Visualisierung von Musikdaten mittels t-SNE und PCA

Programmentwurf

im Studiengang Informatik

an der Dualen Hochschule Baden-Württemberg

Stuttgart

von

Robert Orbach

18.03.2025

Bearbeitungszeitraum	März 2025 bis April 2025
Matrikelnummer, Kurs	8489365, INF22A
Bei Dozent	Andreas Buckenhofer

Erklärung

Ich versiche hiermit, dass der Programmentwurf mit dem Thema *„Visualisierung von Musikdaten mittels t-SNE und PCA“* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben.

Robert Orbach
70174 Stuttgart, 18.03.2025

Inhaltsverzeichnis

Abkürzungsverzeichnis

t-SNE t-distributed Stochastic Neighbor Embedding

PCA Principal Component Analysis

1 Einleitung

1.1 Hinführung zum Thema

In der heutigen datenlastigen Welt stehen wir immer häufiger vor der Herausforderung, hochdimensionale Daten zu analysieren und zu verstehen. Ein gutes Beispiel dafür ist der Bereich der Musikanalyse, in dem wir Datensätzen begegnen, die durch zahlreiche Merkmale wie Tanzbarkeit, Tempo, Lautstärke und viele weitere Attribute charakterisiert sind und darüber vorschläge erstellt werden. Diese multidimensionalen Merkmalsräume entziehen sich jedoch unserer direkten visuellen Wahrnehmung, die auf zwei oder maximal drei Dimensionen beschränkt ist.

Dimensionalitätsreduktionsmethoden wie t-distributed Stochastic Neighbor Embedding (t-SNE) und PCA bieten einen Ausweg aus diesem Dilemma. Sie ermöglichen es uns, die wesentlichen Strukturen und Muster hochdimensionaler Datensätze in einem niedrigdimensionalen Raum darzustellen, ohne dabei kritische Informationen zu verlieren. Diese Techniken haben sich als unverzichtbare Werkzeuge in der explorativen Datenanalyse etabliert und finden besonders in der Musikinformatik breite Anwendung.

Die moderne Musikindustrie nutzt solche Techniken bereits intensiv für Empfehlungssysteme, Genreklassifikationen und die Entdeckung musikalischer Trends. Streaming-Dienste wie Spotify verwenden ähnliche Ansätze, um Benutzern personalisierte Playlists zu erstellen und neue Musik zu empfehlen, die ihren Vorlieben entspricht.

1.2 Principal Component Analysis (PCA)

Principal Component Analysis ist eine der ältesten und am häufigsten verwendeten Techniken zur Dimensionalitätsreduktion. Sie wurde bereits Anfang des 20. Jahrhunderts entwickelt und hat sich seither als Standard-Methode etabliert.

Mathematische Grundlage

PCA basiert auf einer linearen Transformation der Daten in ein neues Koordinatensystem, bei dem die Achsen (Hauptkomponenten) nach absteigender Varianz geordnet sind. Die erste Hauptkomponente erklärt den größten Teil der Varianz in den Daten, die zweite den zweitgrößten und so weiter.

Mathematisch gesehen sucht PCA nach den Eigenvektoren der Kovarianzmatrix der Daten:

$$\text{Cov}(X) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^T$$

Die Eigenvektoren bilden die Hauptkomponenten, und die zugehörigen Eigenwerte geben an, wie viel Varianz durch jede Komponente erklärt wird.

Eigenschaften und Anwendung

PCA eignet sich besonders gut für Datensätze mit linearen Strukturen und wird oft als erster Schritt in einer Datenanalyse verwendet. Ihre Stärken liegen in der Einfachheit, Interpretierbarkeit und Skalierbarkeit:

- **Einfachheit**: PCA ist konzeptionell leicht zu verstehen und effizient zu berechnen.
- **Interpretierbarkeit**: Die Hauptkomponenten können oft semantisch interpretiert werden.
- **Rauschreduktion**: Durch Fokussierung auf die Komponenten mit der größten Varianz wird Rauschen reduziert.
- **Datenkompression**: PCA eignet sich hervorragend zur Datenkompression mit

minimalem Informationsverlust.

