Hochschule Reutlingen
Reutlingen University

Automatisierte Zerlegung von Vorlesungsaufzeichnungen zur Realisierung eines Empfehlungssystems für die Lerneinheiten der Online-Lehre

Master-Thesis
im Studiengang Digital Business Engineering
Herman-Hollerith-Zentrum, Fakultät Informatik
Hochschule Reutlingen
Alteburgstraße 150
72762 Reutlingen

Eingereicht von: Arnaud Zobel Kamlo Ngako
Matrikel Nr. 766608
E-Mail: Arnaud_Zobel.Kamlo_Ngako
@Student.Reutlingen-University.DE

Betreuer/Prüfer: Prof. Dr. Christian Decker
Prof. Dr. Jürgen Münch

Tag der Abgabe: 31. Dezember 2021

Inhaltsverzeichnis

Nomenklatur

ASR	Automatic Speech Recognition
C.a.	Circa
CBR	Case Based Reasoning
CNN	Convolutional Neural Network
CSV	Comma-separated Values
D.h.	Das heißt
DCT	Discrete Cosine Transform
HSV	Hue Saturation Value
LMS	Learning Management System
MAE	Mean Absolute Error
NLP	Natural Language Processing
OCR	Automatic Speech Recognition
RGB	Red Green Blue
RMSE	Root Mean Square
ROI	Region Of Interest
STFT	Short-time Fourier transform
SVD	Single Value Decomposition
Z.B.	Zum Beispiel

Abstract (Deutsch)

Die Anzahl der Online-Vorlesungen ist im Zuge der Corona-Pandemie stark gestiegen. Viele Hochschulen haben die klassische Präsenzlehre auf ein digitales Format umgestellt. Dadurch ergibt sich für Studierende ein zeitlicher und räumlicher Vorteil. Die Onlinelehre bringt zwar neue Impulse, erfordert jedoch auch einen größeren zeitlichen Aufwand. So ist hierfür eine besondere didaktische Aufbereitung der Vorlesungen für das Online-Format erforderlich. Außerdem wird das individuelle Lernen nicht unterstützt.

In der vorliegenden Arbeit werden Ansätze zur automatisierten Zerlegung von Vorlesungsaufzeichnungen untersucht, um auf diese Weise passende Lerneinheiten für die Onlinelehre zu erstellen. Des Weiteren wird auf Basis der erstellten Lerneinheiten ein Empfehlungssystem für die Lerneinheiten der Onlinelehre realisiert.

Der Lösungsprozess für die automatisierte Zerlegung der Aufzeichnungen wird in vier Hauptschritte unterteilt. Im ersten Schritt werden audiovisuelle Features wie das Farbhistogramm und die akustische Salienz extrahiert. Im zweiten Schritt werden diese Features analysiert, um Szeneänderungen zu erkennen. Im dritten Schritt werden Szenen mit dem neuronalen Netzwerk CNN klassifiziert. Im letzten Schritt wird das Eingangsvideo mit dem Programm FFmpeg geschnitten. Der Vorteil dieser Lösung besteht darin, dass für die Zerlegung kein Wissen über den Inhalt des Videos benötigt wird.

Für die Entwicklung des Empfehlungssystems werden die klassischen Lösungsschritte für ein Problem aus dem Bereich des maschinellen Lernens nach dem Standard-Vorgehensmodell CRISP-DM verwendet. Das Empfehlungssystem basiert auf dem Verfahren des kollaborativen Filterns. In diesem Sinne wird mit unterschiedlichen Ansätzen des kollaborativen Filterns experimentiert, um schließlich den geeignetsten auszuwählen. Das Empfehlungssystem wird mit der Kreuzvalidierung evaluiert. Dazu werden Datensätze generiert, die aus Benutzerinteraktionsdaten bestehen.

Abstract (Englisch)

The number of online lectures has risen sharply because of the COVID-19 pandemic. Many universities have converted traditional face-to-face teaching to a digital format. This gives students an advantage in terms of time and space. Online teaching brings new impulses but requires more effort. Preparing a didactic lecture for the online format is time-consuming. Additionally, individualized learning is not supported.

In this thesis, approaches to automated segmentation of lecture recordings to create suitable learning units for online teaching are investigated. Furthermore, a recommendation system is implemented for the created learning units.

The process for the automated decomposition of the recordings is divided into four steps. In the first step, audiovisual features such as the color histogram and acoustic saliency map are extracted. In the second step, these features are analyzed to detect scene changes and interesting scene changes. In the third step, scenes are classified using the convolutional neural network (CNN) model. In the last step, the input video is sliced using the FFmpeg program. The advantage of this solution is that it does not require any knowledge about the content of the video for the segmentation.

To develop the recommender system, the classical solution steps for a machine learning problem according to Industry Standard Process for Data Mining (CRISP-DM) were used. The recommender system is based on collaborative filtering. Various approaches to collaborative filtering were tested, and the most suitable one was selected. The recommender system was evaluated using cross-validation. For this purpose, datasets that consist of user interaction data were generated.

1 Einleitung

1.1 Problemstellung

Die Anzahl der Online-Vorlesungen ist im Zuge der Corona-Pandemie stark gestiegen [1]. Dies verschafft Studierenden eine hohe Flexibilität. So können sie sich die Lerneinheiten beliebig oft, unabhängig von den Vorlesungszeiten und außerhalb der Hörsäle ansehen. Zugleich gestaltet sich jedoch die didaktische Aufbereitung einer Vorlesung für ein Online-Format als eine aufwendige Aufgabe. Hierfür muss die Vorlesung zunächst vom Professor aufgezeichnet werden. Daraufhin wird die Aufzeichnung manuell geschnitten und überarbeitet, bevor die daraus resultierenden Lerneinheiten schließlich in ein Learning-Management-System (LMS) hochgeladen werden.

Diese manuellen Aufgaben werden häufig nicht korrekt durchgeführt. Infolgedessen erfüllen viele Online-Vorlesungen nicht die Anforderungen an ein Online-Format. Die meisten weisen eine lange Übertragungsdauer auf, was die Suche nach relevanten Inhalten für Studierende erschwert. Dabei könnte eine automatisierte Zerlegung der Aufzeichnungen diesen Aufwand reduzieren.

Des Weiteren ist bei einer großen Anzahl an Lernvideos für Studierende unklar, welche Videos als nächste zu betrachten sind. Außerdem basieren viele Online-Vorlesungen auf statischen Lernmaterialien, die die Vielfalt der Lernenden nicht berücksichtigen. Als mögliche Lösung für dieses Problem gilt der Einsatz eines LMS mit integriertem Empfehlungssystem [2]. Derartige Systeme sollen Lernenden eine individuelle Bildung ermöglichen, indem sie diesen auf Grundlage ihrer Ziele, Fähigkeiten und Präferenzen geeignete Lerneinheiten empfehlen. Allgemeine Data-Mining-Tools, wie *Weka* und *Intelligent Miner*, werden heutzutage eingesetzt, um Empfehlungsprobleme zu lösen. Diese Tools sind jedoch nicht speziell für pädagogische Zwecke konzipiert und werden auch nicht entsprechend gepflegt [2].

1.2 Zielsetzung und Einordnung der Arbeit

Das Ziel dieser Arbeit ist es, ein Konzept zu entwickeln, das es erlaubt, Vorlesungsaufzeichnungen automatisiert zu zerlegen, um auf diese Weise passende Lerneinheiten für die Online-Lehre zu erzeugen. Des Weiteren soll auf Basis der erzeugten Lerneinheiten ein Empfehlungssystem realisiert werden. Dabei sollen Lerneinheiten so klassifiziert werden, dass individuelle Lernpfade möglich sind. Individuelle Pfade sind das Resultat von Empfehlungen an einen Studierenden, was er als Nächstes lernen soll. Das zu entwickelnde Empfehlungssystem sollte dabei ähnlich arbeiten wie dasjenige von Amazon oder Netflix, sodass Studierenden, die sich eine bestimmte Lerneinheit angesehen haben, empfohlen wird, sich als Nächstes eine geeignete Folgeeinheit anzusehen.

Konkret soll im Rahmen dieser Arbeit den folgenden Forschungsfragen nachgegangen werden:

- Welche Methoden und Techniken aus dem Bereich des maschinellen Lernens eignen sich für eine audiovisuelle Szeneexploration?
- Wie kann die Interessantheit einer Szeneänderung in einem Vorlesungsvideo definiert und bewertet werden?
- Welche Tools sind für das Schneiden eines Videos geeignet?
- Welche existierende Verfahrensarten und Ansätze aus dem Bereich Statistik und Maschinelles eignen sich für die Realisierung eines Empfehlungssystems für Lerneinheiten der Online-Lehre?
- Wie können Benutzer mit unterschiedlichen Profilen für den Test von kollaborativem Filtern erzeugt werden?
- Wie lässt sich ein Empfehlungssystem evaluieren?

Im Rahmen dieser Arbeit wird kein Softwareprodukt entwickelt, sondern es werden Experimente durchgeführt. Ebenso ist eine inhaltliche Analyse der Vorlesungsvideos nicht Teil dieser Arbeit. Des Weiteren basiert das Empfehlungssystem in dieser Arbeit auf dem Verfahren des kollaborativen Filterns. In dieser Arbeit werden die Ausdrücke ‚Lerneinheit' und ‚Lernabschnitt' synonym verwendet.

