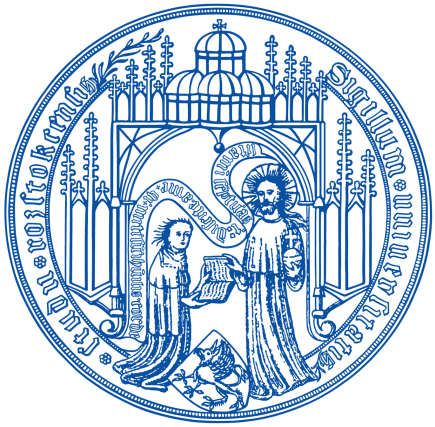
Konzeption eines ETL-Prozesses zur Ermittlung von Kennzahlen aus Videos als Vorbereitungsschritt einer Plagiatserkennung

Bachelorarbeit

Universität Rostock
Fakultät für Informatik und Elektrotechnik
Institut für Informatik



vorgelegt von: Bandar Al-Askari
Matrikelnummer: 219204763
geboren am: 07. Juni 1995 in Sana'a
Erstgutachter: Dr.-Ing. Hannes Grunert
Zweitgutachter: Florian Rose, M.Sc
Abgabedatum: 19. April 2024

Danksagung

An dieser Stelle möchte ich die Gelegenheit nutzen, um all jenen zu danken, die mich während der Erstellung meiner Abschlussarbeit unterstützt und begleitet haben.
Ein besonderer Dank gilt meinen Eltern, die mich stets ermutigt und unterstützt haben. Ihre Geduld, Liebe und ihr Verständnis waren während dieser Zeit mein Anker und haben es mir ermöglicht, mich voll und ganz auf meine Arbeit zu konzentrieren.
Ich möchte auch meinem Betreuer, Herrn Dr.-Ing. Hannes Grunert, meinen tiefsten Dank aussprechen. Seine fachliche Expertise und seine wertvollen Ratschläge waren entscheidend für die erfolgreiche Fertigstellung dieser Arbeit. Seine Unterstützung und sein Engagement waren für mich von unschätzbarem Wert.
Ich bin allen, die mich auf diesem Weg begleitet haben, zutiefst dankbar und erkenne ihre Unterstützung mit großer Wertschätzung an.

Abstract

In the modern digital era, countless videos flood online platforms and social networks at an accelerating pace. Plagiarism, the unauthorized replication of content, stands as a significant societal concern within the domain of internet and multimedia technologies. This study proposes a method for discerning plagiarism by extracting distinctive attributes from videos. Our approach hinges on local features to differentiate between individual videos and those archived in the database. It involves reviewing existing techniques for detecting video plagiarism, evaluating them using relevant performance measures, and pinpointing existing challenges. This thesis outlines an Extract, Transform, Load (ETL) process for uncovering plagiarism in videos. It entails extracting pertinent data from videos including audio, frames, and text, followed by data transformation to derive a dependable set of features for video comparison. Finally, the extracted and transformed data are stored in a suitable database, facilitating rapid comparisons between new and existing videos.

Zusammenfassung

In der heutigen digitalen Welt wird eine enorme Anzahl an Videos auf Videoplattformen und in sozialen Medien mit zunehmender Geschwindigkeit hochgeladen. Plagiat, oder das unautorisierte Kopieren von Inhalten, ist ein ernstes gesellschaftliches Problem im Bereich des Internets und der Multimediatechnologien. In dieser Forschungsarbeit schlagen wir eine Methode vor, um Merkmale aus Videos zu extrahieren und Plagiate zu erkennen. Dieses Framework stützt sich auf lokale Merkmale, um einzelne Videos im Vergleich zu in der Datenbank gespeicherten Videos zu identifizieren. Es erfordert eine Untersuchung der aktuellen Methoden zur Erkennung von Plagiaten in Videos, einen Vergleich dieser Methoden anhand geeigneter Leistungskennzahlen und die Identifizierung offener Herausforderungen. Diese Arbeit stellt einen ETL-Prozess zur Erkennung von Plagiaten in Videos vor. Sie extrahiert relevante Daten aus dem Video wie Audio, Frames und Text. Dann werden die Daten transformiert, um einen zuverlässigen Satz von Merkmalen zu extrahieren, der bei der Vergleich von Videos verwendet wird. Am Ende werden die extrahierten und transformierten Daten in einer geeigneten Datenbank gespeichert, die schnelle Vergleiche zwischen einem neuen Video und den bereits hinzugefügten Videos ermöglicht.

Inhaltsverzeichnis

Abbildungsverzeichnis

Liste der Abkürzungen

Abkürzung	Vollständige Bezeichnung
AI	Artificial Intelligence
API	Application Programming Interface
CNN	Convolutional Neural Network
DB	Database
ER	Entity-Relationship
ETL	Extract, Transform, Load
FK	Foreign Key
FLV	Flash Video
GAN	Generative Adversarial Network
GPL	GNU General Public License
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
ICPR	International Conference on Pattern Recognition
KNN	k-Nearest Neighbors
M4V	MPEG-4 Video
MJPEG	Motion JPEG
MKV	Matroska Video
MOV	QuickTime Movie
MP3	MPEG-1 Audio Layer III
MP4	MPEG-4 Part 14
MPEG	Moving Picture Experts Group
MPG	MPEG Video
OCR	Optical Character Recognition
RNN	Recurrent Neural Network
SIFT	Scale-Invariant Feature Transform
SURF	Speeded-Up Robust Features
SVM	Support Vector Machine
VCA	Video Content Analysis
VCD	Video Compact Disc
VFW	Video for Windows
VGN	Video Game Network
VLC	VideoLAN Client
VOB	DVD Video Object
VOC	Video Object Classification
VSD	Video Structure Description
WAV	Waveform Audio File Format
WMV	Windows Media Video

Kapitel 1

Einleitung

In der heutigen digitalen Welt wachsen Daten in einem bedeutenden und raschen Tempo. Videos, als eine der wichtigsten Datenformen [Smi19, OBAB18] , enthalten reichhaltige Informationen, die extrahiert und in einer Vielzahl von Anwendungen genutzt werden können. Eine solche Anwendung ist die Plagiatserkennung, ein Thema von Bedeutung in vielen Bereichen, einschließlich Bildung und wissenschaftlicher Forschung.
Das Hauptziel dieses Projekts ist es, einen ETL-Prozess (Extrahieren, Transformieren, Laden) zu entwerfen und zu implementieren, um Indikatoren aus Videos als vorbereitenden Schritt für die Plagiatserkennung zu extrahieren. Die extrahierten Daten werden verwendet, um die Videos zu analysieren und Ähnlichkeiten zu erkennen, die auf Plagiate hindeuten könnten. In den letzten Jahren hat mit dem raschen Fortschritt der Videokamera-Technologie das Volumen professioneller Videos und solcher, die von Nutzern erstellt wurden, enorm gestiegen [Smi19, OBAB18] . Täglich werden Zehntausende von Videos erstellt, hochgeladen und veröffentlicht. Angesichts dieser riesigen Menge an visuellen Daten stellt ein großer Teil davon Kopien oder Teil-Kopien dar. Dies wirft zunehmend Bedenken hinsichtlich der Probleme des geistigen Eigentumsrechts aufgrund der geringen Kosten für das Kopieren eines Videos (oder eines Teils davon) und dessen breite Verbreitung im Internet auf. Infolgedessen haben effektive und effiziente Lösungen zur Erkennung von Videokopien, die darauf abzielen, automatisch Kopien in großen Videosammlungen zu erkennen, erhebliches Forschungsinteresse geweckt.
Diese Arbeit befasst sich mit der Problematik der Erkennung und Lokalisierung von Kopien in Videoclips. Die Erkennung von Kopien in Videos stellt aus mehreren Gründen eine Herausforderung dar. Erstens ist die Repräsentation eines Videoframes aufgrund der komplexen Veränderungen des Videoinhalts schwierig. Zweitens erhöht die Möglichkeit von teilweisen oder vollständigen Duplikaten des Kopiermusters die Komplexität der Situation. Drittens können Kopienausschnitte von wenigen Sekunden bis zu vielen Stunden andauern. Tiefe Modelle waren in vielen verschiedenen Anwendungen, einschließlich der Verarbeitung natürlicher Sprache, der Computer Vision und der maschinellen Übersetzung, unglaublich erfolgreich [Bot24, BvE00].
Insbesondere wurden Convolutional Neural Networks (CNNs) in den letzten Jahren als eines der Hauptwerkzeuge im Bereich der Computer Vision angesehen. Sie wurden mit großem Erfolg in einer Vielzahl von Anwendungen eingesetzt, einschließlich der Erkennung von Kopien in Videos.
Es gibt mehrere kritische Fragen, die weiterer Forschung bedürfen, darunter, wie man wiederholte Clips zwischen Videos erkennt und lokalisiert. Diese Details und mehr werden wir in den folgenden Abschnitten besprechen. Diese Methoden betrachten die zeitliche Konsistenz innerhalb von Videoclips und können die Auswirkungen von Unstimmigkeiten bei Keyframes eliminieren. In dieser Forschung schlagen wir einen neuen Ansatz vor, um Kopien in Videos mithilfe von deep CNN-Merkmalen zu erkennen und zu lokalisieren. Die Motivation für diesen Ansatz liegt in der gemeinsamen Berücksichtigung von tiefen CNN-Merkmalen, die auf visueller Ähnlichkeit und zeitlicher Konsistenz basieren. Das Design von CNNs wird traditionell als Merkmalsextraktor auf der Bildebene verwendet.

Abbildung 1.1 veranschaulicht unseren vorgeschlagenen Ansatz. Er besteht aus:

1. **Metadaten:** Angezeigt an der Spitze des Diagramms, beinhalten sie „Name“, „Datum“, „Dauer“ usw. Diese Daten liefern Informationen über das Video, wie seinen Namen, Erstellungsdatum und Dauer.
2. **Video:** Das Video wird in seine Grundkomponenten zerlegt: Bilder und Audio.
3. **Transformation:** Nach der Extraktion der Bilder und des Audios können sie mithilfe verschiedener neuronaler Netze wie CNN, GAN, oder anderer Methoden in Features umgewandelt.
4. **Suche in DB:** Schließlich wird die Datenbank unter Verwendung der Metadaten und extrahierten Merkmale durchsucht, um die erforderlichen Informationen abzurufen.

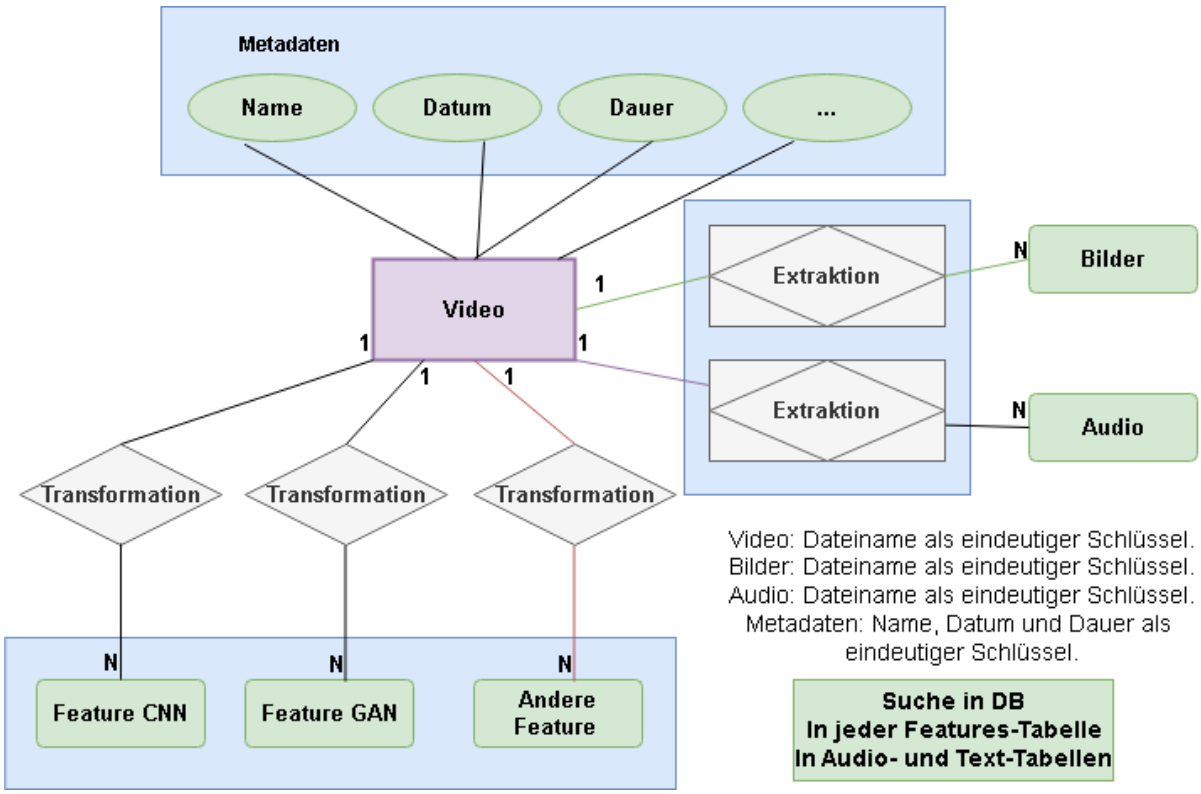


Abbildung 1.1: Das Entity-Relationship-Modell der Videodaten (Metadaten und Merkmale).

Die Hauptbeiträge des vorgeschlagenen Ansatzes sind wie folgt:

1. Es werden deep CNN-Merkmale verwendet, um den visuellen Bildinhalt zu kodieren.
2. Wir etablieren einen vollständigen Rahmen zur Erkennung und Lokalisierung von Kopien in Videos. Er funktioniert besser als andere Methoden, die auf handgefertigten Merkmalen basieren. Handgefertigte Merkmale sind Charakteristika oder Attribute, die von Menschen definiert und ausgewählt werden, um Daten darzustellen. Diese Merkmale basieren auf menschlicher Erfahrung und Wissen und erfordern oft ein tiefes Verständnis des jeweiligen Fachgebiets. Beispielsweise können in der Formerkennung manuell entworfene Merkmale wie Länge, Breite, Umfang, Fläche und geometrische Proportionen festgelegt werden. Diese Merkmale werden von Menschen definiert und verwendet, um

Formen im Modell zu repräsentieren. Im Gegensatz dazu werden in Modellen, die auf tiefem Lernen basieren, Merkmale automatisch aus den Daten während des Trainingsprozesses gelernt. Das bedeutet, dass das Modell selbst herausfindet, welche Merkmale am nützlichsten sind, um das gewünschte Ziel zu erreichen, anstatt sich auf zuvor von Menschen bestimmte Merkmale zu verlassen.

1.1 Das wissenschaftliche Problem

Der Prozess der Plagiatserkennung ist ein wichtiges Thema im akademischen und Forschungsschreiben. Eine der Herausforderungen besteht darin, mit Videoclips als Quelle von Texten und Informationen umgehen zu können. Das wissenschaftliche Problem liegt in der Notwendigkeit, einen effektiven ETL-Prozess zu entwerfen, um wichtige Indikatoren aus Videoclips als vorläufigen Schritt zur Plagiatserkennung zu extrahieren.

1.2 Bedeutung der Forschung

Diese Studie zielt darauf ab, mehrere Hauptziele zu erreichen:

1. Entwurf eines ETL-Prozesses: Einen effektiven ETL-Prozess entwerfen, um wichtige Indikatoren aus Videoclips auf eine effiziente und zuverlässige Weise zu extrahieren.
2. Analyse des visuellen Inhalts: Techniken und Herausforderungen im Zusammenhang mit der Analyse und Extraktion wichtiger Informationen und Schlüsselindikatoren aus Videoinhalten erforschen und analysieren.
3. Anwendungen der Plagiatserkennung: Bewertung der Machbarkeit der Verwendung der aus Videoclips extrahierten Indikatoren in Plagiatserkennungsprozessen und Analyse der Leistungsergebnisse.
4. Systementwurf und Implementierung: Entwurf und Entwicklung des notwendigen Programms zur Implementierung des ETL-Prozesses in einer experimentellen Umgebung und Bewertung seiner Leistung und Wirksamkeit.

1.3 Forschungsfragen

Die Forschungsfragen, die wir in diesem Projekt beantworten möchten, umfassen:

1. Wie können wir effektive Indikatoren aus Videos extrahieren, die für die Plagiatserkennung verwendet werden können?
2. Wie können wir einen effektiven ETL-Prozess entwerfen und implementieren, um diese Indikatoren zu extrahieren?
3. Wie können wir die extrahierten Daten für die Plagiatserkennung in Videos verwenden?

1.4 Aufbau dieser Abschlussarbeit

Die Bachelorarbeit ist systematisch in sieben Hauptkapitel gegliedert:

1. **Einleitung**: Das erste Kapitel umfasst den Projekthintergrund und seine Bedeutung, die Ziele und Forschungsfragen sowie den Aufbau der Arbeit.
2. **Theoretische Grundlagen**: Das zweite Kapitel behandelt die Grundlagen und die Bedeutung von ETL-Prozessen, Techniken und Herausforderungen der Videoanalyse, Kennzahlen im Kontext der Videoanalyse, Grundlagen der Plagiatserkennung sowie ethische Überlegungen und Datenschutz.

3. **Literaturübersicht**: Das dritte Kapitel überprüft die bestehende Literatur, die für unsere Forschung relevant ist.
4. **Methodik**: Das vierte Kapitel umfasst einen Überblick über den vorgeschlagenen ETL-Prozess, Datensammlung (Extraktion), Datenverarbeitung (Transformation) und Datenintegration (Laden).
5. **Systemdesign und -architektur**: Das fünfte Kapitel umfasst die technologische Wahl, Entwicklungs-umgebung und Werkzeuge, Implementierung des ETL-Prozesses, Testen und Validierung.
6. **Ergebnisse**: Das sechste Kapitel umfasst die Ergebnisse der Videoanalyse, die Wirksamkeit der Identifizierung von Kennzahlen und eine Diskussion der Ergebnisse.
7. **Zusammenfassung und Ausblicke**: Das siebte Kapitel umfasst eine Übersicht über die wichtigsten Ergebnisse, Einschränkungen und Herausforderungen, potenzielle Anwendungen und zukünftige Forschungsrichtungen sowie abschließende Bemerkungen.

Es gibt noch zwei Hilfskapitel, die zusätzliche Informationen, wie den Quellcode oder die Quellen, beinhalten:

1. **Anhänge**: Beinhaltet den Quellcode und Beispiele sowie zusätzliche Grafiken.
2. **Referenzen**: Beinhaltet eine Liste der in der Arbeit verwendeten Referenzen.

Kapitel 2

Theoretische Grundlagen

In diesem Kapitel werden die grundlegenden Konzepte von ETL-Prozessen vorgestellt. Es wird erörtert, wie diese Prozesse in der Datenverarbeitung eingesetzt werden, um Daten effizient zu extrahieren, zu transformieren und zu laden. Die Bedeutung dieser Prozesse für die Analyse großer Datensätze wird hervorgehoben, und die Videoanalyse und deren Anwendung in der Plagiatserkennung befasst. Zudem wird auf die Bedeutung des Datenschutzes und ethischer Überlegungen in diesem Kontext eingegangen.

2.1 ETL-Prozesse: Grundlagen und Bedeutung

Der ETL-Prozess (Extract, Transform, Load) ist ein wesentlicher Bestandteil der Datenintegration. Er ermöglicht, Daten aus verschiedenen Quellen zu extrahieren, zu transformieren und in ein einheitliches Datenlager oder in einen anderen Zielspeicher zu laden [Wut22, Wik23] . Dieser Prozess spielt eine entscheidende Rolle bei der Datenerschließung und der Sicherstellung der Datenqualität, da er gewährleistet, dass nur konsistente und bereinigte Daten in Data Warehouses und BI-Tools gelangen [Gam23]. Der ETL-Prozess ist von zentraler Bedeutung für Datenanalyse, maschinelles Lernen und Business Intelligence, da er die Grundlage für Datenanalyse- und Machine-Learning-Workflows bildet. Der ETL-Prozess kann jedoch auch erhebliche Komplexität und operative Probleme mit sich bringen, insbesondere bei unsachgemäß gestalteten ETL-Systemen [Wut23]. Die Vorteile eines ETL-Prozesses liegen in der Schaffung einer zentralen Datenbasis für spätere Analysen der Unternehmens- und Geschäftsdaten. Durch die Extraktion, Säuberung und Transformation ermöglicht der ETL-Prozess die Integration von Daten aus verschiedenen Quellen und schafft somit die Grundlage für aussagekräftige Analysen und Berichte [Gam23, Wut23].

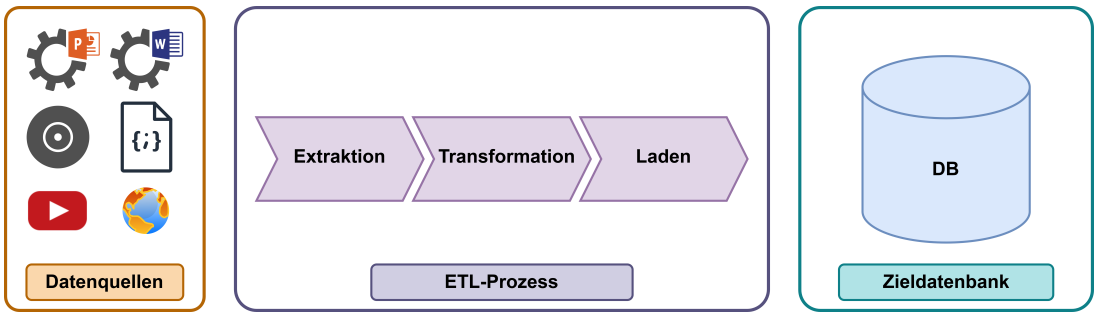


Abbildung 2.1: Schematische Darstellung des ETL-Prozesses.

Abbildung 2.1 zeigt den Prozess der Datenintegration, bekannt als ETL-Prozess. Dieser Prozess setzt sich aus den drei Hauptkomponenten, Extraktion, Transformation und Laden zusammen. Im Folgenden wird eine kurze Erläuterung der einzelnen Komponenten dieses Prozesses gegeben.

2.1.1 Extract/Extraktion

Dieser Schritt beinhaltet die Akquisition von Rohdaten aus diversen Datenquellen. In diesem Schritt erfolgt die Entnahme und Konsolidierung von Daten, die in unterschiedlichen Formaten und Strukturen vorliegen. Zunächst werden die Konventionen für Verbindungen zu verschiedenen Datenquellen sowie die Modalitäten der Datenübertragung festgelegt. Ein wesentlicher Aspekt ist die Definition des Aktualisierungsrhythmus, welcher sich in synchrone und asynchrone Extraktion gliedert. Die synchrone Extraktion ermöglicht eine kontinuierliche Datenaktualisierung, was jedoch zu erhöhter Netzwerkauslastung führt. Daher greifen einige Organisationen auf die asynchrone Extraktion zurück, die in ressourcenschonenden Zeitfenstern (beispielsweise nachts) durchgeführt wird [Gam23]. Die Extraktion variiert ferner in ihrem Umfang:

- Statistische Extraktion: Erstellung eines vollständigen Abbilds der Datenbank (relevant für Erstbefüllung und Wiederherstellbarkeit) [Gam23].
- Inkrementelle Extraktion: Es werden nur die Änderungen gelesen, die zwischen der aktuellen Extraktion und der letzten Extraktion stattgefunden haben [Gam23].

2.1.2 Transform/Transformation

Im Schritt der Transformation werden die extrahierten Daten umgewandelt, um sie in die geforderte Struktur und das erforderliche Format der zentralen Ziel-Datenbank zu überführen. Dieser Prozess beinhaltet häufig Aktivitäten wie die Normalisierung, Bereinigung, Aggregation und Anreicherung der Daten oder Maßnahmen wie [Gam23]:

- Harmonisierung unterschiedlicher Datentypen.
- Umwandlung oder Neukodierung.
- Standardisierung von Textformaten, Zeichenketten, Datumsangaben.
- Umrechnung von Maßeinheiten.

Sowohl strukturelle als auch inhaltliche Diskrepanzen werden mittels eines Korrekturschemas behandelt, das auch Inkonsistenzen, Redundanzen, veraltete Daten und fehlende Werte berücksichtigt [Gam23].

2.1.3 Load/Laden

Das Laden ist der letzte Schritt im ETL-Prozess und es gewährleistet, dass die transformierten Daten aus dem Zwischenspeicher, der „Staging Area“, in das verwendete Data Warehouse transferiert werden. In dieser Phase ist das Data Warehouse normalerweise gesperrt, um die Risiken von fehlerhaften Analysen zu minimieren. Bei Aktualisierungen können existierende Datensätze entweder überschrieben oder als neue Einträge hinzugefügt werden, wobei Änderungen protokolliert werden können. Nach dem Laden der Daten in das Data Warehouse erfolgt zumeist eine Aktualisierung der Analyse- und Business Intelligence-Systeme [Gam23].

