600 Jahre Universität Rostock

Traditio et Innovatio

Bacheloarbeit zum Thema

Konzeption eines ETL-Prozesses zur Ermittlung von Kennzahlen aus Audioaufnahmen als Vorbereitungsschritt einer Plagiatserkennung

Studiengang:	Informatik
Vorgelegt von:	Ghaith Ishak
Matrikelnummer:	219203794
Abgabedatum:	30.09.24
Erstgutachter:	Dr.-Ing. Hannes Grunert
Zweitgutachter:	Dr.-Ing. Holger Meyer

Lehrstuhl für Datenbank- und Informationssysteme
Fakultät für Informatik und Elektrotechnik
Universität Rostock

Gedruckt auf säurefreiem und elementar-chlorfrei gebleichtem Papier.

Stand:: 30. September 2024

Inhalt

Kurzzusammenfassung

Diese Bachelorarbeit befasst sich mit der Konzeption und Implementierung eines ETL-Prozesses zur Ermittlung von Kennzahlen aus Audioaufnahmen, die als Vorbereitungsschritt für die Plagiatserkennung dienen. Ziel der Arbeit war es, einen robusten ETL-Workflow zu entwickeln, der Audiodaten aus verschiedenen Quellen effizient extrahiert, transformiert und in ein Zielsystem lädt. Die Arbeit beleuchtet die Herausforderungen und technischen Aspekte, die mit der Verarbeitung von Audiodaten verbunden sind, und zeigt, wie ein maßgeschneidertes ETL-Framework zur Bewältigung dieser Herausforderungen beitragen kann.

Im theoretischen Teil der Arbeit werden die Grundlagen des ETL-Prozesses sowie die relevanten Technologien und Tools vorgestellt. Darauf aufbauend wird der aktuelle Stand der Technik hinsichtlich der Audiodatenverarbeitung und Plagiatserkennung untersucht. Im praktischen Teil der Arbeit wird das entwickelte Konzept detailliert beschrieben und die Implementierung Schritt für Schritt erläutert. Die Ergebnisse zeigen, dass das entwickelte System in der Lage ist, Audiodaten effizient zu verarbeiten und die Grundlage für eine zuverlässige Plagiatserkennung zu schaffen. Abschließend werden die Ergebnisse diskutiert und Empfehlungen für zukünftige Entwicklungen gegeben.

Abstract

This bachelor thesis presents the design and implementation of an ETL process for extracting metrics from audio recordings as a preparatory step for plagiarism detection. The main objective was to develop a robust ETL workflow capable of efficiently extracting, transforming, and loading audio data from various sources into a target system. The thesis addresses the challenges and technical aspects of audio data processing and demonstrates how a tailored ETL framework can help overcome these challenges.

The theoretical part of the thesis covers the fundamentals of ETL processes and relevant technologies and tools. Building on this foundation, the current state of the art in audio data processing and plagiarism detection is reviewed. The practical part of the thesis details the developed concept and provides a step-by-step implementation guide. The results indicate that the developed system is capable of efficiently processing audio data, laying the groundwork for reliable plagiarism detection. The thesis concludes with a discussion of the results and provides recommendations for future developments.

Kapitel 1

Einleitung und Problemstellung

In diesem Kapitel wird die Problemstellung der Arbeit sowie die Zielsetzung des Projekts vorgestellt. Es wird erläutert, warum die Entwicklung eines ETL-Prozesses zur Verarbeitung von Audiodaten notwendig ist und welche Herausforderungen und Anforderungen dabei im Fokus stehen.

1.1 Problemstellung und Zielsetzung

ETL-Prozesse (Extrahieren, Transformieren, Laden) spielen eine wesentliche Rolle in der Datenverwaltung und -analyse. Ihre Anwendung auf die Verarbeitung von Audiodaten wird zunehmend wichtiger, da der Bedarf an der Analyse und Erkenntnisgewinnung aus Audioquellen stetig wächst. Der ETL-Prozess beginnt mit der Extraktion von Audiodaten aus verschiedenen Quellen wie Live-Feeds, Aufzeichnungen und Streaming-Diensten. Dabei müssen Herausforderungen wie unterschiedliche Codierungsformate und Audioqualitätsstufen bewältigt werden. Nach der Extraktion durchlaufen die Audiodaten eine Transformation, die Schritte wie Normalisierung, Formatkonvertierung, Merkmalsextraktion, Rauschunterdrückung, Anreicherung und Segmentierung umfasst. Diese Prozesse stellen sicher, dass die Audiodaten für die Analyse bereit sind, indem sie konsistent gemacht, relevante Merkmale extrahiert und die Daten bereinigt werden. Im letzten Schritt werden die transformierten Daten in ein Zielsystem geladen, etwa in eine Datenbank, ein spezialisiertes Audioanalysetool ein Data Warehouse. Diese aufbereiteten Audiodaten können dann für verschiedene Anwendungen genutzt werden [Ali18], wie beispielsweise Spracherkennung, Audioanalyse, Musikinformationssuche oder Stimmungsanalyse. Die ETL-Verarbeitung von Audiodaten hat erheblichen Einfluss auf Branchen wie das Gesundheitswesen, die Unterhaltungsindustrie, die Sicherheitsbranche und den Kundendienst, da Unternehmen damit Audiodaten zur Entscheidungsfindung und Verbesserung der Benutzererfahrung einsetzen

können. Mit dem zunehmenden Volumen und der Vielfalt an Audiodaten werden robuste ETL-Frameworks immer wichtiger, um diese Daten in verwertbare Informationen umzuwandeln [See21].

Trotz der zunehmenden Bedeutung von Audiodaten in Bereichen wie dem Gesundheitswesen, der Unterhaltung, der Sicherheit und dem Kundenservice haben viele Unternehmen Schwierigkeiten, diese Daten effizient zu verwalten und zu analysieren. Die Herausforderungen sind vielfältig: Audiodaten müssen aus verschiedenen Quellen mit unterschiedlichen Formaten und Qualitäten extrahiert, umgewandelt, um Konsistenz und Relevanz sicherzustellen, und schließlich in Systeme geladen werden, die in der Lage sind, die Daten effektiv zu nutzen. Herkömmliche Datenverarbeitungstechniken stoßen hierbei oft an ihre Grenzen, da spezielle Algorithmen und Werkzeuge erforderlich sind, um Aufgaben wie die Merkmalsextraktion, Rauschunterdrückung und Segmentierung zu bewältigen. Ohne ein robustes ETL-Framework, das speziell auf Audiodaten abgestimmt ist, sehen sich Unternehmen mit Ineffizienzen konfrontiert, verpassen wichtige Erkenntnisse und sind nicht in der Lage, Audiodaten für strategische Entscheidungen und eine bessere Benutzererfahrung zu nutzen. Es ist entscheidend, diese Herausforderungen zu meistern, um das volle Potenzial von Audiodaten für Innovation und operative Exzellenz auszuschöpfen [SA15; Pin24].

In diesem Zusammenhang wird die Plagiatserkennung immer wichtiger, da immer mehr Audiodaten und gesprochene Inhalte in digitalen Formaten verfügbar sind. Die korrekte Identifizierung von Urheberrechtsverletzungen, insbesondere in der Musik- und Unterhaltungsbranche, erfordert präzise Methoden zur Analyse von Audiodaten. Ein speziell angepasstes ETL-Framework könnte dazu beitragen, Plagiatserkennungssysteme zu verbessern, indem es relevante Merkmale aus Audiodaten extrahiert und die Genauigkeit der Erkennung erhöht [De+12].

Das Ziel dieser Arbeit ist es, ein robustes ETL-Framework für Audiodaten zu entwickeln, das nicht nur die Herausforderungen bei der Datenextraktion, -transformation und -ladung meistert, sondern auch die spezifischen Anforderungen der Plagiatserkennung berücksichtigt. Mit der Implementierung eines solchen Frameworks soll gezeigt werden, wie Unternehmen Audiodaten effizienter nutzen können, um fundierte strategische Entscheidungen zu treffen und rechtliche Risiken im Zusammenhang mit Urheberrechtsverletzungen zu minimieren.

1.2 Aufbau der Arbeit

Diese Arbeit ist in mehrere Kapitel gegliedert, um den Prozess von der theoretischen Grundlagenvermittlung bis zur praktischen Umsetzung nachvollziehbar darzustellen. Kapitel 2 behandelt die theoretischen Grundlagen, die für das Verständnis der folgenden Abschnitte notwendig sind. Kapitel 3 gibt einen Überblick über den aktuellen Stand der Technik im Bereich der ETL-Prozesse und Audiodatenverarbeitung. Darauf aufbauend wird in Kapitel 4 das Konzept für die Entwicklung eines spezifischen ETL-Prozesses zur Audiodatenanalyse vorgestellt. Kapitel 5 beschreibt die konkrete Implementierung und die Ergebinsse dieses Konzepts.

Kapitel 2

Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen vorgestellt, die notwendig sind, um den ETL-Prozess bei der Verarbeitung von Audiodaten zu verstehen und durchzuführen. Zunächst wird der ETL-Prozess detailliert erläutert, gefolgt von einer Einführung in die gängigen ETL-Tools, die in der Praxis häufig zum Einsatz kommen. Außerdem werden grundlegende Konzepte und Techniken im Umgang mit Audiodaten besprochen, einschließlich der verschiedenen Audioformate, der Struktur von Metadaten und der Herausforderungen, die bei der Verarbeitung dieser Daten auftreten können. Zum Abschluss wird auf die speziellen Anforderungen und Methoden der Plagiatserkennung bei Audioaufnahmen eingegangen.

2.1 ETL-Prozess

Der ETL-Prozess (Extrahieren, Transformieren, Laden) ist ein wesentlicher Bestandteil des Datenmanagements und der Datenspeicherung in der Datenwissenschaft und -analyse. Dabei werden Daten aus verschiedenen Quellen extrahiert, gemäß den betrieblichen Anforderungen transformiert und anschließend in eine Zieldatenbank oder ein Data Warehouse geladen. Für Unternehmen ist ETL unverzichtbar, um Daten aus unterschiedlichen Quellen und Formaten in einem konsistenten Datenrepository zusammenzuführen, was umfassende Analysen und fundierte Entscheidungen ermöglicht [VSS02].

2.1.1 Extraktion

Die Extraktionsphase ist der erste Schritt im ETL-Prozess und umfasst das Sammeln von Daten aus verschiedenen, oft heterogenen Quellen. Zu diesen Quellen zählen Datenbanken, Webdienste, Flatfiles und sogar unstrukturierte Datenformate. Das Hauptziel der Extraktion besteht darin, relevante Daten effizient abzurufen, ohne dabei die Leistung der Quellsysteme wesentlich zu beeinträchtigen [SVS05]. Im Kontext von Audiodaten kann die ETL-Phase das Extrahieren von Eigenschaften wie Tempo, Genre oder Stimmung aus Audiodateien beinhalten, die anschließend für Analysen oder Anwendungen im Bereich des maschinellen Lernens genutzt werden.
Wichtige Aspekte der Extraktion [SVS05]:

1.Quellenidentifikation: Verstehen und Identifizieren der verschiedenen Datenquellen, die integriert werden müssen. Dazu gehören relationale Datenbanken, NoSQL-Datenbanken, APIs und Flatfiles.

2.Datenabruf: Einsatz von Methoden zum Extrahieren von Daten, z. B. Abfragen von Datenbanken, Aufrufen von API-Endpunkten oder Lesen von Dateien. Dies kann im vollständigen Extraktionsmodus erfolgen, bei dem alle Daten abgerufen werden, oder im inkrementellen Modus, bei dem nur neue oder aktualisierte Daten extrahiert werden.

3.Datenqualität: Sicherstellen, dass die extrahierten Daten genau und konsistent sind. Dazu gehört die Validierung der Daten anhand vordefinierter Qualitätsmetriken, um Inkonsistenzen im weiteren Verlauf zu vermeiden.

2.1.2 Transformieren

Nachdem die Daten extrahiert wurden, folgt die Transformationsphase. Dieser Schritt ist entscheidend, weil die Rohdaten aus verschiedenen Quellen oft in unterschiedlichen Formaten und Strukturen vorliegen. Der Transformationsprozess sorgt dafür, dass die Daten bereinigt, angereichert und in ein Format gebracht werden, das für Analysen und Berichte geeignet ist [Sou+19]. Im Kontext von Audiodaten kann die Transformation die Umwandlung von Audiodateien in ein geeignetes Format für die Analyse beinhalten. Zudem können ETL-Audiotransformationsprozesse relevante Merkmale aus den Audiodaten extrahieren, um sie für maschinelles Lernen oder andere Analyseanwendungen vorzubereiten.

Wichtige Aspekte der Transformation [Sou+19]:

1.Datenbereinigung: Identifizieren und Korrigieren von Fehlern in den Daten. Dies umfasst das Entfernen von Duplikaten, den Umgang mit fehlenden Werten und die Korrektur von Ungenauigkeiten.

2.Datenstandardisierung: Konvertierung der Daten in ein einheitliches Format oder eine gemeinsame Struktur. Dies ist besonders wichtig, wenn Daten aus unterschiedlichen Quellen mit verschiedenen Formaten zusammengeführt werden.

3.Datenanreicherung: Ergänzen der Daten durch zusätzliche Informationen. Ein Beispiel hierfür wäre die Kombination von Verkaufsdaten mit demografischen Informationen über Kunden, um tiefere Einblicke zu gewinnen.

4.Datenaggregation: Zusammenführen von Daten, um eine übergeordnete Sicht zu erhalten. Dies könnte beispielsweise das Aggregieren von täglichen Verkaufsdaten zu monatlichen oder jährlichen Gesamtwerten umfassen.

5.Datenintegration: Zusammenführen von Daten aus unterschiedlichen Quellen, um einen einheitlichen Datensatz zu schaffen. Dieser Prozess beinhaltet oft komplexe Verknüpfungen und Zuordnungen, um sicherzustellen, dass die integrierten Daten konsistent und nützlich sind.

2.1.3 Laden

Die letzte Phase des ETL-Prozesses besteht darin, die transformierten Daten in ein Zielsystem zu laden, in der Regel ein Data Warehouse, aber es kann auch ein Data Mart, ein Data Lake oder eine andere Speicherlösung sein. Der Ladevorgang muss effizient und präzise ablaufen, um sicherzustellen, dass das Data Warehouse stets aktuelle Informationen enthält [AN08]. Bei Audiodaten kann das Laden das Speichern von Audiofunktionen und Metadaten in einer Datenbank oder einem Data Warehouse beinhalten, um eine effiziente Abfrage und Analyse der Audiodaten zu ermöglichen. Zudem können ETL-Audioladeprozesse auch die Optimierung der Speicherung der Audiodateien umfassen, etwa durch Komprimierung oder die Nutzung von Cloud-basierten Speicherlösungen, um einen schnellen Zugriff und Abruf sicherzustellen.

Wichtige Aspekte des Ladens [AN08]:

1.Auswahl des Zielsystems: Auswahl des geeigneten Ziels für die transformierten Daten. Dies kann von Faktoren wie dem Datenvolumen, der Häufigkeit der Aktualisierungen und der beabsichtigten Verwendung der Daten abhängen.

2. Ladetechniken: Implementierung verschiedener Ladetechniken basierend auf den Anforderungen. Dies kann das vollständige Laden umfassen, bei dem alle Daten von Grund auf geladen werden, oder das inkrementelle Laden, bei dem nur neue oder aktualisierte Daten geladen werden.

3.Datenvalidierung: Sicherstellen, dass die Daten korrekt und vollständig geladen wurden. Dies beinhaltet die Überprüfung von Zeilenzahlen, Prüfsummen und anderen Validierungsmethoden.

2.1.4 Bedeutung von ETL

Der ETL-Prozess ist aus mehreren Gründen von entscheidender Bedeutung [The+16]:

1.Datenkonsolidierung: ETL ermöglicht es Unternehmen, Daten aus verschiedenen Quellen in einem einzigen Repository zu konsolidieren und so eine einheitliche Sicht auf das Unternehmen zu erhalten.

2.Verbesserte Datenqualität: Durch Bereinigung und Standardisierung stellt ETL sicher, dass die für die Analyse verwendeten Daten genau und zuverlässig sind.

3.Verbesserte Entscheidungsfindung: Mit sauberen, integrierten Daten können Unternehmen genauere und umfassendere Analysen durchführen, was zu einer besseren Entscheidungsfindung führt.

4.Betriebseffizienz: Durch die Automatisierung des Datenintegrationsprozesses reduziert ETL den Zeit- und Arbeitsaufwand, der zur Vorbereitung der Daten für die Analyse erforderlich ist.

Im Zusammenhang mit Audiodaten spielt ETL eine entscheidende Rolle, um Analysen und Erkenntnisse zu ermöglichen, die Geschäftsentscheidungen beeinflussen können, wie zum Beispiel die Analyse der Kundenstimmung anhand von Sprachmustern oder das Erkennen von Anomalien in Audiosignalen.

2.2 ETL-Tools

ETL-Tools sind wesentliche Bestandteile des Datenmanagements und der Datenspeicherung. Sie ermöglichen das Extrahieren von Daten aus verschiedenen Quellen, die Umwandlung in ein nutzbares Format und das Laden in eine Zieldatenbank oder ein Data Warehouse. Für Unternehmen sind ETL-Tools unverzichtbar, um Daten aus unterschiedlichen Quellen in einem einzigen, konsistenten Datenrepository zusammenzuführen und dadurch umfassende Analysen und Berichte zu ermöglichen [PP20] .

2.2.1 Bedeutung von ETL-Tools

ETL-Tools bieten mehrere Vorteile, die für eine effiziente Datenverwaltung entscheidend sind [VSS02]:

Vorteil	**Beschreibung**
Automatisierung	ETL-Tools automatisieren den Prozess der Datenextraktion, -transformation und -ladung und reduzieren so den manuellen Aufwand und das Fehlerpotenzial.
Datenqualität	Durch Standardisierung und Bereinigung von Daten stellen ETL-Tools sicher, dass die in das Zielsystem geladenen Daten genau und zuverlässig sind.
Skalierbarkeit	ETL-Tools können große Datenmengen und komplexe Transformationen verarbeiten und eignen sich daher für Unternehmen mit umfangreichen Datenintegrationsanforderungen.
Integration	Diese Tools erleichtern die Integration von Daten aus verschiedenen Quellen und stellen sicher, dass Unternehmen eine einheitliche Sicht auf ihre Daten haben.
Effizienz	ETL-Tools verbessern die Effizienz des Datenintegrationsprozesses und ermöglichen ein schnelleres und zuverlässigeres Laden von Daten.

Tabelle 2.1: Bedeutung von ETL-Tools

2.2.2 Bekannte ETL-Tools

Mehrere ETL-Tools haben aufgrund ihrer robusten Funktionen und Fähigkeiten in der Branche an Popularität gewonnen. in den nächsten fünf Tabellen werden einige der am häufigsten verwendeten ETL-Tools erläutert:

Tool	Beschreibung
Apache Nifi	Apache Nifi ist ein Open-Source-ETL-Tool zur Automatisierung des Datenflusses zwischen Systemen. Es bietet eine intuitive Benutzeroberfläche zum Entwerfen von Datenflüssen und bietet Funktionen zur Echtzeit-Datenaufnahme.
Hauptfunktionen	**Webbasierte Schnittstelle:** Ermöglicht Benutzern das Entwerfen, Steuern und Überwachen von Datenflüssen mithilfe einer webbasierten Benutzeroberfläche. **Skalierbarkeit:** Kann große Datenmengen verarbeiten und horizontal skalieren, indem einem Cluster weitere Knoten hinzugefügt werden. **Echtzeitverarbeitung:** Unterstützt die Echtzeit-Datenaufnahme und -verarbeitung und ist daher für zeitkritische Anwendungen geeignet. **Datenherkunft:** Verfolgt den Datenfluss von der Quelle zum Ziel und gewährleistet so Rückverfolgbarkeit und Überprüfung.
Anwendungsfälle	Datenaufnahme und -verarbeitung in Echtzeit. Datenmigration und -integration zwischen verschiedenen Systemen. Datenerfassung und -analyse im Internet der Dinge (IoT).

Tabelle 2.2: Übersicht zu Apache Nifi [NiF20]

Tool	Beschreibung
Talend	Talend ist ein weit verbreitetes Open-Source-ETL-Tool, das eine Reihe von Anwendungen für Datenintegration, Datenmanagement und Datenqualität bietet. Talend bietet einen umfassenden Satz von Funktionen zum Entwerfen und Bereitstellen von ETL-Prozessen.
Hauptfunktionen	**Benutzerfreundliche Oberfläche:** Bietet eine grafische Benutzeroberfläche zum Entwerfen von ETL-Workflows, wodurch der Codierungsbedarf reduziert wird. **Umfassende Konnektivität:** Unterstützt eine breite Palette von Datenquellen und -zielen, einschließlich Datenbanken, Cloud-Diensten und APIs. **Datenqualitätstools:** Enthält integrierte Tools für Datenprofilierung, -bereinigung und -anreicherung. **Cloud-Integration:** Bietet Funktionen zur Integration mit Cloud-basierten Data Warehouses und Speicherlösungen.
Anwendungsfälle	Integration und Verwaltung von Unternehmensdaten. Cloud-Data-Warehousing und -Migration. Verbesserung und Governance der Datenqualität.

Tabelle 2.3: Übersicht zu Talend [SNR+20]

Tool	Beschreibung
Microsoft SQL Server Integration Services (SSIS)	SSIS ist ein leistungsstarkes ETL-Tool, das von Microsoft als Teil der SQL Server-Suite bereitgestellt wird. Es ist für die Datenextraktion, -transformation und -ladung konzipiert und lässt sich nahtlos in andere Microsoft-Produkte integrieren.
Hauptfunktionen	**Integration mit SQL Server:** Bietet eine enge Integration mit SQL Server und ist daher die bevorzugte Wahl für Organisationen, die Datenbanklösungen von Microsoft verwenden. **Umfangreiches Transformationsset:** Bietet eine breite Palette von Datentransformationsaufgaben, einschließlich Datenbereinigung, aggregation und -zusammenführung. **Skalierbarkeit und Leistung:** Optimiert für hohe Leistung und kann umfangreiche Datenintegrationsaufgaben bewältigen. **Skriptaufgaben:** Ermöglicht Benutzern das Schreiben benutzerdefinierter Skripte in C oder VB.NET, um die Funktionalität zu erweitern.
Anwendungsfälle	Data Warehousing und Business Intelligence. Datenmigration und -konsolidierung. Komplexe Datentransformation und -integration.

Tabelle 2.4: Übersicht zu Microsoft SQL Server Integration Services (SSIS) [MB15]

Tool	Beschreibung
Informatica PowerCenter	Informatica PowerCenter ist eine führende Datenintegrationsplattform für Unternehmen, die für ihre Robustheit und Skalierbarkeit bekannt ist. Sie unterstützt eine breite Palette von Datenintegrationsszenarien, von einfachen ETL-Prozessen bis hin zu komplexen Datenintegrationsprojekten.
Hauptfunktionen	**Hohe Leistung:** Entwickelt für die Verarbeitung großer Datenmengen und komplexe Transformationen. **Umfassende Konnektivität:** Unterstützt eine Vielzahl von Datenquellen, einschließlich lokaler und Cloud-basierter Systeme. **Erweiterte Datentransformation:** Bietet erweiterte Datentransformationsfunktionen, einschließlich Datenmaskierung, Datenvalidierung und Datenprofilierung. **Metadatenverwaltung:** Bietet Tools zur Verwaltung von Metadaten und zur Sicherstellung der Datenherkunft und -verwaltung.
Anwendungsfälle	Datenintegration für Unternehmen im großen Maßstab. Data Warehousing und Analyse. Datenverwaltung und Compliance.