Die vorliegende Thesis lässt sich zwei wissenschaftlichen Forschungsgebieten zuordnen. Aufgrund des engen Zusammenhangs mit der Videozusammenfassung ist der Prozess der automatisierten Zerlegung der Aufzeichnungen diesem Gebiet zuzuordnen. Zudem wird in dieser Thesis das Forschungsfeld des Empfehlungssystems behandelt. Aufgrund des

Bezugs zum Thema Vorlesung wird genauerer von *Empfehlungssystemen im Kontext von E-Learning* gesagt.

1.3 Aufbau der Arbeit

Die vorliegende Arbeit gliedert sich in neun Kapitel. Kapitel 1 bietet grundlegende Überlegungen zu Problemstellung und Zielsetzung der Arbeit. Dabei werden konkrete Forschungsfragen definiert, um die Arbeit dann thematisch in die entsprechenden wissenschaftlichen Kontexte einzuordnen. In Kapitel 2 wird die Methode der Arbeit erläutert. Hierbei wird auf die Methode, das Vorgehen und die Ergebnisse der Literaturrecherche eingegangen. Ebenso werden in diesem Kapitel aktuelle wissenschaftliche Recherchen zu den genannten Forschungsgebieten thematisiert. Schließlich werden die wesentlichen Konzepte der für diese Thesis relevanten Forschungsgebiete dargestellt. Kapitel 3 beschäftigt sich mit den theoretischen Grundlagen des Forschungsbereichs der Videozusammenfassung und der audiovisuellen Szeneanalyse. In diesem Kapitel werden die wesentlichen Konzepte und Methoden zur Videozusammenfassung dargestellt. In Kapitel 4 wird in detaillierter Weise auf die Grundlagen zu Empfehlungssystemen eingegangen. Konkret wird dabei das kollaborative Filtern thematisiert. Ebenso werden die dem zugrunde liegenden Annahmen wiedergegeben und geeignete Evaluationsmethoden präsentiert. In Kapitel 5 wird versucht die vorliegende Arbeit thematisch genauer in das Gebiet des E-Learnings einzuordnen. Hierbei werden einige Annahmen und Methoden aus den vorherigen Kapiteln betrachtet und auf den Kontext der Online-Vorlesung übertragen. Kapitel 6 beschäftigt sich mit dem Systementwurf. Dieser beinhaltet eine Skizze der Systemarchitektur und des Systemablaufs. Ebenso wird hier das Konzept zur Lösung der in dieser Thesis herausgearbeiteten Probleme vorgestellt. Kapitel 7 enthält Ausführungen zur eigentlichen Implementierung. Hierzu gehören die Vorstellung des verwendeten Frameworks, eine Erläuterung der Umsetzung der automatischen Segmentierung und eine Präsentation des Empfehlungssystems. In Kapitel 8 wird der entwickelte Ansatz dann evaluiert. Hier werden insbesondere das Leistungsverhalten und die Genauigkeit überprüft. Schließlich wird in Kapitel 9 eine kurze Zusammenfassung der Arbeit gegeben, das Ergebnis vorgestellt und ein Ausblick aufgezeigt

2 Literaturrecherche und verwandte Untersuchungen

In diesem Kapitel wird eine systematische Literaturrecherche durchgeführt. Hierbei wird auf die Methode, das Vorgehen und die Ergebnisse der Literaturrecherche eingegangen. Zudem werden maßgebliche wissenschaftliche Beiträge aus den relevanten Themengebieten präsentiert.

2.1 Methode der Literaturrecherche

Zur Bestimmung des aktuellen Stands der Technik, wird am Anfang eines wissenschaftlichen Arbeitsprozesses eine systematische Literaturrecherche durchgeführt [3]. Der Prozess der Literaturrecherche lässt sich nach Brocke et al. in fünf Phasen gliedern, wobei der Schwerpunkt auf der Literaturprüfung liegt [3]. Dieses im Folgenden näher geschilderte Vorgehen wurde im Rahmen der vorliegenden Arbeit angewandt (Abbildung 2.1). In der ersten Phase werden Forschungsfragen festgelegt. In der zweiten Phase findet die Organisation der Analyse statt. Hier werden die verwendeten Literaturdatenbanken definiert. In der dritten Phase wird die eigentliche Literatursuche durchgeführt. Hierbei kommen die Methoden *Forward Search* und *Backward Search* zum Einsatz. Die vierte Phase dient zur Analyse der Literaturrecherche. Schließlich wird in der fünften Phase das Ergebnis der Recherche dokumentiert.

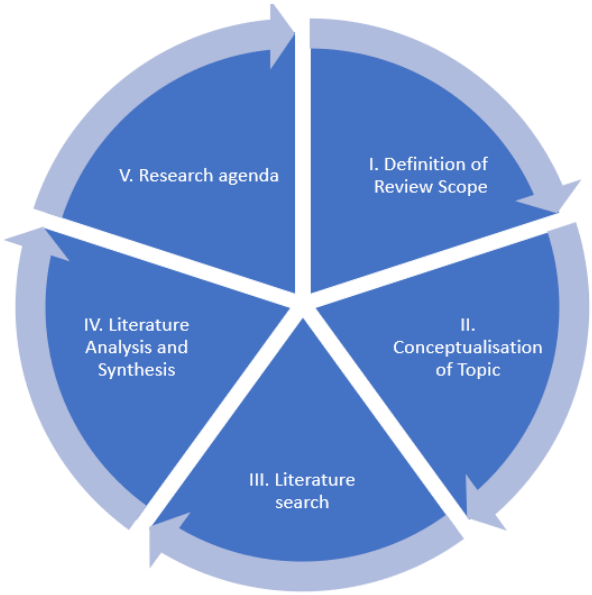


Abbildung 2.1: Methode der Literaturrecherche (Quelle: In Anlehnung an [3])

2.2 Vorgehensweise

Wie bereits erwähnt, werden in der ersten Phase der Literaturrecherche geeignete Forschungsfragen definiert. In der Einleitung wurden zu jedem behandelten Gebiet drei Fragen definiert. Auf diese wird hier nicht mehr weitereingegangen.

Nachdem die Forschungsfragen definiert sind, werden in der zweiten Phase der Literaturrecherche Datenbanken für die Recherche festgelegt. Da die Hochschule die Lizenz zum Zugriff auf folgende Online-Datenbanken besitzt, wurden diese für die Recherche ausgewählt:

- ScienceDirect: Eine digitale Forschungsdatenbank mit wissenschaftlichen und medizinischen Veröffentlichungen.
- Springer Link: Eine große digitale Bibliothek mit Texten aus den Bereichen Engineering, Biomedizin, Sozialwissenschaft etc.
- IEEE Xplore: Eine digitale Forschungsdatenbank für die Suche nach Zeitschriften, Artikeln, Konferenzberichten und technischen Normen aus den Bereichen Informatik, Elektrotechnik und Elektronik.

In der dritten Phase der Recherche findet die eigentliche Literatursuche statt. Dabei ist zunächst zu klären, welche Suchbegriffe infrage kommen. Da im vorliegenden Fall in der ersten Phase entsprechende Forschungsfragen definiert worden waren, bestand nun die Möglichkeit, aus diesen geeigneten Schlüsselwörtern abzuleiten. Anhand dieser wurde dann eine Suche in den oben genannten Datenbanken durchgeführt. Bei der ersten Suche mit deutschen Schlüsselwörtern ergab sich eine niedrige Trefferquote, sodass der Schluss nahelag, dass diese Sprache für eine Recherche der betreffenden Themen nicht geeignet ist. In diesem Sinne wurde für die weitere Suche auf das Englische zurückgegriffen. Dafür wurden passende englische Schlüsselwörter aus den Forschungsfragen abgeleitet. Bei der daraufhin durchgeführten Suchanfrage ergab sich eine hohe Anzahl an Treffern (Abbildung 2.1). Dies lässt sich damit erklären, dass viele wissenschaftliche Arbeiten auf Englisch publiziert werden. Die Trefferquote in IEEE Xplore ist im Vergleich zu den anderen beiden Datenbanken deutlich geringer. Eine Erklärung dafür könnte sein, dass sich IEEE Xplore auf wissenschaftliche Publikationen beschränkt. Bei Springer Link ist die Trefferquote hingegen hoch, jedoch ist der Anteil an wissenschaftlichen Artikeln gering. Um relevante Texte zu extrahieren, wurden die Schlüsselwörter nun in geeigneter Weise mit den logischen Operatoren ‚AND' und ‚OR' verknüpft. Des Weiteren wurden nachfolgende Filter verwendet:

- Zeitraum der Publikation: letzte fünf Jahren
- Themengebiet: künstliche Intelligenz
- Publikationsform: Paper
- Schlüsselwort im Titel enthalten: ja

Nach Einsatz der oben beschriebenen Filter nahm die Zahl der Treffer signifikant ab. In der vierten Phase wurden die verbleibenden Treffer auf ihre Relevanz hin überprüft. Zu diesem Zweck wurden zunächst die Abstracts durchgesehen. Sodann wurden Papers, bei denen das Abstract auf eine thematische Relevanz hindeutete, vollständig gelesen. In der fünften Phase wurden interessante Papers in die Agenda übertragen

2.3 Ergebnis der Literaturrecherche

Tabelle 2.2 stellt das Ergebnis der Recherche dar. Hierbei ist festzustellen, dass zu Beginn des Suchprozesses eine hohe Trefferquote zu verzeichnen war. Nach Anwendung der beschriebenen Filter nahm die Zahl der Treffer dann kontinuierlich ab. Schließlich ergab sich pro Datenbank und Suchstring eine Gesamtzahl von 0 bis maximal zwei relevanten

Suchbegriff	String	Springer Link	IEEE Xplore	ScienceDirect
1	Lecture video segmentation	17964	70	1283
2	Video summarization/lecture	2874	48	85389
3	Scene change detection	73799	2530	57817
4	Audio-visual scene analysis	5380	128	1975
5	Lecture recommender system	8727	14	56449
7	Recommender system evaluation	20197	1455	1115897
8	Video segmentation tool	26512	579	15363

Tabelle 2.1: Trefferquote der ersten Suchstrings

Papers. Insgesamt wurden so für das Themengebiet Videozusammenfassung elf und für das Thema Empfehlungssysteme sieben relevante Papers ermittelt. Diese Papers bilden die Grundlage für die Erarbeitung der vorliegenden Thesis, wobei durch *Forward Search* und *Backward Search* im weiteren Verlauf der Untersuchung weitere Texte hinzugekommen sind.