2.2 Kennzahlen im Kontext der Videoanalyse

Metadaten über ein Video können eine Vielzahl von Informationen enthalten, die Details über den Inhalt, die Erstellung und technische Aspekte des Videos umfassen. In unserem Projekt benötigen wir strukturelle Metadaten, die das Video besser durchsuchbar und anordbar machen. Sie sind wesentlich dafür, dass Suchmaschinen die Videodatei und die damit verbundenen Textdetails erkennen können. Dazu gehören Titel, Beschreibung, Datum, Länge, Codec, Bildrate, Auflösung und Autor gemäß den IPTC-Standards [Int23], welche viele Arten von Metadaten umfassen, die wir hier nicht benötigen. Hier sind einige typische Metadaten, die wir für unsere Arbeit benötigen :

- **Titel**: Der Titel eines Videos ist eine kurze Beschreibung des Inhalts des Videos. Er wird oft im Dateinamen des Videos oder in der Videobeschreibung gespeichert.
- **Beschreibung**: Die Beschreibung eines Videos ist eine längere Beschreibung des Inhalts des Videos.
- **Dauer**: Die Dauer eines Videos ist die Länge des Videos. Sie wird in der Regel in Sekunden oder Millisekunden angegeben.
- **Auflösung**: Die Auflösung eines Videos ist die Anzahl der Pixel pro Zeile und Spalte. Sie wird in der Regel in Pixeln angegeben, z. B. 1920 x 1080.
- **Framerate**: Die Framerate eines Videos ist die Anzahl der Bilder pro Sekunde. Sie wird in der Regel in Bildern pro Sekunde (fps) angegeben, z. B. 24 fps, 30 fps, 60 fps.
- **Dateiformat**: Das Dateiformat eines Videos ist der Standard, in dem das Video gespeichert ist. Es gibt viele verschiedene Dateiformate für Videos, z. B. MP4, AVI, MOV, WMV.
- **Ersteller**: Der Ersteller eines Videos ist die Person oder Organisation, die das Video erstellt hat.
- **Veröffentlichungsdatum**: Das Veröffentlichungsdatum eines Videos ist das Datum, an dem das Video zum ersten Mal veröffentlicht wurde.
- **Urheberrecht**: Das Urheberrecht eines Videos ist das Recht, das Video zu kopieren, zu verbreiten und zu verwenden.
- **Standortinformationen**: Die Standortinformationen wie Aufnahmeort oder Geokoordinaten.

Neben den oben genannten Metadaten können Videos auch weitere Metadaten enthalten, z. B. Tags, Schlagwörter, Sprache des Videos, Untertitel, Transkript-Informationen, Informationen zu verwendeten Bildern oder Musikstücken oder Informationen über die Kamera, die verwendet wurde, um das Video zu erstellen, oder Informationen über die Personen, die im Video zu sehen sind.
Tabelle 2.1 veranschaulicht Metadaten eines Videos sowie detaillierte Angaben dazu.

Tabelle 2.1: Metadaten über ein Video

Metadaten	**Details**
Titel	Das Videospiel
Beschreibung	Ein Videospiel-Trailer
Dauer	2:00 Minuten
Auflösung	1920 x 1080
Framerate	24 fps
Dateiformat	MOV
Ersteller	Sony Interactive Entertainment
Veröffentlichungsdatum	08.02.2023
Urheberrecht	(c) Sony Interactive Entertainment 2023

Die Metadaten eines Videos, lassen sich über mehrere Methoden abrufen. Eine Methode ist die Verwendung eines Video-Metadaten-Editors. Ein Video-Metadaten-Editor ist ein Programm, das ermöglicht, Metadaten für Videos zu bearbeiten oder zu erstellen. Eine weitere Methode ist die Verwendung einer Programmschnittstelle, die es ermöglicht, programmatisch auf die Metadaten des Videos zugreifen.
Diese Metadaten können entweder in einer externen Datenbank oder einem Dateisystem gespeichert werden. Sie spielen eine wichtige Rolle bei der Verwaltung, Sortierung, Suche und Archivierung von Videomaterial sowie bei der Gewährleistung von Urheberrechtskonformität und Zugänglichkeit.

2.3 Grundlagen der Plagiatserkennung

In diesem Abschnitt wird der Begriff Plagiatünd mehrere Verfahren der Plagiatserkennung definiert, dabei werden Entwicklung, Wirksamkeit und Grenzen der Plagiatserkennungsverfahren beleuchtet.

Plagiate: sind die unerlaubte Verwendung fremder Werke ohne angemessene Quellenangabe, und stellen eine große Herausforderung im akademischen und kreativen Bereich dar. Durch die Digitalisierung sind Informationen leichter zugänglich geworden, was einerseits das Plagiieren erleichtert, andererseits aber auch die Aufdeckung von Plagiaten erschwert [Wik24].

Textbasierte Plagiatserkennung: Herkömmliche Verfahren zur Plagiatserkennung konzentrieren sich in erster Linie auf den Textinhalt. Tools wie Turnitin [Tur23] und Copyscape [Ltd23] vergleichen den eingereichten Text mit einer umfangreichen Datenbank von akademischen Arbeiten, Büchern und Internetquellen. Diese Systeme verwenden Algorithmen, die Ähnlichkeiten in Formulierung, Struktur und Wortwahl analysieren. Einige fortgeschrittene Tools erkennen sogar paraphrasierte Inhalte, indem sie die kontextuelle Bedeutung verstehen, anstatt sich nur auf den direkten Textvergleich zu verlassen [BML03].

Multimedia-basierte Plagiatserkennung: Mit der zunehmenden Verbreitung digitaler Inhalte, insbesondere in multimedialen Formaten wie Videos und Bildern, wird eine immer ausgefeiltere Methode zur Plagiatserkennung benötigt. Hierbei kommen Techniken wie digitale Fingerabdrücke und Wasserzeichen zum Einsatz, um Originalinhalte aufzuspüren und zu identifizieren. Beim digitalen Fingerabdruck wird ein eindeutiger Code oder 'Fingerabdruck' für ein Medium erstellt. Dieser kann mit einer Datenbank verglichen werden, um Übereinstimmungen oder Änderungen zu finden. Beim Wasserzeichen hingegen wird ein unsichtbares Zeichen oder Muster in den Datenträger eingebettet, das auf den ursprünglichen Urheber oder Besitzer hinweist [ZYJ06, Lu09].

Maschinelles Lernen und künstliche Intelligenz in der Plagiatserkennung: Die Integration von künstlicher Intelligenz und maschinellem Lernen hat die Plagiatserkennung deutlich verbessert. Diese Technologien unterstützen bei der Mustererkennung, der Verarbeitung natürlicher Sprache und sogar bei der Analyse von Stimmungen. Dadurch wird ein differenzierteres Verständnis des Inhalts und seiner Originalität ermöglicht. KI-Algorithmen können aus großen Datenmengen lernen und werden immer besser darin, nicht nur offensichtliche Kopien, sondern auch subtilere Formen von Plagiaten zu erkennen. Ein Beispiel sind Ideenplagiate, bei denen eine Idee kopiert wird, ohne den ursprünglichen Autor zu nennen [CTKC21, SS14, HJK+19].

Kapitel 3

Literaturüberblick

In diesem Kapitel wird ein Überblick über die einschlägige Forschung gegeben, die durchgeführt wurde, um das vorliegende Problem anzugehen. Zunächst wird erläutert, wie relevante Studien identifiziert wurden. Anschließend werden vielversprechende Ergebnisse aus diesen Untersuchungen diskutiert. Es erfolgt ein Vergleich der verschiedenen Resultate, um den effektivsten Ansatz zu bestimmen. Schließlich werden einige Methoden ausgewählt, um das Problem anzugehen und zu bewältigen.

3.1 Suchprozess

Dieser Abschnitt der Arbeit bietet eine detaillierte Überprüfung des Literatursuchprozesses, der verwendet wurde, um Quellen und Forschungen im Zusammenhang mit dem Problem des Kopierens und Stehlens von Videos aus dem Internet und der Erkennung von Videoplagiaten zu identifizieren. Die Methode der Literatursuche wurde auf eine klare und strukturierte Weise durchgeführt, was den Zugang zu den identifizierten relevanten Werken erleichtert.

3.1.1 Problemformulierung

Eine klare Definition des Problems ist ein kritischer erster Schritt in der Forschung, da sie die Grundlage für die Studie bildet und als Leitfaden für die Auswahl geeigneter Methoden und Techniken dient. In dieser Arbeit beinhaltet die Erkennung von Videokopien im Kontext der Problemformulierung typischerweise die Definition eines Frameworks oder eines Satzes von Methoden zur Identifizierung von Fällen, in denen ein Video oder Teile davon dupliziert und ohne Genehmigung verwendet wurden. Dieser Prozess beinhaltet typischerweise die Entwicklung von Algorithmen zur Erkennung von Ähnlichkeiten zwischen einem gegebenen Video und einer Datenbank existierender Videos. Diese Algorithmen müssen verschiedene Formen der Transformation und Degradation berücksichtigen, die ein kopiertes Video durchlaufen haben könnte, wie Kompression, Zuschneiden oder Formatänderungen. Die Herausforderung besteht darin, ein System zu schaffen, das sowohl genau in der Erkennung von Kopien als auch effizient in der Verarbeitung großer Mengen von Videodaten ist.

3.1.2 Suchanfragen definieren

Nach einer klaren Definition des Problems ist es entscheidend, geeignete Suchanfragen zu identifizieren, um relevante Literatur und Forschungsmaterial zum Problem effektiv zu lokalisieren. Die sorgfältige Auswahl von Suchanfragen, die genau mit dem Problem und seinem Umfang übereinstimmen, ist wesentlich, um Präzision und Relevanz zu gewährleisten. Vor diesem Hintergrund haben wir die folgenden Suchanfragen festgelegt:

- ETL-Prozess
- Erkennung von Videokopien
- Daten aus Videos extrahieren
- Extrahieren von Daten aus Videos mithilfe neuronaler Netze
- Video data analysis
- Erkennung von Video-Plagiaten
- Vorbereitungsschritt einer Plagiatserkennung
- Methoden zur Erkennung von Videoplagiaten
- Plagiatserkennung auf der Grundlage von Algorithmen des Deep Learning
- Erkennung von Kopien in Videos mit Hilfe von deep-CNN Merkmalen

Diese Suchanfragen sind für die Forschung relevant, weil sie darauf abzielen, einen Prozess zur Ermittlung von Kennzahlen aus Videos als vorbereitenden Schritt zur Plagiatserkennung zu identifizieren. Zusätzlich enthalten diese Suchen spezielle Begriffe wie „Methoden zur Erkennung von Videoplagiaten“ und „Extrahieren von Daten aus Videos mithilfe neuronaler Netze“ die für das Forschungsproblem bedeutend sind.

3.1.3 Suchmaschinen auswählen

Nach der Identifizierung der für die Forschung relevanten Begriffe wurde die Auswahl wichtiger Suchmaschinen, die in akademischen und forschungsbezogenen Kontexten bezüglich dieses Problems verwendet werden sollten, als wesentlich betrachtet, da sie erheblichen Einfluss auf die Qualität und Quantität der erzielten Ergebnisse haben können. Aus diesem Grund wurden die folgenden Suchmaschinen ausgewählt:

Tabelle 3.1: Übersicht der Suchmaschinen und deren Links

Suchmaschinen	**Link**
Google Scholar	`https://scholar.google.com/`
ACM Digital Library	`https://dl.acm.org/search/advanced`
Semantic Scholar	`https://www.semanticscholar.org/`
IEEE Xplore	`https://ieeexplore.ieee.org/`
arXiv	`https://arxiv.org/search/cs`
Papers with Code	`https://paperswithcode.com/`

Diese Suchmaschinen sind wertvolle Ressourcen für die Forschung in Informatik und maschinellem Lernen, da sie Forschern den Zugang zu einem umfangreichen Repository von wissenschaftlichen Materialien ermöglichen, die direkt mit ihren spezifischen Herausforderungen zusammenhängen.

3.1.4 Schnellfilterung

Nach Abschluss des initialen Suchprozesses gehen wir zu einer entscheidenden Phase über, die als Schnellfilterung bezeichnet wird. Diese Phase ermöglicht es uns, relevante Forschungsarbeiten und Literatur schnell und effizient zu identifizieren, indem wir rasch die Titel und Zusammenfassungen der gefundenen Papers und Artikel überprüfen. Das primäre Ziel dieses Schrittes ist es, die Relevanz des gefundenen Materials für das aktuelle Forschungsproblem zu bewerten. In dieser Phase konzentrieren wir uns speziell

auf Papers und Artikel, die unser Forschungsproblem direkt ansprechen und vielversprechende Ansätze zu seiner Lösung aufzeigen.
Die Schnellfilterung beginnt mit einer sorgfältigen Untersuchung der Titel und Zusammenfassungen jedes Papers oder Artikels. Der Titel sollte eine klare und prägnante Angabe zum Inhalt des Papers bieten, während die Zusammenfassung eine kompakte Übersicht über die Arbeit liefern sollte. Diese Zusammenfassung sollte ausreichend informativ sein, um zu bestimmen, ob die Forschung für unseren Problemkreis relevant ist. Nach der Analyse von Titel und Zusammenfassung vertiefen wir uns in die angewandten Beispiele des vorgeschlagenen Ansatzes. Diese Erkundung bietet ein tieferes Verständnis dafür, wie der Ansatz implementiert wird und welche Leistungsmerkmale er aufweist. Papers oder Artikel, die nicht mit unseren Problemkriterien übereinstimmen, werden umgehend verworfen.

3.1.5 Auswahl vielversprechender Ergebnisse

Nachdem die Mehrheit der Papers, die die Kriterien für diese Arbeit nicht erfüllen, ausgeschlossen wurde, beginnen wir mit dem Prozess der Auswahl vielversprechender Ergebnisse. Dies beinhaltet eine gründlichere Untersuchung der verbleibenden Werke, um diejenigen zu identifizieren, die das größte Potenzial haben, zur Zielerreichung der Arbeit beizutragen. Um dies zu erreichen, prüfen wir jedes Papers oder jeden Artikel sorgfältig, indem wir uns mit der Methodik, den Ergebnissen und den Schlussfolgerungen auseinandersetzen. Während wir jedes Papers sorgfältig bewerten, beurteilen wir gleichzeitig seine Stärken und Schwächen und berücksichtigen, wie effektiv es das Forschungsproblem angeht. Die ausgewählten Papers oder Artikel werden in Abschnitt 3.2 einer weiteren Analyse und Diskussion unterzogen, in dem wir ihre Feinheiten und Beiträge zum Fachgebiet vertiefen.

3.2 Wissenschaftliche Papers

In diesem Abschnitt werden die wissenschaftliche Papers erläutert und zusammengefasst, die durch Anwendung der im vorherigen Abschnitt vorgestellten Forschungsschritte erzielt wurden. Diese sind wie folgt:
Die Referenz [BML03] verweist auf ein Forschungspaper mit dem Titel „Are We Ready for Widespread Use of Deepfake Detection Tools?“ verfasst von Ruth Barrett und James Malcolm. Das Paper wurde im Jahr 2003 veröffentlicht. Das Paper behandelt das Thema der Deepfake-Erkennung, eine bedeutende Herausforderung im Bereich der digitalen Multimedia. Es präsentiert ein Erkennungssystem, das darauf abzielt, komplexe Deepfakes in Videos der realen Welt effizient und effektiv zu identifizieren. Das System erzielt gute Ergebnisse in einer umfangreichen Datenbank mit Bildern und Videos.

Dieser Abschnitt beschreibt das Paper [ZXL+18] mit dem Titel „Copy Detection in Video Based on Deep CNN Features and Graph-Based Sequence Matching“ von X. Zhang, Y. Xie, X. Luan und anderen. Das Paper behandelt das Thema der Kopienerkennung in Videos, was eine große Herausforderung in der Welt der digitalen Multimedia darstellt. Das Paper präsentiert ein System zur Erkennung von Videokopien, das darauf abzielt, komplexe reale Kopien in Videos effektiv und effizient zu erkennen. Das System erzielt gute Ergebnisse auf der groß angelegten Videotranskript-Datenbank, die im realen Szenario generiert wurde.

Die von El Mostafa, Hambi und Faouzia Benabbou vorgestellte Forschung mit dem Titel „Multi-Level Plagiarism Detection System Based on Deep Learning Algorithms“ [EMB19], weist darauf hin, dass das Thema Plagiatserkennung eine große Herausforderung in der Welt der digitalen Multimedia darstellt. Das Paper präsentiert ein System zur Plagiatserkennung, das darauf abzielt, komplexe reale Plagiate in Videos effektiv und effizient zu erkennen. Das System erzielt gute Ergebnisse auf der groß angelegten Videotext-Datenbank, die im realen Szenario generiert wurde.

Auf der 23. Internationalen Konferenz für Mustererkennung (International Conference on Pattern Recognition (ICPR)) im Dezember 2016 wurde ein Paper von Y. Zhang und X. Zhang mit dem Titel „Efficient video copy detection in real scenarios“ veröffentlicht [ZZ16]. Das Paper befasst sich mit dem Thema der Kopienerkennung in Videos unter realen Bedingungen, was eine große Herausforderung in der Welt der digitalen Multimedia darstellt. Das Paper präsentiert ein System zur Erkennung von Videokopien, das darauf abzielt, komplexe reale Kopien in Videos effektiv und effizient zu erkennen. Das System erzielt gute Ergebnisse auf einer umfangreichen Datenbank von Videotranskripten, die in einem realen Szenario im Jahr 2014 erstellt wurden, und misst das Gleichgewicht zwischen Effektivität und Effizienz.

Das Paper [SPC11] schlägt ein Framework für die Erkennung von Kopien in urheberrechtlich geschützten Videoinhalten oder beim Teilen von Videos online vor. Das Framework verwendet eine bekannte ordinale Anordnung und wird mit einer hierarchischen Bitmap-Indexstruktur kombiniert. Diese Arbeit verbessert ihre vorherigen Arbeiten durch die Einführung einiger Merkmale: ein neuer Matching-Algorithmus, das Entfernen schwarzer Ränder und die Auswahl einer angemessenen Anzahl von Blöcken. Mit diesen neuen Merkmalen hat das System Verbesserungen gezeigt, indem es die Rechenzeit und den Speicherplatz im Video-Matching-Prozess reduziert. Das vorgeschlagene System kann mehrere Video-Transformationen handhaben, einschließlich Farbänderungen, Helligkeit, Auflösung, Kontrast und Gamma, sowie das Hinzufügen von Rauschen, Unschärfe, Logo oder Textsignatur und Umklappen (das vorherige System konnte diese Bedingung nicht erkennen). Darüber hinaus erzielt das System durch das Entfernen schwarzer Striche vor dem Erstellen von Signaturen bessere Ergebnisse in Fällen von veränderten Seitenverhältnisse des Videos wie Letter- und Pillarbox oder auch Vergrößerung und geringfügigem Beschneiden/Verschieben.

In dem von Yeh, M.C., und Cheng, K.T. (2009) [YC09] präsentierten Paper wird vorgeschlagen, die Erkennung von Kopien in Videos als ein lokales Problem zwischen zwei Frame-Sequenzen zu betrachten und führt eine zweistufige Filterstrategie ein, die den Abgleichprozess erheblich beschleunigt. Zunächst empfehlen sie, jedes Frame-Rezept aus der Videodatenbank mithilfe eines anpassbaren Vokabularbaums zu indizieren. In diesem Schritt wird jedes Video als „Sack von Frames“ verarbeitet. Diese Architektur ermöglicht es dem Kernsystem, effektiv ein hierarchisches Matching zwischen Filmen durchzuführen und bietet zudem einen reichhaltigen Wortschatz für die Videodarstellung. Weiterhin bieten sie eine schnelle Sequenzabgleichstechnik an, die auf der Edit-Distanz basiert und unnötigerweise keine unähnlichen Frame-Paare vergleicht. In dieser Phase werden die Längen der verglichenen Sequenzen von einer quadratischen zu einer linearen Laufzeit umgewandelt. Tests, die am MUSCLE VCD-Benchmark durchgeführt wurden, demonstrieren die Wirksamkeit und Effizienz ihrer vorgeschlagenen Methodik. Im Vergleich zu den ursprünglichen Sequenzabgleichstechniken ist sie achtzehnmal schneller. Diese Methode kann für eine Vielzahl weiterer visueller Retrieval-Probleme, wie beispielsweise die Formensuche, verwendet werden. Unter Verwendung eines MPEG-7-Datensatzes zeigen die Forscher, dass die vorgeschlagene Strategie auch die Aufgabe der Formensuche erheblich beschleunigen kann.

Die Referenz [TJN21] bezieht sich auf ein in der „International Journal of Image, Graphics, and Signal Processing“ veröffentlichtes Forschungspaper mit dem Titel „Enhancing Performance Evaluation of Plagiarism Detection in Video Using Local Feature through SVM and KNN Algorithm“ verfasst von Ekta Thirani, Jayshree Jain und Vaibhav Narawade. Das Paper präsentiert eine Methode zur Merkmalsextraktion für die Erkennung von Plagiaten in Videos. Dieses Framework basiert auf lokalen Merkmalen, um Videos Bild für Bild mit in der Datenbank gespeicherten Videos zu vergleichen. Die Funktion zur Videoabrufung und -erkennung wird mit den Klassifizierungsalgorithmen SVM und KNN implementiert. Im ersten Schritt, wenn die Anfrage mit allen Trainingsdaten verglichen wird, berechnet die KNN die Entfernung zwischen der Anfrage und ihren Nachbarn und wählt die K-nächste Nachbarn aus. Es wird im zweiten Schritt für die Objekterkennung den SVM-Algorithmus angewendet. Die Genauigkeit dieser Methode beim Erkennen von plagiierten Videos beträgt 98%.

Diese Arbeiten bieten wertvolle Einblicke in das Feld der Plagiatserkennung und Video-Kopien-Identifikation und konzentrieren sich jeweils auf einzigartige Aspekte und unterschiedliche Methodologien.

3.3 Übersicht zu Algorithmen

In diesem Abschnitt werden einige neuronale Netze und Algorithmen erläutert, die durch die Anwendung der im vorherigen Abschnitt dargelegten Forschungsschritte erzielt wurden und die auch in den im vorherigen Abschnitt vorgestellten wissenschaftlichen Papers und Studien verwendet und umgesetzt wurden. Am Ende werden alle neuronalen Netze tabellarisch verglichen.

3.3.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) sind eine Klasse von tiefen neuronalen Netzen, die am häufigsten zur Analyse visueller Bilder eingesetzt werden. Es ist besonders bekannt für ihre Fähigkeit, Muster und Merkmale in Bildern zu erkennen, was sie für Aufgaben wie Bilderkennung und -klassifizierung sehr effektiv macht. CNNs nutzen eine mathematische Operation namens Faltung, bei der ein Filter oder Kern über die Eingabedaten geführt wird, um räumliche Hierarchien und Muster zu erfassen. Das Netz besteht in der Regel aus mehreren Schichten, die das Eingabebild in zunehmend abstraktere Darstellungen umwandeln und es dem Modell ermöglichen, verschiedene visuelle Objekte und Phänomene zu unterscheiden und zu klassifizieren [ZXL+18, IBM23].
Typischerweise wird die CNN-Architektur auf Frame-Ebene als Merkmalsextraktor eingesetzt. Dabei wird die zeitliche Beziehung des Videoclips berücksichtigt, die Position der Videosegmente bestimmt und das abgerufene Kopie-/Teilkopie-Videosegment mittels graphbasierter Segmentabgleichung gefunden [ZXL+18, IBM23].
Die CNNs bestehen aus mehreren Schichten, wie Faltungsschichten, Pooling-Schichten und vollständig verbundenen Schichten, die lernen, Merkmale und Muster aus den Eingabedaten zu extrahieren.
Einige Arten von Convolutional Neural Networks sind beispielsweise: AlexNet [YC09], VGGNet [SZ15], GoogLeNet [SLJ+14], ResNet [HZRS15]

3.3.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) sind eine Klasse von Algorithmen der künstlichen Intelligenz, die im unüberwachten maschinellen Lernen eingesetzt werden und durch ein System aus zwei miteinander konkurrierenden neuronalen Netzen innerhalb eines Nullsummenspiel-Frameworks implementiert werden. Diese Technik wurde 2014 von Ian Goodfellow und seinen Kollegen eingeführt [GPAM+14]. GANs werden verwendet, um neue Daten zu generieren, die den Trainingsdaten ähnlich sind, was besonders nützlich in der Erzeugung von Bildern, Videos und Sprache ist. Das erste Netz, der Generator, erzeugt neue Daten, während das zweite Nezt, der Diskriminator, ihre Echtheit bewertet. D. h., der Diskriminator entscheidet, ob jede Dateninstanz, die er überprüft, zum tatsächlichen Trainingsdatensatz gehört oder nicht [Sab18, TY20, ASMM22].

