education and Scientific Research, 16(3), pp. 261–273. doi: 10.1016/j.eij.2015.06.005.

Reddy, S. R. S. et al. (2019) 'Content-Based Movie Recommendation System Using Genre Correlation', in Satapathy, S. C., Bhateja, V., and Das, S. (eds) Smart Intelligent Computing and Applications. Singapore: Springer Singapore, pp. 391–397.

Fauzi, M. A., Arifin, A. Z. and Yuniarti, A. (2017) 'Arabic book retrieval using class and book index based term weighting', International Journal of Electrical and Computer Engineering, 7(6), pp. 3705–3710. doi: 10.11591/ijece.v7i6.pp3705-3711.

Wang, L., Chen, Z. and Wu, J. (2017) 'An opportunistic routing for data forwarding based on vehicle mobility.

Perkovic, Ljubomir 2012. Introduction to Computing Using Python: An Application Development Focus.

Koh, C. W. T., Joly, G. L. C., & Chan, K. R. (2021). Gene Updater : A Streamlit web tool that autocorrects and updates for Excel misidentified gene names. 1–8.

K.Nagesh, D.Nageswara Rao, Song K.Choi, 2015, "Python and Matplotlib based Open Source Software System for Simulating Images with point Light Sources in Attenuating and Scattering Media." International Journal of Computer Applications. Vol.131-No.18

Kaggle dataset:
https://www.kaggle.com/datasets/vatsalmavani/spotify-dataset/data

Inspiration for designing streamlit app:
https://www.kaggle.com/code/yannansu/music-recommender-bonus-streamlit-app#Build-a-recommender-system

LAMPIRAN

- Link Presentasi: https://www.youtube.com/watch?v=hG5U7G7nlHQ
- Pembagian Tugas:
    - Ahmad Faiz Ali Azmi:
        - Membuat *design* PPT
        - *Editing* pada *video*
        - Penentuan ide
        - Membantu penentuan algoritma yang cocok
        - Membantu pembuatan makalah
        - Memberikan penjelasan mengenai bab pembahasan saat presentasi
    - Ananda Ravi Kuntadi:
        - Penentuan ide
        - Membantu pembuatan makalah terutama merapikan, membuat kesimpulan dan latar belakang
        - Menentukan algoritma yang ingin digunakan yaitu *content based filtering*
        - Melakukan *recording* pada presentasi
        - Memberikan penjelasan mengenai bab latar belakang dan *content based filtering*
    - Davin Dalana Fidelio Fredra:
        - Penentuan ide
        - Membantu membuat makalah pada bagian latar belakang
        - Memberikan penjelasan mengenai bab kesimpulan
        -
    - Muhammad Yasin Hakim:
        - Pembuatan / Optimalisasi Aplikasi
        - Penentuan Ide
        - Menentukan penggunaan *cosine similarity*
        - Membantu pembuatan makalah
        - Memberikan pembagian tugas
        - Memberikan penjelasan mengenai source code
    - Salsa Zufar Radinka Akmal:
        - Mencari referensi aplikasi
        - Penentuan ide
        - menentukan penggunaan *cosine similarity*
        - Membantu pembuatan makalah pada bagian tinjauan pustaka
        - Memberikan penjelasan mengenai bab tinjauan pustaka
- Link Aplikasi: https://projectkb-spotipuyrekomender.streamlit.app/
- Link Source code: https://github.com/Y716/ProjectKB