Datentank	Suchbegriff	Publikations-form Paper	Zeitraum 2017-2021	Themengebiet	Schlüsselwort im Titel enthalten	Relevant nach Lesen des Abstracts	Relevant
ScienceDirect	1	749	40	6	0	0	1
	2	55898	1430	100	10	3	1
	3	4695	2257	84	14	1	1
	4	80	50	10	8	2	1
	5	27181	1016	45	3	2	1
	7	767682	43559	439	6	4	1
	8	500	100	14	0	0	0
IEEE Xplore	1	66	18	5	5	3	1
	2	39	19	10	8	2	1
	3	2007	483	176	5	1	0
	4	98	30	9	5	0	0
	5	12	6	5	5	3	1
	7	1224	524	111	10	5	2
	8	11	7	4	0	0	0
Springer Link	1	13886	3003	2150	10	3	1
	2	1200	792	60	30	4	2
	3	20834	13187	5	5	2	1
	4	1063	275	60	6	5	1
	5	7234	2507	89	7	5	2
	6	1897	501	40	2	2	0
	7	40	14	10	7	5	2
	8	20	10	8	0	0	0

Tabelle 2.2: Ergebnis der Literaturrecherche

2.4 Verwandte Untersuchungen

Im Bereich der Empfehlungssysteme und der Videozusammenfassung wurde in der letzten Dekade eine Vielzahl an Artikeln publiziert. Der nachfolgende Abschnitt thematisiert aktuelle Ansätze in diesen Bereichen und erörtert, ob bereits Ansätze existieren, die die definierte Forschungsfrage beantworten. Die Publikationen beruhen dabei auf einer Vielzahl von Herangehensweisen.

Empfehlungssystem im E-Learning-Kontext

Dan Fu et al. präsentieren einen Ansatz zur Analyse von Vorlesungsvideos basierend auf dem Inhalt. Die Autoren verwenden das Verfahren der Schlüsselbildextraktion, um Input-Videos zu segmentieren. Die Frames werden in bestimmten Zeitintervallen gelesen. Mithilfe einer Anwendung der Optical-Character-Recognition (OCR)- Technologie auf Schlüsselbilder werden Schlüsselwörter extrahiert. Weiterhin verwenden die Autoren Automatic-Speech-Recognition (ASR)- Techniken, um textuelle Features aus der Audiospur zu extrahieren. Auf Grundlage der gefundenen Schlüsselwörter, werden Weblinks, Bildlinks und YouTube-Links bereitgestellt. Benutzer können über die bereitgestellten Links auf die entsprechenden Videos zugreifen und eine Bewertung für das angesehene Video abgeben. Auf Basis dieser Bewertung werden dann Empfehlungen generiert. Das Empfehlungssystem beruht auf der Pearson-Korrelation und dem Cosinus-Ähnlichkeitsmaß [4].

Fatiha Bousbahi und Henda Chorfi schlagen ein Massive-Open-Online-Course (MOOC)-Empfehlungssystem vor, das auf dem Case-Based-Reasoning (CBR))-Ansatz und Informationssuchtechniken basiert. Auf Basis der Lernprofile, Bedürfnisse und des Wissensstandes der Lernenden, erstellt das System die passende Empfehlung. Bei CBR handelt es sich um eine Art inhaltsbasiertes Empfehlungssystem, das als Basis für die Beschreibung des Empfehlungsobjekts verwendet wird [5].

Harshit Jain und Anika schlagen eine Methode zur Entwicklung eines MOOC Empfehlungssystems vor, wobei sie von Data-Mining-Techniken wie Random Forest, Classification Tree und K-Nearest Neighbors Gebrauch machen. Zunächst werden die Benutzer anhand ihrer Aktivitätsprotokolle auf zwei Kategorien aufgeteilt: aktive und passive Lernende. Nach der Anwendung von Data Mining Ansätzen wird die Kursempfehlung separat für jede Kategorie erstellt [6].

Cristóbal Romero et al. wenden das Data Mining Tool Weka auf das LMS Moodle an, um Empfehlungen für Lehrende und Lernende zu erstellen. Mit diesem Ansatz werden

Aktivitätsprotokolle der Benutzer heruntergeladen und extern mit dem Open Source Tool Weka analysiert. Mithilfe der Visualisierungs- und Klassifikationsfunktion könnten interessante Informationen identifiziert werden [7].

Zameer Gulzar et al. schlagen eine hybride Methodologie in Verbindung mit einer Ontologie vor, um personalisierte Kursempfehlungen basierend auf Benutzerpräferenzen abzugeben [8]. Die Ontologie bietet die Möglichkeit, die Eigenschaften von Objekten und ihre Beziehungen zueinander darzustellen, indem eine Reihe von Konzepten und Kategorien definiert wird, die das Objekt repräsentieren.

Videozusammenfassung

In der Publikation [9] wird ein Ansatz zur Segmentierung von Lernvideos unter Verwendung der natürlichen Sprachverarbeitung (NLP)) gezeigt. Ziel dabei ist es, entscheidende linguistische Features aus dem Video zu extrahieren. Die Autoren nutzen die visuellen, die Audio- und die textuellen Features, um zeitliche Feature-Vektoren zu erstellen, die der verbesserten Segmentierung dienen. Anschließend wird NLP-Kosinusähnlichkeit auf die Cluster angewendet, um verschiedene Themen in Videos zu identifizieren. Für die textuelle und die Audio-Feature-Extraktion nutzen die Autoren ASR und OCR

In der Arbeit [10] wird eine Methode zur Zusammenfassung von Whiteboard-Videos vorgeschlagen, bei der Feature-Repräsentationen von erkannten handschriftlichen Inhaltsregionen extrahiert werden, um die eindeutigen Inhalte zu bestimmen. Dabei wird das Histogramm von Gradienten verwendet, um die erkannten Regionen darzustellen.

In [11] wird eine Methode zur Segmentierung von Videos auf Grundlage des zuvor notierten Beginns und Endes jedes Themas beschrieben. Anschließend werden die einzelnen Themen auf Basis der Audio-Features zusammengefasst.

In der Publikation [12]wird eine Methode zur Extraktion von Inhalten eines Vorlesungsvideos unter Verwendung der Sprecher-Aktionen gezeigt. Dabei wird jedes Vorlesungsvideo in kleine zeitliche Einheiten unterteilt, die als Aktionssegmente bezeichnet werden. Zur Analyse der Pausen des Sprechers werden Körper- und Handskelett-Daten extrahiert, um daraus bewegungsbasierte Merkmale für jedes Segment zu berechnen. Danach wird die dominante Sprecher-Aktion jedes Segments anhand des Random-Forest-Verfahrens und des bewegungsbasierten Merkmals klassifiziert.

3 Videozusammenfassung und audiovisuelle Szeneexploration

Dieses Kapitel beschäftigt sich mit den theoretischen Grundlagen zu den Forschungsbereichen der Videozusammenfassung und der audiovisuellen Szeneanalyse. Hier werden die wesentlichen Konzepte und Methoden dieser Bereiche dargestellt.

3.1 Videozusammenfassung

Bevor das Empfehlungssystem für Lerneinheiten der Online-Lehre realisiert werden kann, müssen zunächst passende Lerneinheiten aus den Vorlesungsaufzeichnungen erzeugt werden. Dazu werden einige Videozusammenfassungstechniken angewendet. Im nachfolgenden Abschnitt werden gängige Methoden aus der Literatur vorgestellt.

Ein Video ist eine Sammlung von Bildern in einer festgelegten Reihenfolge, die mit hoher Geschwindigkeit ablaufen [13]. Die Videozusammenfassung ist eine Technik, mit der eine kurze Zusammenfassung des Inhalts eines längeren Videos erstellt wird, indem die für potenzielle Nutzer informativsten oder interessantesten Bereiche des Videos ausgewählt werden. Der Output einer Videozusammenfassung besteht in der Regel aus einer Reihe von Schlüsselbildern (Frames) oder einem kurzen Video, das aus dem Originalvideo extrahiert wurde. Die Videozusammenfassung zielt darauf ab, das Durchsuchen einer großen Sammlung von Videodaten zu beschleunigen und so eine effiziente Darstellung des Videoinhalts zu ermöglichen [14, 4]. Im Allgemeinen wird der Prozess der Videozusammenfassung in drei Schritte unterteilt. Im ersten Schritt wird das Video analysiert, um audiovisuelle Informationen zu ermitteln. Im zweiten Schritt werden Bilder ausgewählt, die den Inhalt des Videos repräsentieren. Schließlich wird das Ergebnis im dritten Schritt organisiert [15]. In der Literatur werden zwei Arten von Videozusammenfassungen unterschieden: die statische Zusammenfassung, auch Key-Frame-Summarization genannt, und die dynamische Zusammenfassung, die auch als Video-Skimming-Summarization bezeichnet wird [15, 13].