3.3.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) sind eine Klasse von künstlichen neuronalen Netzen, bei denen die Verbindungen zwischen den Knoten ein gerichtetes Graph entlang einer zeitlichen Sequenz bilden. Dies ermöglicht es ihnen, zeitdynamisches Verhalten zu zeigen und ihren internen Zustand (Speicher) zu nutzen, um Sequenzen von Eingaben zu verarbeiten. Das macht RNNs besonders geeignet für Anwendungen wie Spracherkennung, Sprachmodellierung und Übersetzung, bei denen die Abfolge der Datenpunkte entscheidend ist. Im Gegensatz zu vorwärtsgerichteten neuronalen Netzen können RNNs ihren internen Speicher nutzen, um beliebige Sequenzen von Eingaben zu verarbeiten, was sie mächtig für die Modellierung sequenzieller Daten macht [Gro13, SMH11].

3.3.4 k-Nearest Neighbors

Der k-Nearest Neighbors (KNN) Algorithmus ist eine einfache, aber effektive Technik des maschinellen Lernens, der für Klassifizierung und Regression verwendet wird. Er gehört zur Familie der instanzbasierten, kompetitiven Lern- und Lazy-Learning-Algorithmen. Lazy Learning bedeutet, dass er keine Trainingsphase benötigt; stattdessen merkt er sich den Trainingsdatensatz und verwendet ihn zum Zeitpunkt der Vorhersage. KNN klassifiziert einen Datenpunkt basierend darauf, wie seine Nachbarn klassifiziert sind. 'k' in KNN ist ein Parameter, der sich auf die Anzahl der nächsten Nachbarn bezieht, die in den Mehrheitswahlprozess einbezogen werden sollen. Im Wesentlichen betrachtet KNN die 'k' nächstliegenden beschrifteten Datenpunkte im Merkmalsraum, und der Datenpunkt wird der Klasse zugeordnet, die unter seinen 'k' nächsten Nachbarn am häufigsten vorkommt. Es handelt sich um einen nicht-parametrischen Algorithmus, was bedeutet, dass er keine Annahmen über die zugrunde liegende Datenverteilung macht. Diese Eigenschaft macht KNN sehr flexibel einsetzbar in jedem Szenario [Zha16, KK13, Pet09].

3.3.5 Support Vector Machine

Der Support Vector Machine Algorithmus (SVM) ist ein leistungsstarkes Werkzeug für das überwachte maschinelle Lernen, das für Klassifizierungs- und Regressionsherausforderungen verwendet wird. Er erstellt eine Linie oder Hyperebene, um Daten in Klassen zu trennen. Im zweidimensionalen Raum ist diese Hyperebene eine Linie, die eine Ebene in zwei Teile teilt, wobei sich jede Klasse auf einer Seite befindet. SVM ist ideal für Probleme mit vielen Merkmalen, da es eine ähnliche Funktion in höherdimensionalen Räumen ausführt. Das Ziel des SVM-Algorithmus ist es, eine Hyperebene in einem N-dimensionalen Raum zu finden, die die Datenpunkte eindeutig klassifiziert. Es gibt viele mögliche Hyperebenen, um die zwei Klassen von Datenpunkten zu trennen, aber die beste Wahl ist diejenige, die die größte Trennung oder den größten Abstand zwischen den beiden Klassen darstellt. Stützvektoren sind die Datenpunkte, die der Hyperebene am nächsten sind. Die Dimension der Hyperebene hängt von der Anzahl der Merkmale ab. Wenn es zwei Eingabemerkmale gibt, ist die Hyperebene eine Linie. Wenn es drei Eingabemerkmale gibt, ist die Hyperebene eine zweidimensionale Ebene. Mit zunehmender Anzahl an Merkmalen wird die Hyperebene komplexer. SVM ist vorteilhaft, weil es in hochdimensionalen Räumen effektiv ist und relativ speichereffizient ist [MW01, PS20].

3.3.6 Speeded Up Robust Features

Der Speeded Up Robust Features Algorithmus (SURF) ist ein patentierter Computer-Vision-Algorithmus zur Erkennung und Beschreibung lokaler Merkmale in Bildern. Er wurde eingeführt, um die Geschwindigkeit früherer Merkmalsdetektoren und -beschreiber wie SIFT (Scale-Invariant Feature Transform) zu verbessern. SURF ist robust gegenüber Änderungen in Bildrotation, Skalierung und Beleuchtung. Es stützt sich auf Integralbilder für Bildfaltungen, was es ermöglicht, Merkmale schnell und effizient zu berechnen. SURF wird häufig in Aufgaben wie Objekterkennung, 3D-Rekonstruktion und Bildzusammensetzung verwendet [BTVG06, BETVG08].

3.3.7 Scale-Invariant Feature Transform

Scale-Invariant Feature Transform (SIFT) ist ein Computer-Vision-Algorithmus, der zur Erkennung und Beschreibung lokaler Merkmale in Bildern verwendet wird. Er ist bekannt für seine Robustheit gegenüber Änderungen in Maßstab, Rotation und Beleuchtung. SIFT identifiziert Schlüsselpunkte in einem Bild und erzeugt für jeden Schlüsselpunkt einen Deskriptor, der skalen- und rotationsinvariant ist und teilweise invariant gegenüber Änderungen in der Beleuchtung und dem 3D-Kamerablickpunkt. Diese Merkmale machen SIFT zu einer beliebten Wahl für Aufgaben wie Objekterkennung, Bildzusammensetzung und 3D-Modellierung [Ote15, LYT+08, JG09].

3.3.8 Optical Character Recognition

Optical Character Recognition (OCR) steht für „Optische Zeichenerkennung“ . Es handelt sich um eine Technologie, die gedruckten oder handgeschriebenen Text in digitale Daten umwandelt. Mit Hilfe von OCR-Software können Bilder von Dokumenten, z.B. Scans oder Fotografien von Texten, analysiert und die darin enthaltenen Zeichen in maschinenlesbare Zeichenketten umgewandelt werden. Dadurch kann der Text digital bearbeitet, durchsucht und gespeichert werden. OCR wird in verschiedenen Bereichen eingesetzt, von der Automatisierung der Dokumentenerfassung bis hin zur unterstützende Technologie für Sehbehinderte [SBB12, MSY92, Smi07].

3.3.9 Vergleich der neuronalen Netze

In Tabelle 3.2 werden die verschiedenen, in vorherigen Abschnitten beschriebenen neuronalen Netzen verglichen.

3.4 Auswahl von Technologien und Werkzeugen

Die Auswahl der richtigen Technologien und Werkzeuge ist entscheidend für die erfolgreiche Implementierung unseres ETL-Systems. Zuerst werden die gefundenen Technologien für die Extraktion der Metadaten aus Videos präsentiert und verglichen. Anschließend werden die Technologien für die Extraktion der Merkmalen aus Videos betrachtet.

3.4.1 Bibliotheken zur Gewinnung (Extraktion) von Metadaten

Metadaten sind zusätzliche Informationen, die sich auf eine Videodatei beziehen und Informationen wie Titel, Autor, Datum, Beschreibung, Schlüsselwörter, Kodierungsinformationen, Frame-Informationen und viele andere Informationen enthalten, die das Video beschreiben [Con24]. Hier sind einige der bekannten Metadaten-Bibliotheken für Videodateien in den von mir ausgewählten Programmiersprachen (C#, Python, Java) und einigen anderen Sprachen:

FFmpeg

Es handelt sich um ein multifunktionales und leistungsstarkes Werkzeug zur Videobearbeitung in C. Es wird häufig verwendet, um Videometadaten zu lesen und zu schreiben [FFm19]. Es bietet eine breite Palette an Funktionen zur Handhabung verschiedener Videodateien.
FFmpeg ist eine Open-Source-Bibliothek, die auf verschiedenen Betriebssystemen wie Windows, macOS und Linux funktioniert, was es für jedermann zur Nutzung, Modifizierung und Erweiterung verfügbar macht.
FFmpeg zeichnet sich durch seine Fähigkeit aus, Video- und Audioinformationen in mehreren Formaten zu lesen und zu schreiben. Es kann verwendet werden, um verschiedene Video- und Audiodateien zu lesen und deren Informationen wie Auflösung, Bildrate und Audiokodierungsinformationen zu extrahieren. FFmpeg kann auch verwendet werden, um Video- und Audioformate zwischen verschiedenen Formaten zu konvertieren, wie z. B. eine Videodatei von AVI in MP4 zu konvertieren oder eine Audiodatei von WAV in MP3 umzuwandeln.
Darüber hinaus bietet FFmpeg eine breite Palette an Filtern und Effekten, die auf das Video angewendet werden können, wie z. B. das Ändern der Größe des Videos, das Hinzufügen visueller Effekte, das Anpassen der Lautstärke und mehr [US 22].

MediaInfo

Eine Bibliothek, die detaillierte Informationen über Videodateien in verschiedenen Formaten bereitstellt. Anwendungsprogrammierschnittstellen (APIs) sind für viele Sprachen verfügbar, einschließlich C#, Py-

Tabelle 3.2: Vergleich verschiedener Modelle

Modell	**Daten**	**Netztypen**	**Konzept**	**Eingabelänge**	**Einsatzgebiet**	**Layer und Architektur**	**Extrahierte Merkmalen**
CNN	Bilder, Videos, Audio	Faltungsneuronale Netze (Vorwärts gerichtet)	Extrahieren von Merkmalen aus Bildern	Fest	Bildklassifikation, Objekterkennung	Conv2D-layern, Max Pool layern, Vollständig verbundene layern	Eine Matrix arbiträrer Fließkommazahlen
GAN	Bilder, Videos, Audio, Text	Generative Adversarial Networks	Generieren neuer Bilder	Fest	Bildgenerierung, Bildmanipulation	Zwei CNNs, Diskriminator, Generator	Eine Matrix arbiträrer Fließkommazahlen
RNN	Sequenzen (Text, Audio, Zeitreihen)	Rekurrente Neuronale Netze	Verarbeiten sequenzieller Daten	Variabel	Natürliche Sprachverarbeitung, Spracherkennung	Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU)	Eine Matrix arbiträrer Fließkommazahlen
SVM	Vektoren (numerische Daten)	Support Vector Machines	Klassifizieren von Datenpunkten	Fest	Kategorisierung von Daten	Kernel, Entscheidungsfunktion, Parameter, linear, polynomial, radialbasisfunktion usw.	Klasse vom Video
KNN	Vektoren (numerische Daten)	K-Nearest Neighbors	Klassifizieren von Datenpunkten basierend auf ihren Nachbarn	Fest	Empfehlungssysteme, Anomalieerkennung	K nächste Nachbarn	Klasse vom Video
SURF	Bilder	Speeded-Up Robust Features	Extrahieren und Abgleichen von Schlüsselpunkten in Bildern	Fest	Bildabgleich, Objekterkennung usw.	Deskriptoren, Deskriptoren-Abgleich, RANSAC, Hesse-Matrix, Haar-Wavelet usw.	
SIFT	Bilder	Scale-Invariant Feature Transform	Extrahieren und Abgleichen von Schlüsselpunkten in Bildern	Fest	Bildabgleich, Objekterkennung usw.	Deskriptoren, Deskriptoren-Abgleich, RANSAC, Differenz der Gauss-Filter, Gradientenhistogramm usw.	
OCR	Textbilder, gescannte Dokumente	Optical Character Recognition (Nichtneuronal oder neuronal)	Erkennen von gedrucktem oder handgeschriebenem Text	Variabel	Textextraktion aus Bildern, Dokumentenanalyse usw.	Tesseract, CRAFT, CRNN, Klassifizierer, Segmentierung, Erkennung	Text als Zeichenkette

thon und Java. Diese Schnittstellen ermöglichen es, die Funktionalität von MediaInfo in Apps oder Programme zu integrieren [Jer23]. Mit MediaInfo lassen sich folgende Informationen extrahieren:

- Technische Informationen: Beinhalten beispielsweise Auflösung (Breite und Höhe), Bildrate, Kodierungsbitrate, Videoformat, Audioformat und verschiedene Audioinformationen.
- Videometadaten: beinhalten Informationen wie Titel, Autor, Beschreibung, Schlüsselwörter, Produktionsdatum, Zeitmarke und mehr.
- Audiospuren und Untertitel: MediaInfo kann auch Informationen über Audiospuren und Untertitel in der Videodatei anzeigen, wie Untertitelformat, Sprache und Audiokodierung [Med24].

Die in dieser Arbeit vorgestellte Implementierung nutzt die Funktionen von MediaInfo, um Informationen über Videodateien zu sammeln und sie den Benutzern anzuzeigen.

ExifTool

Eine leistungsstarke Bibliothek, die eine Vielzahl unterschiedlicher Metadaten unterstützt, darunter auch für Videodateien. Obwohl Perl als primäre Programmiersprache verwendet wird, lässt sich die Bibliothek auch in C# nutzen, indem die externe Befehlsschnittstelle verwendet wird [Har24a]. ExifTool ist sehr nützlich für das Extrahieren und Analysieren von in Videodateien eingebetteten Metadaten. Die Bibliothek lässt sich verwenden, um auf folgende Informationen zugreifen:

- Technische Informationen: Beinhalten Auflösung (Breite und Höhe), Bildrate, Codec-Bitrate, Videoformat, Audioformat und verschiedene Audioinformationen zuzugreifen [Har24b].
- Videometadaten: Beinhalten Informationen wie Titel, Autor, Beschreibung, Schlüsselwörter, Produktionsdatum, Zeitachsen und mehr.
- Audiospuren und Untertitel: ExifTool kann auch Informationen zu Audiospuren und Untertiteln in der Videodatei anzeigen, wie Untertitelformat, Sprache und Audiocodec.

ExifTool lässt sich in C# nutzen, indem den externen Befehl ExifTool aus dem Programm aufgerufen wird und dann können die zurückgegebenen Ergebnisse analysiert werden. ExifTool stellt eine externe Befehlsschnittstelle zur Verfügung, die zur Interaktion mit Dateien und zum Extrahieren von Metadaten genutzt werden kann [Har24b].

MediaToolkit

Eine C#-Bibliothek basierend auf FFmpeg, die eine einfache Schnittstelle für die Analyse und Bearbeitung von Video- und Audiodateien bietet [Har19]. Diese Bibliothek ermöglicht es, viele medienbezogene Aufgaben durchzuführen, wie das Extrahieren technischer Informationen für Mediendateien, das Schneiden von Videos, das Zusammenführen von Videos, das Ändern des Video- oder Audioformats, das Modifizieren der Video- oder Audioqualitäten und mehr [Ayd17].

OpenCV

OpenCV ist eine Open-Source-Bibliothek zur Bildverarbeitung und gilt als ein leistungsstarkes und beliebtes Werkzeug für die Videoverarbeitung und -analyse. Es wird in C++ entwickelt und bietet fortgeschrittene Funktionen für die Manipulation von Bildern und Videos und das Extrahieren von Informationen aus ihnen. Dank seiner Beliebtheit gibt es interaktive Pakete, um mit OpenCV in Python und Java zu arbeiten und einen einfachen Zugang zu seinen Vorteilen zu ermöglichen [tea23]. OpenCV bietet viele nützliche Funktionen, einschließlich:

- Lesen und Schreiben von Fotos und Videos: OpenCV kann verwendet werden, um Bilder und Videos aus verschiedenen Quellen zu lesen und Änderungen daran vorzunehmen. Es können auch verschiedene Effekte auf Fotos und Videos angewendet und Ergebnisse gespeichert werden.
- Bildverarbeitung: OpenCV bietet viele Technologien für die Bildverarbeitung wie Qualitätsverbesserung, Helligkeits- und Kontrasteinstellung, Rauschreduzierung, Bildfilterung, Kantenerkennung sowie Erkennung von Formen und Objekten.
- Gesichts- und Objekterkennung: OpenCV verfügt über leistungsstarke Werkzeuge zur Erkennung von Gesichtern und Objekten in Fotos und Videos. Technologien zur Gesichts- und Objekterkennung können für Anwendungen wie Gesichtserkennung, Verfolgung, Kategorisierung und Bewegungserkennung verwendet werden.
- Videoanalyse: OpenCV kann für die Videoanalyse verwendet werden, einschließlich der fortgeschrittenen Extraktion von Informationen wie visuellen Bewegungen, Tracking, Frame-Capture und Bewegungsanalyse [BK08].

Vergleich der Technologien

Tabelle 3.3 stellt einen Vergleich der Bibliotheken zur Gewinnung (Extraktion) von Metadaten dar.

3.4.2 Bibliotheken zur Extraktion von Merkmalen aus Videodateien

Es gibt viele Werkzeuge und Bibliotheken, die zur Extraktion von Videomerkmalen und zur Plagiatserkennung verwendet werden können. Hier sind einige beliebte Bibliotheken und Werkzeuge in verschiedenen Programmiersprachen.

OpenCV

Es handelt sich um eine Open-Source-Bibliothek für die Bild- und Videoverarbeitung. die zur Extraktion von Videomerkmalen verwendet werden kann [tea23]. Ihr Ziel ist es, einen umfassenden Satz an Werkzeugen und Funktionen für die Bild- und Videoverarbeitung bereitzustellen. OpenCV wird in der Programmiersprache C++ entwickelt und unterstützt viele andere Programmiersprachen wie Python, Java und andere. OpenCV ist ein aktives Community-Projekt, das von einer Entwicklergemeinschaft auf der ganzen Welt entwickelt und gepflegt wird.
Die OpenCV-Bibliothek bietet eine breite Palette an Funktionen und Werkzeugen für die Bild- und Videoverarbeitung, einschließlich:

- Lesen und Schreiben von Fotos und Videos: OpenCV kann verwendet werden, um Fotos und Videos aus verschiedenen Dateien zu lesen und Fotos und Videos in neuen Dateien in verschiedenen Formaten zu speichern.
- Bildverarbeitung: OpenCV bietet Funktionen für die Bildverarbeitung, wie das Ändern der Größe eines Bildes, das Umwandeln von Farben, das Anwenden von Filtern und das Manipulieren von Kontrast und Helligkeit, das Erkennen von Kanten, das Extrahieren von Merkmalen, das Entfernen von Rauschen und viele andere Operationen.
- Gesichts- und Objekterkennung: OpenCV kann verwendet werden, um Technologien zur Gesichts- und Objekterkennung anzuwenden, einschließlich Gesichtserkennung, Objektverfolgung und Objekterkennung und -klassifizierung.
- Bewegungserkennung und -verfolgung: OpenCV bietet Werkzeuge für die Analyse von Videobewegungen, die Verfolgung beweglicher Objekte und das Extrahieren von Bewegungspfaden.

Tabelle 3.3: Vergleich der Bibliotheken zur Gewinnung von Metadaten für Videodateien

Aspekte	**OpenCV**	**MediaToolkit**	**ExifTool**	**MediaInfo**	**FFmpeg**
Benutzerfreundlichkeit und Erlernbarkeit	Erfordert gute Kenntnisse in C++ und bietet Schnittstellen für Python und Java	Bietet eine benutzerfreundliche Schnittstelle in C#	Erfordert die Nutzung der Kommandozeile und etwas Programmierung für die Interaktion	Erfordert die Interaktion mit der Kommandozeile	Erfordert die Interaktion mit der Kommandozeile
Dokumentation und Support verfügbar	Ausgezeichnete Dokumentation und eine große Community für den Support sind verfügbar	Gute Dokumentation und aktive Support-Community	Gute Dokumentation und aktive Support-Community	Gute Dokumentation und aktive Support-Community	Gute Dokumentation und aktive Support-Gemeinschaft
Leistung und Verarbeitungsgeschwindigkeit	Bietet schnelle Leistung und effiziente Bild- und Videoverarbeitung	Verlässt sich auf das leistungsstarke FFmpeg für schnelle Verarbeitung	Bietet schnelle und effiziente Leistung beim Extrahieren von Metadaten	Bietet schnelle und effiziente Leistung beim Extrahieren von Metadaten	Verlässt sich auf das leistungsstarke FFmpeg für schnelle Verarbeitung
Verfügbare Funktionalität und eine Vielzahl der unterstützten Formate	Bietet fortschrittliche Funktionen für Bild- und Videoverarbeitung sowie Auswertung	Bietet Funktionen zur Analyse und Modifikation von Video- und Audiodateien	Bietet Funktionen zum Lesen und Schreiben von Mediendaten und Metadateien	Bietet Funktionen zum Lesen von Mediendaten und Metadateien	Bietet umfassende Funktionen zur Medienverarbeitung und Formatkonvertierung
Kompatibilität mit verschiedenen Betriebssystemen	Windows, Linux, macOS	Windows, Linux, macOS	Windows, Linux, macOS	Windows, Linux, macOS	Windows, Linux, macOS
Programmiersprache	C++, Python, Java	C#	Perl	C++	C, C++, Python, Java
Arten von extrahierten Metadaten	VideoName, Zeit, Dauer, Codec, Auflösung, Bildrate, Audiokanäle	VideoName, Zeit, Dauer, Codec, Auflösung, Bildrate, Audiokanäle	VideoName, Zeit, Dauer, Codec, Auflösung, Bildrate, Audiokanäle	VideoName, Zeit, Dauer, Codec, Auflösung, Bildrate, Audiokanäle	VideoName, Zeit, Dauer, Codec, Auflösung, Bildrate, Audiokanäle
Unterstützte Video-Dateierweiterungen	AVI, MP4, MOV, MKV, FLV, MPEG	AVI, MP4, MOV, WMV, MKV, FLV, MPEG, , WebM, VOB	AVI, MP4, MOV, WMV, MKV, FLV, MPEG, 3GP, WebM, VOB, M4V, MPG, TS, ASF, RM	AVI, MP4, MOV, WMV, MKV, FLV, MPEG, 3GP, WebM, VOB	AVI, MP4, MOV, WMV, MKV, FLV, MPEG, 3GP, WebM, VOB, M4V, MPG, TS, ASF, RM
Die Fähigkeit, die Informationen im Bild zu ändern	Informationen im Video können modifiziert und verarbeitet werden	Informationen im Video können modifiziert und verarbeitet werden	Informationen im Video können modifiziert und verarbeitet werden	Nicht unterstützt	Informationen im Video können modifiziert und verarbeitet werden
Unterstützte Bildgröße	variabel und basiert auf dem Originalvideo	variabel und basiert auf dem Originalvideo	variabel und basiert auf dem Originalvideo	variabel und basiert auf dem Originalvideo	variabel und basiert auf dem Originalvideo
Anzahl der unterstützten Frames	Unbegrenzt	Unbegrenzt	Unbegrenzt	Unbegrenzt	Unbegrenzt

- Maschinelles Lernen und Computer Vision: OpenCV unterstützt die Implementierung von Techniken des maschinellen Lernens und der Computer Vision wie Bildklassifizierung, Objektverfolgung, Gesichtserkennung, Objekterkennung und Texterkennung.

Scikit-learn-Bibliothek

Es handelt sich um ein Open-Source-Framework für maschinelles Lernen, das in Python arbeitet. Scikit-learn zielt darauf ab, eine breite Palette von Werkzeugen und Techniken für die Entwicklung von Modellen des maschinellen Lernens und Datenanalyse bereitzustellen. Einfach zu verwenden und skalierbar ist scikit-learn eines der beliebtesten und am weitesten verbreiteten Werkzeuge im Bereich des maschinellen Lernens. Scikit-learn kann für eine Vielzahl von Aufgaben verwendet werden, die Datenanalyse, Datenklassifizierung, Aggregation, Hashing, Transformation, Reinigung, merkmalsbezogene Analyse, Mustererkennung, Vorhersage und andere gängige Aufgaben im maschinellen Lernen umfassen [ld24].

Weka

Es ist eine Sammlung von Werkzeugen für maschinelles Lernen und statistisches Lernen. Kann verwendet werden, um Videomerkmale zu analysieren und Plagiate zu erkennen. Weka ist darauf ausgelegt, eine benutzerfreundliche und visuelle Schnittstelle für Datenanalyse und die Entwicklung von Modellen des maschinellen Lernens zu bieten. Weka bietet eine breite Palette von Werkzeugen und Techniken, die in Form von erweiterbaren Modulen, sogenannten „Paketen“ verfügbar sind. Diese Klassen können verwendet werden, um eine Vielzahl von Aufgaben durchzuführen, einschließlich Datenanalyse, Klassifizierung, Aggregation, Segmentierung, Mustererkennung, Merkmalsanalyse, Big-Data-Analyse und anderen [Macss].

Accord.NET

Es handelt sich um ein Framework für maschinelles Lernen und Computer Vision. Es kann verwendet werden, um Videomerkmale zu extrahieren. Entwickelt von einer Gruppe von Entwicklern und Forschern, zielt es darauf ab, einen umfassenden Satz von Werkzeugen und Bibliotheken für Anwendungen des maschinellen Lernens, der Bildverarbeitung und der Computer Vision mit der Programmiersprache C# bereitzustellen. Accord.NET umfasst viele Funktionen, um die Entwicklung von Anwendungen des maschinellen Lernens und der Computer Vision zu erleichtern [Acc17]. Im folgenden werden einige der verfügbaren Funktionen beschrieben:

- Maschinelles Lernen: Accord.NET bietet eine breite Palette von Algorithmen und Modellen für maschinelles Lernen, einschließlich künstlicher neuronaler Netze, Klassifizierung, Segmentierung, Aggregation und statistische Analyse. Diese Algorithmen können für eine Vielzahl von Aufgaben wie Klassifizierung, Prognose, Objekterkennung und Datenanalyse verwendet werden.

