yalinlyrics

October 29, 2024

[7]:
```
pip install python-docx pandas
```

```
Requirement already satisfied: python-docx in /usr/local/lib/python3.10/dist-
packages (1.1.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages
(2.2.2)
Requirement already satisfied: lxml>=3.1.0 in /usr/local/lib/python3.10/dist-
packages (from python-docx) (4.9.4)
Requirement already satisfied: typing-extensions>=4.9.0 in
/usr/local/lib/python3.10/dist-packages (from python-docx) (4.12.2)
Requirement already satisfied: numpy>=1.22.4 in /usr/local/lib/python3.10/dist-
packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-
packages (from pandas) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
```

[9]:
```
from docx import Document
from docx.shared import Pt
import pandas as pd

# Word dosyasını yükle
doc_path = "input.docx"  # Giriş dosyanızın adını buraya yazın
doc = Document(doc_path)

# Şarkı isimleri ve sözlerini saklamak için listeler
song_names = []
song_lyrics = []

# Geçici değişkenler
current_song_name = None
current_lyrics = []

# Word dosyasındaki paragrafları incele
```

```
for paragraph in doc.paragraphs:
    # Şarkı isimlerini bulmak için (bold, italik, font size 18)
    if paragraph.text.strip() and paragraph.runs[0].bold and paragraph.runs[0].
 ↪italic and paragraph.runs[0].font.size == Pt(18):
        # Önceki şarkı ve sözleri kaydet
        if current_song_name and current_lyrics:
            song_names.append(current_song_name)
            song_lyrics.append("\n".join(current_lyrics))

        # Yeni şarkı ismini al ve sözleri sıfırla
        current_song_name = paragraph.text.strip()
        current_lyrics = []
    else:
        # Şarkı sözleri olan 12 pt metinleri topla
        if paragraph.text.strip() and paragraph.runs[0].font.size == Pt(12):
            current_lyrics.append(paragraph.text.strip())

# Son şarkıyı ekle
if current_song_name and current_lyrics:
    song_names.append(current_song_name)
    song_lyrics.append("\n".join(current_lyrics))

# Şarkı isimleri ve sözlerini içeren bir DataFrame oluştur
df_songs = pd.DataFrame({"Song Name": song_names, "Lyrics": song_lyrics})

# DataFrame'i CSV olarak kaydet
output_csv_path = "song_lyrics_separated.csv"  # Çıktı dosyasının adını burada␣
 ↪belirtin
df_songs.to_csv(output_csv_path, index=False)

print(f"CSV dosyası başarıyla kaydedildi: {output_csv_path}")
```

```
CSV dosyası başarıyla kaydedildi: song_lyrics_separated.csv
```

[10]:
```
from docx import Document
from docx.shared import Pt
import pandas as pd

# Word dosyasını yükle
doc_path = "input.docx"  # Giriş dosyanızın adını buraya yazın
doc = Document(doc_path)

# Şarkı isimleri ve sözlerini saklamak için listeler
song_names = []
song_lyrics = []

# Geçici değişkenler
```

```
current_song_name = None
current_lyrics = []

# Word dosyasındaki paragrafları incele
for paragraph in doc.paragraphs:
    # Şarkı isimlerini bulmak için (font size 18)
    if paragraph.text.strip() and paragraph.runs[0].font.size == Pt(18):
        # Önceki şarkı ve sözleri kaydet
        if current_song_name and current_lyrics:
            song_names.append(current_song_name)
            song_lyrics.append("\n".join(current_lyrics))

        # Yeni şarkı ismini al ve sözleri sıfırla
        current_song_name = paragraph.text.strip()
        current_lyrics = []
    else:
        # Şarkı sözlerini toplamak için (herhangi bir yazı olabilir)
        if paragraph.text.strip():
            current_lyrics.append(paragraph.text.strip())

# Son şarkıyı ekle
if current_song_name and current_lyrics:
    song_names.append(current_song_name)
    song_lyrics.append("\n".join(current_lyrics))

# Şarkı isimleri ve sözlerini içeren bir DataFrame oluştur
df_songs = pd.DataFrame({"Song Name": song_names, "Lyrics": song_lyrics})

# DataFrame'i CSV olarak kaydet
output_csv_path = "song_lyrics_separated.csv"  # Çıktı dosyasının adını burada␣
 ↪belirtin
df_songs.to_csv(output_csv_path, index=False)

print(f"CSV dosyası başarıyla kaydedildi: {output_csv_path}")
```

```
CSV dosyası başarıyla kaydedildi: song_lyrics_separated.csv
```

[11]:
```
from docx import Document
from docx.shared import Pt
import pandas as pd
import re

# Word dosyasını yükle
doc_path = "input.docx"  # Giriş dosyanızın adını buraya yazın
doc = Document(doc_path)

# Şarkı isimleri ve sözlerini saklamak için listeler
```

```
song_names = []
song_lyrics = []

# Geçici değişkenler
current_song_name = None
current_lyrics = []

# Word dosyasındaki paragrafları incele
for paragraph in doc.paragraphs:
    # Şarkı isimlerini bulmak için (font size 18)
    if paragraph.text.strip() and paragraph.runs[0].font.size == Pt(18):
        # Önceki şarkı ve sözleri kaydet
        if current_song_name and current_lyrics:
            song_names.append(current_song_name)
            song_lyrics.append("\n".join(current_lyrics))

        # Yeni şarkı ismini al ve sözleri sıfırla
        current_song_name = paragraph.text.strip()
        current_lyrics = []
    else:
        # Şarkı sözlerini toplamak için (herhangi bir yazı olabilir)
        if paragraph.text.strip():
            current_lyrics.append(paragraph.text.strip())

# Son şarkıyı ekle
if current_song_name and current_lyrics:
    song_names.append(current_song_name)
    song_lyrics.append("\n".join(current_lyrics))

# Şarkı isimleri ve sözlerini içeren bir DataFrame oluştur ve numara ekle
df_songs = pd.DataFrame({"Number": range(1, len(song_names) + 1), "Song Name":␣
 ↪song_names, "Lyrics": song_lyrics})

# Fonksiyon: Şarkı sözlerini büyük harfle başlayan her cümleyi yeni satıra␣
 ↪alacak şekilde düzenle
def format_lyrics(lyrics):
    # Büyük harfle başlayan cümleleri yeni satıra alma
    sentences = re.split(r'(?<!\w\.\w.)(?<![A-Z][a-z]\.)(?<=\.|\?|!)(?
 ↪=\s+[A-Z])', lyrics)
    formatted_lyrics = "\n".join(sentence.strip() for sentence in sentences if␣
 ↪sentence)
    return formatted_lyrics

# Şarkı sözlerini formatlama ve yeni DataFrame'e aktarma
df_songs['Formatted Lyrics'] = df_songs['Lyrics'].apply(format_lyrics)

# DataFrame'i CSV olarak kaydet
```

```
output_csv_path = "song_lyrics_formatted.csv"  # Çıktı dosyasının adını burada␣
 ↪belirtin
df_songs.to_csv(output_csv_path, index=False)

print(f"CSV dosyası başarıyla kaydedildi: {output_csv_path}")
```

```
CSV dosyası başarıyla kaydedildi: song_lyrics_formatted.csv
```