Bei der statischen Zusammenfassung besteht das Ergebnis der Zusammenfassung aus einzelnen Schlüsselbildern mit höchster Priorität. Diese werden im ersten Schritt extrahiert, indem Bilder gleichmäßig übersprungen oder zufällig ausgewählt werden. Im nächsten Schritt können die Schlüsselbilder dann durch eine kollaborative Darstellung der benachbarten Bilder bestimmt werden. Mit dieser Methode ist es möglich, Bilder mit minimalen Rekonstruktionsfehlern auszuwählen [15]. Zur Bestimmung der Schlüsselbilder wird am häufigsten von farbbasierten Verfahren Gebrauch gemacht. Dabei wird das Farbhistogramm der einzelnen Bilder aus Red-Green-Blue (RGB) oder Hue-Saturation-Value (HSV) berechnet. Hieraus ergibt sich die Farbverteilung der Bilder, anhand derer zwei aufeinanderfolgende Bilder miteinander verglichen werden können. Dabei wird angenommen, dass Bilder eines Videos in derselben Aufnahme ähnliche Werte haben [15].

Das Ergebnis der dynamischen Zusammenfassung ist ein kurzes Video, in dem die interessantesten Szenen aus dem Eingangsvideo in Form eines Abstracts wiedergeben werden. Hierzu werden Techniken wie das Single-Value-Decomposition (SVD), das Bewegungsmodell und die semantische Analyse angewendet [15, 13]. In Tabelle 3.1 werden die Unterschiede zwischen der statischen und der dynamischen Zusammenfassung aufgeführt.

Statische Zusammenfassung	Dynamische Zusammenfassung
Erstellen einer Reihe von Schlüsselbildern als Zusammenfassung.	Erstellung eines kleinen Videos als Zusammenfassung.
Enthält keine Bewegungsinformationen.	Enthält Bewegungsinformationen.
Eingeschränkte Benutzererfahrung.	Hohe Benutzerfreundlichkeit.
Keine Zeitgrenzen und Synchronisierung erforderlich.	Zeitliche Beschränkung und Synchronisierung ist erforderlich.
Nur Videobilder berücksichtigt.	Video-, Audio- und Textdaten werden berücksichtigt.
Hilft bei der Indizierung und Suche von Videos.	Hilft bei der Indizierung und Suche von Videos, aber die Leistung ist geringer als bei der auf Schlüsselbildern basierenden Zusammenfassung.

Tabelle 3.1: Unterschiede zwischen statischer und dynamischer Zusammenfassung (Quelle: In Ablehnung an [15])

3.2 Audiovisuelle Szeneexploration

Der Vorteil von Videotechniken besteht darin, dass sie sowohl zur Verarbeitung von Audio-Daten als auch zur Verarbeitung von visuellen Daten geeignet sind, sodass beide während einer Szeneexploration untersucht werden können. Bei der audiovisuellen Exploration ist die Salienz ein entscheidender Faktor für die Bestimmung von zentralen Ereignissen in dem Video [16]. In der Literatur wird unterschieden zwischen akustischer Salienz, die ein hörbares Ereignis darstellt, und visueller Salienz, die den sichtbaren Anteil beschreibt. Durch Salienz wird Aufmerksamkeit erzeugt und der Fokus auf das Ereignis gelenkt. Die akustische Salienz kann z. B. durch eine starke Änderung im Audio-Spektrum festgestellt werden. Mit diesem Faktor können Ereignisse wie das Fallen eines Objekts oder Konversationen in Videos dargestellt werden. Die akustische Salienz wird auf Basis der Kurzzeit-Fouriertransformation (STFT) bestimmt. Bei der visuellen Salienz wird eine Salienzkarte generiert, indem für jede Bildstelle die Merkmalsdifferenzen berechnet, normiert und zusammenaddiert werden. Die visuelle Salienz kann verwendet werden, um die spontane Rotation der Aufmerksamkeit und des Blicks zu erklären. Zur Bestimmung der visuellen Salienz wird die diskrete Kosinus-Transformation (DCT) auf Grauwert-Bilder angewendet. Außer der Salienz werden im Prozess der audiovisuellen Analyse typischerweise Merkmale wie Farbe, Schattierung, Form, Bewegung, Textur und Kontext genutzt. In vielen Fällen können Objekte erfolgreich anhand ihrer Form erkannt werden [16, 17].

4 Empfehlungssysteme

In diesem Kapitel wird auf die theoretischen Grundlagen zu Empfehlungssystemen eingegangen. Dies beinhaltet die Definition, die wesentliche Typen, die eingesetzten Methoden und Annahmen sowie die Evaluationsmethoden. Speziell wird auf das Verfahren des kollaborativen Filterns eingegangen, das für die vorliegende Arbeit von zentraler Bedeutung ist.

Das Empfehlungssystem hat seinen Ursprung in den Forschungsfeldern des Data-Minings, des Information-Retrievals, der Statistik und des maschinellen Lernens [18]. Dem ursprünglichen Konzept zufolge handelte es sich bei einem Empfehlungssystem um ein System, dem Empfehlungen als Input zugeführt werden, woraufhin es diese aggregiert und an geeigneten Empfängern weiterleitet. Heutzutage wird dieser Begriff jedoch weiter gefasst. So wird das Empfehlungssystem gegenwärtig als ein System beschrieben, das den Effekt hat, Benutzer auf eine personalisierte Art und Weise bei der Auswahl aus einer großen Menge nützlicher Objekte zu unterstützen [19]. Empfehlungssysteme beeinflussen die Entscheidungen von Menschen im alltäglichen Leben, ohne dass sie selbst über ein Bewusstsein verfügen [20]. So sehen sich Nutzer z. B. beim Online-Händler Amazon.com, einer der bekanntesten Applikation mit integriertem Empfehlungssystem, mit einer großen Auswahl an potenziell interessanten Produkten konfrontiert [20]. Das im Shop integrierte Empfehlungssystem spielt hier die Rolle eines Assistenzsystems und schlägt dem Benutzer Artikel vor, die für ihn interessant sein könnten. Dies erfolgt nach dem Schema: „Kunden, die diesen Artikel gekauft haben, haben auch einen anderen Artikel gekauft“(Amazon.com). Allgemein werden Empfehlungssysteme heute in vielen Bereichen eingesetzt. Abbildung 4.1 bietet eine Übersicht über Realwelt-Empfehlungssysteme.

Empfehlungssysteme	Empfohlene Produkte
Amazon.com	Bücher und andere Produkte
Netflix	DVDs, Videostreaming
Jester	Witze
GroupLens	Nachrichten
MovieLens	Filme
ast.fm	Musik
Google News	Nachrichten
Google Search	Werbeanzeigen
Facebook	Freunde, Werbeanzeigen
Pandora	Musik
YouTube	Online Videos
Tripadvisor	Reiseprodukte
IMDb	Filme

Tabelle 4.1: Beispiele Empfehlungssysteme aus dem Alltag (Quelle: In Anlehnung an [21])

Das Empfehlungsproblem wird häufig entweder als Prediction Version of Problem oder Ranking Version of Problem formuliert [21]:

- Prediction Version of Problem

  Bei diesem Ansatz wird die Bewertung eines Benutzers vorhergesagt, wobei vorausgesetzt wird, dass Daten zum Bewertungsverhalten vorhanden sind, anhand deren die Präferenzen des Benutzers für ein Objekt ermittelt werden können. Dieses Problem lässt sich auch als *Problem der Matrixvervollständigung* beschreiben, da hierbei eine ursprünglich unvollständige Matrix n X m existiert, deren fehlende Werte anhand von Data-Mining-Algorithmen vorhergesagt werden sollen.

- Ranking Version of Problem

  In der Praxis ist es nicht notwendig, die Bewertung eines Benutzers vorherzusagen; vielmehr reicht es aus, die Top-N-Objekte zu finden, die ihn interessieren könnten. Dieser Ansatz, der auch als Top-N-Empfehlungsproblem bezeichnet wird, stellt die am weitesten verbreitete Methode dar.

In der Publikationen [21, 18] wird das Empfehlungsproblem wie folgt beschrieben :

Sei U die Menge aller Benutzer im System, I die Menge aller Objekte im System, R eine vorher festgelegte Menge mit reellen Werten zur Beschreibung der Nutzbarkeit. Sei f_u die Funktion zur Ermittlung der Nutzbarkeit eines Objekts i für einen Benutzer u (Gleichung 4.1).

$$f_u U * I -> R \tag{4.1}$$

Um dem Benutzer $u \in U$ Objekte zu empfehlen, werden Objekte $i \in I$ gesucht, die die Nützlichkeit für den Benutzer u maximieren (Gleichung 4.2).

$$\forall u \in U i_{u=} argmax R(i, u) \tag{4.2}$$

In der Arbeit [21] werden vier Ziele für Empfehlungssysteme beschrieben:

- **Relevance**: Es sollen Objekte empfohlen werden, die für den Nutzer relevant sind.
- **Novelty**: Empfohlene Objekte sollen dem Benutzer zuvor unbekannt sein.
- **Serendipity**: Das empfohlene Objekt soll nicht vom Nutzer erwartet werden.
- **Increasing recommendation diversity**: Die empfohlenen Objekte sollen vielfältig sein, um auf diese Weise die Chance zu erhöhen, dass den Benutzer einige dieser Objekte tatsächlich interessieren.