Tabelle 2.5: Übersicht zu Informatica PowerCenter [MB15]

ETL-Tools sind in der heutigen datengesteuerten Landschaft unverzichtbar und ermöglichen es Unternehmen, Daten effizient und präzise zu integrieren, zu transformieren und zu laden. Werkzeuge wie Apache NiFi, Talend, SSIS und Informatica PowerCenter bieten eine Reihe von Funktionen, die auf unterschiedliche Datenintegrationsanforderungen und Umgebungen zugeschnitten sind. Diese Tools unterstützen jedoch weder Audiodateien noch das Extrahieren von Metadaten aus ihnen, da sie sich auf bestimmte Dateitypen mit festgelegten Formaten wie JSON und CSV konzentrieren. Daher ist es notwendig, ein spezielles Framework zu entwickeln, um Informationen aus Audiodateien extrahieren zu können [MB15].

2.3 Audio

In diesem Abschnitt werden die grundlegenden Konzepte rund um Audiodaten vorgestellt, die für das Verständnis der weiteren Arbeit von Bedeutung sind. Es werden die verschiedenen Arten von Audiodaten, deren Verarbeitung sowie die gängigen Formate und Metadatenstrukturen erläutert. Ziel ist es, ein solides Verständnis der technischen Grundlagen und Herausforderungen im Umgang mit Audiodaten zu schaffen, die im Kontext von ETL-Prozessen und Plagiatserkennung eine wichtige Rolle spielen.

2.3.1 Audiodaten

Audiodaten umfassen die Aufnahme, Verarbeitung, Speicherung und Übertragung von Ton. Schallwellen, also mechanische Schwingungen, die sich durch ein Medium (wie etwa Luft) bewegen, werden erfasst und in elektrische Signale oder digitale Daten umgewandelt. Diese Umwandlung ermöglicht die Analyse, Speicherung und Wiedergabe von Tönen [Wat13].

Digitales Audio verwandelt Schallwellen in diskrete numerische Werte. Dieser Prozess, bekannt als Analog-Digital-Umwandlung (ADC), tastet die Schallwelle in regelmäßigen Abständen ab und quantisiert die Amplitude in eine Reihe endlicher Werte. Zu den gängigen digitalen Audioformaten gehören PCM, MP3, AAC und FLAC. Digitales Audio ist vorteilhaft aufgrund seiner Haltbarkeit, der einfachen Bearbeitung und des Potenzials für eine hochpräzise Wiedergabe [Wat13].

Verarbeitungstechniken zur Audiokomprimierung: Verlustbehaftete Komprimierung Techniken wie MP3 und AAC reduzieren die Dateigröße, indem sie Audioinformationen entfernen, die für das menschliche Ohr weniger wahrnehmbar sind. Dies führt zu kleineren Dateien, jedoch auf Kosten der Audioqualität [Gla+04].

Verlustfreie Komprimierung: Formate wie FLAC und ALAC komprimieren Audiodaten ohne Qualitätsverlust, wobei der exakte Originalton erhalten bleibt, die Dateigröße jedoch im Vergleich zu verlustbehafteten Methoden weniger reduziert wird [Gla+04].

Digitale Signalverarbeitung (DSP) [Gla+04]:
DSP umfasst verschiedene Techniken zur Manipulation von Audiosignalen. Zu den üblichen DSP-Anwendungen gehören:

1. Entzerrung (EQ): Anpassen der Balance zwischen Frequenzkomponenten.
2. Nachhall und Echo: Hinzufügen von Nachhall, um unterschiedliche akustische Umgebungen nachzuahmen.
3. Dynamikbereichskomprimierung: Reduzieren der Lautstärke lauter Töne oder Verstärken leiser Töne,um eine ausgewogene Ausgabe zu erzielen.
4. Rauschunterdrückung: Entfernen unerwünschter Hintergrundgeräusche aus Audioaufnahmen.

Fourier-Transformation: Die Fourier-Transformation ist ein mathematisches Werkzeug, mit dem ein Signal aus seinem ursprünglichen Zeitbereich in eine Darstellung im Frequenzbereich umgewandelt wird. Dies ist für die Analyse des Frequenzinhalts von Audiosignalen unerlässlich und wird häufig in verschiedenen Audioverarbeitungsanwendungen verwendet, einschließlich Spektralanalyse, Filterung und Audiokomprimierung [Gla+04].

Anwendungen von Audiodaten in Musik und Unterhaltung [RDS05] :
Audiodaten sind in der Musikindustrie von zentraler Bedeutung und unterstützen Aufnahme, Produktion, Vertrieb und Wiedergabe. Digitale Audio-Workstations (DAWs) und Software wie Pro Tools, Logic Pro und Ableton Live ermöglichen Musikern und Produzenten, Musik mit hoher Präzision zu erstellen, zu bearbeiten und zu mischen. Auch in der Telekommunikation spielen Audiodaten eine wichtige Rolle. Technologien wie Voice over IP (VoIP) und mobile Kommunikation nutzen Audiodaten, um Sprachsignale über digitale Netzwerke zu übertragen. Durch Techniken wie Komprimierung und Paketierung wird dabei eine effiziente und klare Sprachübertragung ermöglicht, selbst bei begrenzter Bandbreite.
Medizinische Anwendungen und Audiodaten werden in Diagnosewerkzeugen wie Stethoskopen und Ultraschallgeräten verwendet. Die Phonokardiographie beispielsweise zeichnet Herztöne auf, die analysiert werden können, um Anomalien zu erkennen. Audiometrietests messen die Hörschärfe anhand verschiedener Frequenzen und Lautstärken [RDS05].
Sicherheit und Überwachung Audiodaten sind in Sicherheitssystemen zur Überwachung und Aufzeichnung von Umgebungen von entscheidender Bedeutung. Spracherkennung kann zur biometrischen Authentifizierung verwendet werden, und durch Tonanalyse können Anomalien oder unbefugter Zugriff in sensiblen Bereichen erkannt werden [RDS05].

Herausforderungen bei Audiodaten [Ver+16]:

1.Datenspeicherung und Bandbreite: Hochwertige Audiodaten, insbesondere unkomprimierte Formate, können für die Übertragung erheblichen Speicherplatz und Bandbreite erfordern. Effiziente Komprimierungstechniken und Fortschritte in der Speichertechnologie sind unerlässlich, um diese Anforderungen zu erfüllen.
2.Rauschen und Verzerrung: Bei der Aufnahme von Audio in realen Umgebungen muss man sich oft mit Rauschen und Verzerrungen auseinandersetzen. Techniken zur Rauschunterdrückung und Signalverbesserung sind entscheidend für die Verbesserung der Audioqualität.
3.Datenschutz: Die weit verbreitete Verwendung von Audiodaten, insbesondere in Überwachungssystemen und Sprachassistenten, wirft Datenschutzbedenken auf. Die Gewährleistung einer verantwortungsvollen Erfassung, Speicherung und Verarbeitung dieser Daten ist ein wichtiges Anliegen.
4.Geistiges Eigentum: Audiodaten, insbesondere in Form von Musik und Aufnahmen, unterliegen den Gesetzen zum geistigen Eigentum. Der Schutz der Rechte der Urheber bei gleichzeitiger Gewährleistung einer fairen Nutzung und Verbreitung ist eine ständige Herausforderung.

2.3.2 Audioformate

Audioformate sind Container und Kodierungssysteme, die zum Speichern und Verwalten digitaler Audiodaten verwendet werden. Jedes Format verfügt über einzigartige Eigenschaften, die seine Eignung für unterschiedliche Anwendungen bestimmen, von der Hi-Fi-Audioproduktion bis hin zum effizienten Streaming [Dob00].

Audioformate lassen sich allgemein in unkomprimierte, verlustfrei komprimierte und verlustbehaftet komprimierte Formate einteilen. Jede dieser Kategorien hat spezifische Vor- und Nachteile hinsichtlich Audioqualität, Dateigröße und Verarbeitungsanforderungen. Unkomprimierte Audioformate speichern Audiodaten in ihrer Rohform ohne jegliche Komprimierung, wodurch die ursprüngliche Audioqualität vollständig erhalten bleibt. Diese Formate werden häufig in der professionellen Audioproduktion und für Archivierungszwecke verwendet, bei denen höchste Wiedergabetreue von größter Bedeutung ist.
Im Folgenden werden einige Audioformat erwähnt [Beh15]:
WAV (Waveform Audio File Format): WAV wurde von Microsoft und IBM entwickelt und ist ein Standardformat zur Speicherung von unkomprimiertem Audio auf Windows-Systemen. Es unterstützt unterschiedliche Bittiefen, Abtastraten und Kanäle, wodurch es für eine Vielzahl von Audioanwendungen flexibel einsetzbar ist. WAV-Dateien sind zwar groß, bieten jedoch die höchste Audioqualität.

AIFF (Audio Interchange File Format): AIFF wurde von Apple entwickelt und ist WAV sehr ähnlich, wird jedoch hauptsächlich auf Mac-Systemen verwendet. Es speichert ebenfalls unkomprimierte Audiodaten und ist ideal für die Aufnahme und Bearbeitung von hochwertigem Audio.

Verlustfreie komprimierte Audioformate: Verlustfreie komprimierte Audioformate reduzieren die Dateigröße, ohne die Audioqualität zu beeinträchtigen. Diese Formate sind besonders nützlich, wenn Speicherplatz begrenzt ist, die Qualität jedoch erhalten bleiben muss, wie bei der Archivierung und Verteilung von Audio.

FLAC (Free Lossless Audio Codec): FLAC ist ein beliebtes Open-Source-Format, das es ermöglicht, Audiodaten ohne Qualitätsverlust zu komprimieren. Es reduziert die Dateigröße deutlich, während die ursprüngliche Tonqualität vollständig erhalten bleibt. FLAC ist bei Audiophilen sehr geschätzt und wird häufig für hochauflösende Audio-Downloads genutzt.

ALAC (Apple Lossless Audio Codec): ALAC wurde von Apple entwickelt und bietet eine verlustfreie Komprimierung ähnlich wie FLAC, ist jedoch speziell für eine nahtlose Integration in das Apple-Ökosystem, einschließlich iTunes und iOS-Geräten, ausgelegt.

APE (Monkey's Audio): APE ist ein weiteres verlustfreies Audioformat, das hohe Kompressionsraten bietet. Es wird weniger häufig unterstützt als FLAC oder ALAC, findet aber in bestimmten Nischenanwendungen Verwendung.

Verlustbehaftete komprimierte Audioformate: Verlustbehaftete komprimierte Audioformate reduzieren die Dateigröße, indem Audiodaten entfernt werden, die für das menschliche Gehör weniger wichtig sind. Diese Formate bieten eine deutlich kleinere Dateigröße auf Kosten der Audioqualität und sind daher gut für Streaming und mobile Medien geeignet.

MP3 (MPEG-1 Audio Layer III): MP3 ist das am weitesten verbreitete verlustbehaftete Audioformat, das für sein gutes Verhältnis zwischen Dateigröße und Audioqualität bekannt ist. Es wird von fast allen Audioplayern und Geräten unterstützt und ist das Standardformat für Musikverteilung und -streaming.

AAC (Advanced Audio Coding): AAC bietet eine bessere Klangqualität als MP3 bei vergleichbaren Bitraten. Es wird von vielen Streaming-Diensten wie Apple Music und YouTube verwendet und ist das Standardformat für Audio im MP4-Container.

OGG Vorbis: OGG Vorbis ist ein verlustbehaftetes Open-Source-Audioformat, das für seine effiziente Komprimierung und gute Klangqualität bei niedrigen Bitraten bekannt ist. Es wird von einigen Streaming-Diensten genutzt und ist in der Open-Source-Community weit verbreitet.

Opus: Opus ist ein vielseitiger und hocheffizienter verlustbehafteter Audio-Codec, der speziell für interaktive Anwendungen wie VoIP, Videokonferenzen und Live-Streaming entwickelt wurde. Er passt sich verschiedenen Bitraten und Netzwerkbedingungen an und bietet eine solide Leistung für unterschiedliche Audioqualitäten.

Wichtige Merkmale von Audioformaten [Mou05]:

1.Bitrate: Die Bitrate gibt an, wie viele Daten pro Sekunde verarbeitet werden, und wird in Megabit pro Sekunde (Mbit/s) gemessen. Sie variiert je nach Audioformat: Bei CD-DA beträgt sie beispielsweise 1,4112 Mbit/s, während DVD-Audio Bitraten von bis zu 9,6 Mbit/s erreicht. Eine höhere Bitrate führt in der Regel zu besserer Audioqualität, erfordert jedoch auch mehr Speicherplatz und Bandbreite

2.Abtastrate: Die Abtastrate gibt an, wie oft ein Audiosignal pro Sekunde abgetastet wird und wird in Hertz (Hz) gemessen. Für CDs wird meist eine Abtastrate von 44,1 kHz verwendet, die als Standard für CD-Qualität gilt. Höhere Abtastraten, wie etwa 192 kHz bei DVD-Audio, bieten eine bessere Klangtreue, führen jedoch auch zu größeren Dateien.

3.Bittiefe: Die Bittiefe gibt an, wie viele Bits verwendet werden, um einen einzelnen Audiowert darzustellen. Eine höhere Bittiefe ermöglicht einen größeren Dynamikumfang und eine präzisere Aufnahme von Audiosignalen. Während CDs typischerweise eine Bittiefe von 16 Bit nutzen, verwenden Formate wie DVD-Audio eine Bittiefe von 24 Bit, um eine höhere Detailgenauigkeit und einen erweiterten Dynamikbereich zu erreichen.

2.3.3 Audiometadata

Audiometadaten beziehen sich auf die beschreibenden, technischen und administrativen Informationen, die in Audiodateien eingebettet oder mit ihnen verknüpft sind. Diese Metadaten sind für die Organisation, Entdeckung und Verwaltung von Audioinhalten von entscheidender Bedeutung. Das Verständnis der Arten von Audiometadaten, ihrer Anwendungen, Standards und Herausforderungen ist für die effiziente Handhabung von Audiodaten in verschiedenen Kontexten, einschließlich Musik, Rundfunk und Archivierung, von entscheidender Bedeutungen [Gli+11].

Standards für Audio-Metadaten [DNV06]:	**Beschreibung**
ID3	ID3 ist ein Metadaten-Containerformat, das häufig in MP3-Dateien zum Einsatz kommt. Es ermöglicht das Einbetten von Textinformationen wie Titel, Künstler, Album und Genre. Es gibt zwei Versionen von ID3-Tags: ID3v1, das nur begrenzten Platz für Metadaten bietet, und ID3v2, das umfangreichere Tagging-Funktionen ermöglicht.
XMP (Extensible Metadata Platform)	XMP ist ein von Adobe entwickelter Standard zum Einbetten von Metadaten in digitale Mediendateien. Er unterstützt eine breite Palette von Metadatenfeldern und wird in verschiedenen Audiodateiformaten verwendet, darunter WAV und MP3.
APE (Monkey's Audio)	APE-Tags werden in Monkey's Audio-Dateien verwendet und bieten eine flexible und effiziente Möglichkeit zum Speichern von Metadaten. Sie unterstützen eine breite Palette von Metadatenfeldern und sind für ihre Robustheit bekannt.
Vorbis-Kommentare	Vorbis-Kommentare werden in Ogg Vorbis- und FLAC-Dateien verwendet. Sie ermöglichen das Einbetten benutzerdefinierter Metadatenfelder und bieten Flexibilität für verschiedene Arten von Audioinhalten.
BWF (Broadcast Wave Format)	BWF ist eine Erweiterung des WAV-Dateiformats und wurde für Rundfunk- und professionelle Audioanwendungen entwickelt. Es enthält zusätzliche Metadatenfelder zur Verwaltung übertragungsspezifischer Informationen wie Cue-Points und Zeitstempel.

Tabelle 2.6: Standards für Audio-Metadaten

Art der Metadaten [Hor22; DNV06]:	**Beschreibung**
Beschreibende Metadaten	Liefern Informationen über den Inhalt der Audiodatei. Diese Art von Metadaten ist wichtig für die Identifizierung, Kategorisierung und den Abruf von Audiodateien. •Titel: Der Name der Audiodatei oder des Titels. •Künstler/Interpret: Die Person oder Gruppe, die den Audioinhalt aufführt. •Album: Die Sammlung oder das Album, zu dem die Audiodatei gehört. •Genre: Die Kategorie oder Art der Musik, z. B. Rock, Jazz, Klassik usw. •Titelnummer: Die Position der Audiodatei innerhalb eines Albums. Veröffentlichungsdatum: Das Datum, an dem der Audioinhalt veröffentlicht wurde. •Komponist: Die Person, die die Musik komponiert hat. •Liedtext: Der Textinhalt des Lieds, falls zutreffend.
Technische Metadaten	Beschreiben die technischen Attribute der Audiodatei. Diese Informationen sind entscheidend für das Verständnis des Formats, der Qualität und der Wiedergabeanforderungen der Datei. •Dateiformat: Das Format, in dem die Audiodatei gespeichert ist, z. B. MP3, WAV, AAC, FLAC usw. •Bitrate: Die Anzahl der pro Zeiteinheit verarbeiteten Bits, normalerweise gemessen in Kilobit pro Sekunde (kbps). •Abtastrate: Die Anzahl der pro Sekunde übertragenen Audioproben, gemessen in Hertz (Hz). •Bittiefe: Die Anzahl der Bits, die zur Darstellung jeder Audioprobe verwendet werden. •Dauer: Die Gesamtlänge der Audiodatei. •Codec: Der Algorithmus, der zum Komprimieren oder Dekomprimieren der Audiodatei verwendet wird.
Administrative Metadaten	Enthalten Informationen, die zum Verwalten und Aufbewahren von Audiodateien erforderlich sind. Diese Metadaten sind für die Archivierung, das Rechtemanagement und die Gewährleistung der langfristigen Zugänglichkeit von entscheidender Bedeutung. •Copyright-Informationen: Details zu den Eigentums- und Nutzungsrechten der Audiodatei. •Herausgeber: Die für die Verbreitung des Audioinhalts verantwortliche Stelle. •ISRC (International Standard Recording Code): Eine eindeutige Kennung für Tonaufnahmen und Musikvideos. •Erstellungsdatum: Das Datum, an dem die Audiodatei erstellt oder aufgenommen wurde. •Dateigröße: Die Größe der Audiodatei, normalerweise in Bytes gemessen. •Prüfsumme: Ein Wert, der zur Überprüfung der Integrität der Audiodatei verwendet wird.

Tabelle 2.7: Arten von Audio-Metadaten

2.4 Plagiatserkennung von Audio

Plagiate werden traditionell mit schriftlichen Werken in Verbindung gebracht, umfassen im digitalen Zeitalter jedoch mittlerweile verschiedene Medien, darunter auch Audio. Audio-Plagiate sind die unbefugte Verwendung oder Nachahmung von Tonaufnahmen, Musikkompositionen, Voiceovers oder anderen Audioelementen anderer Urheber ohne entsprechende Quellenangabe. Dieses Problem wird durch die Verbreitung digitaler Plattformen, die Audioinhalte leicht zugänglich und teilbar machen, immer bedeutender. Wie bei anderen Formen von Plagiaten verletzt Audio-Plagiat nicht nur die Rechte des ursprünglichen Urhebers, sondern untergräbt auch die Integrität der Kreativbranche [Shi+14].

2.4.1 Arten von Audio-Plagiaten

Audio-Plagiate können sich auf verschiedene Arten manifestieren, wobei jede davon einzigartige Herausforderungen für die Erkennung und Prävention darstellt:
1. Direktes Kopieren: Dabei wird ein Audiosegment genau so verwendet, wie es im Originalwerk vorkommt. Dies kann ein Musikbeispiel, ein Ausschnitt aus einem Podcast oder ein Sounddesign sein, das ohne Erlaubnis verwendet wird. Diese Art ist oft am einfachsten durch digitale Fingerabdrücke und andere technologische Tools zu erkennen [LML22].
2. Melodische und rhythmische Ähnlichkeiten: Diese Art des Plagiats entsteht, wenn jemand die Melodie oder den Rhythmus eines bestehenden Musikstücks imitiert. Obwohl das neue Werk vielleicht keine direkte Kopie ist, können die Ähnlichkeiten dennoch so bedeutend sein, dass Plagiatsansprüche gerechtfertigt sind. Bekannte Fälle wie der Rechtsstreit zwischen Robin Thickes „Blurred Lines“ und Marvin Gayes „Got to Give It Up“ verdeutlichen, wie komplex es sein kann, die Originalität einer musikalischen Komposition zu bestimmen [LML22].
3. Textplagiat: Dies geschieht, wenn Songtexte ohne Erlaubnis kopiert werden. Songtexte gelten als literarische Werke und ihre wörtliche oder leicht abgeänderte Übernahme ohne Nennung des ursprünglichen Songwriters stellt ein Plagiat dar [LML22].
4. Voiceover-Plagiat: Bei dieser Art handelt es sich um die unbefugte Verwendung von Sprachaufnahmen, wie z. B. Erzählungen oder Kommentare, in neuen Audioproduktionen. Dies ist insbesondere im Zusammenhang mit Podcasts, Hörbüchern und Werbung relevant [LML22].

2.4.2 Erkennung und Prävention

Aufgrund der nuancierten Natur von Klang ist das Erkennen von Audioplagiaten oft schwieriger als das Erkennen von Textplagiaten. Es gibt jedoch mehrere Methoden und Tools, die dieses Problem angehen [Par+22]:

1.Digitaler Fingerabdruck: Diese Technologie analysiert und identifiziert einzigartige Audiomuster in einer Aufnahme und erstellt einen "Fingerabdruck", der mit einer Datenbank vorhandener Werke verglichen werden kann. Dienste wie Shazam nutzen diese Technik zur Erkennung von Songs, aber ähnliche Methoden können auch die unbefugte Nutzung von Audiosegmenten aufdecken.

2.Spektralanalyse: Bei der Spektralanalyse wird das Frequenzspektrum einer Audiodatei untersucht, um Ähnlichkeiten mit anderen Aufnahmen festzustellen. Diese Methode ist besonders effektiv bei der Erkennung von kopiertem Sounddesign und Geräuschkreationen.

3.Automatisierte Content-ID-Systeme: Plattformen wie YouTube verwenden Content-ID-Systeme, die hochgeladene Inhalte automatisch mit einer Datenbank urheberrechtlich geschützter Audiodateien abgleichen. Wenn eine Übereinstimmung gefunden wird, kann das System den Upload blockieren, ihn für den ursprünglichen Ersteller monetarisieren oder den Urheberrechtsinhaber benachrichtigen.

4.Rechtliche und ethische Richtlinien: Die Einhaltung von Urheberrechtsgesetzen und ethischen Richtlinien ist entscheidend, um Audioplagiate zu verhindern. Es ist wichtig, die Urheber über die Bedeutung origineller Arbeit und die richtige Verwendung und Quellenangabe bestehender Audiomaterialien aufzuklären. Darüber hinaus kann die Umsetzung strenger Richtlinien und Strafen für Plagiate innerhalb von Organisationen und Plattformen potenzielle Täter abschrecken.