Analiz Aşaması En Çok Tekrar Eden Kelimeleri ve Kelime Bulutu

[12]:
```
# Gerekli kütüphaneleri yükleyin
!pip install wordcloud matplotlib nltk sklearn

# NLTK stop words'i yüklemek için
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
import matplotlib.pyplot as plt
from wordcloud import WordCloud

# song_lyrics_formatted.csv dosyasını oku
df_songs = pd.read_csv("song_lyrics_formatted.csv")  # Colab'e yüklediğiniz␣
 ↪dosyanın ismi

# Türkçe stop words listesini ayarlayın
turkish_stop_words = stopwords.words('turkish')

# Tüm şarkı sözlerini birleştirin
all_lyrics = " ".join(df_songs['Formatted Lyrics'])

# Kelime frekansı analizini başlatın
vectorizer = CountVectorizer(stop_words=turkish_stop_words, max_features=50)  #␣
 ↪En sık kullanılan 50 kelime
word_counts = vectorizer.fit_transform([all_lyrics])
word_counts_array = word_counts.toarray().flatten()

# Kelimeler ve frekanslarını alalım
words = vectorizer.get_feature_names_out()
frequencies = dict(zip(words, word_counts_array))

# Kelime bulutu ile görselleştirme
wordcloud = WordCloud(width=800, height=400, background_color="white").
 ↪generate_from_frequencies(frequencies)
```

```
# Word Cloud'u göster
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```

```
Requirement already satisfied: wordcloud in /usr/local/lib/python3.10/dist-
packages (1.9.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-
packages (3.7.1)
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages
(3.8.1)
Collecting sklearn
  Downloading sklearn-0.0.post12.tar.gz (2.6 kB)
  error: subprocess-exited-with-error

  × python setup.py egg_info did not run successfully.
   exit code: 1
   > See above for output.

  note: This error originates from a subprocess, and is likely not a
problem with pip.
  Preparing metadata (setup.py) … error
error: metadata-generation-failed

× Encountered error while generating package metadata.
 > See above for output.

note: This is an issue with the package mentioned above, not pip.
hint: See above for details.
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
```

[13]:
```
# Gerekli kütüphaneleri yükleyin
!pip install transformers torch pandas matplotlib

from transformers import pipeline
import pandas as pd
import matplotlib.pyplot as plt

# song_lyrics_formatted.csv dosyasını oku
df_songs = pd.read_csv("song_lyrics_formatted.csv")  # Colab'e yüklediğiniz␣
 ↪dosyanın adı

# Duygu analizi için çok dilli model yükle
sentiment_model = pipeline("sentiment-analysis", model="nlptown/
 ↪bert-base-multilingual-uncased-sentiment")

# Duygu analizi fonksiyonu
def get_sentiment_score(text):
    result = sentiment_model(text)[0]
    label = result['label']
    score = int(label.split()[0])  # Puanı al
    return score

# Her şarkı için duygu skorunu hesaplayın
df_songs['Sentiment Score'] = df_songs['Formatted Lyrics'].
 ↪apply(get_sentiment_score)

# Ortalama duygu skorunu hesaplayın ve genel dağılımı inceleyin
average_sentiment = df_songs['Sentiment Score'].mean()
```

```
print(f"Ortalama Duygu Skoru: {average_sentiment}")

# Duygu skorlarını görselleştirin
plt.figure(figsize=(10, 6))
plt.hist(df_songs['Sentiment Score'], bins=5, color='skyblue',␣
 ↪edgecolor='black')
plt.title('Şarkıların Duygu Skor Dağılımı')
plt.xlabel('Duygu Skoru')
plt.ylabel('Şarkı Sayısı')
plt.show()

# En pozitif ve en negatif şarkıları belirleyin
most_positive = df_songs[df_songs['Sentiment Score'] == df_songs['Sentiment␣
 ↪Score'].max()]
most_negative = df_songs[df_songs['Sentiment Score'] == df_songs['Sentiment␣
 ↪Score'].min()]

print("En Pozitif Şarkılar:")
print(most_positive[['Song Name', 'Sentiment Score']])

print("\nEn Negatif Şarkılar:")
print(most_negative[['Song Name', 'Sentiment Score']])
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-
packages (4.44.2)
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages
(2.5.0+cu121)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages
(2.2.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-
packages (3.7.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from transformers) (3.16.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.23.2 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.24.7)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-
packages (from transformers) (1.26.4)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from transformers) (24.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-
packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.10/dist-packages (from transformers) (2024.9.11)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from transformers) (2.32.3)
Requirement already satisfied: safetensors>=0.4.1 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.4.5)
```

```
Requirement already satisfied: tokenizers<0.20,>=0.19 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.19.1)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-
packages (from transformers) (4.66.5)
Requirement already satisfied: typing-extensions>=4.8.0 in
/usr/local/lib/python3.10/dist-packages (from torch) (4.12.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-
packages (from torch) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from torch) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from torch) (2024.6.1)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/dist-
packages (from torch) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from sympy==1.13.1->torch) (1.3.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-
packages (from pandas) (2024.2)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (4.54.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.7)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (3.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch) (3.0.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers)
(2024.8.30)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89:
UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab
(https://huggingface.co/settings/tokens), set it as secret in your Google Colab
and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access
public models or datasets.
  warnings.warn(

config.json:   0%|          | 0.00/953 [00:00<?, ?B/s]

pytorch_model.bin:   0%|          | 0.00/669M [00:00<?, ?B/s]

tokenizer_config.json:   0%|          | 0.00/39.0 [00:00<?, ?B/s]

vocab.txt:   0%|          | 0.00/872k [00:00<?, ?B/s]

special_tokens_map.json:   0%|          | 0.00/112 [00:00<?, ?B/s]