Durch unterschiedliche Anwendungskontexte ergeben sich unterschiedliche Ansätze für Empfehlungssysteme [21, 18]. In den folgenden Abschnitten wird auf einige solche Ansätze eingegangen.

4.1 Kollaboratives Filtern

Das kollaborative Filtern (collaborative filtering) ist eine der erfolgreichsten und am weitesten verbreiteten Technologien zur Gestaltung von Empfehlungssystemen [22]. Dieser Erfolg lässt sich damit erklären, dass die Technologie kein Wissen über das jeweilige Einsatzgebiet erfordert. Generell werden zwei Arten des kollaborativen Filterns unterschieden: das memorybasierte (memory-based) und das modellbasierte (model-based) Verfahren [19]. Beim memorybasierten Ansatz erfolgt die Vorhersage der Bewertung eines Objekts durch einen Benutzer anhand historischer Daten zum Bewertungsverhalten anderer Benutzer. Dagegen erfolgt sie beim modellbasierten Ansatz anhand von Verfahren des maschinellen Lernens. Hierbei wird der Versuch unternommen, Muster in Bewertungen verschiedener Benutzer zu erfassen und aus diesen ein Modell abzuleiten, auf Basis dessen dann neue Objekte vorgeschlagen werden können.

4.1.1 Memorybasiertes kollaboratives Filtern

In der Literatur wird beim memorybasierten kollaborativen Filtern zwischen eine benutzerbasierte (user-based) und eine objektbasierte (item-based) Variante unterschieden [23, 24, 18]:

4.1.1.1 Objektbasiertes kollaboratives Filtern

Dieses Verfahren verwendet zur Erstellung der Empfehlung die Ähnlichkeit von Objekten basierend auf dem Benutzerbewertungsverhalten. Unterschiedliche Objekte, die von unterschiedlichen Benutzern ähnliche Bewertungen erhalten haben, werden als identisch betrachtet, sofern der betreffende Benutzer über die gleichen Präferenzen verfügt. Abbildung 4.1 veranschaulicht den Ablauf des Algorithmus.

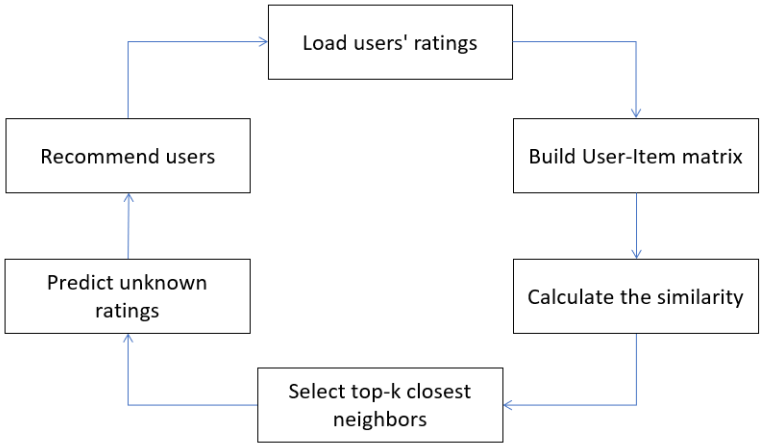


Abbildung 4.1: Ablauf des Algorithmus für das kollaborative Filtern (In Anlehnung an [23])

Im ersten Schritt des Algorithmus werden alle bewertete Objekte I_u eines Benutzers herangezogen. Im zweiten Schritt wird auf Basis dieser Daten die Zuordnungsmatrix Benutzer-Objekte erstellt (Abbildung 4.2). In dieser Matrix repräsentieren die Zeilen $u_1 ... u_n$ die Benutzer ($u \in U$) und die Spalten $i_1 ... i_m$ die Objekte ($i \in I$). In Zellen sind die einzelnen Benutzerbewertungen ($R_{(u,i)}$) für Objekte festgehalten. Unbekannte bzw. unbewertete Objekte sind mit Symbol $\oslash$ gekennzeichnet [23].

Im dritten Schritt des Algorithmus wird die Berechnung der Ähnlichkeit zu einem Zielobjekt $i_u \in I$ durchgeführt. Als Metrik kommt dabei häufig die Kosinusähnlichkeit zum Einsatz (Gleichung 4.3) .

	i_1	i_2	i_3	...	i_j	...	i_m
u_1			R	$\oslash$	R		
u_2			R	$\oslash$	$\oslash$		
u_3			R	$\oslash$	R		
...			...	...	...		
u_j			...	...	...		
...			...	...	...		
u_n			R	$\oslash$	R		

Abbildung 4.2: Benutzer-Objekt-Matrix

$$sim(i,j) = cos(i,j) = \frac{i.j}{\parallel i \parallel \parallel j \parallel} = \frac{\sum_{u \in U} r_{u,i}.r_{u,j}}{\sqrt{\sum_{u \in U} r_{u,i}^2}\sqrt{\sum_{u \in U} r_{u,j}^2}} \tag{4.3}$$

Wie in der vorhergehenden Gleichung dargestellt, erfolgt die Messung der Ähnlichkeit zwischen zwei Objekten i und j wie folgt: beide Objekte werden als Vektoren dargestellt, um dann die Kosinusähnlichkeit der Vektoren zu berechnen, wobei $r_{u,i}$die Bewertung des Benutzers u für ein Objekt i repräsentiert.

Im vierten Schritt des Algorithmus wird die Bewertung $R_{(u,i)}$ des Benutzers vorhergesagt. Dazu werden die k nächsten Nachbarn aus der vorherigen Berechnung herangezogen. Als Nachbarn werden Objekte bezeichnet, die eine starke Ähnlichkeit zu dem Zielobjekt Objekt i aufweisen. Die folgende Formel wird zur Berechnung der Vorhersage verwendet. Dabei enthält die Menge N_k die nächsten k ähnlichsten Objekte zu i und $r_{u,j}$ ihre Bewertung [23, 18].

$$r_{u,i} = \frac{\sum_{j \in N_k} sim(i,j).r_{u,j}}{\sum_{j \in N_k} sim(i,j)} \tag{4.4}$$

Im letzten Schritt des Algorithmus schlägt das System diejenigen Objekte mit der höchsten geschätzten Bewertung vor [18].

4.1.1.2 Benutzerbasiertes kollaboratives Filtern

Im Gegensatz zum objektbasierten Verfahren besteht das Ziel des benutzerbasierten kollaborativen Filterns darin, Benutzer zu finden, die ein ähnliches Bewertungsverhalten wie der aktive Benutzer zeigen, um daraufhin ihre Bewertungen für die Vorhersage der Bewertung des aktiven Benutzers zu verwenden. Dieser Ansatz beruht auf der Grundannahme, dass Benutzer, die unterschiedliche Objekte in ähnlicher Weise bewerten,

dieselben Interessen haben. Um dem aktiven Benutzer eine geeignete Empfehlung vorschlagen zu können, findet das System die k ähnlichsten Nutzer im System, um ihm daraufhin diejenigen Objekte mit der höchsten Bewertung zu empfehlen [24].

Der im vorigen Abschnitt beschriebene Algorithmusablauf gilt generell auch für die benutzerbasierte Methode. Allerdings findet sich hierbei der Unterschied, dass die Berechnung der Ähnlichkeit (Schritt 3) auf Basis der Benutzerähnlichkeit erfolgt. Für diese Berechnung wird häufig die Pearson-Korrelation als Metrik verwendet (Gleichung 4.5) [23, 24, 18].

$$sim(x,y) = \frac{\sum i_{i=1}^{n}(x_i - \overline{x})(yi - \overline{y})}{\sum i_{i=1}^{n}\sqrt{(x_i - \overline{x})^2}\sum i_{i=1}^{n}\sqrt{(y_i - \overline{y})^2}} \tag{4.5}$$

Nach Ermittlung der k nächsten Nachbarn, wird die Nutzbarkeit berechnet (Gleichung 4.6) [23, 24, 18].

$$r(u,i) = \overline{r}_u + \frac{\sum_{v \in N_k} sim(u,v)(r_{v,i}\overline{r}_v)}{\sum_{v \in N_k} sim(u,v)} \tag{4.6}$$

4.1.2 Modellbasiertes kollaboratives Filtern

Wie bereits erwähnt wird bei diesem Ansatz von einem maschinellen Lernverfahren Gebrauch gemacht, um Muster in Bewertungen von Benutzern zu ermitteln und dann auf dieser Basis neue Objekte vorzuschlagen. In diesem Bereich hat sich aufgrund seiner hohen Empfehlungsgenauigkeit das Latent-Faktor-Modell mit der Matrix-Faktorisierung etabliert. In ihrer Grundform charakterisiert die Matrix-Faktorisierung Benutzer und Objekte durch einen Vektor von Einflussfaktoren, die aus Bewertungsinformationen abgeleitet werden [25]. Die Matrix-Faktorisierung, die im Kontext von Empfehlungssystemen häufig auch Singular-Value-Decomposition (SVD) genannt wird, wird formal beschrieben als eine Matrix A von Bewertungen, die in ein Produkt von Matrizen $A = U \sum V^T$ zerlegt wird, wobei $\sum V^T$ die latenten Faktoren für die Objekte und U die Benutzer darstellen (Abbildung 4.3) [26]. Die Fragezeichen in dieser Abbildung repräsentieren die zur Wiederherstellung der ursprünglichen Matrix A zu berechnenden Features.