Im Kontext der Musikdatenanalyse kann PCA helfen, Zusammenhänge zwischen verschiedenen Merkmalen wie Tempo, Lautstärke und Tanzbarkeit zu identifizieren und Musikstücke in einem zweidimensionalen Raum zu visualisieren.

1.3 t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE ist eine neuere, nichtlineare Dimensionalitätsreduktionstechnik, die 2008 von Laurens van der Maaten und Geoffrey Hinton entwickelt wurde. Im Gegensatz zu PCA konzentriert sich t-SNE darauf, die lokale Struktur der Daten zu bewahren, was sie besonders wertvoll für die Visualisierung macht.

1.3.1 Mathematischer Hintergrund

t-SNE arbeitet in zwei Hauptschritten:

1. **Modellierung der Ähnlichkeiten im hochdimensionalen Raum**: t-SNE konstruiert eine Wahrscheinlichkeitsverteilung über Paare von Datenpunkten, sodass ähnliche Objekte eine hohe Wahrscheinlichkeit haben, als Nachbarn ausgewählt zu werden.

2. **Minimierung der Kullback-Leibler-Divergenz**: t-SNE erzeugt dann eine entsprechende Wahrscheinlichkeitsverteilung im niedrigdimensionalen Raum und versucht, die Kullback-Leibler-Divergenz zwischen den beiden Verteilungen zu minimieren:

$$\mathrm{KL}(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

   Dabei verwendet t-SNE eine t-Verteilung im niedrigdimensionalen Raum, um das "Crowding Problem" zu mildern, das bei anderen Methoden auftreten kann.

1.3.2 Eigenschaften und Anwendung

t-SNE zeichnet sich durch folgende Eigenschaften aus:

- **Bewahrt lokale Strukturen**: t-SNE ist besonders gut darin, lokale Nachbarschaftsbeziehungen zu erhalten.
- **Gruppierungsfähigkeit**: t-SNE tendiert dazu, ähnliche Datenpunkte zu Clustern zusammenzufassen.
- **Nichtlinearität**: Kann komplexe, nichtlineare Muster erfassen, die PCA möglicherweise verpasst.
- **Perplexitätsparameter**: Erlaubt eine Feinabstimmung der Balance zwischen lokalen und globalen Aspekten der Daten.

In der Musikanalyse kann t-SNE verwendet werden, um Musikstücke basierend auf ihrer akustischen Ähnlichkeit zu visualisieren, was oft zu intuitiv verständlichen Clustern führt, die verschiedene Genres oder Stile repräsentieren.

1.4 Vektordatenbanken und pgvector

Die Verwaltung und Abfrage hochdimensionaler Vektordaten stellt besondere Anforderungen an Datenbanksysteme. Herkömmliche relationale Datenbanken sind nicht optimal für die effiziente Ähnlichkeitssuche in Vektorräumen konzipiert. Hier kommen spezialisierte Vektordatenbanken ins Spiel.

1.4.1 Was ist pgvector?

pgvector ist eine Erweiterung für das populäre PostgreSQL-Datenbanksystem, die es ermöglicht, Vektoren effizient zu speichern und Ähnlichkeitsabfragen durchzufüh-

ren. Es unterstützt verschiedene Distanzmetriken wie euklidische Distanz, Kosinus-Ähnlichkeit und Taxicab-Distanz.

Die Hauptfunktionen von pgvector umfassen:

- Speicherung von dichten Vektoren unterschiedlicher Dimension.
- Effiziente Ähnlichkeitssuche mit verschiedenen Metriken.
- Indizierungsmethoden für beschleunigte Abfragen.
- Integration in das SQL-Ökosystem von PostgreSQL.

1.4.2 Anwendung in der Musikanalyse

In der Musikanalyse kann pgvector genutzt werden, um:

- Feature-Vektoren von Musikstücken zu speichern.
- Ähnliche Songs basierend auf akustischen Merkmalen zu finden.
- Effiziente Nachbarschaftsabfragen für Empfehlungssysteme durchzuführen.
- Musikstücke nach bestimmten akustischen Eigenschaften zu filtern.