/usr/local/lib/python3.10/dist-
packages/transformers/tokenization_utils_base.py:1601: FutureWarning:
`clean_up_tokenization_spaces` was not set. It will be set to `True` by default.
This behavior will be depracted in transformers v4.45, and will be then set to
`False` by default. For more details check this issue:
https://github.com/huggingface/transformers/issues/31884
  warnings.warn(
Token indices sequence length is longer than the specified maximum sequence
length for this model (518 > 512). Running this sequence through the model will
result in indexing errors
```

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
<ipython-input-13-2680d367c6ce> in <cell line: 22>()
     20
     21 # Her şarkı için duygu skorunu hesaplayın
---> 22 df_songs['Sentiment Score'] = df_songs['Formatted Lyrics'].
 ↪apply(get_sentiment_score)
     23
     24 # Ortalama duygu skorunu hesaplayın ve genel dağılımı inceleyin

/usr/local/lib/python3.10/dist-packages/pandas/core/series.py in apply(self,␣
 ↪func, convert_dtype, args, by_row, **kwargs)
   4922             args=args,
   4923             kwargs=kwargs,
-> 4924         ).apply()
   4925
   4926     def _reindex_indexer(
```

```
/usr/local/lib/python3.10/dist-packages/pandas/core/apply.py in apply(self)
   1425
   1426         # self.func is Callable
-> 1427         return self.apply_standard()
   1428
   1429     def agg(self):

/usr/local/lib/python3.10/dist-packages/pandas/core/apply.py in␣
 ↪apply_standard(self)
   1505         #  Categorical (GH51645).
   1506         action = "ignore" if isinstance(obj.dtype, CategoricalDtype)␣
 ↪else None
-> 1507         mapped = obj._map_values(

   1508             mapper=curried, na_action=action, convert=self.convert_dtyp
   1509         )

/usr/local/lib/python3.10/dist-packages/pandas/core/base.py in _map_values(self ␣
 ↪mapper, na_action, convert)
    919             return arr.map(mapper, na_action=na_action)
    920
--> 921         return algorithms.map_array(arr, mapper, na_action=na_action,␣
 ↪convert=convert)
    922
    923     @final

/usr/local/lib/python3.10/dist-packages/pandas/core/algorithms.py in␣
 ↪map_array(arr, mapper, na_action, convert)
   1741     values = arr.astype(object, copy=False)
   1742     if na_action is None:
-> 1743         return lib.map_infer(values, mapper, convert=convert)
   1744     else:
   1745         return lib.map_infer_mask(

lib.pyx in pandas._libs.lib.map_infer()

<ipython-input-13-2680d367c6ce> in get_sentiment_score(text)
     14 # Duygu analizi fonksiyonu
     15 def get_sentiment_score(text):
---> 16     result = sentiment_model(text)[0]
     17     label = result['label']
     18     score = int(label.split()[0])  # Puanı al

/usr/local/lib/python3.10/dist-packages/transformers/pipelines/
 ↪text_classification.py in __call__(self, inputs, **kwargs)
    154         """
    155         inputs = (inputs,)
--> 156         result = super().__call__(*inputs, **kwargs)
```

```
    157         # TODO try and retrieve it in a nicer way from␣
 ↪_sanitize_parameters.
    158         _legacy = "top_k" not in kwargs

/usr/local/lib/python3.10/dist-packages/transformers/pipelines/base.py in␣
 ↪__call__(self, inputs, num_workers, batch_size, *args, **kwargs)
   1255             )
   1256         else:
-> 1257             return self.run_single(inputs, preprocess_params,␣
 ↪forward_params, postprocess_params)
   1258
   1259     def run_multi(self, inputs, preprocess_params, forward_params,␣
 ↪postprocess_params):

/usr/local/lib/python3.10/dist-packages/transformers/pipelines/base.py in␣
 ↪run_single(self, inputs, preprocess_params, forward_params, postprocess_param
   1262     def run_single(self, inputs, preprocess_params, forward_params,␣
 ↪postprocess_params):
   1263         model_inputs = self.preprocess(inputs, **preprocess_params)
-> 1264         model_outputs = self.forward(model_inputs, **forward_params)
   1265         outputs = self.postprocess(model_outputs, **postprocess_params)
   1266         return outputs

/usr/local/lib/python3.10/dist-packages/transformers/pipelines/base.py in␣
 ↪forward(self, model_inputs, **forward_params)
   1162                 with inference_context():
   1163                     model_inputs = self.
 ↪_ensure_tensor_on_device(model_inputs, device=self.device)
-> 1164                     model_outputs = self._forward(model_inputs,␣
 ↪**forward_params)
   1165                     model_outputs = self.
 ↪_ensure_tensor_on_device(model_outputs, device=torch.device("cpu"))
   1166             else:

/usr/local/lib/python3.10/dist-packages/transformers/pipelines/
 ↪text_classification.py in _forward(self, model_inputs)
    185         if "use_cache" in inspect.signature(model_forward).parameters.
 ↪keys():
    186             model_inputs["use_cache"] = False
--> 187         return self.model(**model_inputs)
    188
    189     def postprocess(self, model_outputs, function_to_apply=None,␣
 ↪top_k=1, _legacy=True):

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/module.py in␣
 ↪_wrapped_call_impl(self, *args, **kwargs)
   1734             return self._compiled_call_impl(*args, **kwargs)  # type:␣
 ↪ignore[misc]
```

```
   1735         else:
-> 1736             return self._call_impl(*args, **kwargs)
   1737
   1738     # torchrec tests the code consistency with the following code

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/module.py in␣
 ↪_call_impl(self, *args, **kwargs)
   1745                 or _global_backward_pre_hooks or _global_backward_hooks
   1746                 or _global_forward_hooks or _global_forward_pre_hooks):
-> 1747             return forward_call(*args, **kwargs)
   1748
   1749         result = None

/usr/local/lib/python3.10/dist-packages/transformers/models/bert/modeling_bert.
 ↪py in forward(self, input_ids, attention_mask, token_type_ids, position_ids,␣
 ↪head_mask, inputs_embeds, labels, output_attentions, output_hidden_states,␣
 ↪return_dict)
   1693         return_dict = return_dict if return_dict is not None else self.
 ↪config.use_return_dict
   1694
-> 1695         outputs = self.bert(
   1696             input_ids,
   1697             attention_mask=attention_mask,

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/module.py in␣
 ↪_wrapped_call_impl(self, *args, **kwargs)
   1734             return self._compiled_call_impl(*args, **kwargs)  # type:␣
 ↪ignore[misc]
   1735         else:
-> 1736             return self._call_impl(*args, **kwargs)
   1737
   1738     # torchrec tests the code consistency with the following code

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/module.py in␣
 ↪_call_impl(self, *args, **kwargs)
   1745                 or _global_backward_pre_hooks or _global_backward_hooks
   1746                 or _global_forward_hooks or _global_forward_pre_hooks):
-> 1747             return forward_call(*args, **kwargs)
   1748
   1749         result = None

/usr/local/lib/python3.10/dist-packages/transformers/models/bert/modeling_bert.
 ↪py in forward(self, input_ids, attention_mask, token_type_ids, position_ids,␣
 ↪head_mask, inputs_embeds, encoder_hidden_states, encoder_attention_mask,␣
 ↪past_key_values, use_cache, output_attentions, output_hidden_states,␣
 ↪return_dict)
   1075                 token_type_ids = torch.zeros(input_shape, dtype=torch.
 ↪long, device=device)
   1076
```

```
-> 1077         embedding_output = self.embeddings(
   1078             input_ids=input_ids,
   1079             position_ids=position_ids,

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/module.py in␣
 ↪_wrapped_call_impl(self, *args, **kwargs)
   1734             return self._compiled_call_impl(*args, **kwargs)  # type:␣
 ↪ignore[misc]
   1735         else:
-> 1736             return self._call_impl(*args, **kwargs)
   1737
   1738     # torchrec tests the code consistency with the following code

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/module.py in␣
 ↪_call_impl(self, *args, **kwargs)
   1745                 or _global_backward_pre_hooks or _global_backward_hooks
   1746                 or _global_forward_hooks or _global_forward_pre_hooks):
-> 1747             return forward_call(*args, **kwargs)
   1748
   1749         result = None

/usr/local/lib/python3.10/dist-packages/transformers/models/bert/modeling_bert.
 ↪py in forward(self, input_ids, token_type_ids, position_ids, inputs_embeds,␣
 ↪past_key_values_length)
    214         if self.position_embedding_type == "absolute":
    215             position_embeddings = self.position_embeddings(position_ids
--> 216             embeddings += position_embeddings
    217         embeddings = self.LayerNorm(embeddings)
    218         embeddings = self.dropout(embeddings)

RuntimeError: The size of tensor a (518) must match the size of tensor b (512)␣
 ↪at non-singleton dimension 1
```