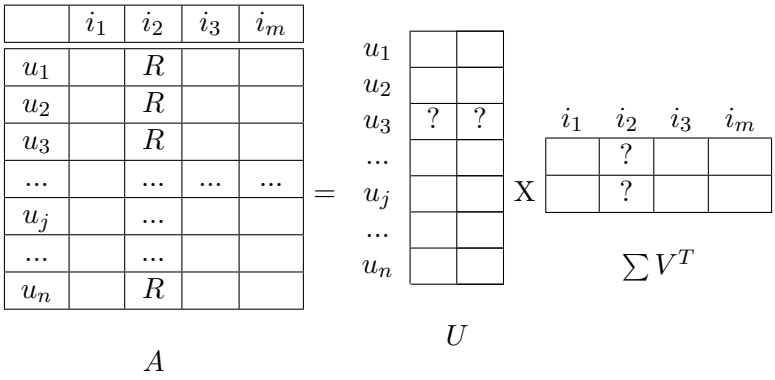


Abbildung 4.3: Matrix-Faktorisierung

4.1.3 Nachteile des kollaborativen Filterns

- **New-Item-Problem**

  Neue Objekte können bei der Empfehlung nicht berücksichtigt werden, da dem System zu diesen keine Bewertungsdaten vorliegen. Eine mögliche Lösung hierfür wäre die Bereitstellung von Empfehlungen auf Basis systemübergreifender Informationen [20].

- **New-User-Problem**

  Ähnlich wie beim New-Item-Problem verfügt das System auch über keine Informationen über neue Benutzer im System. Diese müssen erst eine Anzahl an Objekten bewerten, um vom System berücksichtigt werden zu können. Eine mögliche Lösung hierfür wäre die Erstellung eines initialen Benutzerprofils [20].

- **Sparsity**

  Dieser Begriff beschreibt eine geringe Abdeckung von Bewertungen eines Benutzers zu bestimmten Objekten. Dieses Problem tritt häufig auf, wenn es eine große Anzahl an Benutzern oder Objekten im System gibt. Eine Lösung dafür könnte darin bestehen, bei der Berechnung der Ähnlichkeit zwischen Benutzern nicht nur die Benutzer-Objekt-Matrix, sondern auch benutzerabhängige Faktoren wie demografische Daten einzubeziehen [20].

4.1.4 Implizites und explizites Feedback

Die Informationen, die das Empfehlungssystem von den Benutzern zur Berechnung der Nutzbarkeit benötigt, werden als Feedback bezeichnet. Es gibt zwei Arten von Feedback:

explizites und implizites Feedback. Bei explizitem Feedback gibt der Benutzer explizit eine Bewertung auf einer Skala ab, die der Höhe seines Interesses/seiner Zufriedenheit an bzw. mit einem Objekt entspricht. Bei einer Fünf-Punkte-Skala kann ein Wert aus der Menge $\{-2, -1, 0, 1, 2\}$ gezogen werden, wobei der Wert –2 eine starke Unzufriedenheit und der Wert 2 eine starke Zufriedenheit ausdrückt. Die Anzahl der möglichen Bewertungspunkte kann je nach System variieren. Häufig werden Fünf-Punkte-, Sieben-Punkte- und Zehn-Punkte-Systeme verwendet [21].

Bei implizitem Feedback werden die Benutzerpräferenzen dagegen aus den Aktivitäten der Benutzer abgeleitet. Im E-Commerce-Bereich kann z. B. das Kaufverhalten von Kunden als implizite Bewertung verwendet werden. Wenn der Kunde ein Produkt kauft, ist es wahrscheinlich, dass er Interesse an dem Produkt hat. Ähnlich verhält es sich im Streamingbereich: Wenn ein Nutzer einen Film anschaut, ist es wahrscheinlich, dass er an diesem interessiert ist. Umgekehrt bedeutet die Tatsache, dass ein Kunde ein Produkt aus einer großen Menge nicht kauft, jedoch nicht zwangsläufig, dass er nicht an diesem interessiert ist. So besteht auch die Möglichkeit, dass er sich über dessen Existenz gar nicht im Klaren ist [21].

4.2 Andere Empfehlungssysteme

Eine andere Art von Empfehlungssystem bildet das inhaltsbasierte Filtern (Content-based filtering). Im Gegensatz zum kollaborativen Filtern wird hier zur Erstellung der Empfehlungen auf die Beschreibung von Objekten zurückgegriffen. Dieses Verfahren beruht auf der Annahme, dass sich ein Nutzer, der positiv auf Objekte mit bestimmten Eigenschaften reagiert hat, auch für Objekte interessiert, in deren Beschreibung dieselben Eigenschaften genannt werden [21].

Eine dritte Art von Empfehlungssystem bildet das hybride Filtern, bei dem kollaborative und inhaltsbasierte Methoden miteinander kombiniert werden. Durch die Kombination wird hierbei die Möglichkeit eröffnet, bestimmte Einschränkungen der beiden Methoden umzugehen. In der Publikation [18] werden verschiedene Möglichkeiten einer solchen Kombinierung beschrieben:

- **Combining Separate Recommenders**: Bei diesem Ansatz werden kollaborative und inhaltsbasierte Methoden unabhängig voneinander implementiert und die Ergebnisse kombiniert.
- **Adding Content-Based Characteristics to Collaborative Models**: Bei diesem Ansatz basiert das Empfehlungssystem auf kollaborativen Techniken, wobei

inhaltsbasierte Methoden nur dann verwendet werden, wenn keine Informationen zur Anwendung des kollaborativen Verfahrens vorliegen.

- **Adding Collaborative Characteristics to Content-Based Models**: Bei diesem Ansatz werden kollaborative Methoden in ein inhaltsbasiertes Verfahren integriert. Falls keine Informationen zur Anwendung inhaltsbasierter Verfahren existieren, kann die Empfehlung auf Grundlage der kollaborativen Methode hergeleitet werden.
- **Developing a Single Unifying Recommendation Model**: Bei diesem Verfahren wird ein Modell entwickelt, in dem Bewertungen und Objektattribute miteinander kombiniert werden.

4.3 Evaluation der Empfehlungssysteme

Die Evaluierung von Empfehlungssysteme und ihre Algorithmen wird zu einer herausfordernde Aufgabe. So kann ein Algorithmus bei einem Datensatz gut und bei einem anderen schlecht funktionieren. Außerdem können Empfehlungssysteme für unterschiedliche Zwecke verwendet werden. Für Systeme, bei denen die Empfehlungen zur Unterstützung von Entscheidungen eingesetzt werden, haben Forscher in der Vergangenheit die Genauigkeit als Metrik verwendet, um die Qualität des Empfehlungssystems zu messen. Heutzutage liegt der Fokus unter anderem auf der Benutzerzufriedenheit. Hierbei wird gemessen, wie oft ein System seine Nutzer zu einer falschen Entscheidung führt . Diese beiden Evaluationsmethoden werden auch Online- und Offline-Methoden genannt [22].

Bei der Online-Methode wird die Reaktion des Benutzers auf die im Livesystem präsentierten Empfehlungen gemessen. Bei dieser Methode sind die Klassifikationsmetriken die bedeutendsten Faktoren. Da die Online-Evaluierung eine aktive Beteiligung der Nutzer erfordert, ist diese für Benchmarking und Forschung häufig ungeeignet. Generell ist es von Bedeutung, ein System mit mehreren Datensätzen zu testen, um sicherzustellen, dass der Algorithmus unter verschiedenen Bedingungen funktioniert. In diesen Fällen ist die Offline-Evaluierung geeignet, wobei historische Daten der Benutzer verwendet werden [21]. Diese Methode wird in der Literatur weit häufiger behandelt, obwohl die Online-Methode zum Vervollständigen der Evaluierung notwendig ist. Im Folgenden werden die Evaluationsmetriken beschrieben.

4.3.1 Genauigkeitsmetriken

Genauigkeitsmetriken messen, wie nah die vom Empfehlungssystem vorhergesagten Bewertungen an den tatsächlichen Benutzerbewertungen liegen. Diese Metriken werden von der Offline-Methode verwendet. Als Genauigkeitsmetriken fungieren der mittlere absolute Fehler (MAE) und der mittlere quadratische Fehler (RMSE) [21].

Der **MAE** misst die durchschnittliche absolute Abweichung zwischen einer vorhergesagten Bewertung und der tatsächlichen Bewertung des Nutzers. Zur Berechnung wird der Datensatz in einen Trainings- und einen Testdatensatz aufgeteilt. Daraufhin wird das Modell mit dem Trainingsdatensatz angelernt und mit dem Testdatensatz überprüft. Dabei werden die tatsächlichen Benutzerbewertungen $R_{u,i}$ geheim gehalten und wird der MAE mit der vorhergesagten Bewertung berechnet [21].