Kapitel 3

Stand der Technik

In diesem Kapitel wird der aktuelle Stand der Technik im Bereich der ETL-Prozesse und der Audiodatenverarbeitung betrachtet. Es werden die derzeit gängigen Tools und Technologien vorgestellt, die in der Praxis eingesetzt werden, sowie die Herausforderungen und Fortschritte bei der Extraktion, Transformation und Nutzung von Audiodaten diskutiert. Das Ziel ist es, einen Überblick über die aktuellen Entwicklungen und bewährten Verfahren zu geben, die als Grundlage für die Konzeption und Implementierung der in dieser Arbeit entwickelten Lösung dienen.

3.1 Überblick über ETL-Tools und Prozesse

Der ETL-Prozess ist ein entscheidender Bestandteil im Data Warehousing und der Datenintegration. Er ermöglicht es Unternehmen, ihre Daten zu konsolidieren und für Analysezwecke aufzubereiten. Durch das Extrahieren von Daten aus verschiedenen Quellen, deren Transformation entsprechend den geschäftlichen Anforderungen und das Laden in ein zentrales Repository gewinnen Unternehmen wertvolle Einblicke und können fundierte Entscheidungen auf Basis ihrer Daten treffen. Dieser strukturierte Ansatz stellt die Qualität, Konsistenz und Verfügbarkeit der Daten sicher und unterstützt somit effektive Business Intelligence und Analysen [Kha+24].

Beispiel für einen ETL-Prozess [Pha20] :

Phase	Beschreibung und Aufgaben
1. Extraktionsphase	In dieser Phase werden Audiodateien (z.B. MP3, WAV) aus ihrer Quelle extrahiert. Die Extraktion kann das Lesen der Dateien aus einem lokalen Speicher, einem Cloud-Speicher oder einem Online-Repository umfassen. **Aufgaben:** o Alle Audiodateien aus dem Quellverzeichnis lesen. o Metadaten aus jeder Audiodatei extrahieren (z. B. Künstler, Album, Genre, Titellänge). o Audiofunktionen extrahieren (z.B. Tempo, Tonhöhe, Lautstärke, MFCCs – Mel-Frequency Cepstral Coefficients).
2.Transformationsphase	In dieser Phase werden die extrahierten Daten bereinigt und in ein für die Analyse oder Speicherung geeignetes Format umgewandelt. Datentransformationen können Normalisierung, Kodierung oder das Aggregieren von Funktionen umfassen. **Aufgaben:** o Funktionswerte (z. B. MFCC, Tempo) auf eine gemeinsame Skala normalisieren. o Fehlende oder inkonsistente Metadaten verarbeiten (z.B. fehlende Künstlernamen). o Kategorische Metadaten zur einfacheren Analyse in numerische Werte kodieren (z.B. Genre).
3. Ladephase	In dieser Phase werden die transformierten Daten zur weiteren Verwendung in ein Zielsystem wie eine Datenbank oder ein Data Warehouse geladen. Dabei kann es sich um die Speicherung von Metadaten und Audiofunktionen zur Verwendung in Anwendungen wie Musikempfehlungssystemen, Audioanalyseplattformen oder zur Archivierung handeln. **Ziel:** Eine relationale Datenbank (z.B. PostgreSQL, MySQL),ein Data Warehouse, oder eine NoSQL-Datenbank (z.B. MongoDB).

Tabelle 3.1: Beispiel für einen ETL-Prozess

3.2 Audiodatenextraktion

In diesem Abschnitt wird der Prozess der Audiodatenextraktion im Detail erklärt. Es wird erläutert, wie Audiodaten aus verschiedenen Quellen gesammelt werden und welche Techniken und Herausforderungen dabei auftreten. Der Schwerpunkt liegt auf den Methoden zur effektiven und effizienten Extraktion von Audiodaten, die als Grundlage für die weitere Verarbeitung und Analyse dienen.

3.2.1 Quellen für Audiodaten

Vielfalt der Audioquellen: Audiodaten können aus verschiedenen Quellen gewonnen werden, die jeweils spezifische Merkmale und Anforderungen aufweisen. Zu den wichtigsten Quellen gehören Mikrofone und Aufnahmegeräte, Streaming-Plattformen, Telekommunikationssysteme sowie Podcasts und Radiosendungen [Qam+21; LC05].

Quelle	Beschreibung
Mikrofone und Aufnahmegeräte	Diese Geräte erfassen Schallwellen und wandeln sie in digitale Signale zur Speicherung und Verarbeitung um.
Professionelle Aufnahmegeräte	**Merkmale:** Hochqualitative Tonaufnahme mit unkomprimierten Formaten wie WAV für maximale Klarheit. **Anwendungen:** Musikproduktion, Film- und Fernsehproduktion, Live-Konzertaufnahmen. **Herausforderungen:** Umgang mit großen Dateigrößen und Sicherstellung verzerrungsfreier Aufnahmen.
Unterhaltungselektronik	**Merkmale:** In Alltagsgeräten wie Smartphones und Laptops integrierte Mikrofone mit Funktionen zur Rauschunterdrückung. **Anwendungen:** Sprachassistenten, Videokonferenzen, Erstellung von Social-Media-Inhalten. **Herausforderungen:** Variable Audioqualität je nach Gerät und Umgebung sowie Integration unterschiedlicher Datenquellen.
Outdoor-Aufnahmegeräte	**Merkmale:** Robuste, tragbare Geräte zur Erfassung von Umgebungsgeräuschen und Naturklängen. **Anwendungen:** Sounddesign, Umweltforschung, Dokumentation natürlicher Klanglandschaften. **Herausforderungen:** Einfluss unvorhersehbarer Umgebungsbedingungen und Verwaltung umfangreicher Aufnahmedaten.

Tabelle 3.2: Übersicht zu Mikrofonen und Aufnahmegeräten

Quelle	Beschreibung
Podcasts und Radiosendungen	Strukturierte Audioinhalte wie Podcasts und Radiosendungen bieten umfangreiche Daten für diverse Analysezwecke.
Podcasts	**Merkmale:** On-Demand-Audioinhalte in standardisierten Formaten wie MP3 oder AAC. **Anwendungen:** Inhaltsempfehlungen, Themen- und Stimmungsanalyse, Werbewirksamkeitsstudien. **Herausforderungen:** Unterschiedliche Audioqualitäten und effektive Kategorisierung großer Mengen episodischer Inhalte.
Radiosendungen	**Merkmale:** Live oder aufgezeichnete Inhalte über verschiedene Übertragungswege wie AM/FM oder Internet. **Anwendungen:** Echtzeit-Höreranalysen, Inhaltsarchivierung, Musik- und Spracherkennung. **Herausforderungen:** Kontinuierliche Datenerfassung und Verarbeitung sowie Variabilität in Sendequalität und -formaten.

Tabelle 3.3: Übersicht zu Podcasts und Radiosendungen [Cla+20]

Quelle	Beschreibung
Streaming-Plattformen	Plattformen wie Spotify und YouTube bieten umfangreiche Audiodatenbestände, die jedoch mit spezifischen Herausforderungen verbunden sind.
Musik-Streaming-Dienste	**Merkmale:** Hochqualitative, komprimierte Audiostreams mit proprietären Formaten. **Anwendungen:** Musikanalyse, Empfehlungssysteme, Urheberrechtsmanagement. **Herausforderungen:** Rechtliche Beschränkungen bei der Datennutzung und Vielfalt der Formate und Bitraten.
Video-Streaming-Dienste	**Merkmale:** Audiospuren innerhalb von Videoinhalten in verschiedenen Sprachen und Qualitäten. **Anwendungen:** Spracherkennung,Inhaltsmoderation, Zugänglichkeits verbesserungen. **Herausforderungen:** Synchronisierung von Audio und Video sowie Handling großer Datenmengen.
Live-Streaming-Plattformen	**Merkmale:** Echtzeit-Audiostreams mit nutzergenerierten Inhalten unterschiedlicher Qualität. **Anwendungen:** Echtzeit-Stimmungsanalyse, Content-Moderation, Publikumsanalyse. **Herausforderungen:** Verarbeitung mit geringer Latenz und Variabilität der Audioformate.

Tabelle 3.4: Übersicht zu Streaming-Plattformen [YR22]

Quelle	Beschreibung
Telekommunikationssysteme	Diese Systeme umfassen traditionelle Telefonie, VoIP und moderne Kommunikationsanwendungen und sind essenziell für verschiedene Analysezwecke.
Traditionelle Telefonie	**Merkmale:** Übertragung von Audio über das öffentliche Telefonnetz mit begrenzter Qualität. **Anwendungen:** Anrufqualitätsüberwachung, Kundenzufriedenheitsanalysen, Betrugserkennung. **Herausforderungen:** Integration mit veralteter Infrastruktur und Sicherstellung von Datenschutz.
VoIP-Dienste	**Merkmale:** Digitale Audioübertragung über IP-Netzwerke mit effizienter Bandbreitennutzung. **Anwendungen:** Sprachanalyse, automatische Transkription, Verbesserung der Nutzererfahrung. **Herausforderungen:** Umgang mit Netzwerkinstabilitäten und Integration verschiedener Plattformen.
Moderne Kommunikations-Apps	**Merkmale:** Audioübertragung über Apps wie WhatsApp und Slack mit Ende-zu-Ende-Verschlüsselung. **Anwendungen:** Echtzeit-Kollaborationsanalyse, Nutzerbindungsverbesserung, Compliance-Überwachung. **Herausforderungen:** Verarbeitung verschlüsselter Daten und Vielfalt der Audioformate.

Tabelle 3.5: Übersicht zu Telekommunikationssystemen

3.2.2 Extraktionstechniken

Die Extraktion von Audiodaten kann durch verschiedene Techniken erfolgen, die je nach Quelle und Anwendungsanforderungen angepasst werden. Zu den gängigen Methoden zählen die Nutzung von APIs für Streaming-Dienste, direkte Aufzeichnung und Speicherung sowie die Extraktion von Dateien aus lokalen oder vernetzten Speichersystemen [Kos+23]. Jede dieser Methoden hat ihre eigenen Vorteile, Herausforderungen und Implementierungsstrategien.

APIs für Streaming-Dienste [Kos+23]:
APIs bieten einen effizienten Weg, um auf Audiodaten und Metadaten von Streaming-Plattformen wie Spotify, YouTube und SoundCloud zuzugreifen.

API	Beschreibung
Spotify-API	Die Spotify-API ermöglicht Entwicklern den Zugang zu umfassenden Daten wie Titeln, Wiedergabelisten und Audiofeatures. Durch die Authentifizierung via OAuth 2.0 können Abfragen an verschiedene Endpunkte gesendet und die zurückgegebenen Daten analysiert werden. Ein Beispiel ist der 'Get Audio Features' Endpunkt, der Attribute wie Tempo und Tonart eines Titels liefert. Herausforderungen bestehen in den Lizenz- und Urheberrechtsbeschränkungen, die den direkten Zugriff auf Audiodateien oft einschränken.
YouTube-Daten-API	Diese API bietet Zugriff auf Videometadaten, einschließlich der Audiospuren in den Videos. Entwickler können Videoinformationen und Audiospuren abrufen und Tools wie youtube-dl integrieren, um Audio aus Videos zu extrahieren. Die Einhaltung von YouTubes Richtlinien und Urheberrechtsbestimmungen stellt dabei eine Herausforderung dar.
SoundCloud-API	Die SoundCloud-API ermöglicht den Zugriff auf Trackdaten, Wiedergabelisten und Benutzerprofile. Audiodaten können gestreamt und relevante Informationen extrahiert werden. Wie bei anderen Plattformen gibt es Einschränkungen beim direkten Herunterladen von Audiodateien, und API-Ratenbeschränkungen müssen berücksichtigt werden.

Tabelle 3.6: Übersicht zu Audio-APIs

Direkte Aufzeichnung und Speicherung [Kos+23]:
Diese Methode erfasst Audiodaten entweder kontinuierlich oder ereignisgesteuert über Mikrofone oder andere Aufnahmegeräte.
Kontinuierliche Aufzeichnung:
Bei dieser Methode werden Audiodaten ununterbrochen erfasst und in Segmenten gespeichert, um die Dateigröße und die Verarbeitungsanforderungen zu optimieren. Typische Anwendungen sind Sicherheitssysteme und sprachgesteuerte Assistenten. Die Herausforderung besteht darin, große Datenmengen effizient zu speichern und Rauschunterdrückungstechniken zu integrieren.

Ereignisgesteuerte Aufzeichnung:
Hierbei wird die Aufzeichnung nur bei bestimmten Ereignissen wie Geräuscherkennung oder Sprachaktivierung ausgelöst. Diese Methode spart Speicherplatz und Verarbeitungsleistung. Herausforderungen liegen in der genauen Erkennung der Ereignisse und der Integration von Echtzeitverarbeitung.
Integration von Cloud-Speicher:
Cloud-Speicherlösungen wie AWS S3 und Google Cloud Storage bieten skalierbare Möglichkeiten zur Speicherung von Audiodaten. Die Audiodaten können in Echtzeit oder in Batch-Prozessen hochgeladen werden. Wichtige Aspekte sind die sichere Übertragung und Speicherung sowie das Management der damit verbundenen Kosten.
Dateisystem-Extraktion [Kos+23]:
Die Dateisystem-Extraktion umfasst den Zugriff auf und die Verarbeitung von Audiodateien aus lokalen oder vernetzten Speichersystemen.
Extraktion aus lokalem Speicher:
Diese Methode greift auf Audiodateien zu, die auf Festplatten, SSDs oder tragbaren Speichermedien gespeichert sind. Durch Skripte oder Softwareanwendungen können Verzeichnisse gescannt und relevante Audiodaten extrahiert werden. Herausforderungen bestehen in der Kompatibilität mit verschiedenen Dateisystemen und der effizienten Verarbeitung großer Dateimengen.
Vernetzte Speichersysteme:
Network Attached Storage (NAS) und Storage Area Networks (SAN) bieten über Netzwerke zugängliche Speicherlösungen. Der Zugriff auf Audiodateien erfolgt über Protokolle wie SMB/CIFS oder NFS. Netzwerklatenz und Bandbreitenbeschränkungen können die Extraktion beeinflussen, und die sichere Verwaltung der Zugriffskontrollen ist entscheidend.
Hybride Ansätze:
Kombinationen aus lokaler und Cloud-Speicherung optimieren die Datenverfügbarkeit und Speichereffizienz. Audiodaten werden zunächst lokal gespeichert und später in die Cloud übertragen. Die Verwaltung von Synchronisierungsprozessen und die Sicherstellung der Datenintegrität sind wichtige Herausforderungen.

Die Auswahl der geeigneten Technik zur Audiodatenextraktion hängt von den spezifischen Anforderungen der Anwendung ab. APIs für Streaming-Dienste bieten einen strukturierten Zugang zu umfangreichen Audioinhalten und Metadaten, während direkte Aufzeichnung und Speicherung die Erfassung von Echtzeitdaten ermöglichen. Die Dateisystem-Extraktion ist ideal für die Verarbeitung von vorab aufgezeichneten Audiodaten und die Integration in bestehende Arbeitsabläufe. Jede dieser Methoden bringt eigene Herausforderungen mit sich, von der Handhabung verschiedener Audioformate und Datenmengen bis hin zur Sicherstellung der Datensicherheit und -integrität. Durch die gezielte Auswahl und Implementierung dieser Techniken können Audiodaten effektiv für verschiedenste Anwendungen genutzt werden, von der Medienanalyse bis zur Sicherheitsüberwachung [Kos+23].

3.2.3 Herausforderungen bei der Extraktion

Datenformatvariationen Audiodaten sind von Natur aus vielfältig und stammen oft von verschiedenen Geräten und Anwendungen, die jeweils unterschiedliche Dateiformate verwenden. Zu den gängigen Audioformaten gehören WAV, MP3 und AAC, jedes mit einzigartigen Eigenschaften und Verwendungsmöglichkeiten.

1• WAV (Waveform Audio File Format): Aufgrund der großen Dateigröße sind WAV-Dateien weniger für Anwendungen geeignet, die eine effiziente Speicherung und Übertragung erfordern. Während der Extraktion erhöht die Sicherstellung der Kompatibilität mit Systemen, die komprimierte Formate erwarten, die Komplexität.
2• MP3 (MPEG Audio Layer III): Die verlustbehaftete Komprimierung von MP3 kann die Audioqualität beeinträchtigen, was für Anwendungen problematisch sein kann, die Hi-Fi-Sound erfordern. Beim Extrahieren von MP3-Dateien muss sichergestellt werden, dass die Komprimierungsartefakte die nachfolgende Verarbeitung nicht beeinträchtigen.
3• AAC (Advanced Audio Coding): Obwohl AAC effizienter als MP3 ist, fügt die Vielfalt der Bitraten und Profile dem Extraktionsprozess eine weitere Komplexitätsebene hinzu. Es können Kompatibilitätsprobleme auftreten, die eine Konvertierung in andere Formate erforderlich machen [FS16].

Normalisierungsprozess: Die Normalisierung ist entscheidend, um Audiodaten für eine konsistente Verarbeitung und Analyse zu standardisieren. Der Prozess umfasst die Konvertierung verschiedener Audioformate in ein einheitliches Format. Dieser Schritt stellt die Kompatibilität sicher und vereinfacht nachfolgende Transformationen und Analysen. Tools wie FFmpeg erleichtern die Formatkonvertierung, aber der Prozess muss mit potenziellen Qualitätseinbußen und Datenverlusten sorgfältig umgehen [Sab+17].

Große Dateigrößen [See21]: Die Größe von Audiodateien stellt bei der Extraktion eine erhebliche Herausforderung dar, insbesondere in Bezug auf Speicherung und Verarbeitung. Hochwertige Audiodateien, insbesondere unkomprimierte Formate wie WAV, können viel Speicherplatz beanspruchen und erfordern eine beträchtliche Bandbreite für die Übertragung.

1• Speicherherausforderungen:

• Volumen: Große Mengen an Audiodaten erfordern eine umfangreiche Speicherinfrastruktur. Die Verwaltung der Speicherkosten und die Sicherstellung eines effizienten Zugriffs auf die gespeicherten Dateien stellen dabei kontinuierliche Herausforderungen dar.

• Skalierbarkeit: Systeme müssen skalierbar sein, um wachsende Datenmengen ohne Leistungseinbußen zu bewältigen. Cloud-Speicherlösungen wie AWS S3 oder Google Cloud Storage bieten zwar Skalierbarkeit, bringen aber auch eigene Kosten- und Leistungsüberlegungen mit sich.

2• Verarbeitungsherausforderungen:

• Ressourcenintensität: Die Verarbeitung großer Audiodateien erfordert beträchtliche Rechenressourcen. Aufgaben wie Rauschunterdrückung, Formatkonvertierung und Merkmalsextraktion sind ressourcenintensiv und können die Systemleistung und Reaktionsfähigkeit beeinträchtigen.

• Effizienz: Die effiziente Verwaltung großer Dateien erfordert Techniken wie das Chunking, bei dem Dateien in kleinere Segmente unterteilt werden, um sie parallel zu verarbeiten. Dieser Ansatz hilft, die Ressourcennutzung besser zu steuern, macht jedoch die Wiederzusammenstellung der verarbeiteten Segmente komplexer.

Optimierungsstrategien [See21]:

• Komprimierung: Durch die Anwendung verlustfreier Komprimierungstechniken kann die Dateigröße ohne Qualitätseinbußen reduziert werden. Formate wie FLAC (Free Lossless Audio Codec) bieten Komprimierung ohne Verlust der Audioqualität.

• Stapelverarbeitung: Bei Anwendungen, die nicht in Echtzeit ablaufen, kann die Stapelverarbeitung eingesetzt werden, um die Ressourcennutzung effektiver zu verwalten. Verarbeitungsaufgaben werden in die Warteschlange gestellt und außerhalb der Spitzenzeiten ausgeführt, um die Last auszugleichen.

Extraktion in Echtzeit [See21]:

Die Extraktion von Audiodaten in Echtzeit stellt einzigartige Herausforderungen dar, die in erster Linie mit der Aufrechterhaltung einer geringen Latenz und der Gewährleistung der Datenintegrität während der kontinuierlichen Erfassung und Verarbeitung zusammenhängen.

1. Latenzprobleme:

• Erfassung: Die Erfassung von Audiodaten in Echtzeit erfordert Systeme, die Hochdurchsatz-Streams mit minimaler Verzögerung verarbeiten. Latenz bei der Erfassung kann zu verpassten Audiosegmenten oder Verzögerungen bei der Verarbeitung führen.

• Verarbeitung: Die Echtzeitverarbeitung muss optimiert werden, um Aufgaben wie Rauschunterdrückung, Spracherkennung und Merkmalsextraktion im laufenden Betrieb zu bewältigen. Dies erfordert effiziente Algorithmen und leistungsstarke Hardware, um Engpässe zu vermeiden.

2. Datenintegrität:

• Kontinuierlicher Datenstrom: Die Gewährleistung der Integrität eines kontinuierlichen Audiostroms beinhaltet den Umgang mit potenziellen Unterbrechungen und Datenverlusten. Systeme müssen robust genug sein, um Netzwerkprobleme, Hardwarefehler und andere Störungen zu bewältigen.

• Synchronisierung: In Umgebungen mit mehreren Audioquellen ist die Aufrechterhaltung der Synchronisierung von entscheidender Bedeutung. Das Zeitstempeln und Angleichen von Audioströmen aus verschiedenen Quellen gewährleistet genaue und zusammenhängende Daten.

Techniken zur Echtzeitextraktion [See21]:

• Pufferung: Die Implementierung von Pufferstrategien hilft bei der Verwaltung des Datenflusses und berücksichtigt Schwankungen bei Eingaberaten und Verarbeitungszeiten. Die Puffer müssen jedoch sorgfältig dimensioniert werden, um übermäßige Latenz zu vermeiden.

• Edge Computing: Der Einsatz von Edge-Computing-Lösungen reduziert die Latenz, indem Audiodaten näher an der Quelle verarbeitet werden. Dieser Ansatz minimiert die Verzögerung, die durch die Datenübertragung an zentralisierte Server verursacht wird.

• Optimierte Algorithmen: Die Nutzung optimierter Algorithmen und Hardwarebeschleunigung (z. B. GPUs, spezielle Audioverarbeitungschips) verbessert die Fähigkeit, Audiodaten in Echtzeit zu verarbeiten und sorgt für minimale Latenz und hohen Durchsatz.

3.3 Transformieren der Audiodaten

Die Transformation von Audiodaten ist ein entscheidender Schritt im Prozess verschiedener Audioanalyseaufgaben. Durch die Verbesserung der Audioqualität und das Extrahieren aussagekräftiger Merkmale können wir sicherstellen, dass nachfolgende Analysen wie Spracherkennung, Klassifizierung von Musikgenres und mehr genau und effektiv sind [SKH23; Sub+97].