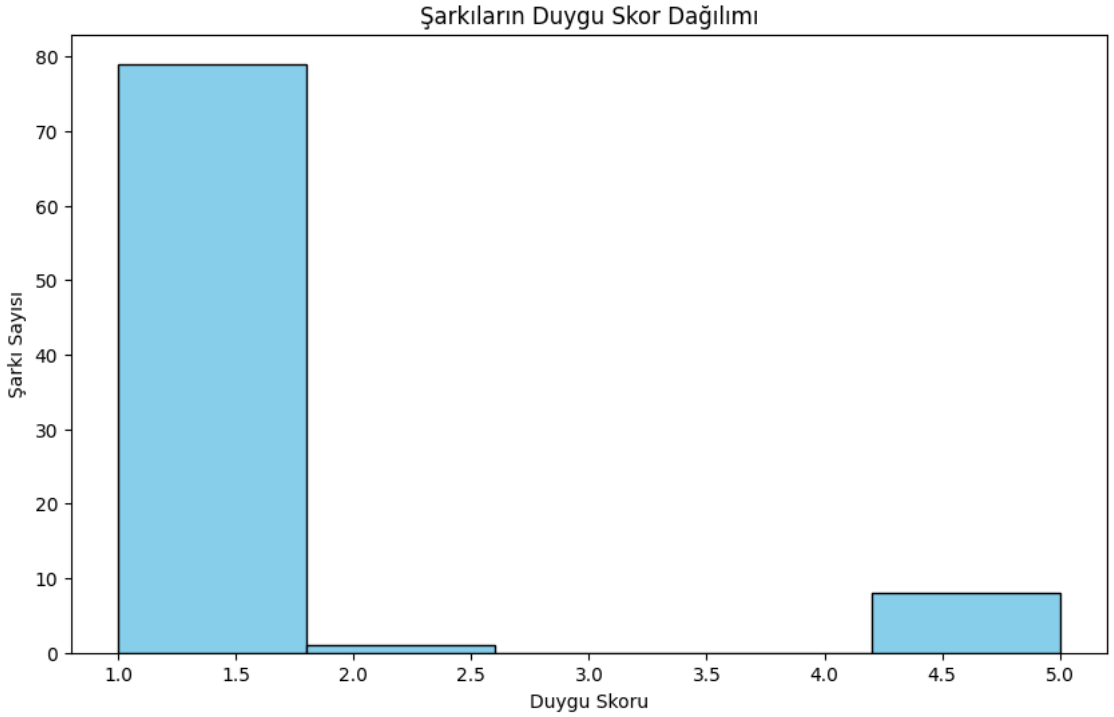
[14]:
```
# Gerekli kütüphaneleri yükleyin
!pip install transformers torch pandas matplotlib

from transformers import pipeline
import pandas as pd
import matplotlib.pyplot as plt

# song_lyrics_formatted.csv dosyasını oku
df_songs = pd.read_csv("song_lyrics_formatted.csv")  # Colab'e yüklediğiniz␣
 ↪dosyanın adı

# Duygu analizi için çok dilli model yükle
```

```
sentiment_model = pipeline("sentiment-analysis", model="nlptown/
 ↪bert-base-multilingual-uncased-sentiment")

# 512 token sınırını aşan şarkı sözlerini kesmek için duygu analizi fonksiyonu
def get_sentiment_score(text):
    truncated_text = text[:512]  # Şarkı sözünü 512 karakter ile sınırlandır
    result = sentiment_model(truncated_text)[0]
    label = result['label']
    score = int(label.split()[0])  # Puanı al
    return score

# Her şarkı için duygu skorunu hesaplayın
df_songs['Sentiment Score'] = df_songs['Formatted Lyrics'].
 ↪apply(get_sentiment_score)

# Ortalama duygu skorunu hesaplayın ve genel dağılımı inceleyin
average_sentiment = df_songs['Sentiment Score'].mean()
print(f"Ortalama Duygu Skoru: {average_sentiment}")

# Duygu skorlarını görselleştirin
plt.figure(figsize=(10, 6))
plt.hist(df_songs['Sentiment Score'], bins=5, color='skyblue',␣
 ↪edgecolor='black')
plt.title('Şarkıların Duygu Skor Dağılımı')
plt.xlabel('Duygu Skoru')
plt.ylabel('Şarkı Sayısı')
plt.show()

# En pozitif ve en negatif şarkıları belirleyin
most_positive = df_songs[df_songs['Sentiment Score'] == df_songs['Sentiment␣
 ↪Score'].max()]
most_negative = df_songs[df_songs['Sentiment Score'] == df_songs['Sentiment␣
 ↪Score'].min()]

print("En Pozitif Şarkılar:")
print(most_positive[['Song Name', 'Sentiment Score']])

print("\nEn Negatif Şarkılar:")
print(most_negative[['Song Name', 'Sentiment Score']])
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-
packages (4.44.2)
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages
(2.5.0+cu121)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages
(2.2.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-
```

```
packages (3.7.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from transformers) (3.16.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.23.2 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.24.7)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-
packages (from transformers) (1.26.4)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from transformers) (24.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-
packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.10/dist-packages (from transformers) (2024.9.11)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from transformers) (2.32.3)
Requirement already satisfied: safetensors>=0.4.1 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.4.5)
Requirement already satisfied: tokenizers<0.20,>=0.19 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.19.1)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-
packages (from transformers) (4.66.5)
Requirement already satisfied: typing-extensions>=4.8.0 in
/usr/local/lib/python3.10/dist-packages (from torch) (4.12.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-
packages (from torch) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from torch) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from torch) (2024.6.1)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/dist-
packages (from torch) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from sympy==1.13.1->torch) (1.3.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-
packages (from pandas) (2024.2)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (4.54.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.7)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
```

```
packages (from matplotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (3.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch) (3.0.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers)
(2024.8.30)

/usr/local/lib/python3.10/dist-
packages/transformers/tokenization_utils_base.py:1601: FutureWarning:
`clean_up_tokenization_spaces` was not set. It will be set to `True` by default.
This behavior will be depracted in transformers v4.45, and will be then set to
`False` by default. For more details check this issue:
https://github.com/huggingface/transformers/issues/31884
  warnings.warn(

Ortalama Duygu Skoru: 1.375
```



```
En Pozitif Şarkılar:
            Song Name  Sentiment Score
19           "Üzülme"                5
40         "Duyulurum"               5
44  "Sen En Güzelsin"                5
45        "Onun Yolu"                5
56  "Nöbetçi Geceler"                5
64         "Aşk Diye"                5
65          "Benimki"                5
75         "İstanbul"                5

En Negatif Şarkılar:
                     Song Name  Sentiment Score
0    "Zalim (Ellerine Sağlık)"                1
1                  "Sonsuz Ol"                1
2                     "Değmez"                1
3                      "Sahte"                1
4                    "Günaydın"                1
..                          …              …
83              "Sensiz Olmaz"                1
84  "Ver O Zaman Gömleklerimi"                1
85                  "Ya Sabır"                1
86                  "Yaz Gülü"                1
87                   "Yeniden"                1

[79 rows x 2 columns]
```