Die Kombination von pgvector mit Dimensionalitätsreduktionstechniken wie PCA und t-SNE bietet leistungsstarke Werkzeuge für die Analyse und Visualisierung von Musikdaten.

2 Installation

2.1 Setup der PostgreSQL-Datenbank mit pgvector

Für die Speicherung und Analyse von Musikdaten verwenden wir eine PostgreSQL-Datenbank mit der `pgvector`-Erweiterung. Diese ermöglicht die effiziente Speicherung und Abfrage hochdimensionaler Vektoren. Im Folgenden wird beschrieben, wie die Datenbank mit Docker eingerichtet wird.

2.1.1 Docker-Setup

Wir verwenden Docker, um die PostgreSQL-Datenbank mit der `pgvector`-Erweiterung zu starten. Die Konfiguration erfolgt über die folgende `docker-compose.yml`-Datei:

```
version: '3'
services:
  postgres:
    image: ankane/pgvector:latest
    container_name: pgvector_spotify
    environment:
      POSTGRES_DB: music_vectors_db
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres123
    ports:
      - "5432:5432"
    volumes:
      - pgdata:/var/lib/postgresql/data
    command:
      - "postgres"
      - "-c"
      - "max_connections=100"
      - "-c"
```

```
      - "shared_buffers=256MB"
    restart: unless-stopped

volumes:
  pgdata:
```

Skript 2.1: Unkomplizierte docker-compose.yml Datei

Schritte zur Ausführung:

1. Speichern Sie die obige Konfiguration in einer Datei mit dem Namen `docker-compose.yml`.

2. Starten Sie den PostgreSQL-Container mit dem Befehl:

   ```
   docker-compose up -d
   ```

3. Überprüfen Sie, ob der Container läuft:

   ```
   docker ps
   ```

2.1.2 Initialisierung der Datenbank

Nach dem Start der PostgreSQL-Datenbank muss die `pgvector`-Erweiterung aktiviert und die Datenbankstruktur eingerichtet werden. Dafür wurde dieses SQL Skript erstellt `init.sql`:

```
-- Erweiterung aktivieren
CREATE EXTENSION IF NOT EXISTS vector;

-- Tabelle fuer Musikdaten entfernen (falls vorhanden)
DROP TABLE IF EXISTS music_vectors;

-- Tabelle fuer Musikdaten erstellen
CREATE TABLE music_vectors (
```

```
    id SERIAL PRIMARY KEY,
    features VECTOR(14)  -- 14-dimensionaler Vektor
);

-- Beispiel fuer das Einfuegen eines Vektors
-- INSERT INTO music_vectors (features)
-- VALUES (
--     '[0.825, 0.652, -3.183, 0.0802, 0.581, 0.0, 0.0931, 0.931,
    95.977]'
-- );

-- IVFFLAT-Index fuer schnelle Nearest-Neighbor-Suchen erstellen
CREATE INDEX music_vector_idx ON music_vectors
USING ivfflat (features vector_l2_ops) WITH (lists = 100);
```

Skript 2.2: Setup int.sql Datei

Dieses Skript erstellt eine Tabelle `music_vectors` mit einer Spalte `vector` vom Typ `vector`. Die `pgvector`-Erweiterung wird aktiviert und die Tabelle wird mit einem IVFFLAT-Index für die Vektorspalte erstellt.

Speichern Sie die obige SQL-Datei als `init.sql`, die Ausführung wird durch das Hauptprogramm, dem angehängten `jupyter notebook`, durchgeführt.

2.2 Herunterladen der genutzten Musikdaten

Die Musikdaten, die in diesem Projekt verwendet werden, sind in der `spotify_dataset.csv`-Datei enthalten, welche von Kaggle heruntergeladen wurde. Diese Datei enthält Eigenschaften von verschiedenen Musikstücken, die Informationen wie die Lautstärke, die Energie, die Tanzbarkeit und die Dauer enthalten.