- Bildverarbeitung und Computer Vision: Accord.NET bietet eine umfassende Bibliothek für Bildverarbeitung und Computer Vision. Diese Bibliothek kann verwendet werden, um Bilder und Videos hochzuladen, Geometrien wie Rotation und Zoom anzuwenden und Operationen wie Bildverbesserung, Farbkonvertierung, Formanalyse, Objekterkennung, Nachverfolgbarkeit und Merkmalsextraktion durchzuführen.

- Merkmalsextraktion: Accord.NET bietet Funktionen zur Extraktion von Merkmalen aus Fotos und Videos. Diese Funktion kann verwendet werden, um visuelle Daten in eine mathematische oder digitale Darstellung umzuwandeln, die in Anwendungen des maschinellen Lernens, der Klassifizierung und Segmentierung verwendet werden kann.

- Datenverarbeitung: Accord.NET bietet Funktionen zum Laden und Analysieren von Daten aus verschiedenen Quellen. Die Bibliothek kann genutzt werden, um Daten aus und in Dateiformaten

wie CSV, Excel, SQL usw. zu lesen und zu schreiben sowie numerische Daten zu analysieren und zu transformieren.

- Unterstützung neuronaler Netze: Accord.NET bietet starke Unterstützung für die Entwicklung und das Training künstlicher neuronaler Netze. Diese Funktion kann verwendet werden, um Deep-Learning-Modelle zu erstellen und Aufgaben im Zusammenhang mit der Klassifizierung durchzuführen.

FFmpeg.NET

Die FFmpeg.NET-Bibliothek ist eine Softwarebibliothek, die eine Anwendungsprogrammierschnittstelle (API) für die Nutzung der FFmpeg-Bibliothek beim Konvertieren, Aufzeichnen und Streamen von Audio und Video bereitstellt. FFmpeg ist eine leistungsstarke, beliebte und Open-Source-Bibliothek, die eine breite Palette von Werkzeugen und Funktionen für die Multimedia-Verarbeitung bietet. FFmpeg.NET bietet eine benutzerfreundliche Schnittstelle, um über die Programmiersprache C# mit der FFmpeg-Bibliothek zu interagieren. Diese Schnittstelle kann sowohl in Web- als auch in Desktop-Anwendungen eingesetzt werden, um Aufgaben wie das Konvertieren von Video- und Audioformaten, das Extrahieren von Frames aus Videos, das Anwenden von Effekten und Filtern auf Medien sowie das Streamen von Multimedia zu ermöglichen [Con24].

JavaCV

Es handelt sich um eine Open-Source-Bibliothek, die eine Anwendungsprogrammierschnittstelle (API) bereitstellt, um eine Vielzahl von Bild-, Video- und Computer-Vision-Verarbeitungsfunktionen mit der Sprache Java zu handhaben. JavaCV basiert auf anderen beliebten Werkzeugen im Bereich der Bild- und Videoverarbeitung wie OpenCV, FFmpeg und anderen. Es ist darauf ausgelegt, eine benutzerfreundliche und leistungsstarke Schnittstelle für Entwickler bereitzustellen, die moderne Bild- und Videoverarbeitungstechniken mit Java anwenden möchten [Ac24].
JavaCV umfasst viele Funktionen, die in verschiedenen Anwendungen genutzt werden können. Hier ist eine detaillierte Erklärung einiger dieser Funktionen:

- Zugriff auf OpenCV und FFmpeg: JavaCV greift auf die Bibliotheken OpenCV und FFmpeg zurück, um Funktionen zur Bild- und Videoverarbeitung zu bieten. Nutzer können OpenCV-Funktionen wie Filterung, Objekterkennung, Formanalyse und vieles mehr verwenden. Darüber hinaus ermöglicht JavaCV den Einsatz von FFmpeg-Funktionen zum Lesen, Schreiben und Konvertieren von Video- und Audiodateien.

- Bild- und Videoverarbeitung: JavaCV stellt eine leistungsstarke Programmierschnittstelle bereit, die eine einfache Verarbeitung von Bildern und Videos ermöglicht. Nutzer können damit Bilder und Videos laden, geometrische Transformationen wie Drehen, Herauszoomen und Transparenzeffekte anwenden sowie verschiedene Operationen wie Bildverbesserung, Farbkonvertierung und Merkmalsextraktion durchführen.

- Objekt- und Gesichtserkennung: JavaCV unterstützt Technologien zur Objekt- und Gesichtserkennung. Entsprechende OpenCV-Funktionen können verwendet werden, um diese Aufgaben umzusetzen.

- Maschinelles Lernen und neuronale Netze: JavaCV stützt sich auf andere Bibliotheken wie DeepLearning4J und TensorFlow, um Techniken des maschinellen Lernens und neuronale Netze zu implementieren. Diese Bibliotheken können genutzt werden, um maschinelle Lernmodelle zu trainieren und Vorhersagen sowie Klassifizierungen durchzuführen.

- Verbindung zur Webcam und Aufnahme von Fotos und Videos: JavaCV kann verwendet werden, um eine Verbindung zur Webcam herzustellen und Fotos sowie Videos aufzunehmen. Ebenso ist

es möglich, Daten zu analysieren, die von der Kamera extrahiert wurden, für Anwendungen wie Bewegungserkennung und Überwachungskontrolle.

PyAV-Bibliothek

Es handelt sich um eine Open-Source-Softwarebibliothek für die Arbeit mit Video- und Audiodateien in der Sprache Python. PyAV zielt darauf ab, eine benutzerfreundliche Schnittstelle bereitzustellen, um Audio- und Videodateien mit der beliebten FFmpeg-Bibliothek zu analysieren, zu bearbeiten und zu konvertieren.

Vergleich der Technologien

Tabelle 3.4 stellt einen Vergleich der Bibliotheken zur Extraktion von Merkmalen in Videos dar.

Tabelle 3.4: Vergleich der Bibliotheken zur Extraktion von Merkmalen

Eigenschaft	**PyAV**	**JavaCV**	**OpenCV**	**FFmpeg.NET**	**Accord.NET**	**Weka**	**Scikit-learn**
Sprache	Python	Java	C++, Python, Java	C#	.NET (C#)	Java	Python
Einsatz-bereiche	Video- und Audiobearbeitung	Computer Video-, Audio- und Bildverarbeitung	Video- und Audiobearbeitung und Bildverarbeitung	Video- und Audiobearbeitung	Maschinelles Lernen und Computer Vision	Maschinelles Lernen und statistische Analyse	Maschinelles Lernen und statistische Analyse
Datenverarbeitung	Analysieren, Bearbeiten und Konvertieren von Audio- und Videodateien	Arbeit mit Video-, Audio- und Bildverarbeitung	Analyse und Verarbeitung von Video-, Audio- und Bildbearbeitung	Konverti-erung, Aufnahme und Streaming von Audio und Video	Extraktion von Videomerkmalen und Plagiatserkennung	Analyse von Videomerkmalen und Plagiatserkennung	Analyse von Videomerkmalen und Plagiatserkennung
Unterstützung für neuronale Netze	Nein	Nein	JA	Nein	JA	Nein	JA
Leistung	PyAV kann in Bezug auf die Leistung etwas niedriger als JavaCV und OpenCV betrachtet werden	Konvergenz zwischen JavaCV und OpenCV, mit dem Vorteil der Verwendung von Java	OpenCV gehört zu den schnellsten Bibliotheken zur Bildverarbeitung	Gut	Die Leistung kann durch die Nutzung beeinflusst werden	Die Leistung wird als gut betrachtet	Die Leistung wird als gut betrachtet

Kapitel 4

Methodik

Dieses Kapitel liefert eine eingehende Erläuterung des ETL-Prozesses, welcher ein zentraler Schritt im Kontext der Plagiatserkennung in Videos darstellt. Der ETL-Prozess umfasst diverse kritische Phasen, welche Präzision sowie höchste Sorgfalt erfordern, um akkurate und zuverlässige Ergebnisse zu gewährleisten. Eine Übersicht des gesamten ETL-Prozesses bildet den Auftakt dieses Kapitels. Anschließend erfolgt eine detaillierte Darlegung jedes einzelnen Schrittes des ETL-Prozesses. Zunächst wird eine ausführliche Beschreibung des Extraktionsprozesses von Bildern, Text und Audiodateien aus der Videodatei gegeben, gefolgt von deren Transformation in geeignete Merkmale für den Vergleich von Videoclips zur Erkennung des Plagiatanteils zwischen ihnen. Abschließend wird erläutert, wie diese Merkmale in einer Datenbank gespeichert werden. Hierzu wird ein Entitäts-Beziehungsdiagramm präsentiert, das in diesem Kapitel zusätzlich zu einem systematischen Diagramm des entwickelten Programms zur Erläuterung der Methode der Plagiatserkennung zwischen Videoclips vorgestellt wird.

4.1 Überblick über den vorgeschlagenen ETL-Prozess

In Abbildung 4.1 wird der ETL-Prozess veranschaulicht. Dabei ist zu beachten, dass die Schritte im Prozess in der Reihenfolge (Extraktion, Transformation, Laden) erfolgen. Die Abbildung stellt alle einzelnen Schritte im ETL-Prozess abstrakt dar und definiert das Konzept, nach dem der tatsächliche ETL-Prozess implementiert wird. Eine detailliertere Beschreibung der Implementierung erfolgt in Kapitel 5. **Hinweis:** Die Audio-Daten der Arbeit werden gemäß dem Konzept von Ghaith Ishak ([Ish24]) bearbeitet, während die Bild-Daten dem Konzept von Muhamad Kurdi [Kur24] folgen.
Zunächst erfolgt die Extraktion von Metadaten aus den Videos. Es wird auch eine Extraktion von Bildern und Audiodateien durchgeführt. Anschließend erfolgt die Transformation der Daten in eine für die Analyse geeignete Form. Dieser Prozess beinhaltet die Verarbeitung, Bereinigung und Vorbereitung der Daten für die Analyse. Die Bewältigung dieser Aufgabe erfordert den Einsatz fortgeschrittener Techniken, um mit den unstrukturierten und vielfältigen Daten umzugehen. Abschließend werden die verarbeiteten Daten in die Zieldatenbank geladen, wo sie für Analysen und Abfragen genutzt werden können. Die Daten umfassen sowohl die Metadaten der Videos als auch die extrahierten Merkmale sowie Relationstabellen zwischen den Merkmalen der Videos und den Merkmalen ihrer Audiodateien. Dieser Schritt gewährleistet eine konsistente Speicherung der Daten, um ihre Verwendung für spätere Abfragen sicherzustellen.

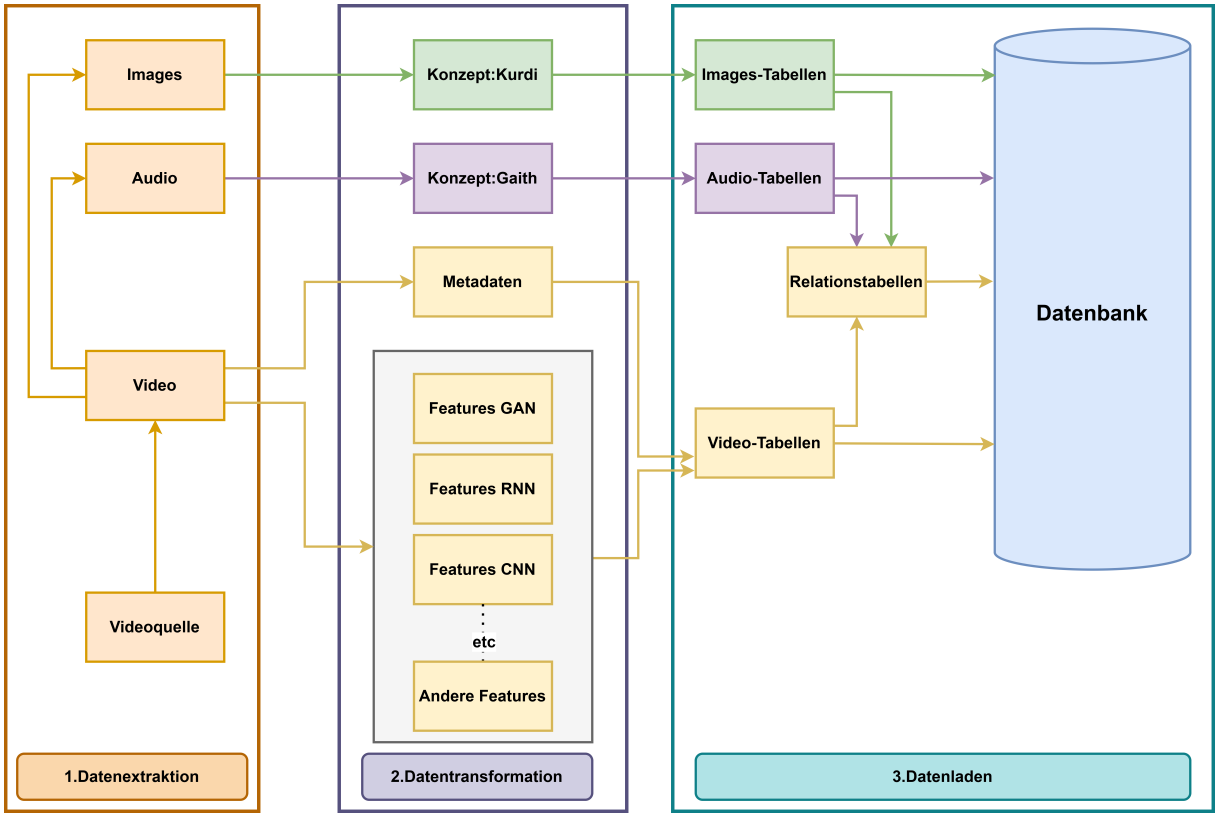


Abbildung 4.1: Das Konzept vom ETL-Prozess.

4.2 Datenextraktion

Zu Erkennungszwecken wurden drei Datentypen extrahiert:

1. **Bilddaten:** Es handelt sich um die aus dem Videoeingang extrahierten Frames.
2. **Audiodaten:** Audio wird durch Umwandlung von Video in Audio extrahiert.
3. **Textdaten:** Jeder Typ von Text, der in den Bilddaten vorhanden sein könnte. Und das Audio wird dann in Text transformiert. Alle diese Datentypen werden in einer Datenbank für weitere Verarbeitung und Forschung gespeichert. Dieser Schritt ist in Abbildung 4.2 dargestellt.

In diesem Schritt müssen wir Daten aus dem Video mithilfe der Accord-, Accord.Video- und Accord.Video.FFMPEG-Bibliotheken extrahieren.

- **Accord:** Es handelt sich um ein Framework für wissenschaftliches Rechnen in .NET [Acc17]. Das Framework umfasst eine Reihe von Bibliotheken, die sowohl in Quellcodeform als auch über ausführbare Installer und NuGet-Pakete verfügbar sind. Zusätzlich gibt es das Accord.NET-Framework, das voll ausgestattete Audio- und Bildverarbeitungsbibliotheken, geschrieben in C#, kombiniert. Es dient als umfassendes Framework für die Erstellung von Anwendungen im Bereich Computer Vision, Computer Audition, Signalverarbeitung und statistische Anwendungen, auch für kommerzielle Nutzung.
- **Accord.Video:** Diese Bibliothek enthält Schnittstellen und Klassen, um auf verschiedene Videoquellen zugreifen zu können, wie z.B. IP-Videokameras (MJPEG-Streams) [Sou17]. Dieses Paket stammt aus dem AForge.NET-Framework und ist Teil des Accord.NET-Frameworks.

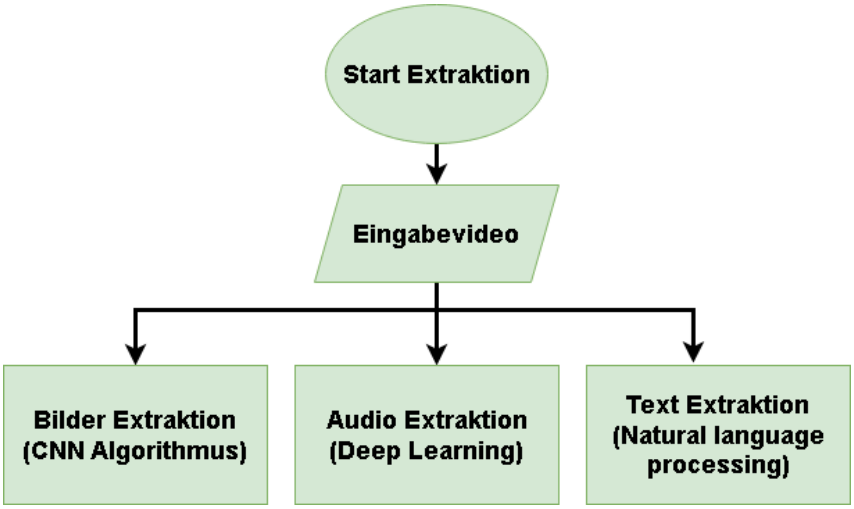


Abbildung 4.2: Der Schritt der Extraktion im ETL-Prozess

- **Accord.Video.FFMPEG:** Diese Bibliothek umfasst Klassen und Methoden zur Handhabung von Videoquellen mit FFMPEG [ces17]. Aufgrund dessen ist das Paket unter der GPL-Lizenz verfügbar. Auch dieses Paket stammt aus dem AForge.NET-Framework und ist Teil des Accord.NET-Frameworks.

Dies kann das Extrahieren von Bildern (d. h. Frames), Audio und Text (falls vorhanden) aus dem Video umfassen. Diese Bibliotheken bieten die notwendigen Werkzeuge, um verschiedene Videoquellen zu verarbeiten und die gewünschten Daten zu extrahieren. Dies geschieht durch Umwandlung des Videos in eine Serie von Frames und anschließendes Extrahieren des Audios und Textes (falls vorhanden) aus jedem Frame. Dieser Prozess erfordert einige Programmierkenntnisse, bietet jedoch eine effiziente Möglichkeit, die große Datenmenge in Videos zu verarbeiten.
Zusätzlich zum Extrahieren von Bildern, Audio und Text (falls vorhanden) aus Videos mit den Accord-Bibliotheken können wir auch Metadaten aus der Videodatei mit der MediaInfoLib-Bibliothek extrahieren. Diese Metadaten umfassen folgende Felder: Name, Zeit, Dauer, Codec, Auflösung, Bildrate und Audiokanäle.
MediaInfoLib ist eine Bibliothek, die eine einheitliche und bequeme Anzeige von technischen und wesentlichen beschreibenden Daten für Video- und Audiodateien bietet. Sie kann verwendet werden, um eine Vielzahl von Informationen über Mediendateien zu erhalten, einschließlich beschreibender Daten. Diese Bibliothek ist besonders nützlich beim Umgang mit großen Mediendateien, da sie Daten effizient analysieren und extrahieren kann.
Es gibt viele andere Werkzeuge und Bibliotheken für die Verarbeitung von Videodaten neben FFMPEG. Hier sind einige Beispiele:

- **GStreamer**: Dies ist ein leistungsstarkes Multimedia-Framework, das eine Pipeline-basierte Architektur für die Handhabung komplexer Multimedia-Aufgaben verwendet. Es unterstützt eine breite Palette von Medienformaten und bietet Möglichkeiten für die Medienaufnahme, -verarbeitung und -wiedergabe.

- **VLC Media Player (libVLC)**: VLC ist ein freier und quelloffener Multimedia-Player, der auch mit einer Kernbibliothek, libVLC, kommt, die für die Einbettung der Videowiedergabe in Anwendungen verwendet werden kann.

- **OpenCV (Open Source Computer Vision Library)**: Dies ist eine hoch optimierte Bibliothek

mit Schwerpunkt auf Echtzeitanwendungen. Sie kann verwendet werden, um Videos aufzunehmen, Frames zu verarbeiten und sogar komplexere Aufgaben wie Objekterkennung durchzuführen.

4.3 Datentransformation

Sobald die Daten extrahiert sind, müssen wir sie in ein Format transformieren, das es uns ermöglicht, damit zu arbeiten. Dies kann das Transformieren von Bildern und Audio in Merkmale beinhalten, die zur Erkennung von Plagiaten verwendet werden können. Deep Neural Networks, wie Convolutional Neural Networks (CNN), können verwendet werden, um Merkmale aus Bildern und Ton zu extrahieren.
Der Grund, warum Video nicht direkt erwähnt wird, liegt darin, dass Video typischerweise als eine Serie von einzelnen Frames verarbeitet wird, von denen jeder ein Bild ist. Daher beziehen wir uns, wenn wir von der Verarbeitung von Video mit CNNs sprechen, oft auf die Verarbeitung dieser einzelnen Frames (oder Bilder). Audio kann separat extrahiert und verarbeitet werden. Es ist jedoch wichtig zu beachten, dass es auch spezifische Techniken und Netzwerkarchitekturen für die Verarbeitung von Videodaten als Ganzes unter Berücksichtigung der zeitlichen Dimension gibt (wie 3D-CNN und auf Recurrent Neural Network (RNN) basierende Ansätze). Diese können die zeitlichen Abhängigkeiten zwischen aufeinander folgenden Frames in einem Video erfassen.

4.4 Datenladen

Das Laden ist der letzte Schritt im ETL-Prozess, bei dem die transformierten Daten in die Datenbank geladen werden. Im Kontext von Videos bedeutet dies das Hochladen der transformierten Daten (Merkmale) aus dem Video in die Datenbank. Dieser Schritt umfasst das Speichern der extrahierten und transformierten Merkmale in der Datenbank zur Verwendung im Plagiatserkennungsprozess.
Das Design der Datenbank hängt von den zu speichernden Daten und deren Verwendung ab. In diesem Zusammenhang müssen möglicherweise Merkmale, die aus Videos extrahiert wurden, sowie andere Informationen wie Video-ID, Zeit, Text und andere gespeichert werden.
Bei der Gestaltung einer Datenbank sollte man in der Lage sein, Videos effizient zu suchen und zu vergleichen. Das bedeutet, dass die Datenbank so gestalten werden müssen, dass man eine schnelle Suche nach Videos auf der Grundlage extrahierter Merkmale ermöglicht und diese zum Erkennen von Plagiaten verglichen werden kann.
Das Design der Datenbank hängt von den Daten ab, die gespeichert werden müssen und wie sie verwendet werden. Wir müssen dieses Design möglicherweise während der praktischen Implementierung und Durchführung von Tests anpassen. In Abbildung 4.3 ist das Design der Datenbank mit allen Tabellen und die Fremdschlüsselbeziehungen dazwischen dargestellt. Im Folgendem werden die Tabellen einzeln beschrieben:

- **Videotabelle** (Videos table): Diese Tabelle kann Informationen und Metadaten über das Video selbst enthalten, wie Video-ID, Name, Zeit und andere grundlegende Informationen.
- **Merkmaltabelle** (Feature table): Diese Tabelle kann aus dem Video extrahierte Merkmale enthalten. Diese Merkmale können unterschiedlich sein, und können durch Methoden wie Convolutional Neural Networks (CNNs) verwenden. Beim Einsatz von CNNs zur Extraktion von Farben aus einem Video kann dieser Prozess auf der Ebene des gesamten Frames oder auf der Ebene einzelner Pixel durchgeführt werden. Zum Beispiel kann das Netzwerk lernen, zwischen verschiedenen im Frame vorhandenen Farben zu unterscheiden und dann die dominante Farbe zu bestimmen oder die Durchschnittsfarbe des Frames zu berechnen. Darüber hinaus kann das Netzwerk lernen, die Farbe von jedem Pixel einzeln zu extrahieren und dann die Durchschnittsfarbe des Frames auf Basis der aus den Pixeln extrahierten Farben zu berechnen.