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|r_i - R_i| \tag{4.7}$$

Der **RMSE** ist ein mit dem MAE zusammenhängendes Maß, das die Fehlerquadrate summiert. Größere Fehler sind mittels RMSE schneller sichtbar als kleinere. Aus diesem Grund wird RMSE am häufigsten zur Evaluierung von Empfehlungssystemen eingesetzt [22].

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_i - R_i)^2} \tag{4.8}$$

4.3.2 Klassifikationsmetriken

Klassifizierungsmetriken messen, wie häufig das Empfehlungssystem den Benutzer aufgrund der Vorhersage zu einer richtigen oder falschen Entscheidung führt. Diese Metrik wird häufig bei der Online-Methode verwendet. Die Evaluation dieser Systeme entwickelt sich zu einem klassischen Klassifikationsproblem mit vier unterschiedlichen Klassen: True-Positive, False-Positive, False-Negative und True-Negative (Abbildung 4.2)[22].

Die bei diesem Ansatz verwendeten Evaluationsmetriken sind *Precision* und *Recall*. Die Precision (Gleichung 4.9) entspricht dabei dem Anteil von gut klassifizierten Objekten unter den insgesamt vorgeschlagenen Objekten. Hieraus ergibt sich die Wahrscheinlichkeit, inwiefern ein vorgeschlagenes Objekt für den Benutzer relevant ist [22]

System/Benutzer	Empfohlen (System)	Nicht empfohlen (System)
Genutzt (Benutzer)	True-Positive	False-Negative
Ungenutzt (Benutzer)	False-Positive	True-Negative

Tabelle 4.2: Klassifikationstabelle

$$Precision = \frac{TP}{TP + FP} \tag{4.9}$$

Der Recall (Gleichung 4.10) lässt sich definieren als das Verhältnis der ausgewählten relevanten Objekte zur Gesamtzahl der verfügbaren relevanten Objekte. Somit wird hierdurch die Wahrscheinlichkeit angegeben, dass ein relevantes Objekt ausgewählt wird [22].

$$Recall = \frac{TP}{TP + TN} \tag{4.10}$$

Es gibt verschiedene Ansätze, um Precision und Recall miteinander zu verbinden. Einer dieser Ansätze ist die F1-Metrik (Gleichung 4.11), die beide Metriken zu einer Zahl kombiniert. Sie dient dazu, das harmonische Mittel beider Werte zu berechnen [22].

$$F1 = 2.\frac{Precision.Recall}{Precision + Recall} \tag{4.11}$$

5 Empfehlungssystem für Online-Vorlesungen

Im vorigen Kapitel wurden die theoretischen Grundlagen zu Empfehlungssystemen dargestellt. Dabei wurden verschiedene Methoden einschließlich der diesen zugrunde liegenden Annahmen erläutert. Ein mögliches Anwendungsgebiet für Empfehlungssysteme bildet der E-Learning-Bereich und insbesondere das Feld der Online-Vorlesungen. In diesem Kapitel wird der Versuch unternommen, die vorliegende Arbeit thematisch genauer in dieses Gebiet einzuordnen. Hierfür werden einige Annahmen und Methoden aus dem Grundlagenkapitel aufgegriffen und auf den untersuchten Kontext übertragen. Dabei wird insbesondere der Bereich des E-Commerce mit demjenigen des E-Learnings verglichen. Außerdem wird in diesem Kapitel die Struktur einer Onlinevorlesung beschrieben, wobei diese Beschreibung auch als Basis für die automatisierte Zerlegung der Aufzeichnungen dient.

5.1 Empfehlungssystem im Vorlesungskontext

Empfehlungssysteme werden im E-Commerce-Bereich bereits erfolgreich eingesetzt. Obwohl die eingesetzten Methoden denjenigen des E-Learnings ähneln, sind mit einige Unterschiede zu benennen [2]:

- Ziel: Das Ziel von Empfehlungssystemen im E-Commerce-Bereich ist es, den Umsatz zu steigern. Dieses Ziel ist messbar, z. B. in Form von Geldbeträgen. Im E-Learning-Bereich besteht das Ziel von Empfehlungssystemen dagegen in einer Verbesserung des Lernprozesses, was eher subjektiv und schwierig zu messen ist.
- Nutzungszweck: Im E-Commerce-Bereich haben Empfehlungssysteme den Zweck, die Kunden beim Kauf zu unterstützen, während beim E-Learning die Aufgabe besteht, die Lernenden beim Lernen zu unterstützen.
- Technik: Die Anwendung von Empfehlungssystemen im E-Learning-Bereich geht mit besonderen Herausforderungen einher, die in anderen Bereichen nicht zu fin-

den sind. Dies umfasst insbesondere die Notwendigkeit, die pädagogischen Aspekte des Lernenden und des Systems zu berücksichtigen. Dementsprechend können einige der traditionellen, im Grundlagenkapitel beschriebenen Techniken auf das E-Learning angewandt werden, während dies bei anderen nicht der Fall ist.

In Kapitel 4 wurde das Empfehlungssystem als ein Softwaresystem definiert, das dem Benutzer nützliche Objekte auf Basis ihres Bewertungsverhaltens empfiehlt. Dabei wurden Objekte als E-Commerce-Produkte gesehen. Im E-Learning-Kontext handelt es sich bei den Objekten dagegen um kontextbezogene Elemente wie Lernmaterialien oder Lernabschnitte [2].

Ebenso wurde bislang nicht zwischen verschiedenen Benutzern eines Empfehlungssystems unterschieden. Im Bereich der Online-Vorlesung werden dagegen zwei Benutzergruppen unterschieden: die Professoren und die Studierenden. Die Professoren sind für die Planung und die Bereitstellung der Lernressourcen zuständig. Sie können das Empfehlungssystem nutzen, um sich einen Überblick über das Studierendenverhalten oder deren Leistung zu verschaffen. Dieser Aspekt soll im Rahmen der vorliegenden Arbeit jedoch nicht weiter berücksichtigt werden. Die zweite Benutzergruppe sind die Studierenden, die als die eigentlichen Endnutzer des Empfehlungssystems gelten können. Aufgrund ihrer aktiven Rolle bilden sie gleichzeitig einen zentralen Bestandteil des Systems [2].

In [2] werden spezifische Ziele für Empfehlungssysteme im E-Learning-Kontext definiert:

- Find Novel Items: Studierenden neue oder passende Lernmaterialien/-inhalte empfehlen.
- Find Peers: Lernpartner empfehlen, die übereinstimmende Interessen oder Lerneigenschaften aufweisen.
- Find Good Pathways: Alternative Lernpfade empfehlen.

Bislang wurde das explizite Feedback als Bewertung von Benutzern in Form einer Skala definiert. In Bereichen wie E-Commerce und Entertainment, in denen der Unterhaltungsfaktor im Vordergrund steht, reicht diese Form von Information aus, da ein Benutzer ein Produkt entweder gut oder schlecht finden kann. Im E-Learning-Bereich ist dies jedoch aufgrund der Komplexität der Lernressourcen nicht ausreichend [27, 28]. Ein Studierender kann einen Lernabschnitt schlecht bewerten, weil der Inhalt ihm nicht detailliert genug fällt. Ebenso kann er jedoch eine schlechtere Bewertung abgeben, weil er den Inhalt als zu schwer oder zu leicht empfindet. Diese auf verschiedenen Interpretationen basierenden Bewertungen können vom Empfehlungssystem nicht berücksichtigt werden. In diesem Zusammenhang hat sich in vielen Bereichen der Begriff des impliziten Feedbacks

etabliert [29]. Im E-Learning-Bereich wird vor allem der Umgang von Studierenden mit Lernressourcen analysiert, um davon dann die Bewertung abzuleiten. Gute Kandidaten hierfür wären die Dauer oder die Nutzungshäufigkeit eines Lernabschnitts [30].

5.2 Die Online-Vorlesung

Unter dem Begriff Vorlesung wird eine Lehrveranstaltung verstanden, deren Zweck darin besteht, Studierende mittels eines Vortrags zu unterrichten [31]. In traditionellen Vorlesungen steht der Dozent im vorderen Teil des Hörsaals, von wo er seinen Vortrag hält und Folien zeigt, während die Studierenden zuhören und sich Notizen machen [32]. Das Ziel der Vorlesung ist es, Wissen an die Studierenden zu vermitteln. Bei der Vorlesung nutzen Dozenten unterschiedliche Lehrmethoden, die alle das Ziel haben, dass Studierende das dargestellte Thema verstehen. Zur Gewährleistung der gewünschten Verständlichkeit beginnen Pädagogen meist mit einer Einleitung, um dann im weiteren Verlauf die Kernideen zu erklären. Dabei werden Hilfsmittel wie Texte, Bilder oder Diagramme verwendet. Eine gute Strukturierung der Vorlesung führt zu einem besseren Verständnis. Um dieses Ziel zu erreichen, nutzen Pädagogen unterschiedliche Wege. Trotzdem lassen sich in vielen Vorlesungen gewisse Gemeinsamkeiten ausmachen. So enthalten sie zu Beginn meist eine Einleitung, in der das Thema der vorangehenden Vorlesung wiederholt wird. Danach folgt der eigentliche Inhalt mit Beispielen, praktischen Übungen, Diskussion etc. Am Ende eines Vorlesungsabschnitts werden häufig Fragen gestellt.