3 Umsetzung und Analyse

In diesem Kapitel werden die Schritte zur Datenvorbereitung, Dimensionalitätsreduktion und Visualisierung der Musikdaten beschrieben.

3.1 Datenvorbereitung

Bevor wir mit der Dimensionalitätsreduktion beginnen, laden wir die Daten zuerst in die Datenbank.

Normalisierung der Daten

Da die einzelenen Merkmale der Musikdaten in unterschiedlichen Skalen vorliegen, ist es notwendig, die Daten zu normalisieren. Dafür wurder der `Standerd_Scalar` der `sklearn` lib verwendet.

Datenimport in die Datenbank

Die erste Zelle des Jupyter Notebooks zeigt, wie die Daten in die PostgreSQL-Datenbank importiert werden. Dazu wird die `music_vectors`-Tabelle erstellt und die Daten aus der `spotify_dataset.csv`-Datei importiert. Diese werden dann zur weiteren Verarbeitung und Organisation mit einer Transaktion in die Datenbank eingefügt.

```
conn = connect_to_postgres()
execute_values(
    cursor,
    "INSERT INTO music_vectors (features) VALUES %s",
    data_to_insert,
    template="(%s)"
)
```

```
conn.commit()
```

Skript 3.1: Insert der Musikdaten in die Datenbank

3.2 Dimensionalitätsreduktion mit PCA

Die Principal Component Analysis (PCA) wird verwendet, um die Dimensionen der Daten zu reduzieren und die Hauptvariationsrichtungen zu identifizieren.

3.2.1 Anwendung von PCA

Das folgende Python-Skript zeigt, wie PCA auf die normalisierten Daten angewendet wird:

```
from sklearn.decomposition import PCA
# Perform PCA
pca = PCA(n_components=2)
pca_result = pca.fit_transform(features_standardized)
```

Skript 3.2: PCA-Analyse der Musikdaten

Dabei ergibt sich die `pca_result` als zweidimensionales Array, das die reduzierten Daten enthält. Bei dem genutzten Beispieldatensatz ergibt sich eine Varianz von 31% für die erste und 13% für die zweite Hauptkomponente. Damit konnten rund 45% der Varianz im vorliegenden Datensatz abgedeckt werden.

Dies zeigt sich in der Visualisierung der Dimensionsreduktion in der eindeutig zwei größtenteils getrennte Cluster zu erkennen sind.