Şarkıların benzerlik analizi

[16]:
```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# TF-IDF matrisini oluştur
tfidf_vectorizer = TfidfVectorizer(stop_words=turkish_stop_words)
tfidf_matrix = tfidf_vectorizer.fit_transform(df_songs['Formatted Lyrics'])

# Benzerlik matrisini hesaplayın
cosine_sim = cosine_similarity(tfidf_matrix)

# En benzer şarkıları bulma
def find_most_similar(song_index, cosine_sim=cosine_sim):
    similarity_scores = list(enumerate(cosine_sim[song_index]))
    similarity_scores = sorted(similarity_scores, key=lambda x: x[1],␣
 ↪reverse=True)
    most_similar = [i[0] for i in similarity_scores[1:6]]  # İlk 5 en benzer␣
 ↪şarkı
```

```
    return df_songs.iloc[most_similar][['Song Name', 'Formatted Lyrics']]

# Örneğin ilk şarkıya en benzer olanları bul
most_similar_songs = find_most_similar(0)
print("En Benzer Şarkılar:")
print(most_similar_songs)
```

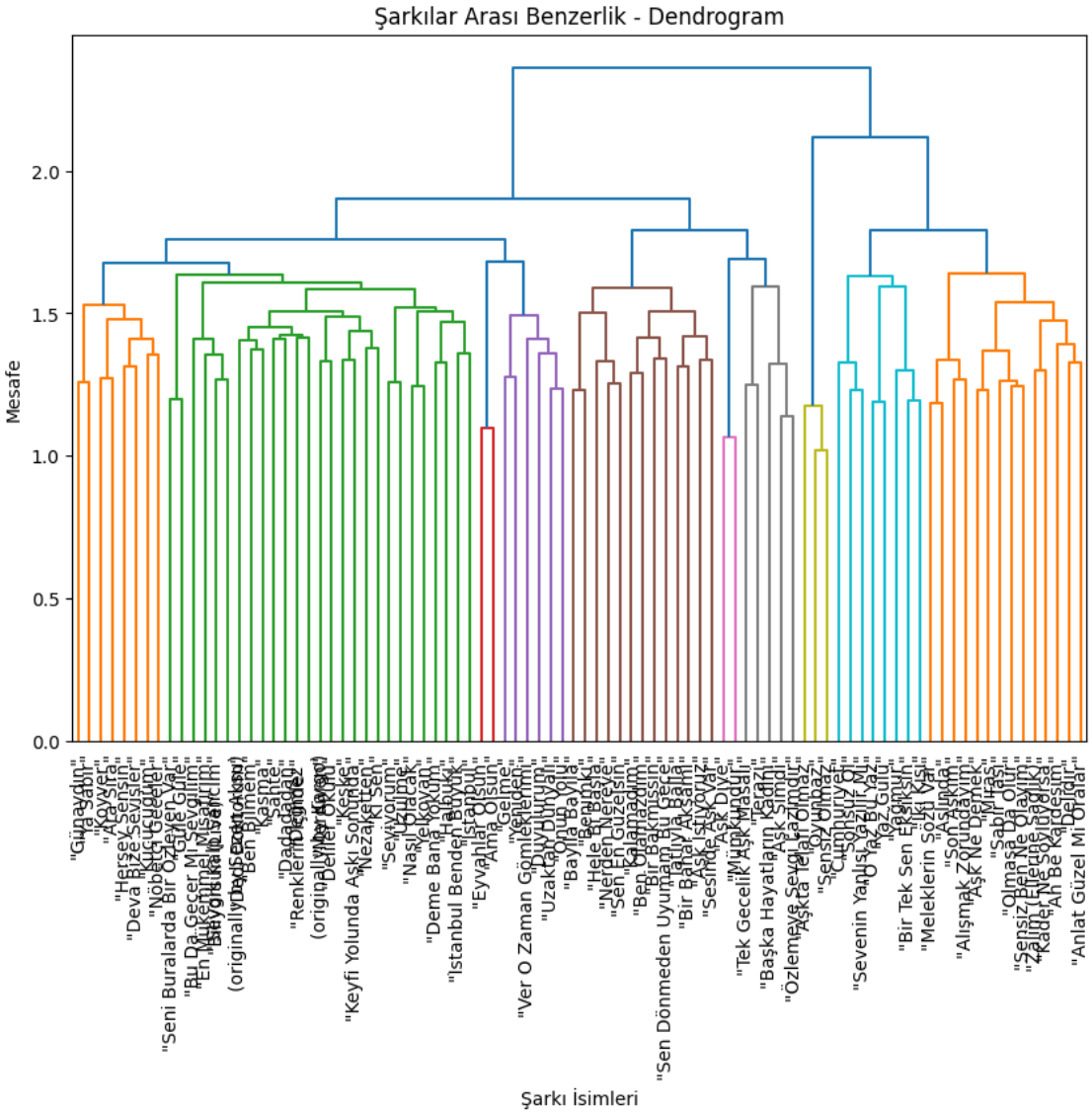
```
En Benzer Şarkılar:
                Song Name                                    Formatted Lyrics
87               "Yeniden"  Yeniden yandı tüm ışıklar\nYeniden nasıl parlı…
54   "Kader Ne Söylüyorsa"  Üzülür müyüm\nİncinir miyim\nBi düşün yanıbaşı…
82  "Sensiz Ben Ne Olayım"  İki mevsim bekledim seni\nSen diyorsun "İki da…
47               "Aslında"  Aslında sen çok uzaklarda durduğumdan mıdır\nG…
61            "Sabır Taşı"  Bi başlasam anlatmaya\nYol olur burdan uzaya\n…
```