3.3.1 Ziele der Transformation

Die Transformationsphase zielt darauf ab, Audiodaten für die Analyse vorzubereiten, indem ihre Qualität verbessert und aussagekräftige Merkmale extrahiert werden.
Die wichtigsten Ziele sind [SKH23]:

Ziel	Beschreibung
Rauschunterdrückung	Die Rauschunterdrückung zielt darauf ab, Hintergrundgeräusche zu entfernen, um die Klarheit des Audiosignals zu verbessern. Solche Geräusche können die Leistung von Analysealgorithmen erheblich beeinträchtigen, besonders bei Anwendungen wie der Spracherkennung, bei denen Klarheit entscheidend ist. Durch die Reduzierung des Rauschens wird das Audiosignal klarer und deutlicher, was eine präzisere Analyse und ein besseres Verständnis ermöglicht.
Normalisierung	Die Normalisierung gewährleistet konsistente Lautstärkepegel für verschiedene Audiodateien. Diese Konsistenz ist entscheidend, wenn mehrere Aufnahmen unter unterschiedlichen Bedingungen gemacht werden. Ohne Normalisierung können Lautstärkeschwankungen zu irreführenden Analyseergebnissen führen, da aus dem Audio extrahierte Merkmale eher von den Aufnahmebedingungen als vom Inhalt des Audios selbst beeinflusst werden können.
Formatkonvertierung	Audiodaten werden oft in verschiedenen Formaten aufgenommen und gespeichert, was eine Konvertierung in ein Standardformat für eine einheitliche Verarbeitung erforderlich macht. Standardformate wie WAV oder MP3 werden von Analysetools weitgehend unterstützt, was Kompatibilität gewährleistet und eine reibungslosere Integration und Verarbeitung über verschiedene Plattformen und Anwendungen hinweg ermöglicht.
Merkmalsextraktion	Die Merkmalsextraktion ist ein wichtiger Prozess, um aus Audiodaten aussagekräftige Erkenntnisse abzuleiten. Merkmale wie Mel Frequency Cepstral Coefficients (MFCCs), Spektrogramme und Chroma-Merkmale fassen wesentliche Eigenschaften des Audiosignals zusammen. Diese Merkmale sind für Aufgaben wie Spracherkennung, Musikinformationsabruf und Audioklassifizierung unverzichtbar und bieten eine kompakte und informative Darstellung des Audioinhalts.

Tabelle 3.7: Übersicht zu Audioverarbeitungstechniken

3.3.2 Transformationstechniken

Im Folgenden werden einige wichtige Transformationstechniken erläutert [Bur+19]:
1. Rauschunterdrückung: Rauschunterdrückungstechniken zielen darauf ab, die Klarheit des Audiosignals zu verbessern, indem unerwünschte Hintergrundgeräusche eliminiert oder reduziert werden.
Zwei gängige Methoden sind:

1• Spektrale Gating: Spektrales Gating reduziert Rauschen, indem es Frequenzen herausfiltert, die nicht zum Hauptsignal passen. Diese Technik analysiert das Frequenzspektrum des Audiosignals und dämpft jene Frequenzen, die als Rauschen erkannt werden. Spektrales Gating ist besonders effektiv in Umgebungen, in denen sich das Rauschprofil deutlich vom gewünschten Signal unterscheidet, zum Beispiel beim Entfernen von konstantem Hintergrundbrummen oder -rauschen.

2• Adaptive Filterung: Adaptive Filterung nutzt Algorithmen, die die Filterparameter dynamisch an die Eigenschaften des Audiosignals anpassen. Diese Algorithmen reagieren kontinuierlich auf die sich ändernde Natur von Signal und Rauschen und bieten so eine robuste Lösung zur Rauschunterdrückung in verschiedenen akustischen Umgebungen. Besonders nützlich ist die adaptive Filterung in Situationen, in denen die Rauscheigenschaften nicht statisch sind und sich im Laufe der Zeit verändern.

2. Normalisierung: Normalisierungstechniken passen die Lautstärke von Audiodateien an, um Konsistenz sicherzustellen.
Zwei primäre Methoden sind:

1• Spitzennormalisierung: Die Spitzennormalisierung passt den Audiopegel so an, dass der höchste Spitzenwert einen festgelegten Zielpegel erreicht. Dadurch wird sichergestellt, dass der lauteste Teil des Audiosignals auf einen vordefinierten Wert eingestellt ist, was zu einem einheitlichen Spitzenpegel bei verschiedenen Audiodateien führt. Diese Methode ist nützlich, um zu verhindern, dass eine Audiodatei einen bestimmten Lautstärkeschwellenwert überschreitet, wodurch Verzerrungen und Übersteuerungen vermieden werden.

2• Lautstärkenormalisierung: Die Lautstärkenormalisierung sorgt dafür, dass die wahrgenommene Lautstärke bei allen Audiodateien konsistent bleibt. Im Gegensatz zur Spitzennormalisierung, die sich auf den höchsten Spitzenwert konzentriert, berücksichtigt sie die insgesamt wahrgenommene Lautstärke und bietet so ein einheitlicheres Hörerlebnis. Diese Methode ist besonders in Bereichen wie dem Rundfunk wichtig, wo eine gleichbleibende Lautstärke für den Hörkomfort unerlässlich ist.

3. Formatkonvertierung : Die Formatkonvertierung ist für die Standardisierung von Audiodateien erforderlich, um die Kompatibilität mit Analysetools sicherzustellen.
Zwei beliebte Tools für diesen Zweck sind [Bur+19]:

1• FFmpeg: FFmpeg ist ein äußerst leistungsstarkes Werkzeug zur Konvertierung von Audioformaten. Es unterstützt eine Vielzahl von Audioformaten und bietet umfangreiche Optionen für die Verarbeitung und Umwandlung von Audiodateien, was es zur bevorzugten Wahl für Aufgaben in der Formatkonvertierung macht. Dank seiner Flexibilität und des umfassenden Funktionsumfangs eignet sich FFmpeg für verschiedene Anforderungen in der Audioverarbeitung, von der einfachen Formatkonvertierung bis hin zu komplexen Audio-Workflows.

2• SoX (Sound eXchange): SoX ist ein weiteres vielseitiges Audioverarbeitungstool, das verschiedene Audioformate unterstützt und umfassende Audioverarbeitungsfunktionen bietet. Aufgrund seiner Flexibilität und Robustheit wird es häufig für Formatkonvertierungen und andere Audioverarbeitungsaufgaben verwendet. SoX bietet eine Reihe von Audioeffekten und -transformationen und ist damit ein wertvolles Tool für die Audiodatentransformation.

4. Merkmalsextraktion: Bei der Merkmalsextraktion werden aus Audiodaten aussagekräftige Merkmale abgeleitet.
Drei gängige Merkmale sind [Bur+19]:

1• MFCCs: Die Mel-Frequenz-Cepstrum-Koeffizienten (MFCCs) erfassen das Leistungsspektrum von Audiosignalen und sind für die Sprach- und Musikanalyse von entscheidender Bedeutung. Sie repräsentieren das kurzzeitige Leistungsspektrum eines Tons und bieten eine kompakte sowie aussagekräftige Darstellung des Audiosignals. Aufgrund ihrer Fähigkeit, die wahrgenommenen spektralen Eigenschaften von Audio abzubilden, werden MFCCs häufig in der Sprach- und Musikanalyse eingesetzt.

2• Spektrogramme: Spektrogramme sind visuelle Darstellungen, die zeigen, wie sich das Frequenzspektrum von Audiosignalen im Laufe der Zeit verändert. Indem sie den Wandel des Frequenzinhalts eines Signals sichtbar machen, sind Spektrogramme ein wertvolles Werkzeug zur Analyse der zeitlichen Dynamik von Audiodaten. Sie werden häufig in Anwendungen wie der Sprach- und Musikanalyse eingesetzt, bei denen das Verständnis der zeitlichen Entwicklung des Signals von entscheidender Bedeutung ist.

3• Chroma-Funktionen: Chroma-Funktionen stellen die zwölf verschiedenen Tonhöhenklassen in einem Audiosignal dar und sind daher besonders nützlich für die Musikanalyse. Diese Funktionen erfassen den harmonischen und melodischen Inhalt des Audios und bieten Einblicke in die musikalische Struktur und Tonalität. Chroma-Funktionen werden bei Aufgaben wie Akkorderkennung, Tonarterkennung und Musiktranskription verwendet, bei denen das Verständnis des Tonhöheninhalts von entscheidender Bedeutung ist.

3.3.3 Herausforderungen bei der Transformation

Die Transformation von Audiodaten bringt mehrere Herausforderungen mit sich, die bewältigt werden müssen, um qualitativ hochwertige Ergebnisse zu gewährleisten. Zu den wichtigsten Herausforderungen gehören [SS20; ZSS+11]:

•**Aufrechterhaltung der Datenintegrität:** Die Aufrechterhaltung der Datenintegrität während der Transformation ist eine erhebliche Herausforderung. Für eine zuverlässige Analyse ist es entscheidend, sicherzustellen, dass die Audiodaten während Prozessen wie Rauschunterdrückung und Normalisierung genau und unverändert bleiben. Jeder Informationsverlust oder die Einführung von Artefakten während der Transformation kann die Qualität der extrahierten Merkmale und der nachfolgenden Analyseergebnisse beeinträchtigen.
•**Verwaltung der Rechenlast:** Die Verwaltung der Rechenlast ist eine weitere Herausforderung, da Audiotransformationstechniken ressourcenintensiv sein können. Effiziente Algorithmen und optimierte Verarbeitungstechniken sind unerlässlich, um große Mengen an Audiodaten ohne Leistungseinbußen zu verarbeiten. Der Kompromiss zwischen Rechenkomplexität und Transformationsqualität ist entscheidend, insbesondere bei Echtzeitanwendungen.
•**Sicherstellen der Kompatibilität mit Analysetools:** Um eine nahtlose Integration und Verarbeitung transformierter Audiodaten sicherzustellen, ist die Kompatibilität mit Analysetools von entscheidender Bedeutung. Das transformierte Audio muss in Formaten und Strukturen vorliegen, die von den verwendeten Analysetools unterstützt werden, um einen reibungslosen Arbeitsablauf und präzise Analysen zu ermöglichen. Kompatibilitätsprobleme können zu Verarbeitungsverzögerungen, Fehlern und suboptimalen Analyseergebnissen führen.
•**Herausforderungen bewältigen:** Um die Herausforderungen bei der Audiodatentransformation zu bewältigen, ist eine Kombination aus fortschrittlichen Algorithmen und effizienten Verarbeitungstechniken erforderlich.
Einige Strategien umfassen:
Fortschrittliche Algorithmen Die Implementierung fortschrittlicher Algorithmen zur Rauschunterdrückung, Normalisierung und Merkmalsextraktion kann die Qualität und Effizienz der Audiodatentransformation verbessern. Mit Techniken des maschinellen Lernens können beispielsweise adaptive Filter und Rauschunterdrückungsalgorithmen entwickelt werden, die unter verschiedenen Bedingungen gut funktionieren.
•**Effiziente Verarbeitungstechniken:** Die Optimierung der Verarbeitungstechniken zur Reduzierung der Rechenlast ist für die Verarbeitung großer Audiodatenmengen unerlässlich. Parallelverarbeitung, Hardwarebeschleunigung und effiziente Codierungspraktiken können die Leistung von Audiotransformationspipelines erheblich verbessern.

•**Standardisierung und Kompatibilität:** Die Einführung von Standardformaten und die Sicherstellung der Kompatibilität mit weit verbreiteten Analysetools können den Transformationsprozess deutlich vereinfachen. Die Nutzung etablierter Bibliotheken und Frameworks für die Audioverarbeitung trägt ebenfalls dazu bei, dass transformierte Audiodaten mit nachfolgenden Analyseanwendungen kompatibel sind. Die Transformation von Audiodaten ist ein vielschichtiger Prozess, der Schritte wie Rauschunterdrückung, Normalisierung, Formatkonvertierung und Merkmalsextraktion umfasst. Jeder dieser Schritte ist entscheidend, um Audiodaten für eine genaue und effektive Analyse vorzubereiten. Um die Herausforderungen bei der Audiodatentransformation zu meistern, sind fortschrittliche Algorithmen, effiziente Verarbeitungstechniken und eine sorgfältige Beachtung der Datenintegrität und -kompatibilität erforderlich. Indem wir diese Herausforderungen angehen, können wir eine qualitativ hochwertige Transformation der Audiodaten gewährleisten und somit den Weg für präzise und aufschlussreiche Audioanalysen ebnen.

3.4 Laden von Audiodaten

Das Laden transformierter Audiodaten in Zielsysteme ist die letzte Phase der Datenverarbeitungspipeline. Diese Phase stellt sicher, dass die Daten für die Analyse und Weiterverarbeitung zugänglich sind. Die Wahl der Zielsysteme und Ladetechniken spielt eine entscheidende Rolle für die Effizienz und Effektivität von Datenanalyse-Workflows [Cla+23].

3.4.1 Zielsysteme

Die Auswahl eines geeigneten Zielsystems zum Speichern von Audiodaten hängt von den spezifischen Anforderungen der Anwendung ab, einschließlich Datenvolumen, Zugriffsmustern und Integration mit anderen Tools. Zu den gängigen Zielsystemen gehören [Cla+23; MZE06]:

Ziel	Beschreibung
Datenbanken	Datenbanken werden zum strukturierten Speichern von Audiometadaten und transformierten Daten verwendet. Zwei Haupttypen von Datenbanken sind: **SQL-Datenbanken:** SQL-Datenbanken wie MySQL, PostgreSQL und Microsoft SQL Server bieten robuste Unterstützung für strukturierte Datenspeicherung. Sie eignen sich gut für Anwendungen, die komplexe Abfragen und Transaktionen erfordern. **NoSQL-Datenbanken:** NoSQL-Datenbanken wie MongoDB, Cassandra und Redis bieten Flexibilität bei der Datenspeicherung ohne vordefiniertes Schema. Sie sind ideal für Anwendungen, die Skalierbarkeit und leistungsstarke Lese- und Schreibvorgänge erfordern.
Data Warehouses	Data Warehouses wie Amazon Redshift, Google BigQuery und Snowflake dienen als zentrale Repositorien für die Speicherung und Analyse großer Datenmengen. Sie sind für analytische Abfragen optimiert und können riesige Datenmengen aus verschiedenen Quellen verarbeiten. Data Warehouses unterstützen komplexe Abfragen und Datenaggregation und sind daher ideal für eine umfassende Audiodatenanalyse.
Cloud-Speicher	Cloud-Speicherlösungen wie AWS S3, Google Cloud Storage und Azure Blob Storage bieten skalierbare und kosteneffiziente Speicheroptionen. Aufgrund ihrer Flexibilität und Skalierbarkeit sind sie besonders nützlich für die Speicherung großer Mengen an Audiodaten. Zudem lassen sich Cloud-Speicherlösungen gut in verschiedene Datenverarbeitungs und Analysetools integrieren, was nahtlose Arbeitsabläufe ermöglicht.

Tabelle 3.8: Übersicht zu Datenbanken, Data Warehouses und Cloud-Speicher

3.4.2 Techniken für das Laden

Das effiziente Laden von Audiodaten in Zielsysteme ist entscheidend, um einen zeitnahen Zugriff und eine zeitnahe Analyse zu gewährleisten. Verschiedene Ladetechniken erfüllen verschiedene Anforderungen, wie z. B. Datenvolumen, Latenz und Verarbeitungsgeschwindigkeit. Zu den gängigen Ladetechniken gehören [Mah+21; Ver04]:

Stapelverarbeitung: Bei der Stapelverarbeitung werden Daten in großen Mengen in geplanten Intervallen geladen. Diese Technik eignet sich für Szenarien, in denen der Echtzeitzugriff auf Daten nicht kritisch ist. Die Stapelverarbeitung kann große Datenmengen effizient verarbeiten, es gibt jedoch eine Verzögerung zwischen der Datengenerierung und der Verfügbarkeit für die Analyse. Sie wird häufig für regelmäßige Aktualisierungen oder die Aggregation von Daten in regelmäßigen Abständen verwendet.

Vorteile der Stapelverarbeitung	**Nachteile der Stapelverarbeitung**
Effizienz: Bewältigt große Datenmengen effizient durch die Verarbeitung von Daten in großen Mengen.	**Latenz:** Es gibt eine Verzögerung zwischen der Datengenerierung und der Verfügbarkeit für die Analyse, was für Echtzeitanwendungen möglicherweise nicht geeignet ist.
Ressourcenverwaltung: Ermöglicht eine bessere Kontrolle über die Ressourcennutzung, da die Verarbeitung außerhalb der Spitzenzeiten geplant werden kann.	**Speicheraufwand:** Erfordert Zwischenspeicherung für Batchdaten vor der Verarbeitung.
Reduzierte Komplexität: Vereinfacht die Fehlerbehandlung und Wiederherstellungsprozesse aufgrund ihrer unkomplizierten Natur.	

Tabelle 3.9: Vorteile und Nachteile der Stapelverarbeitung

Stream-Verarbeitung: Bei der Stream-Verarbeitung werden Daten fortlaufend in Echtzeit geladen. Diese Technik ist unverzichtbar für Anwendungen, die sofortigen Zugriff auf Daten benötigen, sobald sie entstehen. Durch die Unterstützung von Echtzeitanalysen und Überwachung eignet sich die Stream-Verarbeitung besonders für zeitkritische Anwendungen wie Live-Audio-Übertragungen, Echtzeit-Spracherkennung und Überwachungssysteme.

Vorteile der Stream-Verarbeitung	**Nachteile der Stream-Verarbeitung**
Geringe Latenz: Bietet sofortigen Zugriff auf Daten und ermöglicht Echtzeitanalysen und -entscheidungen.	**Komplexität:** Erfordert eine ausgefeilte Infrastruktur und Algorithmen zur Verwaltung des kontinuierlichen Datenflusses und der Datenverarbeitung.
Skalierbarkeit: Kann kontinuierliche Datenströme aus mehreren Quellen gleichzeitig verarbeiten.	**Ressourcenintensiv:** Erfordert eine konstante Ressourcenzuweisung, um eingehende Datenströme effektiv zu verarbeiten.
Aktualität: Stellt sicher, dass immer die aktuellsten Daten für die Analyse verfügbar sind.	

Tabelle 3.10: Vorteile und Nachteile der Stream-Verarbeitung

API-Integration: BDie API-Integration verwendet Anwendungsprogrammierschnittstellen (APIs), um Daten in Cloud-Dienste oder Datenbanken zu laden. APIs bieten einen flexiblen und skalierbaren Ansatz für das Datenladen und ermöglichen eine nahtlose Integration mit verschiedenen Systemen und Plattformen. Diese Methode ist besonders hilfreich für die Einbindung von Diensten Dritter, die Automatisierung von Datenworkflows und die Sicherstellung der Datenkonsistenz.

Vorteile der API-Integration	Nachteile der API-Integration
Flexibilität: Ermöglicht maßgeschneiderte und skalierbare Datenladelösungen, die auf spezifische Anforderungen zugeschnitten sind.	**Abhängigkeit:** Hängt von der Verfügbarkeit und Leistung externer APIs ab, was zu potenziellen Fehlerquellen führen kann.
Automatisierung: Ermöglicht die Automatisierung von Datenladeprozessen und reduziert so manuelle Eingriffe und Fehler.	**Komplexität:** Erfordert die Entwicklung und Wartung von API-Integrationscode, was komplex und zeitaufwändig sein kann.
Interoperabilität: Erleichtert die Integration mit verschiedenen Tools, Plattformen und Diensten und gewährleistet ein zusammenhängendes Datenökosystem.	

Tabelle 3.11: Vorteile und Nachteile der API-Integration

3.4.3 Herausforderungen beim Laden

Das Laden von Audiodaten in Zielsysteme bringt mehrere Herausforderungen mit sich, die bewältigt werden müssen, um Datenkonsistenz, effiziente Verwaltung großer Datenmengen und optimierte Datenabrufgeschwindigkeiten sicherzustellen. Zu den wichtigsten Herausforderungen gehören [XHY17; Yan+17; Ver04]:

1.Sicherstellung der Datenkonsistenz:
Datenkonsistenz ist für genaue Analysen und Entscheidungsfindungen entscheidend. Inkonsistente Daten können zu falschen Schlussfolgerungen und unzuverlässigen Ergebnissen führen. Die Sicherstellung der Datenkonsistenz umfasst:

- Datenvalidierung: Implementierung von Validierungsprüfungen, um die Datenintegrität vor dem Laden sicherzustellen.
- Fehlerbehandlung: Entwicklung robuster Fehlerbehandlungsmechanismen zum Erkennen und Korrigieren von Inkonsistenzen.
- Transaktionsverwaltung: Verwendung von Transaktionen, um Atomizität und Konsistenz bei Datenbankvorgängen sicherzustellen.
- Verwaltung großer Datenmengen Audiodaten, insbesondere in High-Fidelity-Formaten, können groß sein und erhebliche Speicher- und Verarbeitungsressourcen erfordern.

Die Verwaltung großer Datenmengen umfasst:
• Skalierbarkeit: Auswahl skalierbarer Speicherlösungen, die mit zunehmenden Datenmengen wachsen können.
• Komprimierung: Implementierung von Datenkomprimierungstechniken zur Reduzierung des Speicherbedarfs.
• Effiziente Speicherung: Organisation von Daten auf eine Weise, die die Speicher- und Abrufleistung optimiert.
Optimierung der Datenabrufgeschwindigkeit, Schneller Datenabruf ist für eine zeitnahe Analyse und Verarbeitung unerlässlich. Die Optimierung der Datenabrufgeschwindigkeit umfasst:
• Indizierung: Erstellen von Indizes für häufig abgefragte Felder, um den Datenabruf zu beschleunigen.
• Zwischenspeichern: Verwenden von Zwischenspeichermechanismen zum Speichern häufig abgerufener Daten für einen schnellen Abruf.
• Abfrageoptimierung: Optimieren von Abfragen, um die Latenz zu minimieren und die Leistung zu verbessern.

2.Strategien für effizientes Laden:
Um die Herausforderungen beim Laden von Audiodaten zu bewältigen, können mehrere Strategien eingesetzt werden:

Strategie	Beschreibung
Inkrementelles Laden	Beim inkrementellen Laden werden nur die neuen oder aktualisierten Daten geladen, anstatt den gesamten Datensatz neu zu laden. Dieser Ansatz reduziert das übertragene Datenvolumen und beschleunigt den Ladevorgang. Inkrementelles Laden ist besonders nützlich für große Datensätze, bei denen sich nur ein kleiner Teil der Daten häufig ändert.
Parallele Verarbeitung	Bei der parallelen Verarbeitung werden mehrere Prozessoren oder Kerne genutzt, um Datenladeaufgaben gleichzeitig auszuführen. Diese Technik kann den Ladevorgang erheblich beschleunigen, insbesondere bei großen Datensätzen. Parallele Verarbeitung erfordert eine sorgfältige Verwaltung von Abhängigkeiten und Koordination zwischen parallelen Aufgaben, um Datenkonsistenz sicherzustellen.
Datenpartitionierung	Bei der Datenpartitionierung werden die Daten in kleinere, besser handhabbare Blöcke aufgeteilt, die unabhängig voneinander verarbeitet werden können. Die Partitionierung kann die Ladeleistung verbessern und die Verwaltung und Skalierung der Datenspeicherinfrastruktur erleichtern. Partitionierungsstrategien umfassen je nach Anwendungsfall zeitbasierte, bereichsbasierte und hashbasierte Partitionierung.

Tabelle 3.12: Übersicht zu Techniken für effizientes Datenladen und -verarbeitung

3.Robuste Infrastruktur: Die Implementierung einer robusten Infrastruktur ist für effizientes Laden von Daten unerlässlich. Zu den wichtigsten Komponenten einer robusten Infrastruktur gehören:

- Hochleistungsspeicher: Verwendung von Hochleistungsspeicherlösungen mit schnellen Lese- und Schreibgeschwindigkeiten, wie SSDs oder NVMe-Laufwerke.
- Netzwerkoptimierung: Optimierung der Netzwerkkonfigurationen, um eine schnelle und zuverlässige Datenübertragung zwischen Systemen sicherzustellen.
- Überwachung und Warnmeldungen: Implementierung von Überwachungs- und Warnsystemen, um Probleme in Echtzeit zu erkennen und zu beheben und so kontinuierliche Verfügbarkeit und Leistung sicherzustellen.