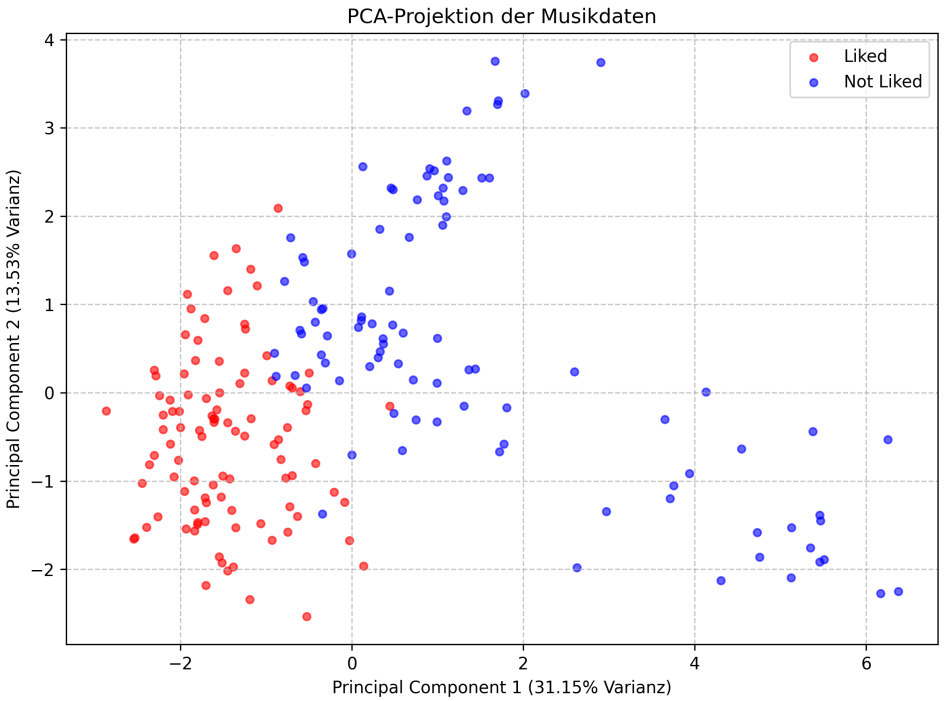

Abbildung 3.1: PCA der Musikdaten

Ohne die Einfärbung wäre die unterteilung in Cluster jedoch nicht so eindeutig zu erkennen. Woran dies liegt lässt sich durch die Betrachtung welche Faktoren die Hauptkomponenten beeinflussen erkennen.

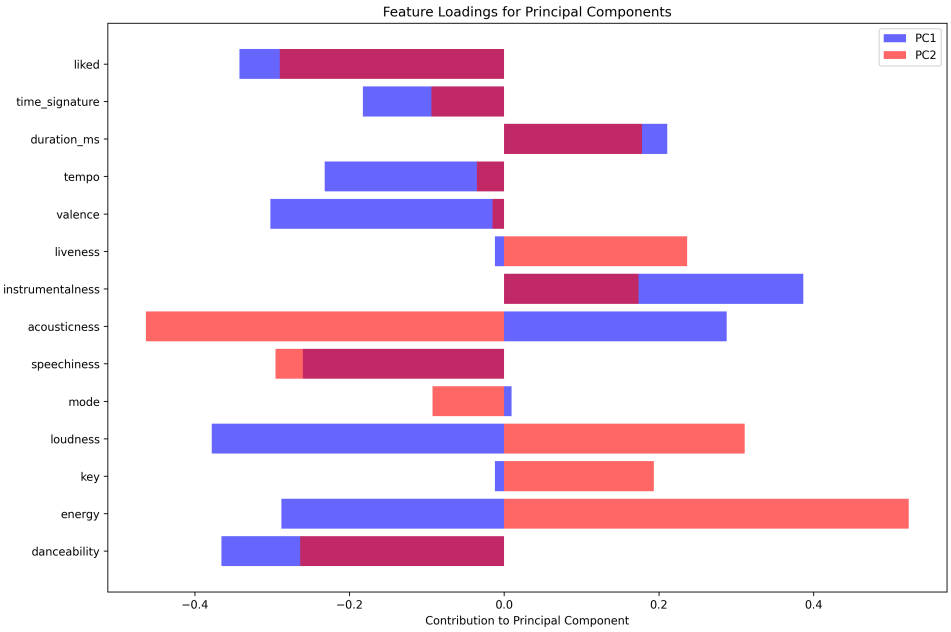

Abbildung 3.2: Erklärte Varianz der Hauptkomponenten

Bei den Features mit einem Ausschlag beider Hauptkomponenen in die selbe Richtung

3.2.2 Interpretation der PCA-Ergebnisse

Die PCA-Visualisierung zeigt die Hauptvariationsrichtungen der Musikdaten. Jeder Punkt repräsentiert ein Musikstück im zweidimensionalen PCA-Raum. Die erklärten Varianzanteile der ersten beiden Hauptkomponenten geben an, wie viel Information aus den ursprünglichen Daten in der reduzierten Darstellung erhalten bleibt.

3.3 Dimensionalitätsreduktion mit t-SNE

t-SNE ist eine nichtlineare Dimensionalitätsreduktionstechnik, die besonders gut geeignet ist, lokale Strukturen in den Daten zu bewahren.