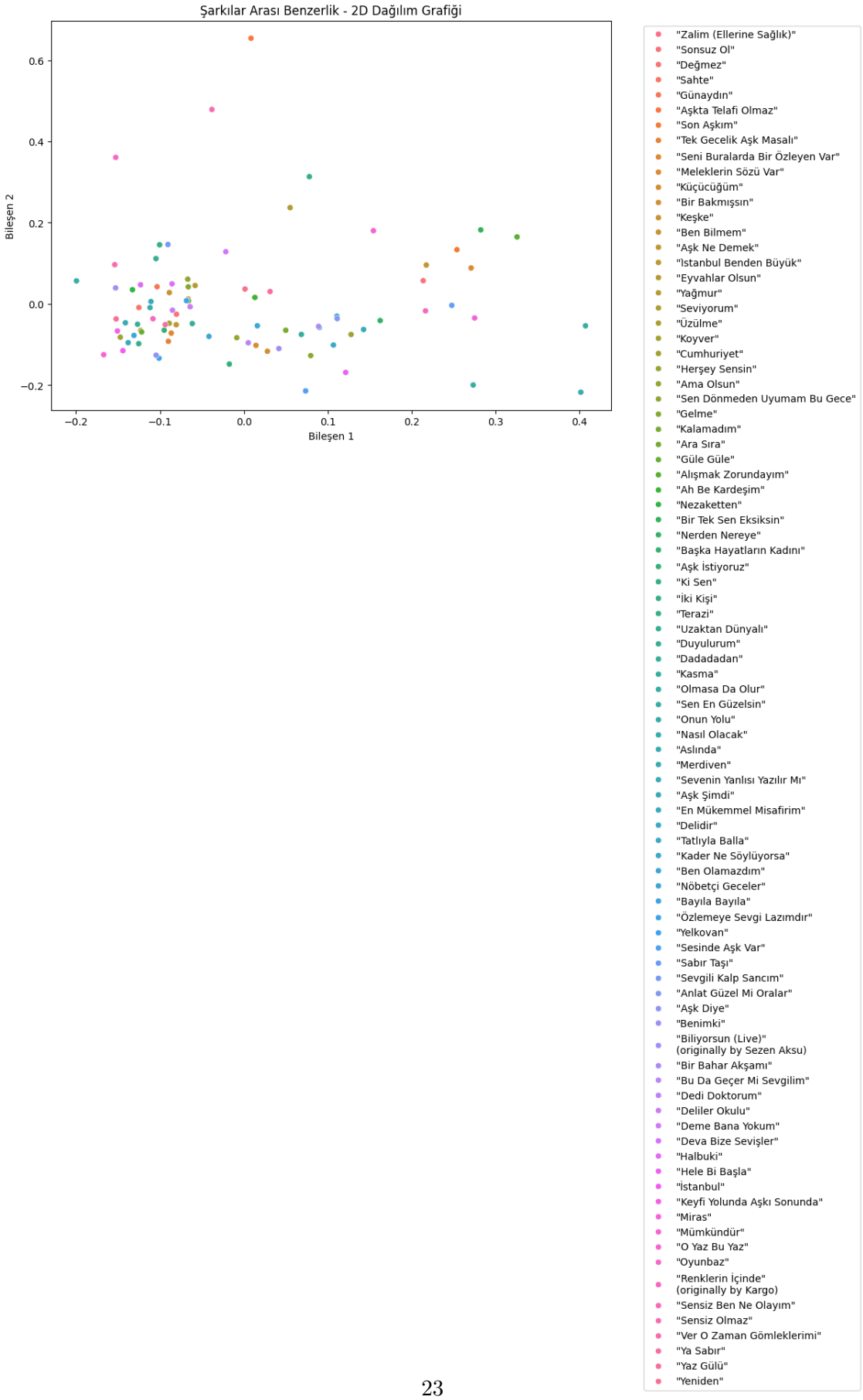
Benzerlik analizi detay ve görselleştirme

[18]:
```
# Gerekli kütüphaneleri yükleyin
!pip install scikit-learn matplotlib seaborn

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from scipy.cluster.hierarchy import dendrogram, linkage
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# CSV dosyasını okuyun
df_songs = pd.read_csv("song_lyrics_formatted.csv")

# Türkçe stop words listesi
turkish_stop_words = ['bir', 've', 'ile', 'bu', 'için', 'de', 'ama', 'çok',␣
 ↪'da', 'ki']

# TF-IDF matrisini oluştur
tfidf_vectorizer = TfidfVectorizer(stop_words=turkish_stop_words,␣
 ↪max_features=1000)
tfidf_matrix = tfidf_vectorizer.fit_transform(df_songs['Formatted Lyrics'])

# Benzerlik matrisini hesaplayın
cosine_sim = cosine_similarity(tfidf_matrix)

# En benzer şarkıları belirleme ve ortak kelimeleri bulma
def find_most_similar(song_index, cosine_sim=cosine_sim, top_n=3):
    similarity_scores = list(enumerate(cosine_sim[song_index]))
    similarity_scores = sorted(similarity_scores, key=lambda x: x[1],␣
 ↪reverse=True)
    most_similar = [i[0] for i in similarity_scores[1:top_n + 1]]
```

```
    similar_songs = df_songs.iloc[most_similar][['Song Name', 'Formatted␣
 ↪Lyrics']]

    # Ortak anahtar kelimeleri bulma
    feature_names = tfidf_vectorizer.get_feature_names_out()
    common_keywords = []
    for idx in most_similar:
        feature_idx = tfidf_matrix[song_index].multiply(tfidf_matrix[idx]).
 ↪nonzero()[1]
        common_keywords.extend([feature_names[i] for i in feature_idx])
    common_keywords = set(common_keywords)

    return similar_songs, common_keywords

# Örneğin ilk şarkıya en benzer olanları bulun
most_similar_songs, common_keywords = find_most_similar(0)
print("En Benzer Şarkılar ve Ortak Anahtar Kelimeler:")
print(most_similar_songs)
print(f"Ortak Anahtar Kelimeler: {common_keywords}")

# Dendrogram ile Şarkı Benzerlikleri Görselleştirme
linked = linkage(1 - cosine_sim, 'ward')  # 1 - cosine similarity ile mesafe␣
 ↪matrisi

plt.figure(figsize=(10, 7))
dendrogram(linked, labels=df_songs['Song Name'].values, leaf_rotation=90,␣
 ↪leaf_font_size=10)
plt.title("Şarkılar Arası Benzerlik - Dendrogram")
plt.xlabel("Şarkı İsimleri")
plt.ylabel("Mesafe")
plt.show()

# 2D dağılım grafiği ile şarkı benzerlikleri
from sklearn.decomposition import PCA

# TF-IDF matrisini 2 boyuta indirgeme
pca = PCA(n_components=2)
tfidf_pca = pca.fit_transform(tfidf_matrix.toarray())

plt.figure(figsize=(10, 7))
sns.scatterplot(x=tfidf_pca[:, 0], y=tfidf_pca[:, 1], hue=df_songs['Song Name'])
plt.title("Şarkılar Arası Benzerlik - 2D Dağılım Grafiği")
plt.xlabel("Bileşen 1")
plt.ylabel("Bileşen 2")
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-
packages (1.5.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-
packages (3.7.1)
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-
packages (0.13.2)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (4.54.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.7)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-
packages (from seaborn) (2.2.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas>=1.2->seaborn) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-
packages (from pandas>=1.2->seaborn) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
En Benzer Şarkılar ve Ortak Anahtar Kelimeler:
                Song Name                                      Formatted Lyrics
45            "Onun Yolu"  O bu yolu daha yürüyecek çok\nKalbini dolduruc…
87              "Yeniden"  Yeniden yandı tüm ışıklar\nYeniden nasıl parlı…
54  "Kader Ne Söylüyorsa"  Üzülür müyüm\nİncinir miyim\nBi düşün yanıbaşı…
Ortak Anahtar Kelimeler: {'yalan', 'onu', 'ne', 'senin', 'hadi', 'yeniden'}
```