Das Laden von Audiodaten in Zielsysteme ist ein entscheidender Schritt in der Datenverarbeitungspipeline. Für eine effiziente und effektive Datenanalyse ist es wichtig, die richtigen Zielsysteme und Ladetechniken auszuwählen und dabei Herausforderungen wie Datenkonsistenz, große Datenmengen und Abrufgeschwindigkeiten zu berücksichtigen. Durch den Einsatz effizienter Ladestrategien und einer robusten Infrastruktur können Unternehmen sicherstellen, dass ihre Audiodaten zugänglich und bereit für die Analyse sind, was wertvolle Erkenntnisse und fundierte Entscheidungen ermöglicht [Raj+20; XHY17].

3.5 Informationsabruf für Audiodateien

Information Retrieval (IR) ist der Prozess, relevante Informationen aus einer Sammlung von Ressourcen zu erhalten. Bei der Anwendung auf Audiodateien umfasst IR das Extrahieren, Indizieren und Durchsuchen von Audiodaten, um bestimmte Informationen abzurufen oder Muster zu erkennen. Ziel ist es, Audioinhalte für verschiedene Anwendungen zugänglich und nützlich zu machen, von Multimedia-Suchmaschinen bis hin zu sprachgesteuerten Systemen [KPM21; CDL21].

Schlüsselkonzepte beim Audio Information Retrieval [CDL21]:

1. Merkmalsextraktion: Der erste Schritt beim Audio-IR ist das Extrahieren relevanter Merkmale aus den Audiodateien. Diese Merkmale können in mehrere Kategorien unterteilt werden:

- Zeitliche Merkmale: Eigenschaften im Zusammenhang mit der Zeit, wie Nulldurchgangsrate und Tempo.
- Spektrale Merkmale: Frequenzbasierte Attribute, wie spektraler Schwerpunkt, Bandbreite und spektraler Kontrast.
- Harmonische Merkmale: Im Zusammenhang mit dem harmonischen Inhalt des Audios, wie Tonhöhe und Harmonie.

2. Indizierung: Nach der Merkmalsextraktion müssen die Audiodaten indiziert werden, um eine effiziente Suche und Abfrage zu ermöglichen. Bei der Indizierung werden die extrahierten Merkmale in ein strukturiertes Format organisiert, das einen schnellen Zugriff und Vergleich ermöglicht.

- Audio-Fingerprinting: Erstellen eindeutiger Kennungen (Fingerabdrücke) für Audiodateien basierend auf ihren Merkmalen. Dies hilft dabei, Audiosegmente schnell zu identifizieren und abzugleichen.
- Metadatenindizierung: Verwenden von Metadaten wie Künstler, Album, Genre und Dauer, um die Suchfunktionen zu verbessern.

3. Suchen und Abrufen: Der Kern der Audio-IR ist die Möglichkeit, relevante Audiodateien basierend auf Benutzerabfragen zu suchen und abzurufen.

- Abfrage per Beispiel (QBE): Ermöglicht Benutzern, einen Beispiel-Audioclip bereitzustellen, um ähnliche Audiodateien zu finden.
- Abfrage per Summen (QBH): Ermöglicht Benutzern, eine Melodie zu summen, um passende Lieder zu finden.
- Textbasierte Suche: Verwenden von Transkriptionen von Sprache oder kommentierten Metadaten, um textbasierte Suchen durchzuführen.

4. Ähnlichkeitsmaße: Zum Vergleichen und Abrufen von Audiodateien werden Ähnlichkeitsmaße verwendet, um zu bewerten, wie genau die Merkmale einer Audiodatei mit denen einer anderen übereinstimmen.

- Euklidische Distanz: Ein gängiges Maß zum Vergleichen von Merkmalsvektoren.
- Dynamic Time Warping (DTW): Wird zum Vergleichen von Sequenzen verwendet, die in Zeit oder Geschwindigkeit variieren können, nützlich zum Abgleichen von Mustern in Sprache und Musik.
- Kosinus-Ähnlichkeit: Misst den Kosinus des Winkels zwischen zwei Vektoren, nützlich zum Bestimmen der Ähnlichkeit in hochdimensionalen Räumen.

Tools und Technologien für Audio-IR [CDL21; TC00]:

Kategorie	Beschreibung
Audioverarbeitungs-bibliotheken	Bibliotheken wie Librosa, PyDub und Essentia bieten Funktionen zur Merkmalsextraktion, Transformation und Analyse von Audiodateien.
Spracherkennungstools	Systeme wie Google Speech-to-Text, IBM Watson und Kaldi zum Konvertieren von Sprache in Audiodateien in Text, wodurch textbasierte Suchen ermöglicht werden.
Machine-Learning-Frameworks	TensorFlow, PyTorch und scikit-learn zum Entwickeln und Bereitstellen von Modellen für Audioklassifizierung, Clustering und Ähnlichkeitserkennung.
Indizierung und Suchmaschinen	Tools wie Apache Lucene, Elasticsearch und Solr können zum Indizieren und Suchen sowohl der Audiofunktionen als auch der zugehörigen Metadaten verwendet werden.
Tools zur Musikinformationsabfrage (MIR)	Spezielle Tools und Bibliotheken wie MARSYAS und MIRtoolbox, die für die Abfrage und Analyse von Musikinhalten entwickelt wurden.

Tabelle 3.13: Übersicht zu Audioverarbeitungswerkzeugen und -technologien

Anwendungen der Audioinformationsabfrage [WKH23; Lou+22]:

Anwendung	Beschreibung
Musikentdeckung	Systeme wie Shazam verwenden Audio-Fingerabdrücke und IR (Information Retrieval), um Songs anhand kurzer Audioclips zu identifizieren.
Sprachaktivierte Assistenten	Verbesserung der Genauigkeit und Funktionalität von Assistenten wie Siri, Alexa und Google Assistant durch effektive Spracherkennung und IR.
Multimedia-Suchmaschinen	Plattformen wie YouTube und Spotify verwenden Audio-IR, um Benutzern das Suchen und Entdecken von Inhalten anhand von Audiofunktionen zu ermöglichen.
Audioarchivierung und -aufbewahrung	Bibliotheken und Archive verwenden IR, um historische Audioaufzeichnungen zu katalogisieren und abzurufen.
Forensische Audioanalyse	Strafverfolgungsbehörden verwenden IR, um Audiobeweise zu identifizieren, zu vergleichen und zu analysieren.

Tabelle 3.14: Anwendungen von Audio-Information Retrieval (IR)

Kapitel 4

Konzept

In diesem Kapitel wird das Konzept für die Entwicklung eines ETL-Prozesses zur Verarbeitung von Audiodaten vorgestellt. Es werden die grundlegenden Überlegungen, Anforderungen und die geplante Architektur beschrieben, die als Basis für die Implementierung dienen. Ziel ist es, ein schlüssiges und durchdachtes Konzept zu präsentieren, das die Grundlage für die praktische Umsetzung bildet.

4.1 Überblick über das Projekt

Dieses Projekt zielt darauf ab, eine webbasierte Anwendung zu entwickeln, die es ermöglicht, Audiodateien hochzuladen, Metadaten zu extrahieren, diese mit einer Datenbank zu vergleichen und mögliche Duplikate anzuzeigen. Es kombiniert verschiedene Programmierbibliotheken und -techniken, um die Kernfunktionen zu realisieren, darunter Flask als Webframework, Librosa und SoundFile für die Audioverarbeitung, sowie PostgreSQL für die Datenbankinteraktion.

4.2 Datenfluss

Das Projekt verarbeitet Audiodatei-Uploads, extrahiert Metadaten, interagiert mit einer Datenbank und gibt die Ergebnisse an den Benutzer zurück. Unten in der **Abbildung** 4.1 befindet sich eine detaillierte Aufschlüsselung des Datenflusses innerhalb der Anwendung, die veranschaulicht, wie Daten durch verschiedene Komponenten von der Benutzerinteraktion zur Backend-Verarbeitung und zurück zum Benutzer gelangen.

Der Prozess ist in mehrere Schritte unterteilt:

1. Benutzerinteraktion und Dateiupload:
Der Datenfluss beginnt, wenn ein Benutzer mit der Webanwendung interagiert:
• Benutzer lädt eine Audiodatei hoch: Der Benutzer navigiert zur Homepage der Anwendung, die mit index.html gerendert wird. Diese Seite enthält wahrscheinlich ein Formular mit einem Dateieingabeelement, mit dem Benutzer eine Audiodatei (z. B. MP3, WAV) hochladen können.
• Formularübermittlung: Wenn der Benutzer eine Audiodatei auswählt und das Formular übermittelt, wird eine POST-Anfrage an die Route /upload gesendet. Diese Anfrage enthält die Audiodateidaten in ihrem Text.
2. Empfangen und Verarbeiten des Uploads:
Nachdem das Formular übermittelt wurde, verarbeitet die Flask-Anwendung die eingehende Anfrage:
• Routenverarbeitung: Die Route /upload ist für die Verarbeitung von POST-Anfragen definiert. In der Funktion upload-file() überprüft die Anwendung, ob tatsächlich eine Datei hochgeladen wurde, indem sie das Objekt request.files überprüft.
• Dateivalidierung: Die Anwendung führt grundlegende Validierungsprüfungen durch: Sie überprüft, ob in der Anfrage eine Datei vorhanden ist. Es wird geprüft, ob die Datei einen gültigen Dateinamen hat (keine leere Zeichenfolge). Es kann auch eine zusätzliche Validierung für unterstützte Dateitypen durchgeführt werden.
3. Audioverarbeitung und Metadatenextraktion:
Nachdem die Datei gespeichert wurde, verarbeitet die Anwendung die Audiodatei, um relevante Informationen zu extrahieren.
• Laden der Audiodatei: Die Anwendung verwendet die librosa-Bibliothek, um die Audiodatei zu laden. Dieser Schritt extrahiert die Wellenform und die Abtastrate des Audios. Außerdem berechnet sie die Dauer des Audios anhand der extrahierten Wellenform.
• Lesen von Audiodaten: Die Soundfile-Bibliothek liest die Audiodaten, um die Bittiefe (z. B. 16 Bit, 24 Bit) zu bestimmen, indem sie den Datentyp der Audiobeispiele untersucht.
• Extrahieren von Metadaten: Die mutagen-Bibliothek wird verwendet, um Metadaten aus der Audiodatei zu extrahieren. Je nach Dateityp (.mp3, .wav) werden unterschiedliche Klassen (MP3, WAVE und EasyID3) zum Lesen von Metadaten wie Titel, Künstler, Album usw. verwendet.Dies stellt den ID3-Standard der Audiostandards dar.
4. Datenbankinteraktion:
Mit den extrahierten Metadaten und Audioattributen interagiert die Anwendung mit der Datenbank, um Informationen zu speichern oder abzurufen:
• Erstellen von SQL-Anfragen: Basierend auf den extrahierten Metadaten (wie dem Titel) erstellt die Anwendung eine SQL-Anfrage, um zu prüfen, ob bereits eine Audiodatei mit demselben Titel in der Datenbank vorhanden ist. Dies geschieht mithilfe der Zeichenfolge sql-query.
• Senden von SQL-Anfragen über HTTP: Die Anwendung verwendet die Bibliothek requests, um die SQL-Anfrage über eine POST-Anfrage an einen Remote-Datenbankserver zu senden.

Der Server führt die Anfrage auf einer PostgreSQL-Datenbank aus.
• Parsen der Anfrageergebnisse: Die Antwort vom Server enthält HTML, das die Ergebnisse der SQL-Anfrage enthält. Die Bibliothek BeautifulSoup parst das HTML, um die Ergebnisse in einem strukturierten Format zu extrahieren. Die Funktion parse-query-result() findet Tabellenelemente und liest Zeilen und Spalten, um Daten abzurufen.
• Auf vorhandene Datensätze prüfen: Die Anwendung prüft, ob die Abfrageergebnisse Datensätze enthalten:
Wenn ein Datensatz gefunden wird, bedeutet dies, dass bereits eine Datei mit demselben Titel vorhanden ist, und die Daten werden als Duplikat betrachtet.
Wenn kein Datensatz gefunden wird, wird die Datei als eindeutig betrachtet. Die Anwendung erstellt und sendet eine INSERT-Anfrage, um die neue Audiodatei zur Datenbank hinzuzufügen. Die Funktion `loaddata()` übernimmt diesen Vorgang und ruft die ID des neu eingefügten Datensatzes ab.

5. Daten für die Antwort vorbereiten:

Nach der Verarbeitung der Audiodatei und der Interaktion mit der Datenbank bereitet die Anwendung Daten vor, die dem Benutzer angezeigt werden sollen:
• Datenvorbereitung: Die Anwendung sammelt Informationen wie Abtastrate, Bittiefe, Dateigröße und Dauer der Audiodatei. Sie sammelt auch Metadaten wie Titel und Jahr.
• Umgang mit Duplikaten und neuen Einträgen: Die Anwendung unterscheidet zwischen doppelten und neuen Einträgen:
Bei Duplikaten bereitet sie Daten vor, um vorhandene Datensätze aus der Datenbank anzuzeigen.
Bei neuen Einträgen bereitet sie Daten vor, um die Details des neu hinzugefügten Datensatzes anzuzeigen.

6. Rendern der Antwort:

Rendern von HTML-Vorlagen: Die Anwendung verwendet die Funktion render-template(), um die Vorlage result.html zu rendern. Sie übergibt die gesammelten Daten und Metadaten als Kontextvariablen an die Vorlage.
• Anzeigen der Ergebnisse für den Benutzer: Die gerenderte HTML-Seite zeigt dem Benutzer die Attribute und Metadaten der Audiodatei an. Sie kann auch Nachrichten enthalten, die angeben, ob die Datei als Duplikat erkannt oder als neuer Eintrag hinzugefügt wurde.

7. Bereinigen:

Entfernen von Dateien: Nach der Verarbeitung entfernt die Anwendung die temporäre Audiodatei mithilfe von os.remove(file-path) vom Server. Dies hilft bei der Speicherverwaltung und stellt sicher, dass der Server keine unnötigen Dateien ansammelt.

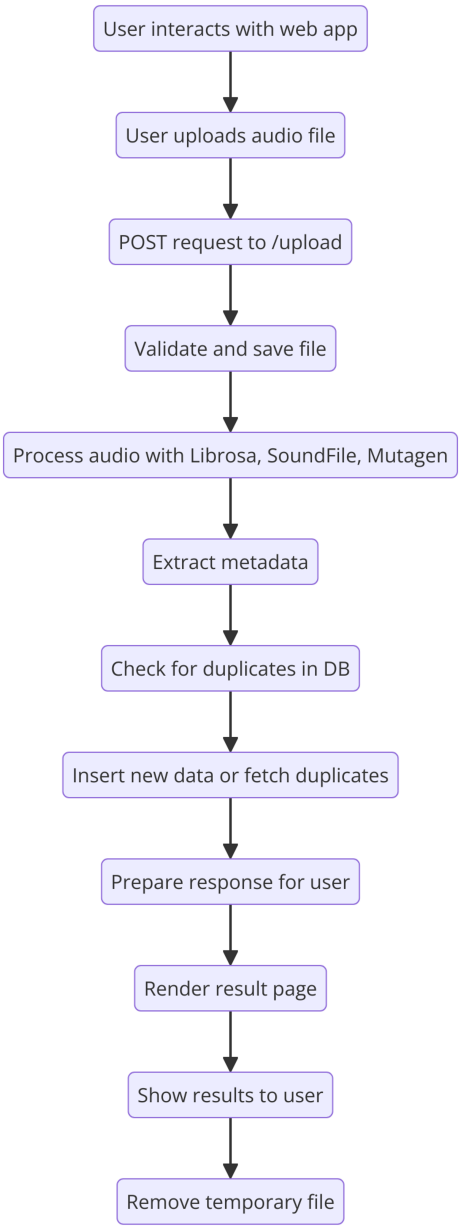

Abbildung 4.1: Datenfluss zur Audiodatei-Verarbeitung und Duplikatsprüfung

Beschreibung der Abbildung 4.1:
Diese Abbildung veranschaulicht den Datenfluss innerhalb einer Flask-basierten Webanwendung, die Audiodatei-Uploads verarbeitet. Der Ablauf beginnt mit der Interaktion des Benutzers und dem Upload einer Audiodatei, gefolgt von der Validierung und Speicherung der Datei. Die Audiodatei wird mithilfe von Librosa, SoundFile und Mutagen verarbeitet, um Audioattribute und Metadaten zu extrahieren. Anschließend wird die Datei in einer Datenbank auf Duplikate überprüft.
Falls keine Duplikate gefunden werden, wird die Datei als neuer Eintrag in die Datenbank eingefügt. Die Anwendung bereitet die Ergebnisse auf, rendert die Resultatseite und zeigt die Informationen dem Benutzer an. Abschließend wird die temporäre Datei vom Server gelöscht, um Speicherplatz freizugeben.

4.3 Projektstruktur

Die Projektstruktur ist modular und skalierbar und nutzt die Kernfunktionen von Flask sowie andere Python-Bibliotheken für bestimmte Funktionen. Nachfolgend befindet sich eine detaillierte Aufschlüsselung der Architektur, in der hervorgehoben wird, wie verschiedene Komponenten interagieren und zur Gesamtfunktionalität der Anwendung beitragen.

Architektsstruktur:

Das Projekt folgt einer typischen Webanwendungsarchitektur, die aus mehreren Schichten besteht:
• Präsentationsschicht: Dazu gehören die HTML-Vorlagen und statischen Dateien (wie CSS und JavaScript), die die Benutzeroberfläche der Anwendung definieren.
• Anwendungslogikschicht: Diese Schicht wird von Flask verwaltet und übernimmt das Routing, die Anforderungsverarbeitung und die Antwortgenerierung. Sie enthält auch die Logik für die Verarbeitung von Audiodateien, die Interaktion mit der Datenbank und das Rendern von Vorlagen.
• Datenzugriffsschicht: Dazu gehören Funktionen, die über SQL-Abfragen mit der Datenbank interagieren. Diese Funktionen übernehmen das Abrufen, Einfügen und Aktualisieren von Daten.

Die **Abbildung** 4.2 veranschaulicht die oben genannten drei Schichten und deren nächsten drei Schritte.

1.Präsentationsebene:
• HTML-Vorlagen (Verzeichnis /templates): Das Verzeichnis templates enthält HTML-Dateien, die die Struktur und das Layout von Webseiten definieren. Flask verwendet die Jinja2-Vorlagen-Engine, um Daten dynamisch in diese Vorlagen einzufügen.
• In diesem Projekt:
• index.html: Dient wahrscheinlich als Startseite oder Upload-Formular, wo Benutzer Audiodateien einreichen können.
• result.html: Zeigt die Ergebnisse nach der Dateiverarbeitung an und zeigt extrahierte Metadaten, Dateiattribute und Datenbanksuchergebnisse.
• Statische Dateien (Verzeichnis /static): Enthält CSS-, JavaScript- und Bilddateien.Diese Dateien helfen beim Stylen der Webseiten und bieten clientseitige Interaktivität. Beispielsweise können CSS-Dateien das Erscheinungsbild des Upload-Formulars definieren, während JavaScript zur Validierung oder Verbesserung der Benutzerinteraktion verwendet werden kann.

2.Anwendungslogikebene:
• Flask-Anwendungssetup (app.py): Dies ist der Kern der Anwendung, in dem die Flask-App initialisiert wird. Es definiert Routen, die verschiedenen URLs und ihren zugehörigen Handlern entsprechen. Die Handler verarbeiten eingehende HTTP-Anfragen,

führen Geschäftslogik aus und geben Antworten zurück.
Die Stammroute, die die Vorlage index.html rendert. Sie dient als Einstiegspunkt für Benutzer.
• ('/upload', methods=['POST']): Verarbeitet Datei-Uploads. Sie verarbeitet die hochgeladene Audiodatei, extrahiert Metadaten, interagiert mit der Datenbank und rendert dann die Vorlage result.html mit den relevanten Daten.
• Dateihandhabung und Metadatenextraktion: Das Projekt verwendet mehrere Python-Bibliotheken zur Handhabung von Audiodateien.

3.Datenzugriffsebene: Datenbankinteraktion: Das Projekt verwendet PostgreSQL als Datenbanksystem und interagiert mit diesem über SQL-Anfragen, die über HTTP-POST-Anfragen gesendet werden.
• SQL-Abfragen: SQL-Anfragen werden dynamisch erstellt, um neue Audiodateien in die Datenbank einzufügen oder anhand von Metadaten (wie dem Titel) nach vorhandenen Dateien zu suchen. Die Funktion loaddata() ist für die Erstellung und Ausführung einer INSERT-Abfrage verantwortlich. Der Upload-Handler der Anwendung erstellt eine SELECT-Abfrage, um nach doppelten Einträgen zu suchen.

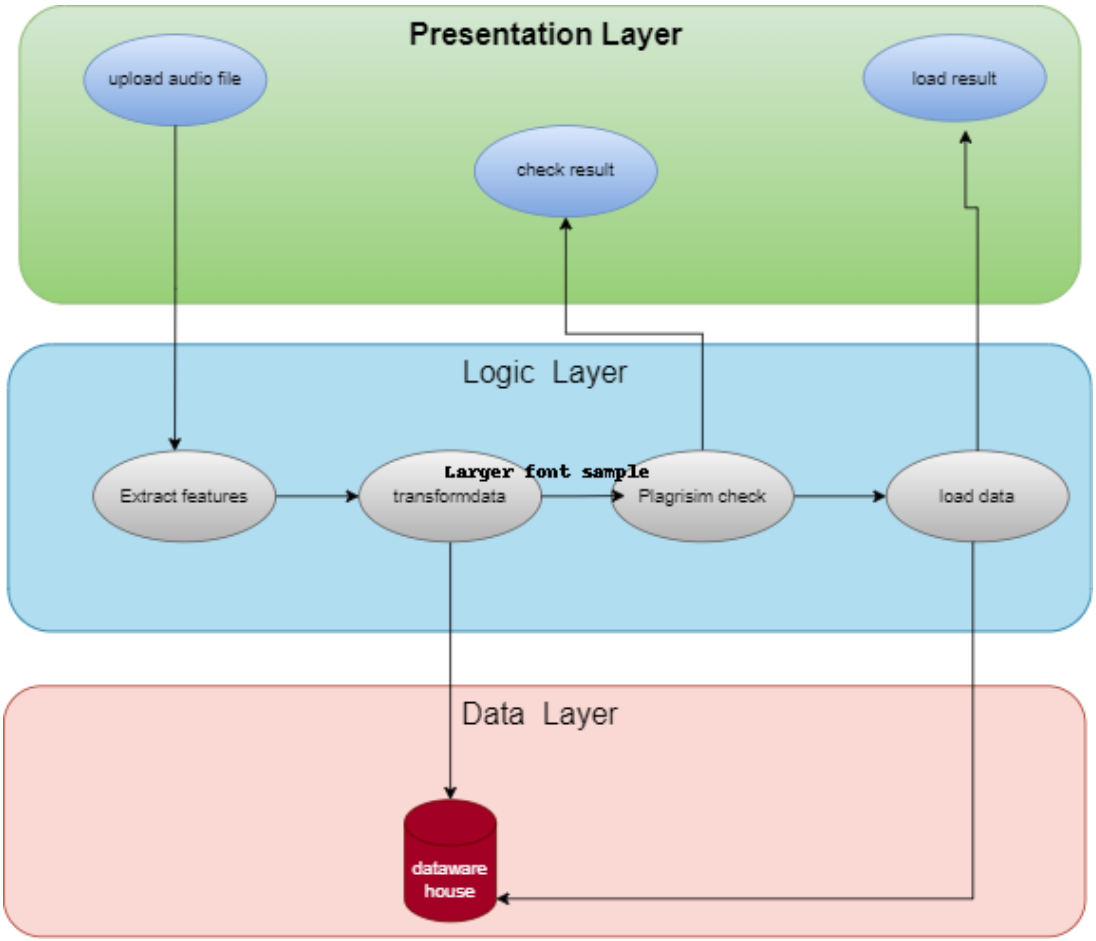

Abbildung 4.2: Architektur der Audioverarbeitungsanwendung

Beschreibung der Abbildung 4.2:
Dieses Diagramm stellt die Schichtenarchitektur der Audioverarbeitungsanwendung dar. Die Präsentationsschicht erlaubt Benutzern das Hochladen von Audiodateien und das Abrufen der Ergebnisse. In der Logikschicht werden wichtige Verarbeitungsschritte wie das Extrahieren von Audioeigenschaften, das Transformieren von Daten und die Plagiatsprüfung durchgeführt. Die Datenzugriffsschicht interagiert mit dem Data-Warehouse, um die verarbeiteten Daten zu speichern und abzurufen.