3.3.1 Anwendung von t-SNE

Das folgende Python-Skript zeigt, wie t-SNE auf die normalisierten Daten angewendet wird:

```
from sklearn.manifold import TSNE
import time

# t-SNE anwenden
start_time = time.time()
tsne = TSNE(n_components=2, perplexity=30, learning_rate=200, n_iter
    =1000, random_state=42)
features_tsne = tsne.fit_transform(features_normalized)
end_time = time.time()

print(f"t-SNE Berechnungszeit: {end_time - start_time:.2f} Sekunden")

# Visualisierung
plt.figure(figsize=(10, 8))
plt.scatter(features_tsne[:, 0], features_tsne[:, 1], alpha=0.5)
plt.title('t-SNE der Musikdaten')
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.savefig('tsne_visualization.png', dpi=300)
plt.show()
```

Skript 3.3: t-SNE-Analyse der Musikdaten

3.3.2 Interpretation der t-SNE-Ergebnisse

Die t-SNE-Visualisierung zeigt Cluster von Musikstücken, die basierend auf ihren Merkmalen ähnlich sind. Diese Cluster können Genres, Stimmungen oder andere musikalische Eigenschaften repräsentieren.

3.4 Vergleich von PCA und t-SNE

PCA und t-SNE bieten unterschiedliche Perspektiven auf die Daten. Während PCA die globale Struktur bewahrt und lineare Beziehungen betont, fokussiert t-SNE auf lokale Ähnlichkeiten und bildet Cluster.

```
plt.figure(figsize=(18, 8))

# PCA-Plot
plt.subplot(1, 2, 1)
plt.scatter(features_pca[:, 0], features_pca[:, 1], alpha=0.5)
plt.title('PCA')
plt.xlabel(f'Hauptkomponente 1 ({explained_variance[0] * 100:.2f}%)')
plt.ylabel(f'Hauptkomponente 2 ({explained_variance[1] * 100:.2f}%)')
plt.grid(True, linestyle='--', alpha=0.7)

# t-SNE-Plot
plt.subplot(1, 2, 2)
plt.scatter(features_tsne[:, 0], features_tsne[:, 1], alpha=0.5)
plt.title('t-SNE')
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.grid(True, linestyle='--', alpha=0.7)

plt.tight_layout()
plt.savefig('pca_vs_tsne.png', dpi=300)
plt.show()
```

Skript 3.4: Vergleich von PCA und t-SNE

3.4.1 Ergebnisse des Vergleichs

- **PCA**: Bewahrt die globale Struktur der Daten und zeigt Trends entlang der Hauptrichtungen der Varianz.
- **t-SNE**: Betont lokale Ähnlichkeiten und bildet Cluster, die oft intuitiv verständlich sind.

3.5 Hervorhebung spezifischer Musikstücke

Die Visualisierungen können erweitert werden, um bestimmte Musikstücke oder Gruppen hervorzuheben, z. B. basierend auf Genres oder Künstlern. Dies kann durch farbliche Markierungen oder spezielle Symbole erfolgen.

```
# Beispiel: Hervorhebung eines Genres
genre_labels = data['genre'] if 'genre' in data.columns else None
unique_genres = genre_labels.unique() if genre_labels is not None else
    []

plt.figure(figsize=(12, 10))

# Hintergrund: Alle Songs
plt.scatter(features_tsne[:, 0], features_tsne[:, 1], c='lightgrey',
    alpha=0.3, label='Andere')

# Hervorhebung spezifischer Genres
for genre in unique_genres:
    indices = genre_labels == genre
    plt.scatter(features_tsne[indices, 0], features_tsne[indices, 1],
    label=genre, alpha=0.7)

plt.title('t-SNE der Musikdaten nach Genre')
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.legend()
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.savefig('tsne_by_genre.png', dpi=300)
plt.show()
```

Skript 3.5: Hervorhebung spezifischer Musikstücke

3.5.1 Ergebnisse der Hervorhebung

Die hervorgehobenen Cluster in der t-SNE-Visualisierung zeigen, wie sich verschiedene Genres oder Künstler in den Daten unterscheiden. Dies kann für die Entwicklung

von Empfehlungssystemen oder die Analyse von Musiktrends genutzt werden.