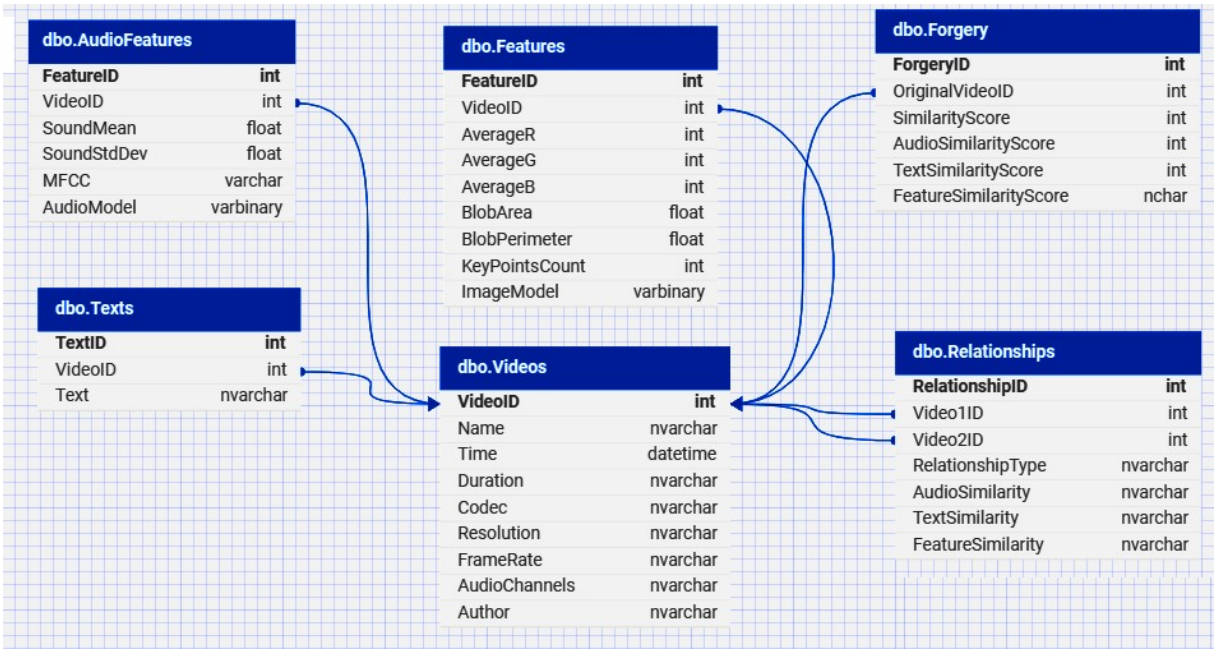


Abbildung 4.3: Datenbankschema

- **Texttabelle** (Text table): Wenn Ihr Video Text enthält (wie Untertitel oder eingebetteten Text), kann dieser Text extrahiert und in einer separaten Tabelle gespeichert werden.
- **Audiotabelle** (audio table): Zuerst wird das Audio aus dem Video extrahiert, dann werden die gewünschten Merkmale vom Audio berechnet, die abschließend in der Datenbank gespeichert werden.
- **Plagiattabelle** (Plagiarism table): Diese Tabelle kann Informationen über Videos enthalten, die als Duplikate oder ähnlich erkannt wurden. Jeder Eintrag in der Tabelle enthält einen Ähnlichkeitswert, der ausdrückt, wie ähnlich ein Video einem anderen Video ist. Fälschung wird in einer separaten Tabelle dargestellt, die „forgeryid“, „originalvideoid“, „duplicatevideoid“ und „similarityscore“ umfasst. Dies bezieht sich auf den Prozess der Identifizierung und Bewertung von Videofälschungen.
- **Beziehungstabelle** (Relationship table): Diese Tabelle kann Informationen über die Beziehungen zwischen Videos enthalten, wie Videos, die miteinander verknüpft sind oder als Duplikate erkannt wurden.

Die Tabellen sind miteinander durch Fremdschlüsselbeziehungen verknüpft. Die Video-ID in den Tabellen `Features`, `Texts` und `Audio` ist als Fremdschlüssel (FK) mit der Tabelle `Videos` verknüpft, ebenso wie die Originalvideo-ID in der Tabelle `Forgery`. Die IDs der beiden Videos in der Tabelle `Relationships` sind ebenfalls Fremdschlüssel.

Kapitel 5

Systemdesign und Architektur

In diesem Kapitel befassen wir uns mit dem Design unseres Systems und erkunden die architektonischen Aspekte, die mit unserem ETL-Prozess (Extrahieren, Transformieren, Laden) zusammenhängen. Das Ziel ist es, relevante Indikatoren aus Videos zu identifizieren, die als entscheidender vorbereitender Schritt für die Plagiatserkennung dienen werden.
Um dieses Ziel zu erreichen, verwendeten wir eine Reihe von Bibliotheken, die vielfältige Werkzeuge und Methoden zur Verarbeitung von Audio, Bildern aus Videos bereitstellen. Zu diesen Bibliotheken gehören Accord.Video.FFMPEG, Accord.Video, MediaInfoDotNet, NAudio.FileFormats, NAudio.Wave, OpenCvSharp.Extensions, OpenCvSharp.XFeatures2D und OpenCvSharp. Es ist erwähnenswert, dass diese Bibliotheken auf neuronalen Netzwerken (CNN) basieren.
In unserer Anwendung nutzten wir die von der Programmierumgebung bereitgestellten Bibliotheken, um die gewünschten Ziele des Systems zu erreichen. Diese Bibliotheken bieten eine Reihe von Werkzeugen und Methoden, die uns helfen, Daten auf effiziente und präzise Weise zu verarbeiten und zu analysieren.

5.1 Verwendete Bibliotheken/Werkzeuge

In diesem Abschnitt werden die von unserem Programm verwendeten Bibliotheken aufgeführt, zusammen mit einer Erläuterung des Zwecks jeder Bibliothek. Es wurden noch weitere weniger wichtige Bibliotheken verwendet, die in Unterabschnitt A.1.1 beschrieben sind.

5.1.1 Bibliotheken zur Audiobearbeitung

- **Accord.Audio.ComplexFilters** *(version 3.8.2-alpha)*: Bietet fortgeschrittene Filter zur Verarbeitung komplexer Audiosignale, wie die schnelle Fourier-Transformation (FFT) und Gabor-Transformationen.
- **Accord.Audio.Filters** *(version 3.8.2-alpha)*: Bietet grundlegende Filter zur Verarbeitung von Audiosignalen, wie Tiefpass-, Hochpass- und Rauschfilter.
- **Accord.Audio.Formats** *(version 3.8.2-alpha)*: Bietet Funktionen zum Lesen und Schreiben verschiedener Audio-Dateiformate, wie WAV und MP3.
- **Accord.Audio** *(version 3.8.2-alpha)*: Bietet grundlegende Klassen zur Verarbeitung von Audiosignalen, wie die Darstellung von Sprachsignalen und DFT-Operationen.

5.1.2 Bibliotheken zur Bildverarbeitung

- **Accord.Imaging.Filters** *(version 3.8.2-alpha)*: Bietet grundlegende Filter zur Bildverarbeitung, wie Glättung, Rauschfilterung und Größenänderung.
- **Accord.Imaging** *(version 3.8.2-alpha)*: Bietet grundlegende Kategorien zur Bildverarbeitung, wie Bildrepräsentation und schnelle Fourier-Transformation (FFT).

5.1.3 Bibliotheken zur Videobearbeitung

- **Accord.Video.FFMPEG** *(version 3.8.2-alpha)*: Bietet Funktionen zum Lesen und Schreiben verschiedener Video-Dateiformate, wie AVI und MP4, unter Verwendung der FFMPEG-Bibliothek.
- **Accord.Video** *(version 3.8.2-alpha)*: Bietet grundlegende Klassen zur Videobearbeitung, wie das Extrahieren von Frames und das Konvertieren von Videoformaten.

5.2 Entwurf des ETL-Workflows

Zur Durchsuchung des Internets nach plagiierten Videos kommen diverse Werkzeuge und Methoden zum Einsatz, darunter die Google-Bildersuche, das Durchsuchen von Textdaten sowie die Anwendung von Techniken der natürlichen Sprachverarbeitung auf Textinhalte.

5.2.1 Verwendung der Google-Bildersuche

Eine der gängigsten Techniken bei der Suche nach plagiierten Videos ist die Verwendung der Google Bildersuche. Diese Technik beinhaltet die Erstellung einer benutzerdefinierten Bildsuchmaschine, die Googles Bildersuchfunktionen nutzt, um nach verarbeiteten Videoframes im Internet zu suchen. Die Suchmaschine ist darauf ausgelegt, Bild mit Bild oder Bild mit Video abzugleichen. Für die ersten zehn Ergebnisse der Suche werden dann URLs zurückgegeben. Die Idee hinter dieser Technik ist, dass verarbeitete Videoframes oder einzelne aus dem Video entnommene Frames einzigartig genug sind, um anderen Instanzen desselben Frames im Internet übereinzustimmen. Diese Methode kann hilfreich sein, um plagiierte Videos zu identifizieren, die online gestellt wurden, da gleichen Frames aus dem Originalvideo in der plagiierten Version erscheinen werden.

5.2.2 Textdaten durchsuchen

Die Verwendung von Textdaten zur Suche nach plagiierten Videos ist eine einfache und effektive Methode. Durch die Nutzung von Informationen, die aus in einem Video erfassten Bildern gewonnen werden, wie z. B. dem Titel des Videos oder dem Namen des Erstellers, kann diese Methode sehr effektiv sein, um plagiierte Videos zu identifizieren. Das Beibehalten des Originaltitels in plagiierten Videos zeigt, dass die Titelsuche eine schnelle und effektive Möglichkeit ist, solche Videos zu identifizieren.
Die Verwendung von Textdaten ist jedoch immer noch wertvoll, da sie die Möglichkeit bietet, den Titel des Videos offenzulegen und es einfacher zu finden. Im Wesentlichen nutzt diese Art der Forschung die Kraft der extrahierten Textdaten, um beim Identifizieren von plagiierten Videos online zu helfen.

5.2.3 Natürliche Sprachverarbeitung auf Textdaten anwenden

Die Verwendung von natürlicher Sprachverarbeitung bei der Suche nach plagiierten Videos ist eine anspruchsvollere Methode. Dies umfasst den Einsatz von Werkzeugen zur natürlichen Sprachverarbeitung, um Metatags, Titel und Namen aus Textdaten, die mit dem Originalvideo verbunden sind, zu extrahieren. Die extrahierte Information wird separat als Textstrings gespeichert, die später im Web mit einem

Python-Skript durchsucht werden können. Der Zweck der Verwendung von natürlicher Sprachverarbeitung bei der Suche nach plagiierten Videos besteht darin, ein genaueres und relevanteres Ergebnis zu erhalten, indem Nomen, Verben und Adjektive aus Textdaten extrahiert werden. Diese Methode kann effektiver sein als traditionelle textbasierte Suchmethoden, da sie den Kontext von Textdaten und die Beziehungen zwischen verschiedenen Wörtern und Phrasen berücksichtigt.

5.3 Architekturkomponenten

Abbildung 5.1 zeigt eine Programmschnittstelle, die entworfen wurde, um eine ETL-Operation durchzuführen, um Merkmale und Metadaten aus Videos als vorbereitenden Schritt zur Plagiatserkennung zu extrahieren. Die Schnittstelle umfasst mehrere Schaltflächen und Optionen, die die verschiedenen Stufen des Prozesses darstellen, einschließlich:

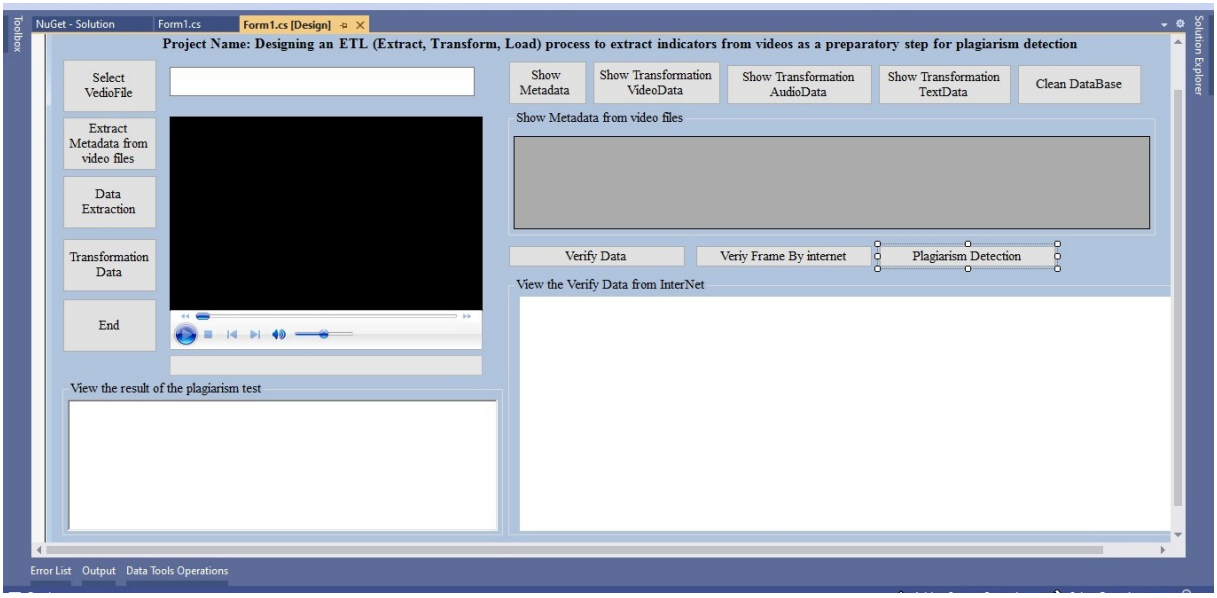


Abbildung 5.1: Programmoberfläche

1. `Select VideoFile` **(Videodatei auswählen):** Das Programm beginnt mit der Auswahl der gewünschten Videodatei.

2. `Extract Metadata from video files` **(Metadaten aus Videodateien extrahieren):** Grundlegende Metadaten werden aus ausgewählten Videodateien extrahiert.

3. `Data Extraction` **(Datenextraktion):** Grafische Elemente in Videos werden extrahiert, einschließlich visueller und auditiver Komponenten wie Frames, Text und Audio.

4. `Transformation Data` **(Transformation der Daten):** Die extrahierten Daten werden verarbeitet und in ein strukturiertes Format umgewandelt, das für die Analyse geeignet ist. Das bedeutet, diese Frames, Audio und Text in Merkmale umzuwandeln, die im Plagiatserkennungsprozess verwendet werden können, diese Merkmale werden in den folgenden Absätzen detailliert erklärt.

5. `Show Metadata` **(Metadaten anzeigen):** Die extrahierten Metadaten werden angezeigt.

6. `Show Transformation VideoData` **(Transformation der Videodaten anzeigen):** Die verarbeiteten Videodaten werden mit Schwerpunkt auf Schlüsselmerkmalen und -eigenschaften angezeigt. Hier werden alle Merkmale, die im Plagiatserkennungsprozess verwendet werden, dargestellt.

7. `Show Transformation AudioData` **(Verarbeitete Audiodaten anzeigen):** Verarbeitete Audiodaten werden präsentiert. In der Benutzeroberfläche des Programms gibt es einen Button, der die Merkmale anzeigt, die aus dem Audio, wie beispielsweise SoundMean, SoundStdDev und MFCC extrahiert wurden.

8. `Show Transformation TextData` **(Verarbeitete Textdaten anzeigen):** Verarbeitete Textdaten werden angezeigt.

9. `Clean DataBase` **(Datenbank bereinigen):** Bereinigung der Datenbank: Dieser Befehl kann verwendet werden, wenn alte Daten vorhanden sind, die wir entfernen möchten, da beim Testen von Betrug für einen Videoclip die Datenbank viele Daten für Videoclips enthalten kann, die nichts mit diesem Clip zu tun haben. Daher vergleicht das Programm nur sie und zeigt das Ergebnis in einem Feld an. Somit vergleichen wir nicht alles, um optimale Leistung und Genauigkeit der Analyse zu gewährleisten.

10. `Verify Data` **(Überprüfung des Videos anhand von Metadaten):** Wir suchen mit der Google-Suchmaschine anhand von Metadaten nach der Präsenz ähnlicher Videodateien, bevor wir mit der Plagiatserkennung fortfahren. Wenn wir Videoclips in der Google-Suchmaschine finden, laden wir diese Clips herunter und verarbeiten sie dann im Plagiatserkennungsprogramm, um die erforderlichen Merkmale zu erhalten. Anschließend verwenden wir den Plagiatserkennungsknopf im Programm, um das Erkennungsergebnis zu erfahren.

11. `Verify Frame By Internet` **(Frame im Internet überprüfen):** Wir suchen mit der Google-Suchmaschine anhand von Frames nach ähnlichen Videos.

12. `Plagiarism Detection` **(Plagiatserkennung):** Geeignete Algorithmen werden angewendet, um Videos zu erkennen, die verdächtigt werden, Fälschungen zu sein. Die vorherigen Absätze werden in den folgenden Absätzen detailliert erklärt.

In diesem Kapitel konzentrieren wir uns auf die Software-Design-Methodik, die darauf abzielt, Plagiate in Videos zu erkennen. Die Software wurde unter Verwendung der Programmiersprache C# version 9 entwickelt, und als Datenbank wurde Microsoft SQL Server Local DB 2022 verwendet. Das Programm umfasst mehrere Module.

5.3.1 Datenextraktion

In diesem Abschnitt wird der Schritt der Datenextraktion implementiert. Dieser Abschnitt ist in zwei Teile unterteilt. Der erste Teil ist fürs Extrahieren der Metadaten, der zweite Teil ist fürs Extrahieren der Frames aus dem Video.
Der erste Teil umfasst das Extrahieren von Metadaten aus der ausgewählten Videodatei. Zu den Metadaten gehören folgende Elemente: VideoID, Name, Zeit, Dauer, Codec, Auflösung, Bildrate, Audiokanäle und Autor. Diese Daten werden in einer Datenbanktabelle namens *Videos* gespeichert.
Listing 5.1 zeigt die Funktion zur Extraktion von Metadaten aus einem Videoclip:

Listing 5.1: Extraktion der Metadaten

```
1 static VideoMetadata ExtractMetadata(string videoFilePath)
2 {
3     // Create a MediaInfo object
4     MediaInfo mediaInfo = new MediaInfo();
```

```
5
6       // Open the media file
7       mediaInfo.Open(videoFilePath);
8
9       // Extract Metadata
10      var metadata = new VideoMetadata
11      {
12          Duration = mediaInfo.Get(StreamKind.Video, 0, "Duration/String3"),
13          Codec = mediaInfo.Get(StreamKind.Video, 0, "Codec/String"),
14          Resolution = $"{mediaInfo.Get(StreamKind.Video, 0,
            "Width")}x{mediaInfo.Get(StreamKind.Video, 0, "Height")}",
15          FrameRate = mediaInfo.Get(StreamKind.Video, 0, "FrameRate/String"),
16          AudioChannels = mediaInfo.Get(StreamKind.Audio, 0, "Channels"),
17          Author = mediaInfo.Get(StreamKind.General, 0, "EncodedBy")
18      };
19      // Close the media file
20      mediaInfo.Close();
21
22      return metadata;
23  }
```

Listing 5.1 illustriert, wie Metadaten aus einem Video mit C# extrahiert werden können. Die Funktion „ExtractMetadata“ erhält den Dateipfad der Videodatei als Parameter und erzeugt ein Objekt des Typs „MediaInfo“. „MediaInfo“ ist eine Klasse, die eine Schnittstelle zum Umgang mit Metadaten für Mediendateien bietet. Anschließend öffnet die Funktion die Videodatei mit der „Open“ Methode. Die Metadaten des Videos werden extrahiert und in einem Objekt des Typs „VideoMetadata“ gespeichert. Diese Daten umfassen Dauer, Codec, Auflösung, Bildrate, Audiokanäle und den Autor.

Der zweite Teil dieses Abschnitts beinhaltet das Extrahieren von Frames aus dem Video und deren Speicherung in einem Ordner auf der Festplatte C (dies wird durch den Code festgelegt) mit demselben Datei-Namen, dann das Extrahieren des Audios und dessen Speicherung im selben Ordner, gefolgt von der Transformation des Audios in das WAV-Format. Danach wird eine Nachricht angezeigt, die besagt, dass der Prozess abgeschlossen ist. Dieser Abschnitt umfasst auch mehrere andere Operationen, wie die Überprüfung, ob die Datei eine Audiodatei ist, die Bestimmung des Pfads des Ordners auf der Festplatte C, die Überprüfung der Existenz des Ordners und, falls der Ordner existiert, dessen Löschung.
Diese Module werden in einer bestimmten Reihenfolge durchgeführt, um sicherzustellen, dass der Prozess reibungslos und effizient abläuft. Die extrahierten und verarbeiteten Daten werden in der Datenbank gespeichert und für nachfolgende Schritte des Prozesses verwendet.
Listing 5.2 demonstriert die Funktion, Rahmen und Audio aus einem Videoclip zu extrahieren. Dies geschieht durch die Erstellung eines Objekts vom Typ VideoFileReader, welches eine Klasse ist, die eine Schnittstelle für den Umgang mit Videodateien bereitstellt.

Listing 5.2: Frame-Extraktion

```
1   private void ExtractData_Frame()
2   {
3       // Create a VideoCapture object for the video
4       var videoCapture = new Accord.Video.FFMPEG.VideoFileReader();
5
6       // Open the video file
7       videoCapture.Open(filePath);
```

```
8
9       // Specify the folder path on disk C
10      string directoryPath = Path.Combine("C:\\",
        Path.GetFileNameWithoutExtension(filePath));
11
12      // Extract frames from video
13      int i = 0;
14      while (true)
15      {
16          // Extract the frame
17          Bitmap frame = videoCapture.ReadVideoFrame();
18
19          // Check if the frame is null
20          if (frame == null)
21          {
22              // If the frame is null, break the loop
23              break;
24          }
25
26          // Save the frame as an image in the new folder
27          frame.Save(Path.Combine(directoryPath, $"frame{i}.bmp"));
28
29          // free up memory
30          frame.Dispose();
31      }
32  }
```

Listing 5.3 zeigt nur einen Ausschnitt von der Methode, die das Audio aus dem Video extrahiert. Es wird gezeigt, wie das extrahierte Audio mit der Funktion CreateWaveFile in das WAV-Format konvertiert wird.

Listing 5.3: Audio-Extraktion

```
1  // Convert audio to WAV format
2  WaveFileWriter.CreateWaveFile(audioFilePath, audioFileReader);
3  // Close the video file
4  videoCapture.Close();
```

5.3.2 Datentransformation

Das Modul zur Datentransformation ist zentral für die Software, da alle Plagiatsprüfungen mit den Merkmalen zusammenhängen, die aus den Videodateien extrahiert werden. Dieser Abschnitt umfasst die folgenden Stufen:

Audiokonvertierung

Diese Funktion verarbeitet die im vorherigen Modul extrahierte Audiodatei, um die folgenden Merkmale gemäß der definierten Tabellenstruktur zu erhalten:

- **FeatureID**: Ein eindeutiger Identifikator für jedes extrahierte Merkmal. Es kann verwendet werden, um Merkmale nachzuverfolgen und sie den entsprechenden Videos zuzuordnen.

- **VideoID**: Ein eindeutiger Identifikator für jedes Video. Es wird verwendet, um die extrahierten Merkmale dem Video zuzuordnen, aus dem sie extrahiert wurden.
- **SoundMean**: Der arithmetische Mittelwert der Audiodaten. Er spiegelt den Durchschnittswert des Audiosignals wider.
- **SoundStdDev**: Die Standardabweichung der Audiodaten. Es spiegelt das Ausmaß des Kontrasts oder der Streuung im Audiosignal wider.
- **MFCC (Mel Frequency Cepstral Coefficients)**: werden zur automatischen Spracherkennung eingesetzt und ermöglichen eine kompakte Darstellung des Frequenzspektrums und auch die Darstellung von Ton in Videos. MFCCs werden auch zur Analyse von Musik herangezogen. MFCCs finden ebenfalls Anwendung in der Musikanalyse [con23].
- **AudioModel**: Ein Modell, das aus Audiodaten erstellt wurde und zur Erkennung von Klängen, Wörtern oder anderen Mustern in Audio verwendet werden kann.

Diese Merkmale werden verarbeitet und in der Datenbank in einer Tabelle namens *AudioFeatures* gespeichert, um in nachfolgenden Schritten des Plagiatserkennungsprozesses verwendet zu werden.
Listing 5.4 zeigt einen Ausschnitt aus einer Methode. Die Methode konvertiert Audiodaten in ein Array von Dezimalzahlen und wendet eine Normalisierung auf den Bereich $[-1, 1]$ an. Anschließend wird ein Objekt vom Typ Signal mit den modifizierten Audiodaten und der Abtastrate der Audiodatei erstellt. Der Code extrahiert dann Audio-Merkmale mithilfe der Funktion `MelFrequencyCepstrumCoefficient` aus der Accord.NET-Bibliothek und konvertiert die Ergebnisse in ein Array von Dezimalzahlen. Abschließend berechnet der Code den Mittelwert und die Standardabweichung der Audiodaten, konvertiert die Ergebnisse in Text und gibt sie zurück.