4.4 Projektanwendungsfälle

Die Anwendung bietet mehrere wichtige Funktionen, in der untenstehenden **Abbildung** 4.3 dargestellt, die verschiedene praktische Anwendungen unterstützen. Die folgenden Anwendungsfälle demonstrieren die Hauptfunktionen des Systems:

1. Hochladen und Analysieren von Audiodateien:
Ein Benutzer möchte eine Audiodatei analysieren, um Metadaten zu extrahieren und ihre technischen Eigenschaften zu verstehen.
Schritte: Der Benutzer navigiert zur Homepage der Webanwendung. Er wählt eine Audiodatei (z. B. MP3 oder WAV) von seinem Gerät aus und lädt sie mithilfe des bereitgestellten Formulars hoch. Die Anwendung verarbeitet die Datei, um Metadaten (wie Titel, Künstler, Album) und technische Eigenschaften (wie Bittiefe, Abtastrate, Dateigröße und Dauer) zu extrahieren. Die Ergebnisse werden dem Benutzer auf einer neuen Seite angezeigt, auf der sowohl die Metadaten als auch die technischen Informationen angezeigt werden.
Ergebnis: Der Benutzer erhält detaillierte Informationen über die Audiodatei, einschließlich ihrer technischen Eigenschaften und eingebetteten Metadaten.

2. Erkennen doppelter Audiodateien:
Ein Benutzer möchte überprüfen, ob eine Audiodatei, die er hochlädt, bereits in der Datenbank vorhanden ist, um Duplikate zu vermeiden.
Schritte: Der Benutzer lädt eine Audiodatei über die Schnittstelle der Webanwendung hoch. Die Anwendung extrahiert die Metadaten (insbesondere den Titel) und fragt die Datenbank ab, um festzustellen, ob bereits eine Datei mit demselben Titel vorhanden ist. Wenn ein Duplikat gefunden wird, benachrichtigt die Anwendung den Benutzer und zeigt die Details der vorhandenen Datei an. Wenn kein Duplikat gefunden wird, speichert die Anwendung die Informationen der neuen Datei in der Datenbank.
Ergebnis: Der Benutzer wird informiert, ob es sich bei der hochgeladenen Datei um ein Duplikat oder einen neuen Eintrag handelt, was dabei hilft, eine saubere und organisierte Datenbank aufrechtzuerhalten.

3. Speichern neuer Audiodateidaten in der Datenbank:
Ein Benutzer lädt eine neue, eindeutige Audiodatei hoch und das System muss ihre Daten in der Datenbank speichern.
Schritte: Der Benutzer lädt die Audiodatei hoch. Die Anwendung extrahiert relevante Metadaten und technische Informationen. Nach der Bestätigung, dass die Datei eindeutig ist (keine Duplikate in der Datenbank), speichert die Anwendung die Daten der Datei in der PostgreSQL-Datenbank. Das System bestätigt die erfolgreiche Speicherung und gibt dem Benutzer die Details des neuen Eintrags zurück.
Ergebnis: Die Daten der Audiodatei werden sicher in der Datenbank gespeichert und der Benutzer erhält eine Bestätigung, dass seine Datei hinzugefügt wurde.

4. Anzeigen von Audiodateidetails:
Ein Benutzer möchte die Details einer bestimmten Audiodatei anzeigen, die in der Datenbank gespeichert ist.
Schritte: Der Benutzer fragt die Datenbank über die Webanwendung ab, entweder durch Hochladen einer Datei (um nach Duplikaten zu suchen) oder über eine Suchoberfläche. Die Anwendung ruft die relevanten Daten aus der Datenbank ab. Die Details der Audiodatei, einschließlich ihrer Metadaten und technischen Attribute, werden dem Benutzer angezeigt.
Ergebnis: Der Benutzer kann detaillierte Informationen zu einer bestimmten Audiodatei anzeigen, die in der Datenbank gespeichert ist, was bei der Katalogisierung und Verwaltung hilft.

5. Umgang mit nicht unterstützten Audiodateiformaten:
Ein Benutzer versucht, eine Audiodatei in einem Format hochzuladen, das die Anwendung nicht unterstützt.
Schritte: Der Benutzer wählt eine Audiodatei in einem nicht unterstützten Format aus (z. B. ein anderes Format als MP3 oder WAV). Die Anwendung prüft den Dateityp und erkennt, dass dieser nicht unterstützt wird. Das System gibt dem Benutzer eine Fehlermeldung zurück, die ihn über das nicht unterstützte Format informiert und ihm empfiehlt, eine Datei in einem unterstützten Format hochzuladen.
Ergebnis: Der Benutzer wird darüber informiert, dass sein Dateiformat nicht unterstützt wird, wodurch Fehler bei der Verarbeitung vermieden und die Anwendungsstabilität aufrechterhalten wird.

6. Fehlerbehandlung beim Hochladen von Dateien:
Ein Benutzer stößt beim Hochladen von Dateien auf einen Fehler, z. B. eine unvollständige Datei oder ein Serverproblem.
Schritte: Der Benutzer lädt eine Datei hoch, aber während der Verarbeitung tritt ein Problem auf (z. B. die Datei ist beschädigt oder es liegt ein Serverfehler vor). Die Anwendung erkennt den Fehler, stoppt die Verarbeitung und entfernt alle teilweise hochgeladenen Dateien, um Unordnung zu vermeiden. Das System gibt dem Benutzer eine Fehlermeldung zurück, die

das Problem erklärt und möglicherweise Lösungen vorschlägt (z. B. erneuter Uploadversuch der Datei).
Ergebnis: Der Benutzer wird über den Fehler informiert und die Anwendung behandelt ihn ordnungsgemäß, ohne abzustürzen, wodurch ein reibungsloses Benutzererlebnis gewährleistet wird.

7. Integration von Audio-Metadaten in ein größeres Datenbanksystem:
Eine Organisation möchte Audiodatei-Metadaten zur Katalogisierung und Analyse in ein größeres Datenbanksystem integrieren.
Schritte: Die Organisation lädt Audiodateien über die Weboberfläche oder programmgesteuert über eine API hoch. Die Anwendung verarbeitet die Dateien, extrahiert Metadaten und interagiert mit der vorhandenen PostgreSQL-Datenbank der Organisation, um Datensätze zu speichern oder zu aktualisieren. Die Organisation ruft diese Daten ab und verwendet sie für weitere Analysen, z. B. Katalogisierung, Suche oder statistische Analysen.
Ergebnis: Die Organisation integriert erfolgreich Audiodatei-Metadaten in ihr größeres Datenbanksystem und verbessert so ihre Fähigkeit, Audio-Assets zu verwalten und zu analysieren.

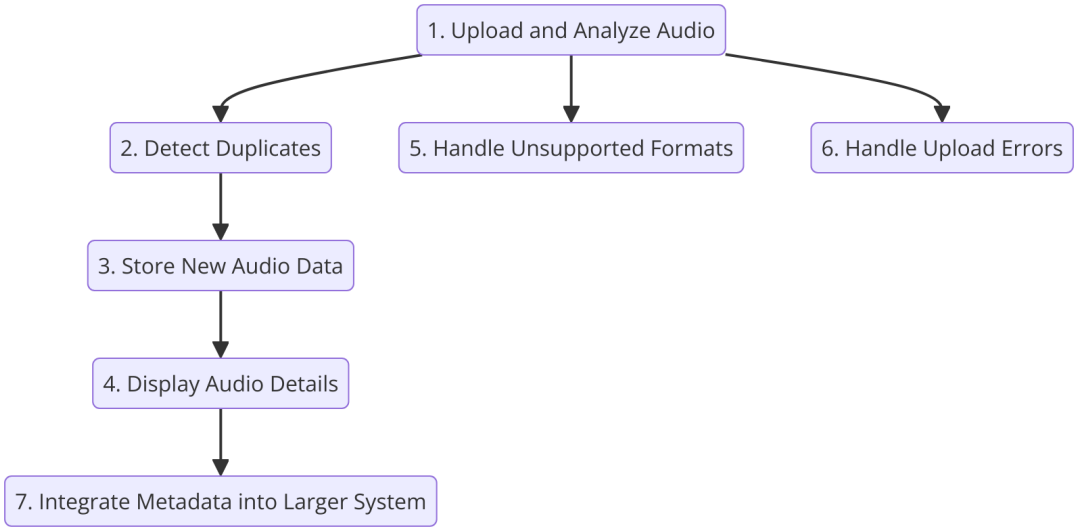

Abbildung 4.3: Anwendungsfälle der Audiodateiverarbeitung und -analyse

Beschreibung der Abbildung 4.3:
Diese Abbildung illustriert die wichtigsten Anwendungsfälle einer Audiodateiverarbeitungsanwendung. Der Prozess beginnt mit dem Hochladen und Analysieren von Audiodateien. Das System überprüft, ob es sich um eine doppelte Datei handelt, speichert neue Audiodaten und zeigt die Ergebnisse dem Benutzer an. Es verarbeitet außerdem Fehler und nicht unterstützte Formate, um eine reibungslose Benutzererfahrung sicherzustellen. Abschließend werden die Audiodaten in ein größeres Datenbanksystem integriert, was eine umfassende Verwaltung und Analyse ermöglicht.

Kapitel 5

Implementierung

Im folgenden Abschnitt wird die praktische Implementierung des beschriebenen Konzepts anhand von Screenshots und Beispielen aus der realisierten Anwendung veranschaulicht. Diese Bilder zeigen die Kernfunktionen der Anwendung, wie den Upload von Audiodateien, die Verarbeitung von Metadaten, die Anzeige der Ergebnisse sowie die Verwaltung der Datenbank. Die dargestellten Schritte geben einen Einblick in die Benutzeroberfläche und die technische Umsetzung, um das Verständnis für den Ablauf der Audiodateiverarbeitung und -speicherung zu vertiefen.

5.1 Audio-Duplikatsprozess

Die **Abbildung** 5.1 veranschaulicht den Prozessablauf der Audioverarbeitung und Duplikatsprüfung.

Zu den zentralen Komponenten gehören:

1. Python-Webframework:
• Flask ist ein leichtgewichtiges Python-Webframework zur Erstellung von Webanwendungen. Mit Flask können Routen (URLs) definiert und mit Python-Funktionen verknüpft werden, was die Verarbeitung von Anfragen und die Rückgabe von Antworten vereinfacht.
In diesem Code wird die Flask-Anwendung für unser Projekt initialisiert, und Routen werden mit dem Dekorator app.route definiert.
• Die beiden Hauptrouten in diesem Skript sind:
/: Zeigt die Homepage (index.html) an.
/upload: Ermöglicht den Dateiupload, verarbeitet die Datei, sucht in der Datenbank nach Duplikaten und gibt die Ergebnisse zurück.

2. Handhabung des Datei-Uploads:
Flask bietet eine einfache Schnittstelle zur Handhabung von Datei-Uploads über request.files. Der Code prüft, ob eine Datei hochgeladen wurde, speichert sie in einem temporären Verzeichnis (Uploads) und verarbeitet sie dann. Die Datei wird mit file.save gespeichert, wodurch die Anwendung auf den Inhalt und die Metadaten der Datei zugreifen kann.

3. Audioverarbeitung mit Librosa und SoundFile
• Librosa ist ein Python-Paket für Musik- und Audioanalyse. Es bietet Tools zum Extrahieren von Merkmalen aus Audiodateien, wie Wellenformdaten und Dauer.
Im Code lädt librosa.load die Audiodatei und gibt die Wellenformdaten und die Abtastrate zurück. Die Funktion librosa.get- duration berechnet die Dauer der Audiodatei in Sekunden.
• SoundFile ist eine weitere Python-Bibliothek zum Lesen und Schreiben von Sounddateien. Sie wird hier verwendet, um die Bittiefe der Audiodatei zu bestimmen, indem auf den Datentyp der Audiosamples zugegriffen wird.

4. Metadatenextraktion mit Mutagen:
• Mutagen ist eine Python-Bibliothek zur Handhabung von Audiometadaten und unterstützt Formate wie MP3 und WAV.Die Funktion extract-metadata file-path verwendet Mutagen, um Metadaten aus der hochgeladenen Audiodatei zu extrahieren.
• Je nach Dateityp MP3 oder WAV werden Metadaten wie Titel, Künstler, Album und Jahr abgerufen.Für MP3-Dateien werden die Klassen MP3 und EasyID3 verwendet, um ID3-Tags zu extrahieren. Für WAV-Dateien werden die Klassen WAVE und EasyID3 verwendet, um Metadaten zu extrahieren.Wenn der Dateityp nicht unterstützt wird, wird eine Fehlermeldung zurückgegeben.

5. Datenbankinteraktion und SQL-Abfragen:
• Der Code interagiert mit einer PostgreSQL-Datenbank und verwendet HTTP-POST-Anfragen, um SQL-Abfragen auf einem Remote-Server auszuführen.
• Die Funktion loaddata (title, bitrate, samplerate, length, size) erstellt eine SQL INSERT-Abfrage, um der TRACKS-Tabelle einen neuen Titel hinzuzufügen, und gibt die ID des neu eingefügten Datensatzes zurück.
• Um nach Duplikaten zu suchen, sendet der Code eine SELECT-Abfrage an den Remote-Server und ruft Datensätze ab, die mit dem Titel der hochgeladenen Audiodatei übereinstimmen. Die Funktion requests.post() sendet die SQL-Abfrage an den Remote-Server und die Funktion parse-query-result() analysiert die HTML-Antwort, um Abfrageergebnisse zu extrahieren.

6. HTML-Parsing mit BeautifulSoup:
• BeautifulSoup ist eine Python-Bibliothek zum Parsen von HTML- und XML-Dokumenten. Sie erstellt einen Parse-Baum aus dem Quellcode der Seite und ermöglicht so eine einfache Datenextraktion aus HTML-Elementen.
• In der Funktion parse-query-result(html) wird BeautifulSoup verwendet, um die erste Tabelle in der HTML-Antwort zu finden, Zeilen und Spalten zu extrahieren und die Daten in einem strukturierten Format zurückzugeben.

7. Vorlagen-Rendering:
• Die Funktion render-template() von Flask wird zum Rendern von HTML-Vorlagen mit dynamischen Daten verwendet.
• Die Funktion index() rendert die Vorlage index.html, wenn auf die Stamm-URL zugegriffen wird.Nach der Verarbeitung der hochgeladenen Datei rendert die Funktion upload-file() die Vorlage result.html und übergibt Daten wie die Abtastrate des Audios, Bittiefe, Dateigröße, Länge, Metadaten und ob die Datei ein Duplikat ist.

8. Plagiatsprüfung und Datentransformation:
• Der Code enthält einen grundlegenden Mechanismus zum Erkennen doppelter Titel in der Datenbank, die im Kontext dieses Skripts als „Plagiat“ bezeichnet werden.
• Wenn ein Duplikat gefunden wird, werden die entsprechenden Daten extrahiert und zur Anzeige an die Vorlage result.html übergeben.
• Wenn kein Duplikat gefunden wird, wird der Titel in die Datenbank eingefügt und seine Daten werden abgerufen und angezeigt.

9. Fehlerbehandlung:
Der Code enthält eine grundlegende Fehlerbehandlung zur Verwaltung von Problemen wie nicht unterstützten Dateitypen, fehlenden Dateien oder SQL Anfragefehlern.
Wenn während der Dateiverarbeitung ein Fehler auftritt, versucht der Code, die temporäre Datei zu entfernen und gibt eine JSON-Antwort mit der Fehlermeldung zurück.

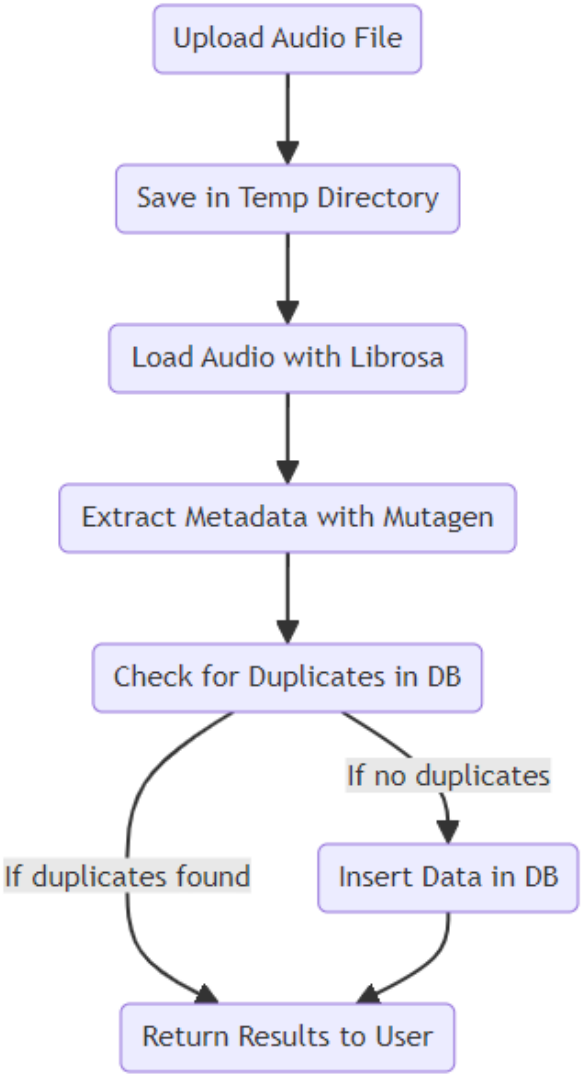

Abbildung 5.1: Prozess der Audioverarbeitung und Duplikatsprüfung

Beschreibung der Abbildung 5.1:
Diese Abbildung zeigt den Arbeitsablauf des Projekts, das den Upload und die Verarbeitung von Audiodateien sowie die Überprüfung auf Duplikate umfasst. Der Prozess beginnt mit dem Hochladen einer Audiodatei, die temporär gespeichert und dann mit der Bibliothek "Librosa"geladen wird, um akustische Merkmale zu extrahieren. Mit der Mutagen-Bibliothek werden anschließend die Metadaten der Datei ausgelesen. Diese Daten werden dann mit einer Datenbank verglichen, um Duplikate zu identifizieren. Falls Duplikate gefunden werden, erhält der Benutzer eine entsprechende Rückmeldung. Andernfalls wird der Datensatz in die Datenbank eingefügt und das Ergebnis an den Benutzer zurückgegeben.

5.2 Verwendete Technologien

Das Projekt nutzt eine Vielzahl von Technologien, in der **Abbildung** 5.2 unten dargestellt, um seine Funktionalität zu erreichen. Diese Technologien erstrecken sich über verschiedene Bereiche wie Webentwicklung, Audioverarbeitung, Datenbankinteraktion und mehr. Hier ist ein umfassender Überblick über die in diesem Projekt verwendeten Schlüsseltechnologien und ihre jeweiligen Rollen.

1. Flask: (Version 3.0.3) ist ein schlankes und flexibles Webframework für Python. Dank seiner Einfachheit, seines Minimalismus und seiner Benutzerfreundlichkeit ist es sowohl bei Anfängern als auch bei erfahrenen Entwicklern beliebt. Da Flask auf dem WSGI-Standard (Web Server Gateway Interface) basiert, eignet es sich hervorragend zum Entwickeln von Webanwendungen und APIs.
•**Rolle im Projekt:** In diesem Projekt dient Flask als zentrales Webframework, das Routing, Anforderungsverarbeitung und Antwortgenerierung verwaltet. Es initialisiert die Anwendung, definiert Routen (wie / für die Homepage und /upload für die Verarbeitung von Datei-Uploads), verarbeitet Benutzeranforderungen und rendert HTML-Vorlagen, um dynamische Webseiten zu generieren. Die Flexibilität von Flask ermöglicht die Integration verschiedener anderer Bibliotheken und Tools, die im Projekt verwendet werden.

2. Jinja2: (Version 3.1.4) ist eine moderne und designerfreundliche Template-Engine für Python. Sie ermöglicht Entwicklern die Erstellung dynamischer HTML-Inhalte durch Einbettung von Python-ähnlichen Ausdrücken in HTML-Dateien.
•**Rolle im Projekt:** Jinja2 wird in Verbindung mit Flask zum Rendern von HTML-Vorlagen verwendet. Es ermöglicht dem Projekt, dynamisch Webseiten zu generieren, die serverseitig verarbeitete Daten anzeigen. Beispielsweise verwenden die Vorlagen index.html und result.html die Jinja2-Syntax, um Daten wie Audiometadaten und Dateiattribute einzufügen, wodurch die Webseiten interaktiv und informativ werden.

3. Librosa: (Version 0.10.2.post1) ist eine leistungsstarke Python-Bibliothek für Audio- und Musikanalyse. Sie bietet eine breite Palette von Dienstprogrammen für die Arbeit mit Audiodaten, darunter das Laden von Audiodateien, das Extrahieren von Funktionen und die Analyse von Ton.
•**Rolle im Projekt:** In diesem Projekt wird Librosa verwendet, um Audiodateien zu laden und grundlegende Attribute wie Wellenform und Abtastrate zu extrahieren. Es berechnet auch die Dauer der Audiodatei. Diese Funktionen sind wichtig, um den Audioinhalt zu analysieren und wichtige Merkmale zu bestimmen, die später in der Datenbank gespeichert werden.

4. SoundFile (PySoundFile) : (Version 0.12.1) ist eine Python-Bibliothek, die eine Schnittstelle zur C-Bibliothek libsndfile bietet und das Lesen und Schreiben von Sounddateien in verschiedenen Formaten ermöglicht. Es unterstützt WAV, FLAC und andere Audiodateiformate.
•**Rolle im Projekt:** SoundFile wird verwendet, um Audiodaten aus Dateien zu lesen und die

Bittiefe zu bestimmen. Die Fähigkeit der Bibliothek, verschiedene Audioformate zu verarbeiten, ergänzt Librosa, das sich mehr auf die Extraktion von Audiomerkmalen konzentriert. SoundFile stellt sicher, dass das Projekt Audiodateimerkmale wie die Bittiefe genau abrufen kann, was für die Metadatenanalyse entscheidend ist.