Şarkılar Arası Benzerlik - Dendrogram
Mesafe
0.0
0.5
1.0
1.5
2.0
Şarkı İsimleri

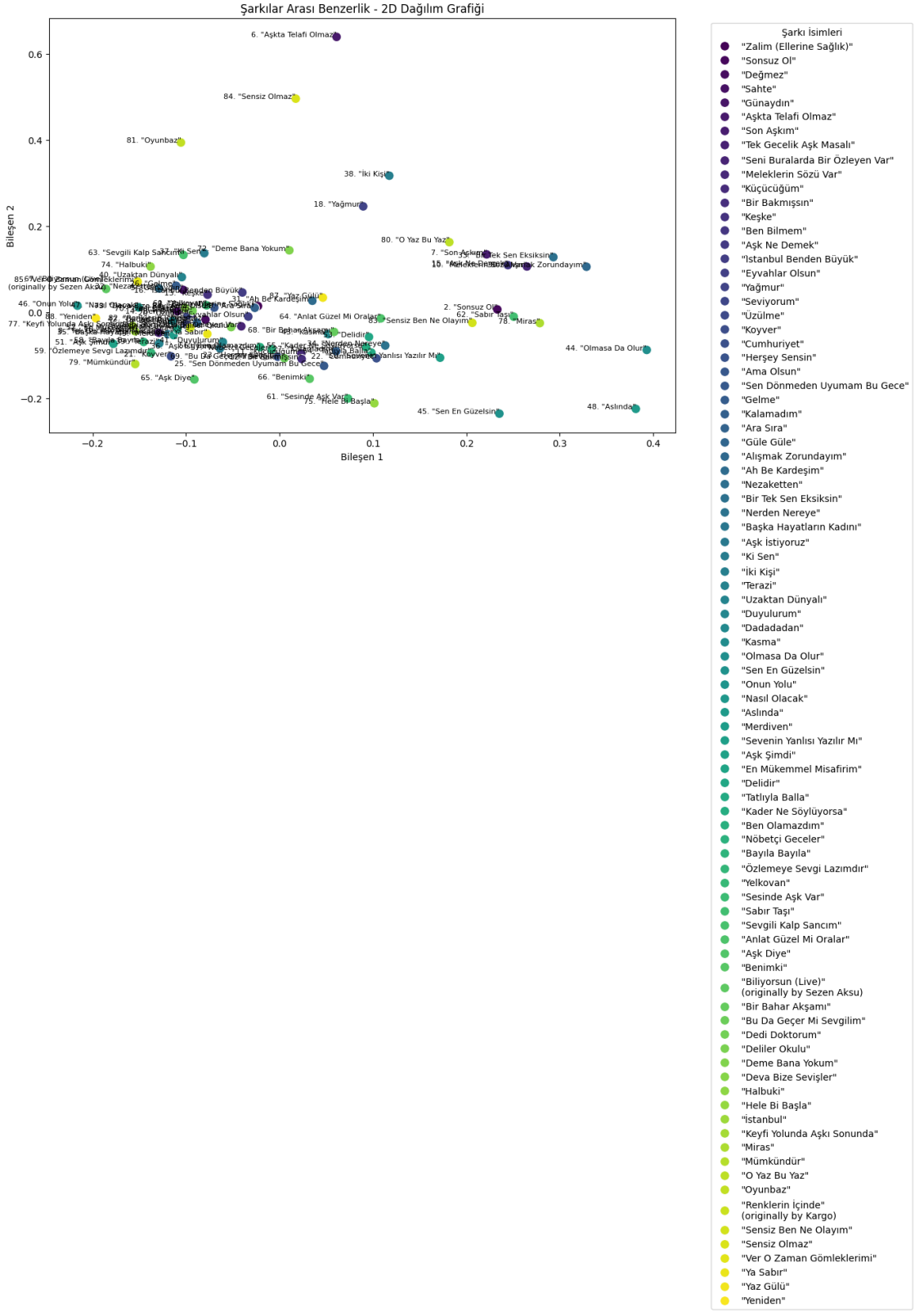

Şarkılar Arası Benzerlik - 2D Dağılım Grafiği
Bileşen 1
Bileşen 2
0.6
0.4
0.2
0.0
−0.2
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
"Zalim (Ellerine Sağlık)"
"Sonsuz Ol"
"Değmez"
"Sahte"
"Günaydın"
"Aşkta Telafi Olmaz"
"Son Aşkım"
"Tek Gecelik Aşk Masalı"
"Seni Buralarda Bir Özleyen Var"
"Meleklerin Sözü Var"
"Küçücüğüm"
"Bir Bakmışsın"
"Keşke"
"Ben Bilmem"
"Aşk Ne Demek"
"İstanbul Benden Büyük"
"Eyvahlar Olsun"
"Yağmur"
"Seviyorum"
"Üzülme"
"Koyver"
"Cumhuriyet"
"Herşey Sensin"
"Ama Olsun"
"Sen Dönmeden Uyumam Bu Gece"
"Gelme"
"Kalamadım"
"Ara Sıra"
"Güle Güle"
"Alışmak Zorundayım"
"Ah Be Kardeşim"
"Nezaketten"
"Bir Tek Sen Eksiksin"
"Nerden Nereye"
"Başka Hayatların Kadını"
"Aşk İstiyoruz"
"Ki Sen"
"İki Kişi"
"Terazi"
"Uzaktan Dünyalı"
"Duyulurum"
"Dadadadan"
"Kasma"
"Olmasa Da Olur"
"Sen En Güzelsin"
"Onun Yolu"
"Nasıl Olacak"
"Aslında"
"Merdiven"
"Sevenin Yanlısı Yazılır Mı"
"Aşk Şimdi"
"En Mükemmel Misafirim"
"Delidir"
"Tatlıyla Balla"
"Kader Ne Söylüyorsa"
"Ben Olamazdım"
"Nöbetçi Geceler"
"Bayıla Bayıla"
"Özlemeye Sevgi Lazımdır"
"Yelkovan"
"Sesinde Aşk Var"
"Sabır Taşı"
"Sevgili Kalp Sancım"
"Anlat Güzel Mi Oralar"
"Aşk Diye"
"Benimki"
"Biliyorsun (Live)"
(originally by Sezen Aksu)
"Bir Bahar Akşamı"
"Bu Da Geçer Mi Sevgilim"
"Dedi Doktorum"
"Deliler Okulu"
"Deme Bana Yokum"
"Deva Bize Sevişler"
"Halbuki"
"Hele Bi Başla"
"İstanbul"
"Keyfi Yolunda Aşkı Sonunda"
"Miras"
"Mümkündür"
"O Yaz Bu Yaz"
"Oyunbaz"
"Renklerin İçinde"
(originally by Kargo)
"Sensiz Ben Ne Olayım"
"Sensiz Olmaz"
"Ver O Zaman Gömleklerimi"
"Ya Sabır"
"Yaz Gülü"
"Yeniden"

Detaylı ve anlaşılabilir görsel

[19]:
```
# Gerekli kütüphaneleri yükleyin
!pip install scikit-learn matplotlib seaborn

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from scipy.cluster.hierarchy import dendrogram, linkage
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA

# CSV dosyasını okuyun
df_songs = pd.read_csv("song_lyrics_formatted.csv")

# Türkçe stop words listesi
turkish_stop_words = ['bir', 've', 'ile', 'bu', 'için', 'de', 'ama', 'çok',␣
 ↪'da', 'ki']

# TF-IDF matrisini oluştur
tfidf_vectorizer = TfidfVectorizer(stop_words=turkish_stop_words,␣
 ↪max_features=1000)
tfidf_matrix = tfidf_vectorizer.fit_transform(df_songs['Formatted Lyrics'])

# Benzerlik matrisini hesaplayın
cosine_sim = cosine_similarity(tfidf_matrix)

# TF-IDF matrisini 2 boyuta indirgeme
pca = PCA(n_components=2)
tfidf_pca = pca.fit_transform(tfidf_matrix.toarray())

# Şarkı numaralarını oluşturun
df_songs['Song Number'] = range(1, len(df_songs) + 1)

# 2D dağılım grafiği ile şarkı benzerlikleri (numara ve isim ile)
plt.figure(figsize=(12, 8))