Listing 5.4: Datentransformation

```
1   // Convert the audio data to float array and normalize to the range
    [-1, 1]
2   float[] audioDataFloat = new float[bytesRead / 2];
3   for (int i = 0, j = 0; i < bytesRead; i += 2, j++)
4   {
5       short sample = BitConverter.ToInt16(buffer, i);
6       audioDataFloat[j] = sample / 32768.0f;
7   }
8   // Create Accord.Audio.Signal object
9   var signal = Signal.FromArray(audioDataFloat,
    audioFileReader.WaveFormat.SampleRate);
10
11  // Extract sound features using Accord.NET
12  var mfcc = new MelFrequencyCepstrumCoefficient();
13  var descriptors = mfcc.Transform(signal);
14
15  // Convert the descriptors to double array
16  mfccResult = descriptors.Select(x => x.Descriptor.Select(y =>
    (double)y).ToArray()).ToArray();
17
18  // Calculate the mean and standard deviation of the audio data
19  for (int i = 0; i < audioDataFloat.Length; i++)
20  {
21      float sample = audioDataFloat[i];
```

```
22          sum += sample;
23          sumOfSquares += sample * sample;
24          count++;
25      }
26  }
27  double mean = sum / count;
28  double standardDeviation = Math.Sqrt(sumOfSquares / count - mean * mean);
29  string mfccResult1 = string.Join(", ", mfccResult.Select(x =>
    string.Join(", ", x)));
30  // Return the mean, standard deviation, and MFCC coefficients as text
31  return (mean, standardDeviation, mfccResult1);
```

Transformation des Textes

Diese Stufe beinhaltet die Transformation der Audiodatei in Text und dessen Speicherung gemäß der folgenden Tabellenstruktur:

- **TextID**: Ein eindeutiger Identifikator für jeden extrahierten Text. Er kann verwendet werden, um Texte zu verfolgen und sie den entsprechenden Videos zuzuordnen.
- **VideoID**: Ein eindeutiger Identifikator für jedes Video. Er wird verwendet, um die extrahierten Texte dem Video zuzuordnen, aus dem sie extrahiert wurden.
- **Text**: Der aus der Audiodatei extrahierte Text.

Diese Merkmale werden verarbeitet und in der Datenbank in der Tabelle *Texts* gespeichert, um in nachfolgenden Schritten des Prozesses verwendet zu werden. Dieser Prozess erfordert fortgeschrittene Techniken im Bereich der automatischen Spracherkennung, um Audio in Text umzuwandeln. Dies geschieht durch die folgenden Schritte:

1. **Einrichten der Spracherkennungs-Engine:** Die Spracherkennungs-Engine wird mit dem IETF-Sprach-Tag „en-US“ eingerichtet. Dies bedeutet, dass die Engine darauf ausgelegt ist, englischsprachige Eingaben zu erkennen. Es können auch weitere IETF-Sprach-Tags verwendet werden. Der verwendete Wert „en-US“ steht dabei für die englische Sprache, wie in Listing 5.5 dargestellt.

Listing 5.5: Erkennen von englischsprachigen Eingaben

```
1  using (var recognizer = new SpeechRecognitionEngine(new
   System.Globalization.CultureInfo("en-US")))
```

2. **Laden der Sprachgrammatik:** Die Sprachgrammatik wird mit DictationGrammar geladen. Das bedeutet, dass die Engine die Sprache erkennen wird, die als Teil der natürlichen Sprache präsentiert wird.
3. **Zuweisen von Eingaben an die Engine:** Die Eingaben werden der Engine anhand der Audiodatei zugewiesen, die der Funktion als Parameter übergeben wird.
4. **Implementierung der Spracherkennung:** Die Spracherkennung wird durchgeführt und die Ergebnisse werden gesammelt.
5. **Ergebnisse sammeln:** Die Ergebnisse werden gesammelt und mit string.Join in Text umgewandelt.

6. **Überprüfung der Ergebnisse:** Wenn der Text leer ist, wird der Wert auf **null** gesetzt.

7. **Speichern des Textes in der Datenbank:** Es wird eine Verbindung zur Datenbank hergestellt und eine SQL-Anfrage ausgeführt, um die VideoID aus der Tabelle Videos anhand des Videonamens zu ermitteln. Wenn das Video gefunden wird, wird eine weitere SQL-Anfrage ausgeführt, um den Text in die Tabelle *Texte* einzufügen.

   Die in Listing 5.6 verwendeten Bibliotheken sind System.Globalization zur Behandlung von IETF-Sprach-Tags, System.Data.SqlClient zur Behandlung von SQL-Datenbanken und System.Speech.Recognition zur Behandlung von Spracherkennung. Diese Bibliotheken werden verwendet, um verschiedene Operationen durchzuführen, die für die Extraktion von Text aus Videos nötig sind.

   Listing 5.6 konvertiert die Sprache in der Videodatei mithilfe der .NET-Spracherkennungsbibliothek in Text. Dies wird durch das Erstellen eines Objekts vom Typ „SpeechRecognitionEngine“ erreicht, welches eine Klasse ist, die eine Schnittstelle für den Umgang mit Spracherkennungsdiensten bietet. „en-US“ wird als Parameter an den Konstruktor übergeben, der die für die Spracherkennung zu verwendende Sprache spezifiziert. Dann wird die Spracherkennungsgrammatik mit der Funktion „LoadGrammar“ geladen. Hier wird „DictationGrammar“ verwendet, was bedeutet, dass der erwartete Text freier Text ist und nicht auf einen bestimmten Satz von Wörtern oder Phrasen beschränkt ist. Die Videodatei wird dann als Eingabequelle mit der Funktion „SetInputToWaveFile“ ausgewählt. Die Sprache in der Videodatei wird mit der Funktion „Recognize“ erkannt. Diese Funktion analysiert Audiodaten und konvertiert sie in Text. Anschließend werden die Ergebnisse mit der Funktion „string.Join“ in Text umgewandelt. Diese Funktion fasst alle erkannten Texte zu einem einzigen Text zusammen, wie in Listing 5.6 dargestellt.

Listing 5.6: Spracherkennungs-Engine

```
1 using (var recognizer = new SpeechRecognitionEngine(new
  System.Globalization.CultureInfo("en-US")))
2             {
3     recognizer.LoadGrammar(new DictationGrammar());
4     recognizer.SetInputToWaveFile(videoFilePath);
5     var result = recognizer.Recognize();
6     var fullText = string.Join(" ", result.Alternates.Select(a =>
      a.Text));
```

Bildtransformation

Nachdem Frames aus der Videodatei extrahiert wurden, wandeln wir diese Frames in Merkmale um und speichern sie gemäß der folgenden Struktur in einer Tabelle:

- **FeatureID**: Ein eindeutiger Identifikator für jedes extrahierte Merkmal.
- **VideoID**: Ein eindeutiger Identifikator für jedes Video. Er wird verwendet, um die extrahierten Merkmale dem Video zuzuordnen, aus dem sie extrahiert wurden.
- **AverageR, AverageG, AverageB**: Dies sind die Durchschnittswerte der roten, grünen und blauen Farben im Frame.
- **BlobArea**: Die Summe der Flächen aller Blöcke im Frame.
- **BlobPerimeter**: Die Summe der Umfänge aller Blöcke im Frame. Ein **Block** bezieht sich üblicherweise auf eine Gruppe benachbarter Pixel in einem Frame, die ein bestimmtes Objekt oder Gebiet

bilden. Diese Blöcke können verschiedene Teile des Bildes darstellen, wie Objekte, Personen oder Hintergrundbereiche. Daher ist der **BlobPerimete** die Summe der Umfänge all dieser Blöcke im Frame. Der **Perimeter** wird berechnet, indem die Länge der Kanten für jedes Pixel an den äußeren Grenzen des Blocks addiert wird.

- **KeyPointsCount**: Die Anzahl der Schlüsselpunkte im Frame.
- **ImageModel**: Ein aus Bilddaten erstelltes Modell, das zur Erkennung anderer Muster im Bild verwendet werden kann.

Diese Merkmale werden verarbeitet und in der Datenbank in der Tabelle „Features“ für die Verwendung in nachfolgenden Schritten des Prozesses gespeichert. In Listing 5.7, Listing 5.8 und Listing 5.9 sind Implementierungen der Funktionen „ExtractColorFeatures“, „ExtractShapeFeatures“ , „CalculatePerimeter“ und „CalculateDistance“ , welche die folgenden Schritte durchführen:

1. **ExtractColorFeatures**: Diese Funktion berechnet die Durchschnittswerte der roten, grünen und blauen Farben im Frame.
2. **ExtractShapeFeatures**: Diese Funktion berechnet die Summe der Flächen und die Summe der Umfänge aller Blöcke im Frame.
3. **CalculatePerimeter**: Diese Funktion berechnet den Umfang des Blocks anhand der Randpunkte.
4. **CalculateDistance**: Diese Funktion berechnet den Abstand zwischen zwei Punkten.

So vervollständigt dieses Softwaremodul die Arbeit der Transformation der Videodatei in Merkmale, die im Plagiatsprozess verwendet werden können.
Listing 5.7 zeigt die Methode „ExtractShapeFeatures“, die dazu dient, durchschnittliche RGB-Farbmerkmale aus einem übergebenen Bildrahmen (Bitmap) zu extrahieren. Der Code durchläuft jedes Pixel im Bild, sammelt die Gesamtsumme der Rot-, Grün- und Blauwerte und berechnet anschließend den Durchschnitt dieser Werte.
Die Funktion wandelt Formmerkmale aus dem Rahmen um. Dies wird durchgeführt, indem der Rahmen in ein monochromes Bild umgewandelt wird, dann wird „BlobCounter“ verwendet, um die Regionen (Blobs) im Bild zu extrahieren. Anschließend werden die Gesamtfläche und der Gesamtumfang aller Flächen berechnet.

Listing 5.7: Farbmerkmale aus einem übergebenen Bildrahmen zu extrahieren

```
1  private (int, int, int) ExtractColorFeatures(Bitmap frame)
2  {
3      long totalR = 0, totalG = 0, totalB = 0;
4      int pixelCount = frame.Width * frame.Height;
5      var bitmapData = frame.LockBits(new Rectangle(0, 0, frame.Width,
       frame.Height), ImageLockMode.ReadOnly, frame.PixelFormat);
6
7      unsafe
8      {
9          byte* ptrFirstPixel = (byte*)bitmapData.Scan0;
10         for (int y = 0; y < frame.Height; y++)
11         {
12             byte* currentLine = ptrFirstPixel + (y * bitmapData.Stride);
13             for (int x = 0; x < frame.Width; x++)
14             {
15                 totalB += currentLine[x * 4];
```

```
                totalG += currentLine[x * 4 + 1];
                totalR += currentLine[x * 4 + 2];
            }
        }
    }

    frame.UnlockBits(bitmapData);
    int averageR = (int)(totalR / pixelCount);
    int averageG = (int)(totalG / pixelCount);
    int averageB = (int)(totalB / pixelCount);

    return (averageR, averageG, averageB);
}
```

In Listing 5.8 ist die Methode ExtractShapeFeatures dargestellt und sie dient dazu, Formmerkmale aus einem gegebenen Bildrahmen (Bitmap) zu extrahieren. Der Code wandelt das Bild zunächst in ein monochromes Bild um und nutzt dann den BlobCounter, um Informationen über Bereiche (Blobs) im Bild zu extrahieren. Abschließend berechnet er Fläche und Umfang der identifizierten Blobs.

Listing 5.8: Formmerkmale aus einem gegebenen Bildrahmen zu extrahieren

```
private (double, double) ExtractShapeFeatures(Bitmap frame)
{
     //Convert the frame to a monochrome frame
    Bitmap grayFrame = Grayscale.CommonAlgorithms.BT709.Apply(frame);

    // Use BlobCounter to extract area information (blobs) in an image
    BlobCounter blobCounter = new BlobCounter();
    blobCounter.ProcessImage(grayFrame);
    Blob[] blobs = blobCounter.GetObjectsInformation();

    // Use shape features for extracted objects (blobs)
    double totalArea = 0;
    double totalPerimeter = 0;
    foreach (Blob blob in blobs)
    {
        // You can use Blob  properties to get shape features
        double area = blob.Area;
        totalArea += area;

        // Calculate the perimeter of a blob manually using edge points
        List<IntPoint> edgePoints = blobCounter.GetBlobsEdgePoints(blob);
        double perimeter = CalculatePerimeter(edgePoints);
        totalPerimeter += perimeter;
    }

    return (totalArea, totalPerimeter);
}
```

In Listing 5.9 sind die beiden Methoden CalculatePerimeter und CalculateDistance bereitgestellt. Diese beiden Methoden sind Teile eines Bildverarbeitungsprozesses, der darauf abzielt, den Umfang eines

Objekts (Blob) zu berechnen, indem die Entfernung zwischen aufeinanderfolgenden Punkten entlang des Randes des Objekts gemessen wird. Hier ist eine detaillierte Beschreibung und Aufschlüsselung dieser Methoden:

CalculatePerimeter Diese Methode berechnet den Umfang eines Objekts (Blob), indem sie die Entfernungen zwischen allen aufeinanderfolgenden Kantenpunkten summiert. Zum Abschluss der Umfangsberechnung wird die Entfernung zwischen dem letzten Punkt und dem ersten Punkt der Liste hinzugefügt, um die Schleife zu schließen und den Umfang vollständig zu erfassen.

CalculateDistance Diese Funktion berechnet die euklidische Distanz zwischen zwei Punkten. Die Distanz wird mittels des Satzes des Pythagoras berechnet, wobei deltaX und deltaY die Differenzen der X- und Y-Koordinaten der beiden Punkte sind.

Listing 5.9: Berechnung des Umfangs und der euklidischen Distanz geometrischer Formen

```
1  // Function to calculate the perimeter of a Blob using edge points
2  private double CalculatePerimeter(List<IntPoint> edgePoints)
3  {
4      double perimeter = 0;
5
6      for (int i = 0; i < edgePoints.Count - 1; i++)
7      {
8       perimeter += CalculateDistance(edgePoints[i], edgePoints[i + 1]);
9      }
10
11     // Closing the loop by calculating the distance between the last point
       and the first point
12     perimeter += CalculateDistance(edgePoints[edgePoints.Count - 1],
       edgePoints[0]);
13
14     return perimeter;
15 }
16
17 // A function to calculate the distance between two points
18 private double CalculateDistance(IntPoint point1, IntPoint point2)
19 {
20     int deltaX = point2.X - point1.X;
21     int deltaY = point2.Y - point1.Y;
22
23     return Math.Sqrt(deltaX * deltaX + deltaY * deltaY);
24 }
```

5.3.3 Datenladen

Der Zweck dieses Moduls besteht darin, die umgewandelten Daten aus der Videodatei in einer Datenbank zu speichern. Die Datenbank besteht aus den folgenden Tabellen.

1. **AudioFeatures-Tabelle**: Sie enthält die aus dem Video extrahierten Audio-Merkmale. Die Tabelle umfasst die folgenden Spalten: FeatureID, VideoID, SoundMean, SoundStdDev, MFCC und AudioModel.

Listing 5.10: AudioFeatures

```
CREATE TABLE [dbo].[AudioFeatures] (
    [FeatureID]   INT             IDENTITY (1, 1) NOT NULL,
    [VideoID]     INT             NOT NULL,
    [SoundMean]   FLOAT (53)      NOT NULL,
    [SoundStdDev] FLOAT (53)      NOT NULL,
    [MFCC]        VARCHAR (MAX)   NOT NULL,
    [AudioModel]  VARBINARY (MAX) NULL,
    PRIMARY KEY CLUSTERED ([FeatureID] ASC),
    FOREIGN KEY ([VideoID]) REFERENCES [dbo].[Videos] ([VideoID])
);
```

2. **Features-Tabelle**: Sie enthält die aus den Frames im Video extrahierten Merkmale. Die Tabelle umfasst die folgenden Spalten: FeatureID, VideoID, AverageR, AverageG, AverageB, BlobArea, BlobPerimeter, KeyPointsCount und ImageModel.

Listing 5.11: Features

```
CREATE TABLE [dbo].[Features] (
    [FeatureID]      INT             IDENTITY (1, 1) NOT NULL,
    [VideoID]        INT             NOT NULL,
    [AverageR]       INT             NOT NULL,
    [AverageG]       INT             NOT NULL,
    [AverageB]       INT             NOT NULL,
    [BlobArea]       FLOAT (53)      NOT NULL,
    [BlobPerimeter]  FLOAT (53)      NOT NULL,
    [KeyPointsCount] INT             NOT NULL,
    [ImageModel]     VARBINARY (MAX) NULL,
    PRIMARY KEY CLUSTERED ([FeatureID] ASC),
    FOREIGN KEY ([VideoID]) REFERENCES [dbo].[Videos] ([VideoID])
);
```

3. **Forgery-Tabelle**: enthält Informationen über Videos, die als Plagiat erkannt wurden. Die Tabelle umfasst die folgenden Spalten: ForgeryID, OriginalVideoID und SimilarityScore.

Listing 5.12: Forgery

```
CREATE TABLE [dbo].[Forgery] (
    [ForgeryID]       INT IDENTITY (1, 1) NOT NULL,
    [OriginalVideoID] INT NOT NULL,
    [SimilarityScore] INT NOT NULL,
    PRIMARY KEY CLUSTERED ([ForgeryID] ASC),
    FOREIGN KEY ([OriginalVideoID]) REFERENCES [dbo].[Videos]
    ([VideoID])
);
```

4. **Relationships-Tabelle**: Sie enthält Informationen über die Beziehungen zwischen Videos. Die Tabelle umfasst die folgenden Spalten: RelationshipID (der eindeutige Identifikator der Beziehung), Video1ID (der eindeutige Identifikator des ersten Videos), Video2ID (der eindeutige Identifikator des zweiten Videos) und RelationshipType (der Typ der Beziehung).

Listing 5.13: Relationships

```
CREATE TABLE [dbo].[Relationships] (
    [RelationshipID]   INT            IDENTITY (1, 1) NOT NULL,
    [Video1ID]         INT            NOT NULL,
    [Video2ID]         INT            NOT NULL,
    [RelationshipType] NVARCHAR (255) NOT NULL,
    PRIMARY KEY CLUSTERED ([RelationshipID] ASC),
    FOREIGN KEY ([Video1ID]) REFERENCES [dbo].[Videos] ([VideoID]),
    FOREIGN KEY ([Video2ID]) REFERENCES [dbo].[Videos] ([VideoID])
);
```

5. **Texts-Tabelle**: Sie enthält die aus dem Video extrahierten Texte. Die Tabelle umfasst die folgenden Spalten: TextID, VideoID und Text.

Listing 5.14: Texts

```
CREATE TABLE [dbo].[Texts] (
    [TextID]  INT            IDENTITY (1, 1) NOT NULL,
    [VideoID] INT            NOT NULL,
    [Text]    NVARCHAR (MAX) NOT NULL,
    PRIMARY KEY CLUSTERED ([TextID] ASC),
    FOREIGN KEY ([VideoID]) REFERENCES [dbo].[Videos] ([VideoID])
);
```

6. **Videos-Tabelle**: enthält Informationen über Videos. Die Tabelle umfasst die folgenden Spalten: VideoID, Name, Zeit, Dauer, Codec, Auflösung, FrameRate, AudioChannels und Autor.

Listing 5.15: Videos

```
CREATE TABLE [dbo].[Videos] (
    [VideoID]       INT            IDENTITY (1, 1) NOT NULL,
    [Name]          NVARCHAR (MAX) NOT NULL,
    [Time]          DATETIME       NOT NULL,
    [Duration]      NVARCHAR (50)  NULL,
    [Codec]         NVARCHAR (50)  NULL,
    [Resolution]    NVARCHAR (20)  NULL,
    [FrameRate]     NVARCHAR (20)  NULL,
    [AudioChannels] NVARCHAR (20)  NULL,
    [Author]        NVARCHAR (MAX) NULL,
    PRIMARY KEY CLUSTERED ([VideoID] ASC)
);
```

Dieser Schritt ist trotz seiner großen Bedeutung nicht unabhängig von den vorgerigen Schritten. Dies liegt daran, dass wir dieses Modul mit dem vorherigen Modul kombiniert haben, wo die umgewandelten Daten unmittelbar nach ihrer Gewinnung in den dafür vorgesehenen Tabellen gespeichert werden.
Damit endet der vorherige ETL-Prozess. Natürlich können diese vorherigen Schritte bei Bedarf weiterentwickelt und modifiziert werden.

Kapitel 6

Ergebnisse

In diesem Kapitel stellen wir eine Methode zur Erkennung von Videos, die Plagiate enthalten, vor und wie man Ähnlichkeitsergebnisse zwischen Videodateien mittels der folgenden Methodik berechnet:

6.1 Suche nach ähnlichen Videodateien im Internet

Der erste Schritt in diesem Prozess besteht darin, online nach einer ähnlichen Videodatei zu suchen, indem nach Videodateien gesucht wird, die dieselben Metadaten enthalten.
Der in Abschnitt A.2 angehängte Code implementiert die Funktion, die die folgenden Schritte durchführt:

1. Der Dateiname ohne die Erweiterung wird mit der Funktion `Path.GetFileNameWithoutExtension` bestimmt.
2. Eine Verbindung zur Datenbank wird geöffnet und eine SQL-Anfrage ausgeführt, um den Namen, die Dauer und den Codec des Videos aus der Tabelle Videos zu erhalten.
3. Die Internetsuche wird mit dem Namen, der Dauer und dem Codec des Videos durchgeführt.
4. Suchergebnisse werden in `richTextBox1` angezeigt. Es wird angegeben, dass ein passendes Video basierend auf den Videometadaten gefunden wurde.

Die Funktion `SearchOnline` führt eine Internetsuche mit `HttpClient` durch. Die Suchanfrage wird mit dem Videonamen, der Dauer und dem Codec gebildet und dann an Google gesendet. Wenn die Antwort erfolgreich ist, werden Suchergebnisse zurückgegeben. Falls keine Ergebnisse gefunden werden, wird eine leere Liste zurückgegeben. Dem Benutzer wird dann eine Nachricht angezeigt, dass keine Ergebnisse gefunden wurden.
Die umgewandelten Daten werden in der Datenbank gespeichert, um in nachfolgenden Schritten des Prozesses (Plagiatserkennung) verwendet zu werden. Dieser Prozess erfordert fortgeschrittene Techniken im Bereich der Internetsuche und der Datenbankbearbeitung.
Wenn wir eine Videodatei mit demselben Namen und denselben Metadaten bei Google finden, beginnen wir den Vergleichsprozess gemäß den folgenden Schritten.

6.1.1 Vergleich basierend auf Metadaten

Diese Art des Vergleichs wird durchgeführt, indem das erste Video aus der *Videotabelle* ausgewählt und der Vergleich mit den Videos in der Datenbank auf Basis der Metadaten gestartet wird. Wenn ein Video mit denselben Metadaten gefunden wird, werden die Namen beider Videos in der Tabelle *Relationships* gespeichert, wo ähnliche Daten im Feld *RelationshipType* gespeichert werden. Zusätzlich

wird der Name des ersten Videos in der Tabelle *Forgery* gespeichert, wo der Ähnlichkeitsprozentsatz im Feld *SimilarityScore* gespeichert wird. In diesem Fall werden die fünf Spalten verglichen (Dauer, Codec, Auflösung, Bildrate und Audiokanäle), damit der Wert in jeder Spalte bestimmt wird. Dabei gilt, die Werte einer Spalte gleich sind, dann hat fügt die Spalte 20% zum Endergebnis hinzu, ansonsten 0%. Das Endergebnis ist dann zwischen 0% und 100%. Diese Methode stellt sicher, dass jede Spalte im Endergebnis gleich bewertet wird. Das Ergebnis wird in *richTextBox1* angezeigt. Z. B. wenn sich die Codec, Auflösung, Bildrate und Audiokanäle in beiden Videos ähneln, aber nicht die Dauer, dann ist das Endergebnis 80%.

Weiter zum nächsten Video: Nachdem der Vergleich mit dem ersten Video abgeschlossen ist, geht das Programm zum nächsten Video und wiederholt den Prozess bis zum Ende der Videos in der Videos-Tabelle.
Dieser Prozess erfordert den Einsatz fortgeschrittener Techniken im Bereich der Internetrecherche und der Datenbankbearbeitung, um Genauigkeit bei der Erkennung von Plagiaten zu erreichen.
Um diesen Prozess durchzuführen, folgen wir den folgenden Schritten:

1. Datenbankabfrage für alle Videos: Die Funktion *GetAllVideos* wird aufgerufen, um alle Videos aus der Datenbank abzurufen.

2. Jedes Video mit jedem anderen Video vergleichen: Der Prozess wird für jedes Video in der Liste wiederholt und mit jedem anderen Video in der Liste verglichen.

3. Berechnung der Ähnlichkeit zwischen Videos: Die Funktion *CalculateSimilarity* wird aufgerufen, um den Grad der Ähnlichkeit zwischen den beiden ausgewählten Videos zu berechnen.

4. Aktualisierung der Beziehungs- und Stellvertretungstabellen, falls die Videos ähnlich sind:

   Die Funktionen *UpdateRelationshipsTable* und *UpdateForgeryTable* werden aufgerufen, um die entsprechenden Tabellen in der Datenbank zu aktualisieren, wobei:

   - *UpdateForgeryTable*: Diese Funktion aktualisiert die Plagiatstabelle in der Datenbank, indem der eindeutige Identifikator des Originalvideos und der Ähnlichkeitswert hinzugefügt werden.
   - *UpdateRelationshipsTable*: Diese Funktion aktualisiert die Beziehungstabelle in der Datenbank, indem der eindeutige Identifikator für das erste und zweite Video sowie der Beziehungstyp hinzugefügt werden.