5. Mutagen: (Version 1.47.0) ist ein Python-Modul zum Verarbeiten von Audiometadaten. Es unterstützt verschiedene Audiodateiformate, darunter MP3, MP4, OGG, FLAC und mehr. Mutagen ermöglicht das Lesen und Schreiben von Metadaten-Tags wie Titel, Künstler, Album und andere.
•**Rolle im Projekt:** Mutagen wird verwendet, um Metadaten aus Audiodateien zu extrahieren. Abhängig vom Dateityp (.mp3, .wav) ruft Mutagen Informationen wie Titel, Künstler und Album ab. Diese Metadaten werden dann verwendet, um nach doppelten Dateien in der Datenbank zu suchen und neue Einträge zu speichern. Die Unterstützung mehrerer Dateiformate durch Mutagen macht es zu einer vielseitigen Wahl für das Projekt.

6. BeautifulSoup: (Version 4.12.3) ist eine Python-Bibliothek zum Parsen von HTML- und XML-Dokumenten. Sie erstellt Parsebäume aus dem Seitenquellcode, wodurch sich Daten aus HTML-Tags und -Attributen leicht extrahieren lassen.
•**Rolle im Projekt:** BeautifulSoup wird zum Parsen der vom Datenbankserver empfangenen HTML-Antworten verwendet. Nachdem die Flask-Anwendung eine SQL-Abfrage über eine HTTP-POST-Anfrage gesendet hat, erhält sie eine HTML-Antwort mit den Abfrageergebnissen. BeautifulSoup verarbeitet diese Antwort, um die in HTML-Tabellen enthaltenen Daten zu extrahieren, wodurch sie in der Anwendung leicht zu handhaben und zu verwenden ist.

7. Requests: (Version 2.32.2) ist eine einfache und elegante HTTP-Bibliothek für Python. Sie ermöglicht das Senden von HTTP-Anfragen mit Python und erleichtert so die Interaktion mit Webdiensten und APIs.
•**Rolle im Projekt:** Requests wird zum Senden von HTTP-POST-Anfragen an den Remote-Server verwendet, der SQL-Abfragen verarbeitet. Die Bibliothek ist für die Schnittstelle mit dem Datenbankserver über HTTP von entscheidender Bedeutung, da sie es der Anwendung ermöglicht, SQL-Befehle auszuführen und Ergebnisse abzurufen. Die Benutzerfreundlichkeit und Zuverlässigkeit von Requests machen es für diese Aufgabe gut geeignet.

8. PostgreSQL : ist ein leistungsstarkes relationales Datenbankmanagementsystem (RDBMS) mit Open Source. Es ist bekannt für seine Robustheit, Skalierbarkeit und Unterstützung erweiterter SQL-Funktionen.
•**Rolle im Projekt:** PostgreSQL dient als Backend-Datenbank zum Speichern und Abrufen von Audiodateidaten und Metadaten. Das Projekt verwendet SQL-Abfragen zur Interaktion mit der PostgreSQL-Datenbank und führt Aufgaben wie die Überprüfung auf Duplikate und das Einfügen neuer Datensätze aus. Aufgrund der Fähigkeit von PostgreSQL, komplexe Abfragen und große Datensätze zu verarbeiten, eignet es sich für die Verwaltung der Audiometadaten und Attribute in diesem Projekt.

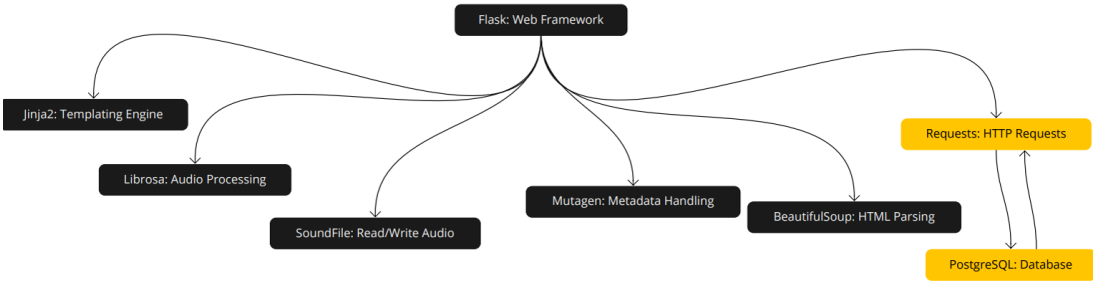

Abbildung 5.2: Verwendete Technologien in der Webanwendung zur Audiodateiverarbeitung

Beschreibung der Abbildung 5.2:
Die Abbildung zeigt ausführlich den Datenfluss innerhalb einer Webanwendung, die Audiodateien hochlädt, Metadaten extrahiert, Datenbankabfragen durchführt und dem Benutzer die Ergebnisse präsentiert. Der Prozess beginnt damit, dass ein Benutzer über die Weboberfläche eine Audiodatei hochlädt. Diese Datei wird anschließend validiert, gespeichert und mithilfe von Bibliotheken wie Librosa, SoundFile und Mutagen verarbeitet, um wichtige Audioattribute und Metadaten zu extrahieren. Danach wird in der Datenbank geprüft, ob bereits ein Duplikat der Datei existiert. Die Ergebnisse dieser Überprüfung—ob also ein Duplikat vorliegt oder die Datei neu hinzugefügt wird—werden auf einer Ergebnisseite angezeigt und dem Benutzer präsentiert. Abschließend wird die temporäre Datei vom Server gelöscht, um Speicherplatz freizugeben.

5.3 Projektfunktionen

Initialisieren der Anwendung:

1. Zu Beginn von app.py wird die Flask-Anwendung initialisiert:

```
from flask import Flask, request, jsonify, render_template
app = Flask(__name__)
```

2. File Upload Handling: Die /upload Route übernimmt den Datei-Upload:

```
@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return jsonify({'error': 'No file part'})

    file = request.files['file']
    if file.filename == '':
        return jsonify({'error': 'No selected file'})

```

```
    if file:
        file_path = os.path.join('uploads', file.filename)
        try:
            file.save(file_path)
            # Process the file after saving
        except Exception as e:
            if os.path.exists(file_path):
                os.remove(file_path)
            return jsonify({'error': str(e)})

    return jsonify({'error': 'Unknown error'})
```

3. Metadatenextraktion: Die Funktion `extract_metadata` wird verwendet, um Metadaten aus der Audiodatei zu extrahieren:

```
def extract_metadata(file_path):
    try:
        metadata = {}
        file_extension = os.path.splitext(file_path)[1].lower()
        if file_extension == '.mp3':
            audio = MP3(file_path, ID3=EasyID3)
            metadata = {key: audio[key] for key in audio.keys()}
        elif file_extension == '.wav':
            audio = WAVE(file_path)
            for key in audio.keys():
                metadata[key] = audio[key]
        else:
            metadata = {'error': 'Unsupported file type'}
        return metadata
    except Exception as e:
        return {'error': str(e)}
```

Diese Funktion öffnet die Audiodatei mithilfe der Mutagen-Bibliothek und extrahiert Metadaten wie Titel, Künstler und Albuminformationen.

4. Datenbankintegration: Die Datenbankinteraktion wird über HTTP-Anfragen mithilfe der Anforderungsbibliothek abgewickelt:

```
def loaddata(title, bitrate, samplerate, length, size):
    sql_query = f"INSERT INTO TRACKS (title, bitrate, length,
        samplerate, filesize) VALUES ('{title}', {bitrate}, {length}, {
        samplerate}, {size}) RETURNING id"

    post_data = {
        'dbSystem': 'PostgreSQL',
        'query': sql_query
```

```
    }

    response = requests.post("http://dbis-lehre.informatik.uni-rostock.
        de/db1/ergebnis-db1-spielwiese.php", data=post_data)
    query_result = parse_query_result(response.text)
    return query_result[0][0]
```

Diese Funktion erstellt eine SQL-Abfrage zum Einfügen der Audio-Metadaten in die Datenbank und sendet sie über eine HTTP-POST-Anforderung. Die Funktion `parse_query_result` wird dann verwendet, um das Abfrageergebnis zu analysieren.

5. HTML-Ergebnisse analysieren: `BeautifulSoup` wird zum Analysieren der vom Server zurückgegebenen HTML-Ergebnisse verwendet:

```
def parse_query_result(html):
    soup = BeautifulSoup(html, 'html.parser')
    table = soup.find('table')
    values = []

    if table:
        rows = table.find_all('tr')
        for row in rows:
            cols = row.find_all('td')
            values.append([col.text.strip() for col in cols])

    return values
```

Diese Funktion lokalisiert und extrahiert Daten aus Tabellen auf einer HTML-Seite und ermöglicht der Anwendung, die Ergebnisse von Datenbankabfragen zu verarbeiten.

5.4 DatenbanksSchema

Für die Verwaltung der Audiodaten wurde eine relationale Datenbank verwendet, um eine effiziente Speicherung und Verarbeitung der extrahierten und transformierten Daten zu gewährleisten. Das Schema der Datenbank umfasst mehrere Tabellen, die spezifische Informationen zu den Audiodateien und ihren Attributen enthalten. Eine der zentralen Tabellen ist die Tabelle tracks, welche die wesentlichen Metadaten und Kennzahlen zu den verarbeiteten Audiodateien speichert.

Um detaillierte Informationen über die Struktur der Tabelle tracks zu erhalten, kann die folgende **SQL-Anfrage** verwendet werden:

```
SELECT *
FROM INFORMATION-SCHEMA.COLUMNS
WHERE TABLE-NAME = 'tracks';
```

Diese Abfrage liefert eine Übersicht über alle Spalten der Tabelle, einschließlich ihrer Datentypen, Einschränkungen und anderen relevanten Informationen, die für die Datenverarbeitung und das Laden in das System wichtig sind.
Nachfolgend ist die Struktur der Tabelle tracks dargestellt:

Feldname	Datentyp	Beschreibung
id	INTEGER	Dies ist die eindeutige Kennung für jeden Titel. Sie wird automatisch von der Datenbank generiert und dient zur eindeutigen Unterscheidung jedes Datensatzes in der Tabelle.
title	CHARACTER VARYING	Dieses Feld speichert den Titel des Audiotitels. Er wird mithilfe der Mutagen-Bibliothek aus den Metadaten der Audiodatei extrahiert. Der Titel hilft Benutzern, den Titel leicht zu identifizieren.
Bitrate	INTEGER	Dieses Feld speichert die Bitrate der Audiodatei, gemessen in Kilobit pro Sekunde (kbps). Die Bitrate ist ein wichtiger Indikator für die Audioqualität.
Samplerate	INTEGER	Dieses Feld speichert die Samplingrate der Audiodatei, gemessen in Hertz (Hz). Sie stellt die Anzahl der Audiosamples pro Sekunde dar, was für das Verständnis der Audioqualität entscheidend ist.
Länge	INTEGER	Dieses Feld speichert die Dauer der Audiodatei in Millisekunden. Die Dauer hilft beim Verständnis der Länge des Inhalts.
Dateigröße	INTEGER	Dieses Feld speichert die Größe der Audiodatei in Bytes. Die Kenntnis der Dateigröße ist für die Speicherverwaltung und Überlegungen zur Netzwerkbandbreite nützlich.
Jahr	INTEGER	Dieses optionale Feld speichert das Jahr der Veröffentlichung oder Erstellung des Audiotitels. Es ist Teil der aus der Datei extrahierten Metadaten und kann für die Kategorisierung und Filterung nützlich sein.
Künstler	INTEGER	Dieses Feld speichert den Namen des Künstlers oder Erstellers des Audiotitels. Es wird verwendet, um den Titel seinem rechtmäßigen Ersteller zuzuordnen.
Album	INTEGER	Dieses optionale Feld speichert den Namen des Albums, zu dem der Audiotitel gehört. Es hilft dabei, Titel nach ihren jeweiligen Alben zu organisieren.

Tabelle 5.1: Übersicht der Felder in der TRACKS-Tabelle

5.5 Ergebnisse

Hier werden die Resultate des entwickelten ETL-Prozesses zur Audiodatenverarbeitung vorgestellt. Zunächst wird eine Zusammenfassung des implementierten Codes gegeben, wobei die wichtigsten Programmteile und ihre Funktionen erläutert werden. Abschließend werden die einzelnen Schritte der Audiodateiverarbeitung detailliert beschrieben, um aufzuzeigen, wie die Daten extrahiert, transformiert und schließlich für die Plagiatserkennung aufbereitet werden

5.5.1 Zusammenfassung des Codes

Der Code enthält alle Schritte von der Verarbeitung der Audiodatei, der Extraktion von Metadaten, der Überprüfung auf Duplikate, bis hin zur Speicherung der Daten in der Datenbank und der Darstellung der Ergebnisse für den Benutzer.

1. Flask-App und Routing: Der erste Abschnitt des Codes initialisiert die Flask-Webanwendung und definiert die Routen für den Dateiupload und die Anzeige der Ergebnisse. Hier wird die Logik für das Empfangen von Dateien und das Rendern der HTML-Seiten implementiert.
2. Dateihandhabung und Validierung: Dieser Teil des Codes behandelt den Upload der Audiodatei, die Validierung des Dateiformats und das Speichern der Datei im temporären Verzeichnis. Zudem werden mögliche Fehler beim Upload oder bei der Dateiverarbeitung abgefangen.
3. Verarbeitung der Audiodatei: Mithilfe von Bibliotheken wie Librosa, SoundFile und Mutagen werden die Audiodateien analysiert, um technische Informationen wie Samplerate, Bitrate und Dauer sowie eingebettete Metadaten wie Titel, Künstler und Album zu extrahieren.
4. Datenbankinteraktion: Der nächste Teil des Codes zeigt die Verbindung zur PostgreSQL-Datenbank und führt SQL-Abfragen durch. Dabei werden neue Audiodateien hinzugefügt oder Duplikate erkannt, um eine ordnungsgemäße Verwaltung der Audiodaten sicherzustellen.
5. Ergebnisanzeige: Schließlich wird der Code beschrieben, der die gerenderten Ergebnisse auf einer HTML-Seite anzeigt. Hier werden die extrahierten Daten und Metadaten dem Benutzer visuell präsentiert.

5.5.2 Schritte der Audiodateiverarbeitung

Die folgenden Abbildungen veranschaulichen den gesamten Ablauf der „Etl-Audioprojektt“ Webanwendung. Sie zeigen die wichtigsten Schritte der Benutzerinteraktion, beginnend mit dem Hochladen einer Audiodatei, über die Extraktion von Metadaten und technischen Informationen, bis hin zur Plagiatserkennung und dem Speichern der Datei in der Datenbank. Dieser visuelle Ablauf bietet einen umfassenden Überblick über die Funktionsweise der Anwendung.

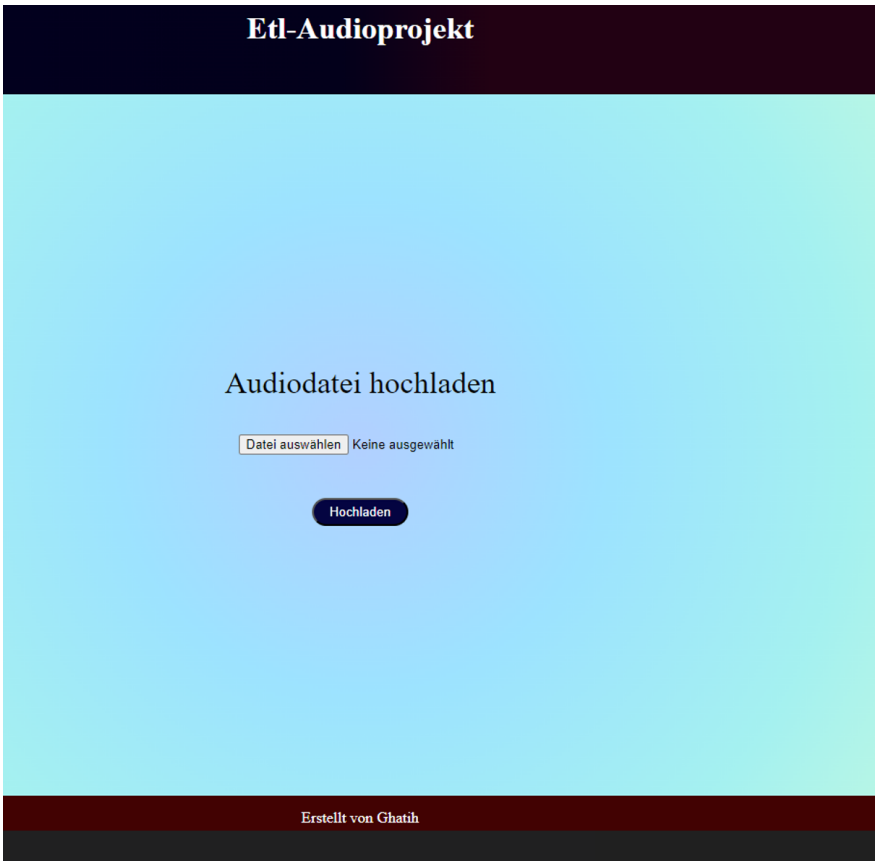

Abbildung 5.3: Startseite der Audiodateiverarbeitungsanwendung

Beschreibung der Abbildung 5.3:
Die Abbildung zeigt die Startseite der Webanwendung „Etl-Audioprojekt“. Auf dieser Seite haben Benutzer die Möglichkeit, Audiodateien hochzuladen, um sie analysieren zu lassen. Der Benutzer wird aufgefordert, eine Audiodatei auszuwählen und diese durch Drücken des „Hochladen“-Buttons an den Server zu senden, wo die Datei verarbeitet wird. Die Seite ist minimalistisch gestaltet, mit einem einfachen Formular, das die Benutzerinteraktion erleichtert. Am unteren Rand der Seite befindet sich ein Hinweis auf den Ersteller der Anwendung.

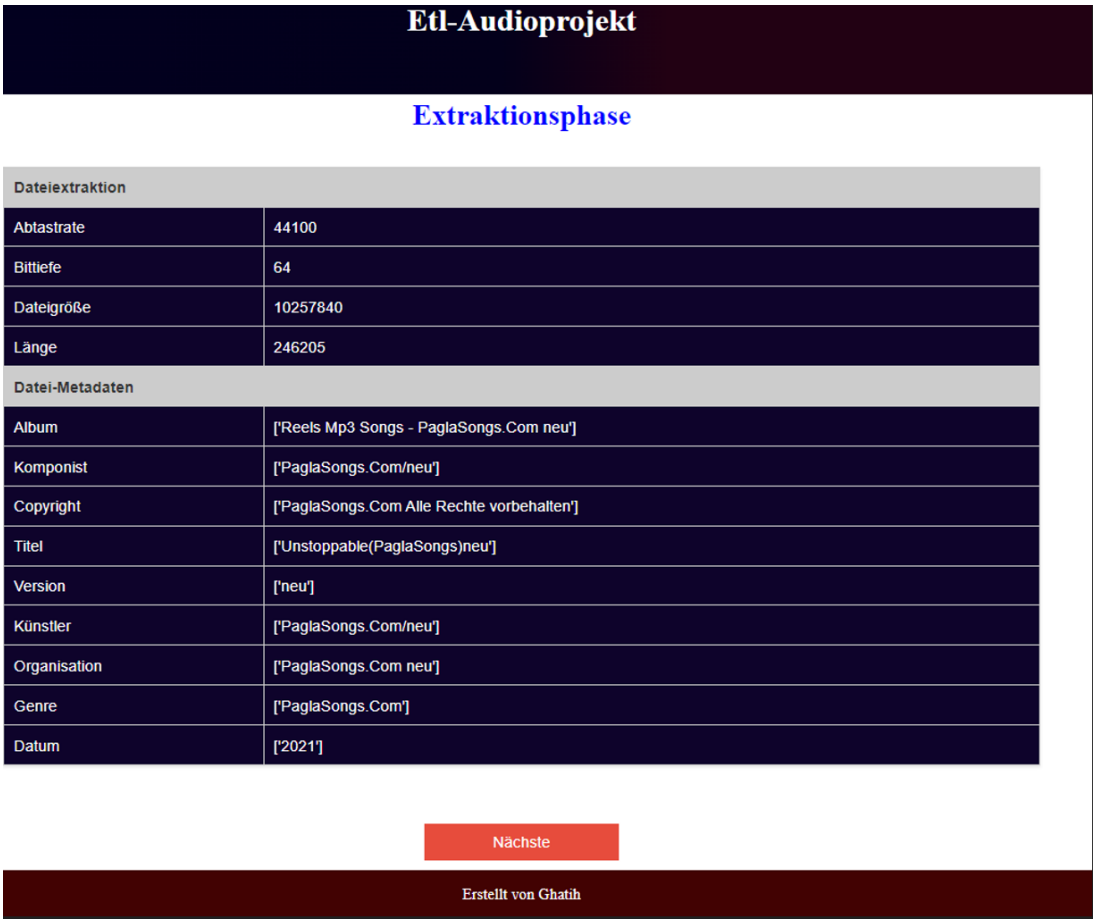

Abbildung 5.4: Extraktionsphase der Audiodateiverarbeitungsanwendung

Beschreibung der Abbildung 5.4:
Diese Abbildung zeigt die Ergebnisse der Extraktion von technischen Informationen und Metadaten nach dem Hochladen einer Audiodatei. Unter Dateiextraktion sind technische Attribute wie die Samplerate (sampling-rate), Bit-Tiefe (bit-depth), Dateigröße (filesize) und Dauer (length) der Audiodatei aufgelistet. Darunter werden unter Datei-Metadaten die Metadaten der Datei angezeigt, die Informationen wie das Album, den Komponisten, den Titel und das Veröffentlichungsjahr enthalten. Diese Phase ermöglicht es dem Benutzer, die vollständigen Eigenschaften und Metadaten der hochgeladenen Audiodatei einzusehen.

Abbildung 5.5: Transformationsphase der Audiodateiverarbeitungsanwendung

Beschreibung der Abbildung 5.5:
Diese Abbildung zeigt die Transformationsphase der Webanwendung „Etl-Audioprojekt“, in der die Datei nach der Verarbeitung weiter untersucht wird. Hier wird dem Benutzer angezeigt, dass bei der hochgeladenen Audiodatei kein Plagiat erkannt wurde (”Kein Plagiat festgestellt”). Der Status der Datei wird in zwei Abschnitten angezeigt: ”Dateiextraktionünd ”Datei-Metadaten”. Diese Phase markiert den Abschluss des Verarbeitungsprozesses, bei dem die Datei entweder als neues Element gespeichert oder als Duplikat erkannt wird. Der Benutzer kann mit den Schaltflächen „Vorherige“ und „Nächste“ durch die Phasen navigieren.