sns.scatterplot(x=tfidf_pca[:, 0], y=tfidf_pca[:, 1], hue=df_songs['Song␣
 ↪Name'], palette='viridis', s=100)

# Her noktanın yanına şarkı numarasını ve ismini ekleme
for i in range(len(df_songs)):
    plt.text(tfidf_pca[i, 0], tfidf_pca[i, 1], f"{df_songs['Song Number'][i]}.␣
 ↪{df_songs['Song Name'][i]}",
             fontsize=8, ha='right', color='black')
```

```
plt.title("Şarkılar Arası Benzerlik - 2D Dağılım Grafiği")
plt.xlabel("Bileşen 1")
plt.ylabel("Bileşen 2")
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', title="Şarkı İsimleri")
plt.show()
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-
packages (1.5.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-
packages (3.7.1)
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-
packages (0.13.2)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (4.54.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.7)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-
packages (from seaborn) (2.2.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas>=1.2->seaborn) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-
packages (from pandas>=1.2->seaborn) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```


Şarkılar Arası Benzerlik - 2D Dağılım Grafiği
Bileşen 1
Bileşen 2
Şarkı İsimleri
"Zalim (Ellerine Sağlık)"
"Sonsuz Ol"
"Değmez"
"Sahte"
"Günaydın"
"Aşkta Telafi Olmaz"
"Son Aşkım"
"Tek Gecelik Aşk Masalı"
"Seni Buralarda Bir Özleyen Var"
"Meleklerin Sözü Var"
"Küçücüğüm"
"Bir Bakmışsın"
"Keşke"
"Ben Bilmem"
"Aşk Ne Demek"
"İstanbul Benden Büyük"
"Eyvahlar Olsun"
"Yağmur"
"Seviyorum"
"Üzülme"
"Koyver"
"Cumhuriyet"
"Herşey Sensin"
"Ama Olsun"
"Sen Dönmeden Uyumam Bu Gece"
"Gelme"
"Kalamadım"
"Ara Sıra"
"Güle Güle"
"Alışmak Zorundayım"
"Ah Be Kardeşim"
"Nezaketten"
"Bir Tek Sen Eksiksin"
"Nerden Nereye"
"Başka Hayatların Kadını"
"Aşk İstiyoruz"
"Ki Sen"
"İki Kişi"
"Terazi"
"Uzaktan Dünyalı"
"Duyulurum"
"Dadadadan"
"Kasma"
"Olmasa Da Olur"
"Sen En Güzelsin"
"Onun Yolu"
"Nasıl Olacak"
"Aslında"
"Merdiven"
"Sevenin Yanlısı Yazılır Mı"
"Aşk Şimdi"
"En Mükemmel Misafirim"
"Delidir"
"Tatlıyla Balla"
"Kader Ne Söylüyorsa"
"Ben Olamazdım"
"Nöbetçi Geceler"
"Bayıla Bayıla"
"Özlemeye Sevgi Lazımdır"
"Yelkovan"
"Sesinde Aşk Var"
"Sabır Taşı"
"Sevgili Kalp Sancım"
"Anlat Güzel Mi Oralar"
"Aşk Diye"
"Benimki"
"Biliyorsun (Live)" (originally by Sezen Aksu)
"Bir Bahar Akşamı"
"Bu Da Geçer Mi Sevgilim"
"Dedi Doktorum"
"Deliler Okulu"
"Deme Bana Yokum"
"Deva Bize Sevişler"
"Halbuki"
"Hele Bi Başla"
"İstanbul"
"Keyfi Yolunda Aşkı Sonunda"
"Miras"
"Mümkündür"
"O Yaz Bu Yaz"
"Oyunbaz"
"Renklerin İçinde" (originally by Kargo)
"Sensiz Ben Ne Olayım"
"Sensiz Olmaz"
"Ver O Zaman Gömleklerimi"
"Ya Sabır"
"Yaz Gülü"
"Yeniden"