5. Anzeige der Ergebnisse in *richTextBox1*: Ähnlichkeitsergebnisse zwischen Videos werden in *richTextBox1* angezeigt.

6.1.2 Vergleich basierend auf den Merkmalen der Audiodatei

Diese Stufe beginnt mit einem Vergleich basierend auf den Daten in der Tabelle *AudioFeatures* in der Datenbank, d. h. basierend auf diesen Spalten:

- **SoundMean**: Es handelt sich um den arithmetischen Mittelwert des Tons.
- **SoundStdDev**: Es ist die Standardabweichung des Tons.
- **MFCC**: Mel Frequency Cepstral Coefficients.

Um Plagiate und Ähnlichkeiten zwischen Videos zu bestimmen, verwenden wir eine Methodik, die die absolute Differenz zwischen den Werten von *SoundMean* und *SoundStdDev* für beide Videos berechnet und dann die *MFCC*-Werte für beide Videos vergleicht. Die Unterschiede werden zu einer einzigen Ähnlichkeitsbewertung zusammengefasst. Wenn die *MFCC*-Werte nicht ähnlich sind, wird der Ähnlichkeitsgrad reduziert. Das Ergebnis wird in *richTextBox1* angezeigt. Danach gehen wir zum nächsten Schritt des Vergleichsprozesses über.

6.1.3 Vergleich basierend auf Frame-Merkmalen

In dieser Stufe vergleichen wir basierend auf den Daten in der Tabelle *Features* in der Datenbank. Basierend auf den folgenden Spalten:

- **AverageR, AverageG, AverageB**: Dies sind die Durchschnitte der Werte der roten, grünen und blauen Farben jeweils im Frame.
- **BlobArea**: Es ist die Summe der Flächen aller Blöcke im Frame.
- **BlobPerimeter**: Es ist die Summe der Umfänge aller Blöcke im Frame.
- **KeyPointsCount**: Die Anzahl der Schlüsselpunkte im Frame.
- **ImageModel**: Ein aus Bilddaten erstelltes Modell, das zur Erkennung anderer Muster im Bild verwendet werden kann.

Um den Prozentsatz von Plagiaten und Ähnlichkeiten zwischen Videodateien zu bestimmen, verwenden wir die folgende Beziehung:

$$\text{Plagiatsrate} = \frac{\text{Anzahl identischer Frames}}{\text{Gesamtzahl der Frames im zu überprüfenden Video}} \cdot 100 \tag{6.1}$$

Das Ergebnis wird in *richTextBox1* angezeigt. Danach gehen wir zum nächsten Schritt des Vergleichsprozesses über.

6.1.4 Vergleich basierend auf Textmerkmalen

In dieser Phase vergleichen wir basierend auf den Daten in der Tabelle *Texts* in der Datenbank, d. h. basierend auf der folgenden Spalte:

- **Text**: Es ist der Text, der aus dem Video extrahiert wurde.

Um Plagiate und Ähnlichkeiten zwischen Videodateien zu bestimmen, verwenden wir einen Code, der Texte in Wörter unterteilt, dann die Anzahl der gemeinsamen Wörter im zu überprüfenden Text zählt. Die Gesamtanzahl der Wörter wird gezählt, und dann wird die Ähnlichkeit als Prozentsatz der gemeinsamen Wörter berechnet. Das Ergebnis wird in *richTextBox1* angezeigt.

$$\text{Ähnlichkeit} = \frac{\text{Anzahl identischer Wörter}}{\text{Gesamtzahl der Wörter im zu überprüfenden Text}} \cdot 100 \tag{6.2}$$

Damit endet der Vergleichsprozess, der wie folgt zusammengefasst werden kann:

1. Die Funktionen *GetAllVideos()*, *GetAllAudioFeatures()*, *GetAllTexts()* und *GetAllFeatures()* werden aufgerufen, um alle Video-, Audio-, Text- und Frame-Merkmale aus der Datenbank abzurufen.
2. Zwei verschachtelte Schleifen werden ausgeführt, um jedes Video mit allen anderen zu vergleichen.
3. Die Ähnlichkeit zwischen jedem Paar von Videos wird basierend auf Metadaten, Audio, Text und Frame-Merkmalen berechnet.
4. Wenn eine Ähnlichkeit zwischen den Videos besteht (d. h. einer der Ähnlichkeitswerte größer als Null ist), werden die Tabelle *Relationships* und die Tabelle *Forgery* mit den berechneten Werten aktualisiert.
5. Die Ergebnisse werden in *richTextBox1* angezeigt. Enthält Informationen über die Ähnlichkeit zwischen einem Paar von Videos. Es beinhaltet:

- Name des ersten und zweiten Videos.
- Der Grad der Ähnlichkeit zwischen den Videos basierend auf Metadaten, der als Prozentsatz ausgedrückt wird.
- Der Grad der Ähnlichkeit zwischen den Videos basierend auf dem Audio, der als Prozentsatz ausgedrückt wird.
- Der Grad der Ähnlichkeit zwischen den Videos basierend auf dem Text, der als Prozentsatz ausgedrückt wird.
- Der Grad der Ähnlichkeit zwischen den Videos basierend auf Frame-Merkmalen, ausgedrückt als Prozentsatz.

Hinweis: Bei der Verwendung dieser Art von Analyse ist Vorsicht geboten. Es kann viele Faktoren geben, die berücksichtigt werden müssen, um zu bestimmen, ob tatsächlich ein Plagiat vorliegt. Beispielsweise können Videos aufgrund der Verwendung von Schlagworten oder gemeinsamen Themen zu ähnlich sein, aber nicht aufgrund von Plagiaten. Daher sollten diese Werkzeuge sorgfältig und vernünftig eingesetzt werden.
Die Ergebnisse dieser Verhältnisse können in einem Projekt zur Plagiatserkennung verwendet werden. Eine höhe Ähnlichkeit könnte zwischen Videos auf mögliches Plagiat hinweisen.

6.1.5 Vergleich basierend auf Suchen in Suchmaschinen

Diese Vergleichsmethode behandelt eine Strategie zur Überprüfung von Originalität und Ähnlichkeit zwischen Videodateien unter Verwendung der umgekehrten Bildersuche online. Der Prozess beinhaltet das Extrahieren spezifischer Frames aus dem Video und deren Online-Suche, um zu bestimmen, ob sie in anderen Videos vorhanden sind.
Beispielsweise kann man mit einem Bild zu Beginn des Videoclips, einem Bild in der Mitte und einem Bild am Ende suchen. Wenn eine genauere Untersuchung erforderlich ist, kann man alle aus der Videodatei extrahierten Bilder in Suchmaschinen suchen.
Im Folgenden sind die Schritte zur Implementierung dieser Methodik aufgeführt:

1. Spezifiziert den Dateipfad, den Dateinamen und die Dateierweiterung aus `textpathVideo.Text`.
2. Spezifiziert den Verzeichnispfad, der die extrahierten Frames enthält.
3. Wenn das Verzeichnis existiert, erhält es die Liste der Frame-Dateipfade.
4. Überprüft, ob genügend Frames vorhanden sind. Wenn nicht, wird eine Warnmeldung angezeigt und die Ausführung gestoppt.
5. Auswahl der Frames (erster, mittlerer und letzter): Um den Suchprozess in der Google-Suchmaschine durchzuführen, wird der erste Frame ausgewählt und nach Videos gesucht, in denen dieser Frame vorkommt, ebenso wie der Frame in der Mitte und der Frame am Ende. Außerdem kann der Nutzer des Programms jeden Frame auswählen, den er für geeignet oder charakteristisch für diesen Videoclip hält, um den Suchprozess durchzuführen.

Danach kann jedes Video, das mit den Frames übereinstimmt, aus dem Internet heruntergeladen und der oben genannte Vergleichsprozess durchgeführt werden. Diese Methode kann nützlich sein, um gefälschte oder ähnliche Videos zu erkennen.
Hinweis: Die Suche muss manuell erfolgen, was bedeutet, dass der Benutzer des Programms die Bilder (Frame) auswählt, nach denen er suchen möchte, und wenn er eine Videodatei findet, die übereinstimmt oder ihr nahe kommt, lädt er diese Videodatei herunter und beginnt dann mit dem Vergleichen von Videodateien, da die automatische Suche mit dem Programm blockiert wurde.

6.2 Ergebnisanzeige des Plagiatstests

Nach Durchführung des Vergleichsprozesses mit der Plagiatserkennungssoftware erscheinen die Ergebnisse als Prozentsätze, die die Ähnlichkeit zwischen den Videoclips anzeigen, wie folgt:
Die Ergebnisse der Suche nach Videoclips erscheinen unter Verwendung von Metadaten. Das Programm vergleicht die Metadaten der ausgewählten Videodatei mit allen Videoclips, deren Metadaten in der Datenbank gefunden werden, berechnet das Ergebnis, wie im vorherigen Kapitel erklärt, und gibt das entsprechende Ergebnis als Prozentsatz an, was bedeutet, je größer der Prozentsatz, desto größer die Ähnlichkeit.

Das Programm fährt dann fort, den Prozentsatz der Ähnlichkeit in Bezug auf die Merkmale der Audiodatei, die aus der Videodatei extrahiert wurde, zu berechnen und gibt den Ähnlichkeitswert als Prozentsatz an. Je höher dieser Prozentsatz, desto größer die Ähnlichkeit.
Anschließend geht das Programm dazu über, den Prozentsatz der Ähnlichkeit in Bezug auf den Text zu berechnen, da es das Audio in Text umwandelt und den Text mit den restlichen in der Datenbank gespeicherten Videodateien vergleicht und den Ähnlichkeitswert als Prozentsatz angibt. Je höher dieser Prozentsatz, desto größer die Ähnlichkeit.
Das Gleiche gilt für die Angabe des Ähnlichkeitswertes durch den Vergleich der Merkmale von Frames, die aus der Videodatei extrahiert wurden, wie in Abbildung 6.1 dargestellt.

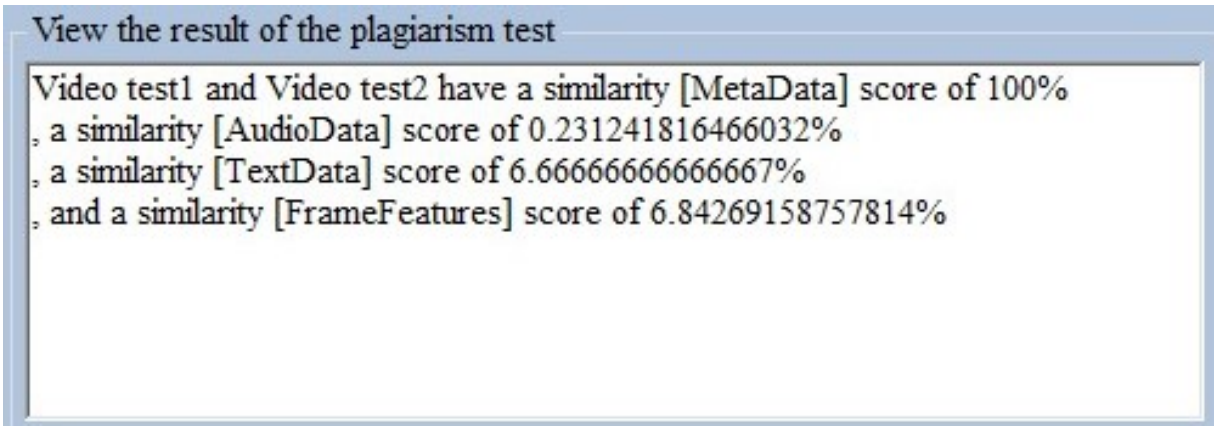


Abbildung 6.1: Ergebnisse

6.3 Ergebnisse des ETL-Prozesses

In dieser Arbeit wurde ein ETL-Prozess zur Plagiatserkennung in Videoinhalten entwickelt. Die Forschung zeigte, dass dieser Prozess für die Forschungsfrage geeignet ist, indem die Hauptfunktionalitäten erfolgreich implementiert wurden. In der Umsetzung wurden nur einfache Ähnlichkeitsmethoden verwendet, um einen Proof-of-Concept des ETL-Prozesses zu demonstrieren. Allerdings besteht die Möglichkeit, das Programm um komplexere Ähnlichkeitsmethoden zu erweitern. Die Erweiterbarkeit des ETL-Prozesses wurde als wichtige Eigenschaft bei seiner Konzeption betrachtet. Es wurde jedoch festgestellt, dass der ETL-Prozess nicht gut skalierbar ist, da die Videos linear verglichen werden. Dies könnte in zukünftigen Arbeiten verbessert werden. Darüber hinaus traten Probleme bei der Implementierung einer Schnittstelle zur Google-Suche auf, da das Programm nach mehreren Bildsuchanfragen von Google blockiert wurde. Es besteht die Notwendigkeit, alternative Ansätze zur Suche nach Internetvideos zu erforschen.

Kapitel 7

Zusammenfassung und Ausblick

Soziale Medien und die digitale Welt expandieren heute rasch. Im Internet werden täglich viele Videos veröffentlicht. Da viele Zugang zum Internet haben, ist das Herunterladen und erneute Veröffentlichen von Inhalten einfach. Plagiat tritt auf, wenn jemand ein Video aus dem Internet stiehlt und es unter seinem Namen veröffentlicht. In dieser Forschung extrahieren wir Merkmale zur Erkennung von Plagiaten in Videos. Dabei fügen wir den Videoclip ein. Dann extrahieren wir die Metadaten, danach extrahieren wir Frames mit einer Rate von 0,33 pro Sekunde, wir extrahieren die Audiodatei und dann konvertieren wir die Audiodatei in Text. Dieser Prozess wird als ETL (Extrahieren, Transformieren, Laden) bezeichnet. Anschließend sucht Googles Plagiatserkennungsprogramm mit Hilfe der Metadaten nach der eingegebenen Videodatei, und wenn es einen Videoclip gibt, laden wir diesen Clip herunter, um den Vergleichsprozess zu beginnen, der im vorherigen Kapitel untersucht wurde.

7.1 Zusammenfassung

Nach der Verwendung mehrerer Techniken zur Identifizierung eines urheberrechtsverletzenden Videos konnte diese Studie Plagiate lokal im Video identifizieren. Es bestimmt die Prozentsätze der Ähnlichkeit zwischen Videoclips. Urheberrechtserkennung und Inhaltsdiebstahl sind zwei große gesellschaftliche Probleme, die eine praktische Lösung erfordern. Wir empfehlen, zusätzliche Studien durchzuführen und das Projekt auszuweiten.
Man muss bei der Verwendung der Ergebnisse dieser Arbeit vorsichtig sein. Es gibt nämlich viele Faktoren, die berücksichtigt werden müssen, um festzustellen, ob tatsächlich ein Plagiat vorliegt oder nicht. Beispielsweise können Videos aufgrund der Verwendung von Schlagwörtern oder gemeinsamen Themen zu ähnlich sein, und nicht aufgrund von Plagiaten. Daher sollt dieses Werkzeug vorsichtig und sinnvoll eingesetzt werden.

7.2 Zukünftige Perspektiven

1. Die Erstellung eines öffentlichen Webportals für ein System zur Erkennung von Videoplagiaten kann in vielerlei Hinsicht mit Plagiaten zusammenhängen. So wie Websites dabei helfen, Plagiate in Texten zu erkennen, zum Beispiel Duplichecker [tea24] und Turnitin [Tur23].

2. Der Einsatz von Bildersuchmaschinen ist ebenfalls genauer, da die derzeit verwendeten nicht ausreichend geschult sind, um Ergebnisse zu liefern, die einem Bild im Frame ähnlich sind.

3. Bei der Transformation von Audiodaten in Text werden die Textwörter, wird eine zufällige Zeichenfolge davon ausgewählt und dann verwendet, um im Internet nach Videotiteln oder dem Namen und

Titel von Videoinhalten zu suchen. Dadurch können wir bessere Ergebnisse bei der Plagiatsprüfung erzielen.

4. Verwendung von OCR-Methoden zur Extraktion von Textinformationen aus Frames.

5. Spezialisierte Suchmaschinen können zukünftig mit fortgeschrittenen Techniken entworfen werden, um das Problem der Urheberrechtsverletzung effektiver zu lösen

Literaturverzeichnis

[Ac24] Audet, Samuel und contributors: *JavaCV: Java interface to OpenCV, FFmpeg, and more*. https://github.com/bytedeco/javacv, 2024. Zugriff am 12.04.2024.

[Acc17] Accord.NET, Team: *Accord.NET Framework*. https://accord-framework.net, 2008–2017. Zugriff am 16.04.2024.

[ASMM22] Aldausari, Nuha, Arcot Sowmya, Nadine Marcus und Gelareh Mohammadi: *Video generative adversarial networks: a review*. ACM Computing Surveys (CSUR), 55(2):1–25, 2022.

[Ayd17] AydinAdn: *MediaToolkit*. https://github.com/AydinAdn/MediaToolkit, 2017. Zugriff am 01.03.2024.

[BETVG08] Bay, Herbert, Andreas Ess, Tinne Tuytelaars und Luc Van Gool: *Speeded-up robust features (SURF)*. Computer vision and image understanding, 110(3):346–359, 2008.

[BK08] Bradski, Gary und Adrian Kaehler: *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 2008.

[BML03] Barrett, Ruth, James Malcolm und Caroline Lyon: *Are we ready for large scale use of plagiarism detection tools*. NUI Galway, 01 2003.

[Bot24] Botwright, R.: *Data Warehousing: Optimizing Data Storage and Retrieval for Business Success*. Pastor Publishing Ltd, 2024.

[BTVG06] Bay, Herbert, Tinne Tuytelaars und Luc Van Gool: *Surf: Speeded up robust features*. In: *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*, Seiten 404–417. Springer, 2006.

[BvE00] Böhnlein, Michael und Achim Ulbrich vom Ende: *Grundlagen des Data Warehousing: Modellierung und Architektur*. Otto-Friedrich-Universität Bamberg, Bamberg, 2000.

[ces17] cesarsouza: *NuGet Gallery | Accord.Video.VFW 3.8.0*. https://www.nuget.org/packages/Accord.Video.VFW, Letzte Aktualisierung 19.10.2017. Zugriff am 03.02.2024.

[con23] contributors, Wikipedia: *Mel Frequency Cepstral Coefficients*. https://de.wikipedia.org/wiki/Mel_Frequency_Cepstral_Coefficients, 2023. Zugriff am 16.04.2024.

[Con24] Contributors, FFmpeg: *FFmpeg: A complete, cross-platform solution to record, convert, and stream audio and video*. https://ffmpeg.org/, 2024. Zugriff am 20.02.2024.

[CTKC21] Chavan, Hiten, Mohd Taufik, Rutuja Kadave und Nikita Chandra: *Plagiarism detector using machine learning*. International Journal of Research in Engineering, Science and Management, 4(4):152–154, 2021.

[EMB19] El Mostafa, Hambi und Faouzia Benabbou: *A Multi-Level Plagiarism Detection System Based on Deep Learning Algorithms*, 10 2019.

[FFm19] *The ffmpeg Open Source Project on Open Hub: Licenses Page*. https://openhub.net/p/ffmpeg, 2019. Zugriff am 20.01.2024.

[Gam23] Gambit: *ETL Prozess*. https://www.gambit.de/wiki/etl-prozess/, 2023. Zugriff am 11.12.2023.

[GPAM+14] Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville und Yoshua Bengio: *Generative Adversarial Networks*, 2014.

[Gro13] Grossberg, Stephen: *Recurrent neural networks*. Scholarpedia, 8(2):1888, 2013.

[Har19] Harvey, Phil: *ExifTool: A Platform-Independent Perl Library plus a Command-Line Application for Reading, Writing and Editing Meta Information in Image, Audio and Video Files*. In: *Proceedings of the 14th International Conference on Digital Preservation (iPRES 2017)*, Seiten 339–348. ACM, 2019.

[Har24a] Harvey, Phil: *ExifTool*. https://exiftool.org/, 2024. Zugriff am 01.03.2024.

[Har24b] Harvey, Phil: *exiftool/exiftool*. https://github.com/exiftool/exiftool, 2024. Zugriff am 01.03.2024.

[HJK+19] Hunt, Ethan, Ritvik Janamsetty, Chanana Kinares, Chanel Koh, Alexis Sanchez, Felix Zhan, Murat Ozdemir, Shabnam Waseem, Osman Yolcu, Binay Dahal et al.: *Machine learning models for paraphrase identification and its applications on plagiarism detection*. In: *2019 IEEE International Conference on Big Knowledge (ICBK)*, Seiten 97–104. IEEE, 2019.

[HZRS15] He, Kaiming, Xiangyu Zhang, Shaoqing Ren und Jian Sun: *Deep Residual Learning for Image Recognition*, 2015.

[IBM23] IBM: *Was sind konvolutionale neuronale Netze*. https://www.ibm.com/de-de/topics/convolutional-neural-networks, 2023. Zugegriffen am 07.01.2024.

[Int23] International Press Telecommunications Council (IPTC): *IPTC Video Metadata Hub - Recommendation 1.5 / Properties*. https://www.iptc.org/std/videometadatahub/recommenda, 10 2023. Zugriff am 10.04.2024.

[Ish24] Ishak, Ghaith: *Konzeption eines ETL-Prozesses zur Ermittlung von Kennzahlen aus Bildern als Vorbereitungsschritt einer Plagiatserkennung*. Bachelorarbeit, Universität Rostock, 18055 Rostock, 2024. Noch in Bearbeitung.

[Jer23] Jerome, Martinez: *MediaInfo*. https://github.com/MediaArea/MediaInfo, 2023. Zugriff am 20.01.2024.

[JG09] Juan, Luo und Oubong Gwun: *A comparison of sift, pca-sift and surf*. International Journal of Image Processing (IJIP), 3(4):143–152, 2009.

[KK13] Kramer, Oliver und Oliver Kramer: *K-nearest neighbors*. Dimensionality reduction with unsupervised nearest neighbors, Seiten 13–23, 2013.

[Kur24] Kurdi, Muhamad: *Konzeption eines ETL-Prozesses zur Ermittlung von Kennzahlen aus Audios als Vorbereitungsschritt einer Plagiatserkennung*. Bachelorarbeit, Universität Rostock, 18055 Rostock, 2024. Noch in Bearbeitung.

[ld24] DEVELOPERS, SCIKIT LEARN: *Scikit-learn: Machine Learning in Python.* `https://scikit-learn.org/stable/index.html`, 2024. Zugriff am 16.04.2024.

[Ltd23] LTD, INDIGO STREAM TECHNOLOGIES: *Copyscape.* `https://www.copyscape.com/`, 2023. Zugriff am 07.01.2024.

[Lu09] LU, JIAN: *Video fingerprinting for copy identification: from research to industry applications.* Media Forensics and Security, 7254:725402, 2009.

[LYT+08] LIU, CE, JENNY YUEN, ANTONIO TORRALBA, JOSEF SIVIC und WILLIAM T FREEMAN: *Sift flow: Dense correspondence across different scenes.* In: *Computer Vision–ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part III 10*, Seiten 28–42. Springer, 2008.

[Macss] MACHINE LEARNING GROUP AT THE UNIVERSITY OF WAIKATO: *WEKA: The workbench for machine learning.* Website, Year of Access. Zugriff am 21.03.2024.

[Med24] MEDIAAREA.NET: *MediaInfoLib.* `https://mediaarea.net/`, 2024. Zugriff am 01.03.2024.

[MSY92] MORI, S., C.Y. SUEN und K. YAMAMOTO: *Historical review of OCR research and development.* Proceedings of the IEEE, 80(7):1029–1058, 1992.

[MW01] MEYER, DAVID und FT WIEN: *Support vector machines.* R News, 1(3):23–26, 2001.

[OBAB18] OUSSOUS, AHMED, FATIMA-ZAHRA BENJELLOUN, AYOUB AIT LAHCEN und SAMIR BELFKIH: *Big Data technologies: A survey.* Journal of King Saud University - Computer and Information Sciences, 30(4):431–448, 2018.

[Ote15] OTERO, IVES REY: *Anatomy of the SIFT Method.* Doktorarbeit, École normale supérieure de Cachan-ENS Cachan, 2015.

[Pet09] PETERSON, LEIF E: *K-nearest neighbor.* Scholarpedia, 4(2):1883, 2009.

[PS20] PISNER, DEREK A und DAVID M SCHNYER: *Support vector machine.* In: *Machine learning*, Seiten 101–121. Elsevier, 2020.

[Sab18] SABSCH, TIM: *Eine kurze Einführung in Generative Adversarial Networks.* `https://www.codecentric.de/wissens-hub/blog/eine-kurze-einfuehrung-in-generative-adversarial-networks`, 2018. Zugriff am 13.04.2024.