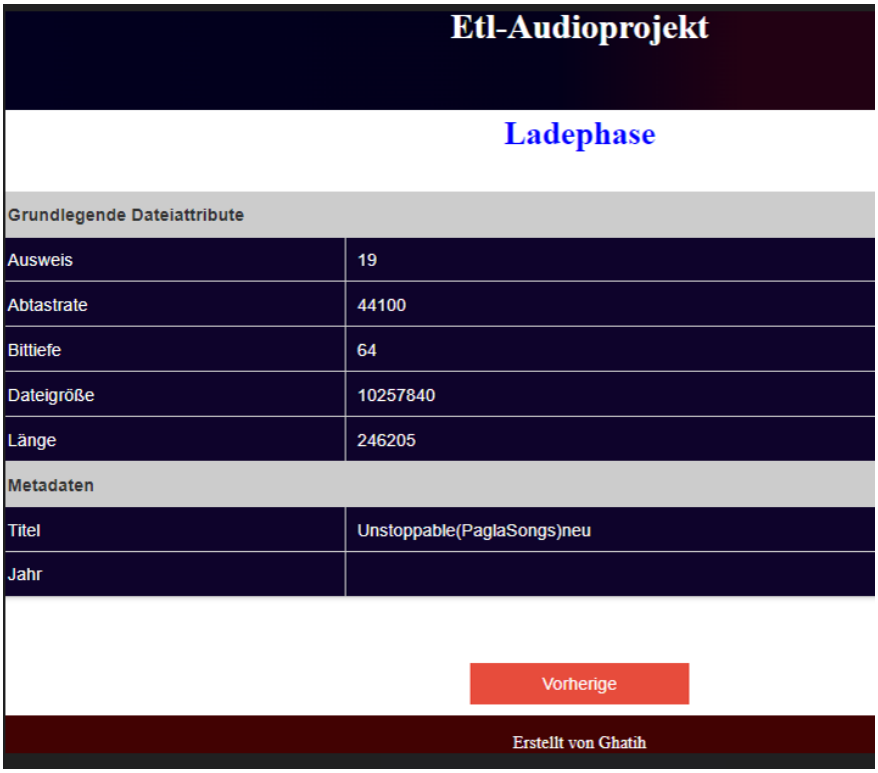

Abbildung 5.6: Ladephase der Audiodateiverarbeitungsanwendung

Beschreibung der Abbildung 5.6:
Die Abbildung zeigt die Ladephase der Webanwendung, in der die Datei nach erfolgreicher Verarbeitung und Speicherung in der Datenbank dargestellt wird. Es werden sowohl die grundlegenden Dateiattribute wie die ID, Samplerate (sampling-rate), Bit-Tiefe (bit-depth), Dateigröße (filesize) und Dauer (length) als auch die Meta-Daten wie der Titel und das Jahr angezeigt. Diese Phase dient der Übersicht über die gespeicherten Daten in der PostgreSQL-Datenbank. Der Benutzer kann die Darstellung der Datei mit der Schaltfläche „Vorherige“ verlassen.

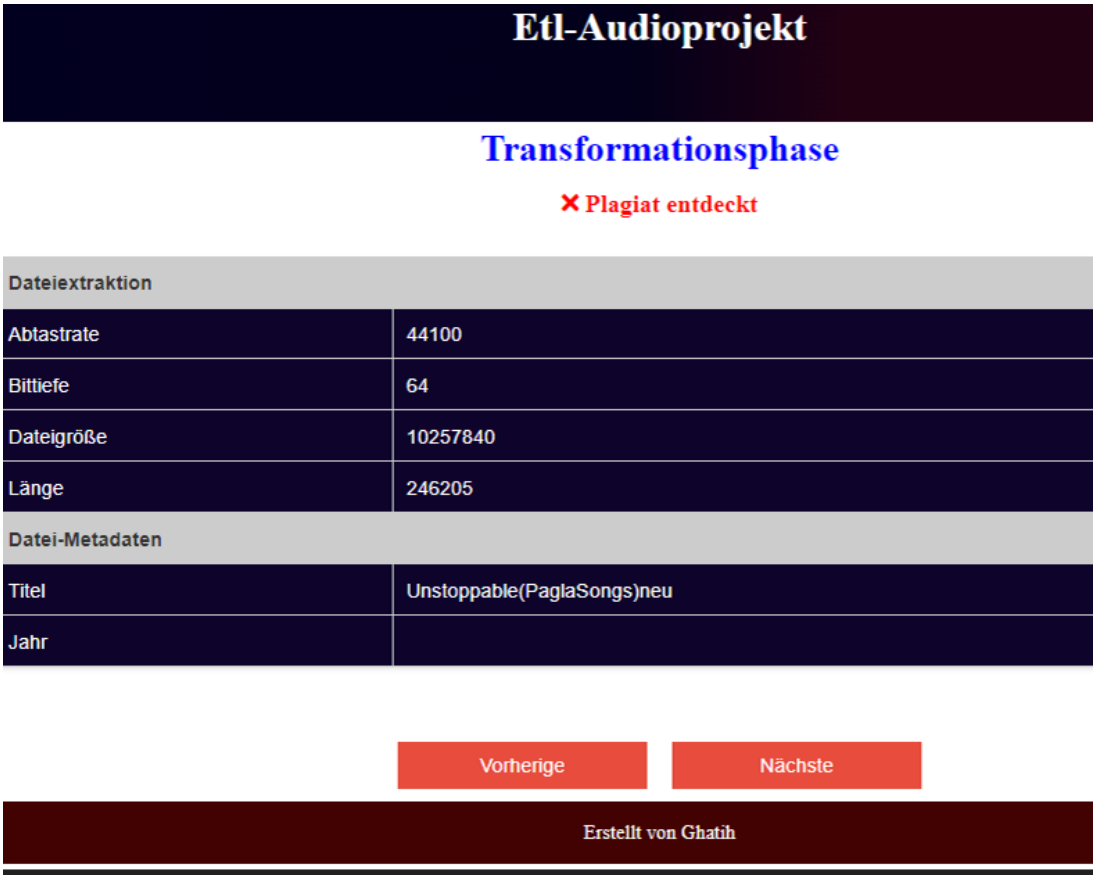

Abbildung 5.7: Transformationsphase mit Plagiatserkennung

Beschreibung der Abbildung 5.7:
In dieser Abbildung wird in der Transformationsphase der Webanwendung „Etl-Audioprojektt“ angezeigt, dass ein Plagiat erkannt wurde (”Plagiat entdeckt”). Die Anwendung stellt die technischen Details der hochgeladenen Datei dar, einschließlich der Samplerate (sampling-rate), Bit-Tiefe (bit-depth), Dateigröße (filesize) und Dauer (length). Zusätzlich werden die Metadaten der Datei angezeigt, darunter der Titel und das Veröffentlichungsjahr. Diese Phase dient dazu, den Benutzer darauf hinzuweisen, dass die hochgeladene Datei bereits in der Datenbank existiert.

Abbildung 5.8: Ladephase mit Plagiatserkennung

Beschreibung der Abbildung 5.8:
Diese Abbildung zeigt die Ladephase der Webanwendung „Etl-Audioprojekt“ nach der Verarbeitung einer hochgeladenen Audiodatei. In dieser Phase wird deutlich, dass ein Plagiat erkannt wurde, wie durch die Meldung „Plagiat entdeckt“ in roter Schrift angezeigt. Der Benutzer wird darüber informiert, dass die hochgeladene Datei bereits in der Datenbank vorhanden ist. Der „Vorherige“-Button ermöglicht es dem Benutzer, zur vorherigen Phase zurückzukehren.

Zusammenfassung der Schritte:

1.Startseite der Anwendung: Der Benutzer wird aufgefordert, eine Audiodatei hochzuladen. Das Interface ist einfach gestaltet, mit einem Button zur Auswahl der Datei und einem „Hochladen“-Button, um den Upload-Prozess zu starten.
2.Extraktionsphase: Nach dem Hochladen der Datei werden technische Daten wie Samplerate, Bit-Tiefe, Dateigröße und Länge extrahiert und zusammen mit den Metadaten wie Titel, Künstler und Album angezeigt.
3.Transformationsphase (kein Plagiat): Die Datei wird weiter untersucht, und es wird bestätigt, dass kein Plagiat vorliegt. Die Datei wird somit als einzigartig betrachtet und weiterverarbeitet.
4.Ladephase (neue Datei): Die Audiodatei wird erfolgreich in der Datenbank gespeichert, und ihre Attribute sowie Metadaten werden auf einer Übersichtsseite dargestellt. Der Benutzer erhält die Bestätigung über die erfolgreiche Speicherung.
5.Transformationsphase (Plagiat): Es wird festgestellt, dass die hochgeladene Datei bereits in der Datenbank vorhanden ist. Dies wird durch die Meldung „Plagiat entdeckt“ angezeigt, und die technischen Attribute sowie Metadaten der Datei werden zur Einsicht präsentiert.
6.Ladephase (Plagiat): In dieser abschließenden Phase wird die Plagiatserkennung erneut bestätigt, und der Benutzer hat die Möglichkeit, zur vorherigen Phase zurückzukehren.

Kapitel 6

Fazit

In dieser Bachelorarbeit wurde ein ETL-Prozess zur Ermittlung von Kennzahlen aus Audiodaten entwickelt, der als Vorbereitungsschritt für die Plagiatserkennung dient. Ziel war es, ein robustes System zu entwerfen, das Audiodaten aus verschiedenen Quellen effizient extrahiert, transformiert und in eine strukturierte Datenbank lädt. Die Arbeit hat gezeigt, dass ein maßgeschneiderter ETL-Prozess essenziell ist, um die spezifischen Herausforderungen bei der Verarbeitung von Audiodaten zu bewältigen.

Die Implementierung des ETL-Workflows führte zu einer deutlichen Verbesserung der Datenkonsistenz und -qualität, was die Grundlage für präzise und zuverlässige Plagiatserkennungen bildet. Durch die Anwendung von Techniken wie der Merkmalsextraktion und Datenbereinigung konnten relevante Informationen aus den Audiodaten gewonnen werden, die für die weitere Analyse und Erkennung von Plagiaten entscheidend sind.

Die Ergebnisse verdeutlichen, dass die eingesetzten Technologien und Verfahren in der Lage sind, große Mengen an Audiodaten effizient zu verarbeiten. Dennoch gibt es Bereiche, in denen Optimierungen möglich sind, insbesondere in Bezug auf die Skalierbarkeit und die Genauigkeit der Plagiatserkennung. Zukünftige Arbeiten könnten sich darauf konzentrieren, den Prozess weiter zu optimieren und zusätzliche Audiodatenquellen zu integrieren.

Insgesamt zeigt diese Arbeit, dass der entwickelte ETL-Prozess einen wichtigen Schritt hin zu einer effektiven und automatisierten Plagiatserkennung in Audiodaten darstellt. Die Ergebnisse bieten eine solide Basis für zukünftige Forschungen und Weiterentwicklungen auf diesem Gebiet.

Abbildungen

Literatur

[Ali18] Syed Muhammad Fawad Ali. „Next-generation ETL Framework to Address the Challenges Posed by Big Data.“ In: *DOLAP*. 2018 (siehe S. 3).

[AN08] Alexander Albrecht und Felix Naumann. „Managing ETL Processes.“ In: *NTII* 8.2008 (2008), S. 12–15 (siehe S. 9, 10).

[Beh15] Harrison Behl. „Audio formats: Characteristics and deterioration“. In: *ARSC Guide to Audio Preservation* (2015), S. 14 (siehe S. 16).

[Bur+19] Juan José Burred, Emmanuel Ponsot, Louise Goupil, Marco Liuni und Jean-Julien Aucouturier. „CLEESE: An open-source audio-transformation toolbox for data-driven experiments in speech and music cognition“. In: *PloS one* 14.4 (2019), e0205943 (siehe S. 33, 34).

[CDL21] Rodrigo Castellon, Chris Donahue und Percy Liang. „Codified audio language modeling learns useful representations for music information retrieval“. In: *arXiv preprint arXiv:2107.05677* (2021) (siehe S. 41–43).

[Cla+20] Christopher GD Clarke, Uzoma Nnajiuba, Jamie Howie, Muhammad Khan, Daniel Pinto dos Santos und Erik Ranschaert. „Giving radiologists a voice: a review of podcasts in radiology“. In: *Insights into Imaging* 11 (2020), S. 1–9 (siehe S. 25).

[Cla+23] Ivandro Claudino, Thiago Gadelha, Tiago Vinuto, José Wellington Franco, José Maria Monteiro und Javam Machado. „A Real-Time Platform to Monitoring Misinformation on Telegram“. In: (2023) (siehe S. 36, 37).

[De+12] Soham De, Indradyumna Roy, Tarunima Prabhakar, Kriti Suneja, Sourish Chaudhuri, Rita Singh und Bhiksha Raj. „Plagiarism Detection in Polyphonic Music using Monaural Signal Separation.“ In: *INTERSPEECH*. 2012, S. 1744–1747 (siehe S. 4).

[DNV06] Robbie De Sutter, Stijn Notebaert und Rik Van de Walle. „Evaluation of metadata standards in the context of digital audio-visual libraries“. In: *International Conference on Theory and Practice of Digital Libraries*. Springer. 2006, S. 220–231 (siehe S. 19, 20).

[Dob00] Richard W Dobson. „Developments in audio file formats“. In: *ICMC*. 2000 (siehe S. 16).

[FS16] Luthfi Firmansah und Erwin Budi Setiawan. „Data audio compression lossless FLAC format to lossy audio MP3 format with Huffman shift coding algorithm“. In: *2016 4th International Conference on Information and Communication Technology (ICoICT)*. IEEE. 2016, S. 1–5 (siehe S. 29).

[Gla+04] James Glass, Timothy J Hazen, Lee Hetherington und Chao Wang. „Analysis and processing of lecture audio data: Preliminary investigations“. In: *Proceedings of the Workshop on Interdisciplinary Approaches to Speech Indexing and Retrieval at HLT-NAACL 2004*. 2004, S. 9–12 (siehe S. 14, 15).

[Gli+11] Riste Gligorov, Michiel Hildebrand, Jacco Van Ossenbruggen, Guus Schreiber und Lora Aroyo. „On the role of user-generated metadata in audio visual collections“. In: *Proceedings of the sixth international conference on Knowledge capture*. 2011, S. 145–152 (siehe S. 19).

[Hor22] John Horodyski. *Metadata matters*. Auerbach Publications, 2022, S. 6–15 (siehe S. 20).

[Kha+24] Bilal Khan, Saifullah Jan, Wahab Khan und Muhammad Imran Chughtai. „An Overview of ETL Techniques, Tools, Processes and Evaluations in Data Warehousing.“ In: *Journal on Big Data* 6 (2024) (siehe S. 23).

[Kos+23] Ferdinand Kossmann, Ziniu Wu, Eugenie Lai, Nesime Tatbul, Lei Cao, Tim Kraska und Samuel Madden. „Extract-transform-load for video streams“. In: *arXiv preprint arXiv:2310.04830* (2023) (siehe S. 27, 28).

[KPM21] Parminder Kaur, Husanbir Singh Pannu und Avleen Kaur Malhi. „Comparative analysis on cross-modal information retrieval: A review“. In: *Computer Science Review* 39 (2021), S. 100336 (siehe S. 41).

[LC05] Ruei-Shiang Lin und Ling-Hwei Chen. „A new approach for classification of generic audio data“. In: *International Journal of Pattern Recognition and Artificial Intelligence* 19.01 (2005), S. 63–78 (siehe S. 25).

[LML22] Aarón López-Garcıa, Brian Martınez-Rodrıguez und Vicente Liern. „A Proposal to Compare the Similarity Between Musical Products. One More Step for Automated Plagiarism Detection?“ In: *International Conference on Mathematics and Computation in Music*. Springer. 2022, S. 192–204 (siehe S. 21).

[Lou+22] Siyu Lou, Xuenan Xu, Mengyue Wu und Kai Yu. „Audio-text retrieval in context“. In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, S. 4793–4797 (siehe S. 43).

[Mah+21] Amena Mahmoud, Mahmoud Y Shams, OM Elzeki und Nancy Awadallah Awad. „Using semantic web technologies to improve the extract transform load model“. In: *Computers, Materials & Continua* 68.2 (2021), S. 2711–2726 (siehe S. 37).

[MB15] Nilesh Mali und Sachin Bojewar. „A survey of ETL tools“. In: *International Journal of Computer Techniques* 2.5 (2015), S. 20–27 (siehe S. 13, 14).

[Mou05] John N Mourjopoulos. „The evolution of digital audio technology“. In: *Communication acoustics*. Springer, 2005, S. 299–319 (siehe S. 18).

[MZE06] Dalibor Mitrovic, Matthias Zeppelzauer und Horst Eidenberger. „Analysis of the data quality of audio features of environmental sounds“. In: *Journal of Universal Knowledge Management* 1.1 (2006), S. 4–17 (siehe S. 37).

[NiF20] Apache NiFi. *Apache NiFi Overview*. 2020 (siehe S. 12).

[Par+22] Kyuwon Park, Seungyeon Baek, Jueun Jeon und Young-Sik Jeong. „Music Plagiarism Detection Based on Siamese CNN“. In: *Hum.-Cent. Comput. Inf. Sci* 12 (2022), S. 12–38 (siehe S. 22).

[Pha20] Phuong Pham. „A case study in developing an automated ETL solution: concept and implementation“. In: (2020) (siehe S. 24).

[Pin24] Filipe de Morais Teixeira Pinto. „A Framework for Integrating Heterogeneous Data Sources“. In: (2024) (siehe S. 4).

[PP20] Monika Patel und Dhiren B Patel. „Progressive growth of ETL tools: A literature review of past to equip future“. In: *Rising Threats in Expert Applications and Solutions: Proceedings of FICR-TEAS 2020* (2020), S. 389–398 (siehe S. 11).

[Qam+21] Mustafa A Qamhan, Hamdi Altaheri, Ali Hamid Meftah, Ghulam Muhammad und Yousef Ajami Alotaibi. „Digital audio forensics: microphone and environment classification using deep learning“. In: *Ieee Access* 9 (2021), S. 62719–62733 (siehe S. 25).

[Raj+20] Aiswarya Raj, Jan Bosch, Helena Holmström Olsson und Tian J Wang. „Modelling data pipelines“. In: *2020 46th Euromicro conference on software engineering and advanced applications (SEAA)*. IEEE. 2020, S. 13–20 (siehe S. 41).

[RDS05] Regunathan Radhakrishnan, Ajay Divakaran und A Smaragdis. „Audio analysis for surveillance applications“. In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005.* IEEE. 2005, S. 158–161 (siehe S. 15).

[SA15] Dumin Sahiet und PD Asanka. „ETL framework design for NoSQL databases in dataware housing“. In: *Int. J. Res. Comput. Appl. Rob* 3 (2015), S. 67–75 (siehe S. 4).

[Sab+17] Adilah Sabtu, Nurulhuda Firdaus Mohd Azmi, Nilam Nur Amir Sjarif, Saiful Adli Ismail, Othman Mohd Yusop, Haslina Sarkan und Suriayati Chuprat. „The challenges of Extract, Transform and Loading (ETL) system implementation for near real-time environment“. In: *2017 International Conference on Research and Innovation in Information Systems (ICRIIS)*. IEEE. 2017, S. 1–5 (siehe S. 29).

[See21] Dhamotharan Seenivasan. „ETL in a World of Unstructured Data: Advanced Techniques for Data Integration“. In: *Journal Homepage: http://www. ijmra. us* 11.01 (2021) (siehe S. 4, 29–31).

[Shi+14] Mi-Hae Shin, Eui-Jeong Kim, Su-Seok Seo und Young-Chul Kim. „Implementation of Plagiarism Analysis System through Digital Conversion Processing of Sound Source“. In: *International Journal of Multimedia and Ubiquitous Engineering* 9.10 (2014), S. 33–44 (siehe S. 21).

[SKH23] Atefe Sedaghat, Masood Jafari Kang und Maryam Hamidi. „A Heuristic ETL Process to Dynamically Separate and Compress AIS Data“. In: *2023 Systems and Information Engineering Design Symposium (SIEDS)*. IEEE. 2023, S. 159–164 (siehe S. 31, 32).

[SNR+20] J Sreemathy, S Nisha, Gokula Priya RM u. a. „Data integration in ETL using TALEND“. In: *2020 6th international conference on advanced computing and communication systems (ICACCS)*. IEEE. 2020, S. 1444–1448 (siehe S. 12).

[Sou+19] Manel Souibgui, Faten Atigui, Saloua Zammali, Samira Cherfi und Sadok Ben Yahia. „Data quality in ETL process: A preliminary study“. In: *Procedia Computer Science* 159 (2019), S. 676–687 (siehe S. 8, 9).

[SS20] Ian Scoones und Andy Stirling. *The politics of uncertainty: Challenges of transformation*. Taylor & Francis, 2020 (siehe S. 35).

[Sub+97] SR Subramanya, Rahul Simha, Bhagirath Narahari und Abdou Youssef. „Transform-based indexing of audio data for multimedia databases“. In: *Proceedings of IEEE International Conference on Multimedia Computing and Systems*. IEEE. 1997, S. 211–218 (siehe S. 31).

[SVS05] Alkis Simitsis, Panos Vassiliadis und Timos Sellis. „Optimizing ETL processes in data warehouses“. In: *21st International Conference on Data Engineering (ICDE'05)*. Ieee. 2005, S. 564–575 (siehe S. 8).

[TC00] George Tzanetakis und Perry Cook. „Audio information retrieval (AIR) tools“. In: *Proc. International Symposium on Music Information Retrieval*. 2000 (siehe S. 43).

[The+16] Vasileios Theodorou, Alberto Abelló, Wolfgang Lehner und Maik Thiele. „Quality measures for ETL processes: from goals to implementation“. In: *Concurrency and computation: practice and experience* 28.15 (2016), S. 3969–3993 (siehe S. 10).

[Ver+16] Jai Prakash Verma, Smita Agrawal, Bankim Patel und Atul Patel. „Big data analytics: Challenges and applications for text, audio, video, and social media data“. In: *International Journal on Soft Computing, Artificial Intelligence and Applications (IJSCAI)* 5.1 (2016), S. 41–51 (siehe S. 16).

[Ver04] Barry Vercoe. „Audio-pro with multiple DSPs and dynamic load distribution“. In: *BT Technology Journal* 22.4 (2004), S. 180–186 (siehe S. 37, 39).

[VSS02] Panos Vassiliadis, Alkis Simitsis und Spiros Skiadopoulos. „Conceptual modeling for ETL processes“. In: *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*. 2002, S. 14–21 (siehe S. 7, 11).

[Wat13] John Watkinson. *Art of digital audio*. Routledge, 2013 (siehe S. 14).

[WKH23] Stefan Wagenpfeil, Paul Mc Kevitt und Matthias Hemmje. „Smart Multimedia Information Retrieval“. In: *Analytics* 2.1 (2023), S. 198–224 (siehe S. 43).

[XHY17] Yong Xiang, Guang Hua und Bin Yan. *Digital audio watermarking: fundamentals, techniques and challenges*. Springer, 2017 (siehe S. 39, 41).

[Yan+17] Chaowei Yang, Qunying Huang, Zhenlong Li, Kai Liu und Fei Hu. „Big Data and cloud computing: innovation opportunities and challenges“. In: *International Journal of Digital Earth* 10.1 (2017), S. 13–53 (siehe S. 39).

[YR22] YK Yee und M Raheem. „Predicting music popularity using spotify and youtube features“. In: *Indian Journal of Science and Technology* 15.36 (2022), S. 1786–1799 (siehe S. 26).

[ZSS+11] Udo Zölzer, Xavier Serra, Mark Sandler u. a. *Digital audio effects*. 2011 (siehe S. 35).

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Bachelorarbeit selbstständig und ohne unzulässige Hilfe verfasst habe. Alle Stellen, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, sind als solche kenntlich gemacht.

Ich erkläre weiterhin, dass die vorliegende Arbeit in gleicher oder ähnlicher Form noch nicht im Rahmen eines anderen Prüfungsverfahrens eingereicht wurde.

Rostock, den 30. September 2024

Ghaith Ishak