[SBB12] SINGH, AMARJOT, KETAN BACCHUWAR und AKSHAY BHASIN: *A survey of OCR applications.* International Journal of Machine Learning and Computing, 2(3):314, 2012.

[SLJ+14] SZEGEDY, CHRISTIAN, WEI LIU, YANGQING JIA, PIERRE SERMANET, SCOTT REED, DRAGOMIR ANGUELOV, DUMITRU ERHAN, VINCENT VANHOUCKE und ANDREW RABINOVICH: *Going Deeper with Convolutions*, 2014.

[SMH11] SUTSKEVER, ILYA, JAMES MARTENS und GEOFFREY E HINTON: *Generating text with recurrent neural networks.* In: *Proceedings of the 28th international conference on machine learning (ICML-11)*, Seiten 1017–1024, 2011.

[Smi07] SMITH, R.: *An Overview of the Tesseract OCR Engine.* In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, Band 2, Seiten 629–633, 2007.

[Smi19] Smith, D.: *The Exponential Growth of Big Data.* https://tweakyourbiz.com/posts/big-data-growth, 2019. Zugriff am 17.04.2024.

[Sou17] Souza, C.: *Accord.NET Machine Learning Framework.* https://accord-framework.net/intro.html, 2017. Lizenziert unter einer Creative Commons Attribution 3.0 Unported License.

[SPC11] Sainui, Janya, Ladda Preechaveerakul und Lekha Chaisorn: *An image-based video copy detection using ordinal bitmap signature.* In: *2011 8th International Conference on Information, Communications & Signal Processing*, Seiten 1–5, 2011.

[SS14] Subroto, Imam Much Ibnu und Ali Selamat: *Plagiarism detection through internet using hybrid artificial neural network and support vectors machine.* TELKOMNIKA (Telecommunication Computing Electronics and Control), 12(1):209–218, 2014.

[SZ15] Simonyan, Karen und Andrew Zisserman: *Very Deep Convolutional Networks for Large-Scale Image Recognition*, 2015.

[tea23] team, OpenCV: *OpenCV.* https://opencv.org/, Letzte Aktualisierung 28.12.2023. Offizielle Website, abgerufen von OpenCV am 20.01.2024.

[tea24] team, Dupli Checker: *Dupli Checker.* https://www.duplichecker.com/de, 2024. Zugriff am 21.03.2024.

[TJN21] Thirani, Ekta, Jayshree Jain und Vaibhav Narawade: *Enhancing Performance Evaluation for Video Plagiarism Detection Using Local Feature through SVM and KNN algorithm.* International Journal of Image, Graphics and Signal Processing, 13(5):41–50, 2021.

[Tur23] Turnitin: *Turnitin.* https://www.turnitin.com/, 2023. Zugriff am 07.01.2024.

[TY20] Tran, Quang Nhat und Shih-Hsuan Yang: *Efficient video frame interpolation using generative adversarial networks.* Applied Sciences, 10(18):6245, 2020.

[US 22] US National Archives: *Preservation Action Plan: Digital Cinema.* https://github.com/usnationalarchives/digital-preservation/blob/3df6add5bfdbf654996dfc2642671695c737bb7b/Moving_Image_Formats/NARA_PreservationActionPlan_DigitalCinema_20220714.pdf, Juli 2022. Zugriff über GitHub.

[Wik23] Wikipedia: *ETL-Prozess.* https://en.wikipedia.org/wiki/ETL-Prozess, 2023. Zugriff am 10.12.2023.

[Wik24] Wikipedia: *Plagiat — Wikipedia, die freie Enzyklopädie.* https://de.wikipedia.org/w/index.php?title=Plagiat&oldid=243955447, 2024. Zugriff am 14.04.2024.

[Wut22] Wuttke, Laurenz: *ETL Prozess.* https://datasolut.com/wiki/etl-prozess/, 2022. Zugriff am 10.12.2023.

[Wut23] Wuttke, Laurenz: *was ist ein etl-prozess.* https://datasolut.com/wiki/etl-prozess/, 2023. Zugriff am 10.12.2023.

[YC09] Yeh, Mei-Chen und Kwang-Ting Cheng: *Video copy detection by fast sequence matching.* In: *Proceedings of the ACM International Conference on Image and Video Retrieval*, CIVR '09, New York, NY, USA, 2009. Association for Computing Machinery.

[Zha16] Zhang, Zhongheng: *Introduction to machine learning: k-nearest neighbors.* Annals of translational medicine, 4(11), 2016.

[ZXL+18] Zhang, X., Y. Xie, X. Luan et al.: *Video Copy Detection Based on Deep CNN Features and Graph-Based Sequence Matching.* Wireless Personal Communications, 103:401–416, 2018.

[ZYJ06] Zdziarski, Jonathan, Weilai Yang und Paul Judge: *Approaches to phishing identification using match and probabilistic digital fingerprinting techniques.* In: *Spam Conference*, 2006.

[ZZ16] Zhang, Yue und Xinxiang Zhang: *Effective real-scenario Video Copy Detection.* In: *2016 23rd International Conference on Pattern Recognition (ICPR)*, Seiten 3951–3956, 2016.

Anhang A

A.1 Systemdesign und Architektur

A.1.1 Andere Bibliotheken

- **MediaInfoDotNet:** Bietet Funktionalitäten zur Information über Multimedia-Dateien, wie Videodauer und Bitrate.
- **NAudio.FileFormats** *(version 2.2.1)*: Bietet Funktionen zum Lesen und Schreiben verschiedener Audio-Dateiformate, wie WAV und MP3.
- **NAudio.Wave** *(version 2.2.1)*: Bietet grundlegende Klassen zur Verarbeitung von Audiosignalen, wie Sprachsignalrepräsentation und DFT-Operation
- **OpenCvSharp.Extensions** *(version 4.9.0.20240103)*: Bietet zusätzliche Funktionalitäten für die OpenCV-Bibliothek
- **OpenCvSharp.XFeatures2D** *(version 4.9.0.20240103)*: Die Algorithmen ermöglichen die Extraktion von Merkmalen aus Bildern, wie SURF und ORB.
- **OpenCvSharp** *(version 2.4.0.1)*: Bietet eine umfassende Bibliothek zur Bild- und Videobearbeitung, einschließlich Funktionen zur Gesichtserkennung und Objekterkennung.
- **System.Collections.Generic** *(version 4.3.0)*: Bietet grundlegende Datenklassen, wie Listen und Wörterbücher.
- **System.ComponentModel** *(version 4.3.0)*: Bietet Funktionalitäten zur Erstellung von UI-Komponenten.
- **System.Data.SqlClient** *(version 4.3.0)*: Bietet Funktionalitäten zur Verbindung mit SQL Server-Datenbanken.
- **System.Data** *(version 4.3.0)*: Bietet grundlegende Klassen zur Datenhandhabung, wie Datasets und Tabellenkalkulationen.
- **System.Drawing.Imaging** *(version 4.3.0)*: Bietet Funktionen zur Bildverarbeitung, wie das Hochladen und Speichern von Bildern.
- **System.Drawing** *(version 4.5.4)*: Bietet grundlegende Klassen zur Erstellung von Grafiken, wie Formen und Text.
- **System.IO** *(version 4.3.0)*: Bietet Funktionen zur Handhabung von Dateien und Ordnern.
- **System.Linq** *(version 4.3.0)*: Bietet LINQ-Abfragen zur Datenhandhabung.

- **System.Net.Http** *(version 4.3.0)*: Bietet Funktionalitäten zum Senden von HTTP-Anfragen und zum Empfangen von Antworten.
- **System.Speech.Recognition** *(version 9.0.0-preview.1.24080.9)*: Bietet Funktionen zur Spracherkennung.
- **System.Text** *(version 7.0.3)*: Bietet grundlegende Klassen zur Arbeit mit Text, wie Textkodierung und -dekodierung.
- **System.Threading.Tasks** *(version 4.5.4)*: Bietet Funktionalitäten für die Programmierung mit mehreren Threads.
- **System.Threading** *(version 4.5.4)*: Bietet Funktionen zur Handhabung von Threads, wie das Erstellen und Stoppen von Threads.
- **System.Windows.Forms** *(version 4.5.4)*: Bietet grundlegende Klassen zur Erstellung der Benutzeroberfläche, wie Fenster und Knöpfe.

A.2 Quellcode

Die im Anhang der Abschlussarbeit dargestellten Codes wurden durch einen Refactoring-Prozess verbessert, um die Verständlichkeit zu steigern. Damit soll erreicht werden, dass jeder Leser, unabhängig von seinen Programmierkenntnissen, den Code mühelos verstehen und ihm folgen kann.

A.2.1 Metadaten aus Videodateien extrahieren

```
1  private void SelectVedioFile_Click(object sender, EventArgs e)
2  {
3      OpenFileDialog ofd = new OpenFileDialog();
4      ofd.Filter = "Video files (*.mp4;*.mkv)|*.mp4;*.mkv";
5      if (ofd.ShowDialog() == DialogResult.OK)
6      {
7          textpathVideo.Text = ofd.FileName;
8          axWindowsMediaPlayer1.URL = ofd.FileName; // Play video using
           Windows Media Player
9      }
10 }
11 private void ExtractMetadatafromvideo_Click(object sender, EventArgs e)
12 {
13     // Replace "path/to/your/video/file.mp4" with your video path
14     string videoFilePath = textpathVideo.Text;
15     dataGridView1.Columns.Clear();
16     // Replace "YourConnectionString" with your database connection
17
18     // Extract Metadata
19     var metadata = ExtractMetadata(videoFilePath);
20
21     // Save Metadata to Database
22     SaveMetadataToDatabase(connectionString, metadata, videoFilePath,
       this);
23 }
```

A.2.2 Data Extraction

```
1  private void ExtractData()
2  {
3      string filePath = textpathVideo.Text;
4      string fileExtension = Path.GetExtension(filePath);
5
6
7      // Check if the file is a video
8      if (fileExtension.ToLower() == ".mp4" || fileExtension.ToLower() ==
       ".avi")
9      {
10
11         // Create a VideoCapture object for the video
12         var videoCapture = new Accord.Video.FFMPEG.VideoFileReader();
13
14
15         // Open the video file
16         videoCapture.Open(filePath);
17
18
19         // Specify the folder path on disk D
20         string directoryPath = Path.Combine("C:\\",
           Path.GetFileNameWithoutExtension(filePath));
21
22
23         // Check if the folder exists
24         if (Directory.Exists(directoryPath))
25         {
26
27             // If the folder exists, delete it
28             Directory.Delete(directoryPath, true);
29         }
30
31
32         // Recreate the folder
33         Directory.CreateDirectory(directoryPath);
34
35
36         // Set the maximum value of progressBar1
37         progressBar1.Maximum = (int)videoCapture.FrameCount;
38
39
40         // Extract frames from video
41         for (int i = 0; i < videoCapture.FrameCount; i++)
42         {
43
44             // Extract the frame
45             Bitmap frame = videoCapture.ReadVideoFrame();
46
```

```
47
48              // Check if the frame is null
49              if (frame == null)
50              {
51
52                  // If the frame is null, break the loop
53                  break;
54              }
55
56
57              // Save the frame as an image in the new folder
58              frame.Save(Path.Combine(directoryPath, $"frame{i}.bmp"));
59
60
61              // free up memory
62              frame.Dispose();
63
64
65              //Update progressBar1
66              progressBar1.Value = i + 1;
67          }
68
69          // extract audio
70          using (var audioFileReader = new MediaFoundationReader(filePath))
71          {
72              string audioFilePath = Path.Combine(directoryPath,
                "audio.wav");
73
74
75              // Convert audio to WAV format
76              WaveFileWriter.CreateWaveFile(audioFilePath, audioFileReader);
77          }
78
79
80          // Close the video file
81          videoCapture.Close();
82
83          // Display a message that the process has finished
84          MessageBox.Show("Data extraction completed successfully!!");
85      }
86
87      // Check if the file is audio
88      else if (fileExtension.ToLower() == ".mp3" || fileExtension.ToLower()
        == ".wav")
89      {
90          // Specify the folder path on disk D
91          string directoryPath = Path.Combine("C:\\",
            Path.GetFileNameWithoutExtension(filePath));
92
93
94          // Check if the folder exists
```

```
95          if (Directory.Exists(directoryPath))
96          {
97
98              // If the folder exists, delete it
99              Directory.Delete(directoryPath, true);
100         }
101
102
103         // Recreate the folder
104         Directory.CreateDirectory(directoryPath);
105
106
107         // Copy the audio file to the new folder
108         string newFilePath = Path.Combine(directoryPath, "audio" +
            fileExtension);
109         File.Copy(filePath, newFilePath);
110
111
112         // Display a message that the process has finished
113         MessageBox.Show("Data extraction completed successfully!!");
114     }
115     else
116     {
117
118         // Display a message in an unsupported format
119         MessageBox.Show("Unsupported file format!");
120     }
121 }
```

A.2.3 Datentransformation

```
1  private void TransformationData_Click(object sender, EventArgs e)
2          {
3              int i = 0;//counter
4              string filePath = textpathVideo.Text;
5              string videoName =
               Path.GetFileNameWithoutExtension(textpathVideo.Text);
6              string fileExtension = Path.GetExtension(filePath);
7              string directoryPath = Path.Combine("C:\\",
               Path.GetFileNameWithoutExtension(filePath));
8              if (Directory.Exists(directoryPath))
9              {
10                 string[] frameFiles = Directory.GetFiles(directoryPath,
                   "frame*.bmp");
11                 progressBar1.Maximum = frameFiles.Length;
12                 string audioFile = Path.Combine(directoryPath,
                   "audio.wav");
13                 AnalyzeAudio(directoryPath, "audio.wav", connectionString,
                   videoName);
```

```
                ExtractTextFromVideo(audioFile, connectionString,
                videoName);
                foreach (string frameFile in frameFiles)
                {
                    Bitmap frame = new Bitmap(frameFile);
                    using (var audioFileReader = new
                    MediaFoundationReader(audioFile))
                    {
                        AnalyzeFrame(directoryPath, "audio.wav", frame,
                        connectionString, videoName);
                    }
                    progressBar1.Value = i++;
                }
            }
            MessageBox.Show("Data transformation completed
            successfully!!");
        }
```

A.2.4 Plagiatserkennung

```
private void btnReporting_Click(object sender, EventArgs e)
        {
            richTextBox1.Text = "";
            // Query the database for all videos
            List<Video> videos = GetAllVideos();

            // Compare each video with every other video
            for (int i = 0; i < videos.Count; i++)
            {
                for (int j = i + 1; j < videos.Count; j++)
                {
                    // Calculate the similarity between the two videos
                    double similarity = CalculateSimilarity(videos[i],
                    videos[j]);

                    // If the videos are similar, update the Relationships
                    and Forgery tables
                    if (similarity > 0)
                    {
                        UpdateRelationshipsTable(videos[i], videos[j],
                        similarity);
                        UpdateForgeryTable(videos[i], similarity);
                    }

                    // Display the results in richTextBox1
                    richTextBox1.Text += $"Video {videos[i].Name} and
                    Video {videos[j].Name} have a similarity [MetaData]
                    score of {similarity * 100}%\n";
                }
```

```
            }
        }
```

A.2.5 Überprüfung des Bildrahmens über das Internet.

```
private async void VeriyFrameByinternet_Click(object sender, EventArgs e)
{
    string filePath = textpathVideo.Text;
    string videoName =
    Path.GetFileNameWithoutExtension(textpathVideo.Text);
    string fileExtension = Path.GetExtension(filePath);
    string directoryPath = Path.Combine("C:\\",
    Path.GetFileNameWithoutExtension(filePath));
    if (Directory.Exists(directoryPath))
    {
        // Get the list of all frame file paths in the folder
        string[] frameFilePaths = Directory.GetFiles(directoryPath);

        // Check if there are enough frames
        if (frameFilePaths.Length < 3)
        {
            MessageBox.Show("Not enough frames in the folder.");
            return;
        }
        // Select the first, middle, and last frames
        string[] selectedFrames = new string[3];
        selectedFrames[0] = frameFilePaths[0 + 1];

        // Indicate that the search is starting
        progressBar1.Value = 0;
        progressBar1.Style = ProgressBarStyle.Marquee;

        // Open Google Images in ChromiumWebBrowser
        webBrowser1.Navigate("https://www.google.com/imghp");

        // Indicate that the search is complete
        progressBar1.Style = ProgressBarStyle.Continuous;
        progressBar1.Value = 100;
    }

}
```

A.2.6 Überprüfung des Videos anhand von Metadaten

```
private async void button2_Click(object sender, EventArgs e)
{
    try
```

```
4       {
5           webBrowser1.ScriptErrorsSuppressed = true;
6           string videoName =
            Path.GetFileNameWithoutExtension(textpathVideo.Text);
7
8
9           using (SqlConnection connection = new
            SqlConnection(connectionString))
10          {
11              connection.Open();
12              using (SqlCommand command = new SqlCommand())
13              {
14
15                  command.Connection = connection;
16                  command.CommandText = "SELECT Name, Duration, Codec FROM
                    [dbo].[Videos] WHERE Name = @VideoName";
17                  command.Parameters.AddWithValue("@VideoName", videoName);
18
19                  using (SqlDataReader reader = command.ExecuteReader())
20                  {
21                      if (reader.Read())
22                      {
23                          string retrievedVideoName =
                            reader["Name"].ToString();
24                          string videoDuration =
                            reader["Duration"].ToString();
25                          string videoCodec = reader["Codec"].ToString();
26
27                          // Indicate that the search is starting
28                          progressBar1.Value = 0;
29                          progressBar1.Style = ProgressBarStyle.Marquee;
30
31                          // Perform online search using retrievedVideoName,
                            videoDuration, videoCodec
32                          string searchResults = await
                            SearchOnline(retrievedVideoName, videoDuration,
                            videoCodec);
33                          webBrowser1.DocumentText = searchResults;
34
35                          // Indicate that the search is complete
36                          progressBar1.Style = ProgressBarStyle.Continuous;
37                          progressBar1.Value = 100;
38
39                          // Display the search results in richTextBox1
40                          if (!string.IsNullOrEmpty(searchResults))
41                          {
42                              richTextBox1.Text = $"Search results for video
                                '{retrievedVideoName}':\n There is no matching
                                video based on this video's metadata";
43                          }
44                          else
```

```
45                          {
46                              richTextBox1.Text = " There is an identical
                                video according to the metadata of this
                                video... Please work on downloading this
                                video, comparing it with this video, and
                                finding the percentage of plagiarism.";
47                          }
48
49                      }
50                      else
51                      {
52                          MessageBox.Show("Video not found in the
                            database.");
53
54                      }
55                  }
56              }
57          }
58      }
59      catch (Exception ex)
60      {
61          MessageBox.Show($"Error: {ex.Message}");
62      }
63  }
64
65
66  private async Task<string> SearchOnline(string videoName, string
    videoDuration, string videoCodec)
67  {
68      // Implement your online search logic here
69      // You can use HttpClient to perform web searches or query APIs
70      // Example:
71      using (HttpClient client = new HttpClient())
72      {
73          string searchQuery = $"{videoName} {videoDuration} {videoCodec}";
74          HttpResponseMessage response = await
            client.GetAsync($"https://google.com/search?q={searchQuery}");
75          if (response.IsSuccessStatusCode)
76          {
77              return await response.Content.ReadAsStringAsync();
78          }
79          else
80          {
81              return "Error fetching search results.";
82          }
83      }
84  }
```

A.3 Methodik zur Erkennung von Videokopien

Sie kann mehrere Schritte umfassen, wie in Abbildung A.1 dargestellt.

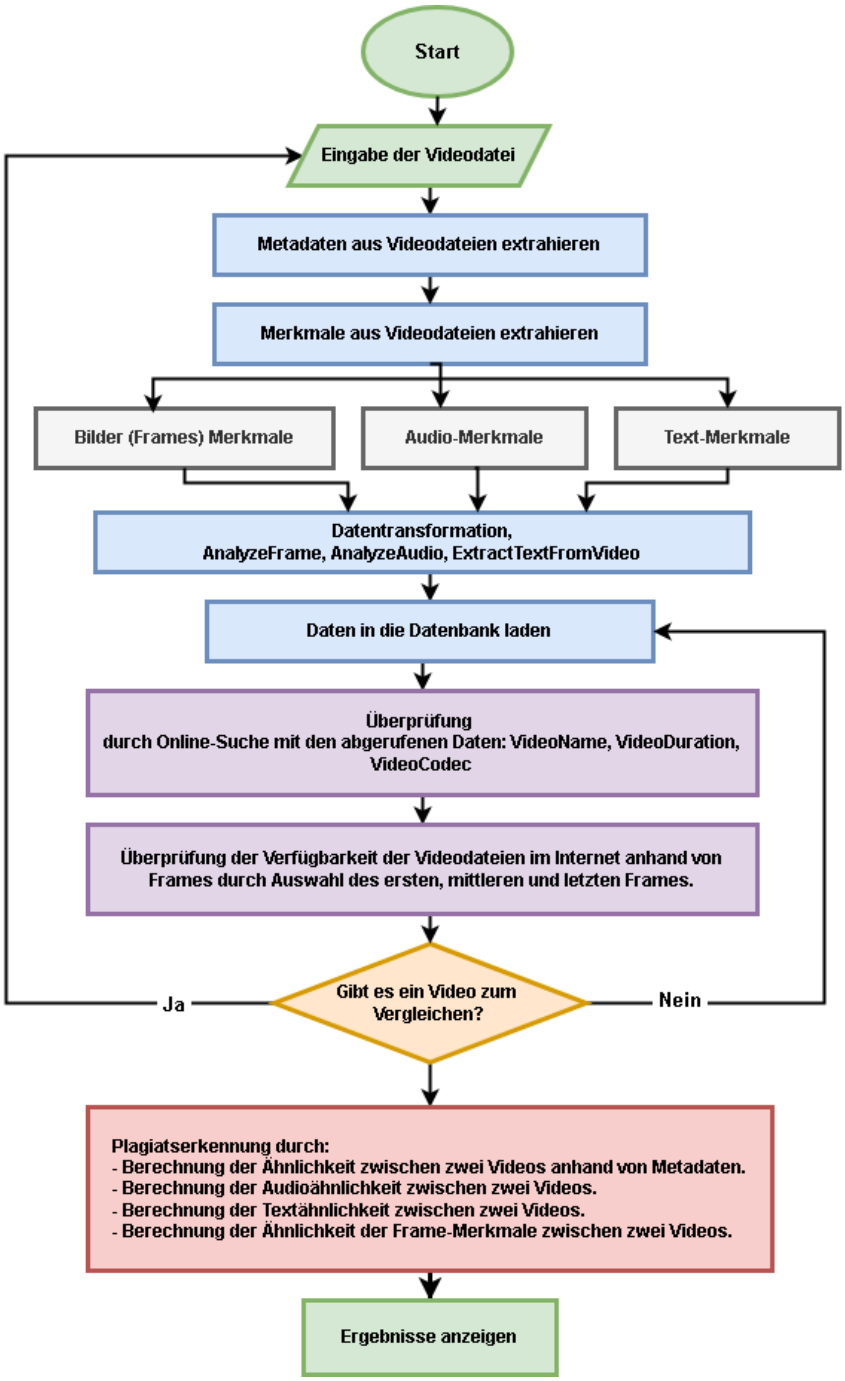


Abbildung A.1: Methode zur Erkennung von urheberrechtlich geschützten Videos

Erläuterung des Schemas in Abbildung A.1

1. **Start**: Der Beginn des Programms.
2. **Eingabe der Videodatei** (`Video file entry`): Hier wird die zu analysierende Videodatei eingegeben.
3. **Merkmale extrahieren** (`Extract features`): In diesem Schritt werden Merkmale aus dem Video extrahiert, einschließlich Bilder (Frames) mit dem CNN-Algorithmus, Audio mit Deep Learning und Text mit der Verarbeitung natürlicher Sprache.
4. **Vergleich mit der Datenbank** (`Comparison with the database`): Nachdem die Merkmale extrahiert wurden, werden sie mit den Merkmalen aus anderen Videos in der Datenbank verglichen. Wenn in der Datenbank keine Übereinstimmung gefunden wird, kehren Sie zum Anfang zurück. Wenn eine Übereinstimmung gefunden wird, fahren Sie mit dem nächsten Schritt fort.
5. **Überprüfung** (`Verification`): Wenn der Verdacht auf Plagiat besteht, kann dies durch Überprüfung und Vergleich ähnlicher Videos verifiziert werden.
6. **Meldung** (`Reporting`): Wenn ein Plagiat erkannt wird, kann dies gemeldet und entsprechende Maßnahmen ergriffen werden.

Selbständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbständig und nur unter Vorlage der angegebenen Literatur und Hilfsmittel angefertigt habe. ChatGPT [1] wurde lediglich für die Umformulierung und Übersetzung ins Deutsche verwendet.

Rostock, den 19. April 2024

Bandar Al-Askari

[1] https://chat.openai.com/