Während der Vorlesung kann der Informationsfluss durch ein Direktgespräch erfolgen. Ebenso kann der Dozent auf diverse Hilfsmittel zurückgreifen. Beim Direktgespräch steht der Professor im Vollbild und die Informationsquelle ist die Audiospur. Je nach Inhalt der Vorlesung können bestimmte Hilfsmittel wie PowerPoint-Folien und Whiteboards verwendet werden. Ebenso kommen auch häufig Hilfsmittel wie externe Programme zum Einsatz.

Die Online-Vorlesung ist eine multimediabasierte Form der Vorlesung. Bei der Erstellung einer Online-Vorlesung wird meist mit der Aufzeichnung einer Vorlesung begonnen, um dann auf dieser Grundlage einzelne Lerneinheiten zu erzeugen. Lerneinheiten dienen der thematischen Unterteilung der Vorlesungsaufzeichnung, auf der die Online-Vorlesung basiert. Zur Aufzeichnung wird häufig ein Screen-Recording verwendet. Beim Screen-Recording wird der Bildschirminhalt des Präsentationsrechners aufgezeichnet. Ein Nachteil dieses Verfahrens besteht darin, dass die Skalierung der Pixelrepräsentation unter Qualitätsverlust erfolgt [33]. Für die Aufzeichnung einer Vorlesung bietet sich

unter Umständen auch das Format einer Videokonferenz an, wobei Kameras und Mikrophone zum Einsatz kommen.

6 Konzeption

Nachdem im bisherigen Verlauf der Arbeit die zentralen Probleme beschrieben und die theoretischen Grundlagen erarbeitet wurden, soll in diesem Kapitel nun eine Lösung konzipiert werden. Die vorgeschlagene Lösung wird anhand der Systemarchitektur und des Systemablaufs dargestellt.

6.1 Systemarchitektur

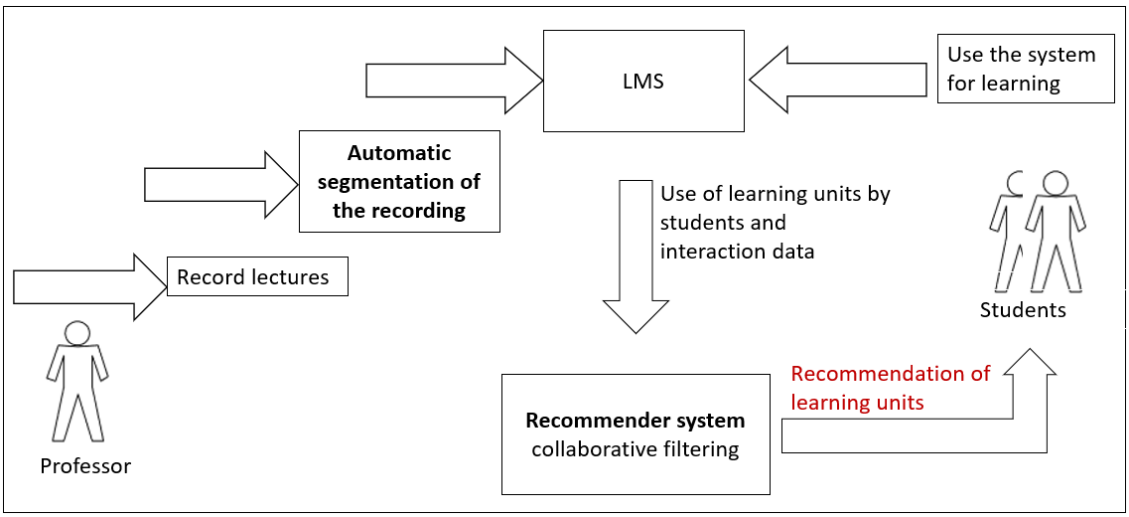


Abbildung 6.1: Systemarchitektur

Abbildung 6.1 stellt die Systemarchitektur dar. Zu Beginn des Prozesses zeichnet der Dozent die Vorlesungen auf. Dann wird diese Aufzeichnung mit dem System zerlegt, um passende Lerneinheiten zu erstellen. Schließlich werden diese Lerneinheiten im LMS zur Verfügung gestellt. Im Rahmen ihrer Lernprozesse greifen Studierende darauf zu, um sich das dargestellte Wissen anzueignen. Die Nutzungsdaten werden vom LMS-System protokolliert, um in der Folge in das Empfehlungssystem einzufließen, das einen Teil des LMS bildet. Nach der Anwendung von kollaborativen Algorithmen auf diese Daten

werden den Studierenden geeignete Lerneinheiten empfohlen. Eine Herausforderung bei diesem Prozess besteht in der automatisierten Zerlegung der Aufzeichnungen und der Realisierung des Empfehlungssystems. Eine detaillierte Beschreibung der beiden Komponenten erfolgt im nachfolgenden Abschnitt zum Systemablauf.

6.2 Systemablauf

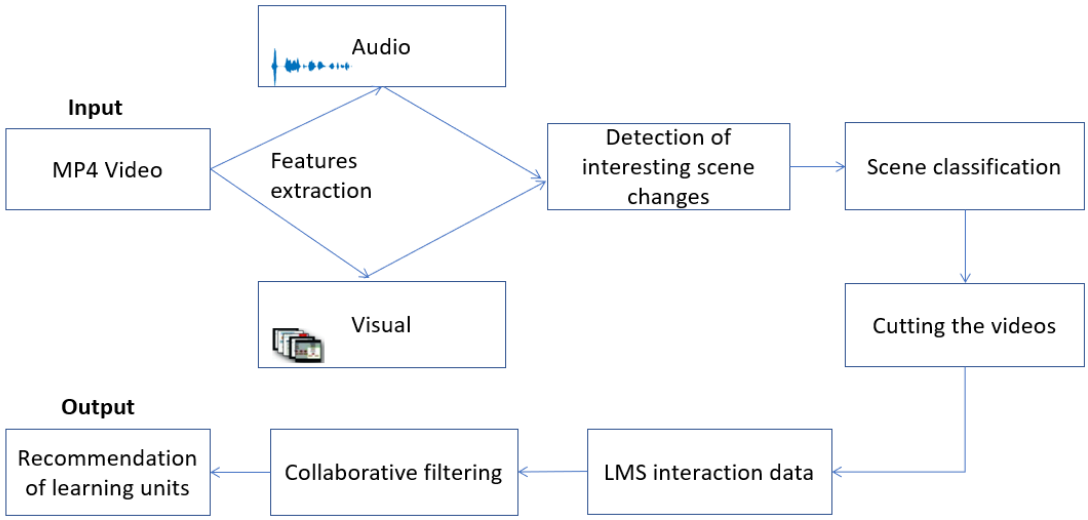


Abbildung 6.2: Systemablauf

Abbildung 6.2 stellt den Ablauf des zu entwickelnden Systems dar. Die dazugehörigen Komponenten können auf zwei Prozesse aufgeteilt werden. Die Komponenten *MP4 Video* bis *Cutting the videos* gehören zum Prozess der automatisierten Zerlegung der Aufzeichnungen. Dagegen gehören die Komponenten *Collaborative Filtering* und *Recommendation of learning units* zum Prozess des Empfehlungssystems für Lerneinheiten. Was die Komponente *LMS interaction data* betrifft, so handelt es sich hierbei um die Schnittstelle zwischen beiden Prozessen. Im Folgenden wird auf die einzelnen Komponenten eingegangen.

6.3 Input MP4 and Visual Features Extraction

Als Input des Systems fungiert eine Videodatei im MP4-Format, die die Aufzeichnung der Vorlesung enthält. Dieses Format wird häufig zum Speichern von Video-, Text- und Audi-

odaten verwendet. Nachdem diese Datei ausgelesen wurde, beginnt die visuelle Feature-Extraktion. Hierfür wird die in Kapitel 3 beschriebene Schlüsselbildextraktionstechnik verwendet. Das Video wird Bild für Bild ausgelesen, um dann die RGB-Farbhistogramme der einzelnen Bilder zu berechnen. Diese geben die Anzahl der Pixels im Bild für einen bestimmten Helligkeitswert von 0 bis 256 an [34]. In der Literatur wurde nachgewiesen, dass eine signifikante Unähnlichkeit der Histogramme zwischen den Einzelbildern eines Videos auf einen schnellen Szenenwechsel hinweist. Auf Basis der Farbhistogramme wird der euklidische Abstand zwischen zwei aufeinanderfolgenden Bildern berechnet. Anhand dieses Abstands wird entschieden, ob die Bilder signifikante Unähnlichkeiten aufweisen. Wenn der euklidische Abstand eines Bildes den Durchschnitt aller Distanzen übersteigt, wird das Bild im Rahmen der vorliegenden Arbeit als Schlüsselbild betrachtet. Für die Optimierung der Szeneänderungserkennung werden Audio-Features benötigt.

6.4 Audio Features Extraction

In Kapitel 3 wurde beschrieben, dass die Salienz einen entscheidenden Faktor für die audiovisuelle Szeneanalyse darstellt. Dabei wurde zwischen akustischer und visueller Salienz unterschieden. In dieser Arbeit wird die akustische Salienz angewendet, um Pausen in den Videos zu erkennen. Dazu wird die Audiospur vom Eingangsvideo getrennt und separat analysiert. Mithilfe von Fouriertransformationsalgorithmen werden die Audiodaten in Amplituden umgewandelt. Schließlich werden die Amplituden analysiert, um Stillstände festzustellen. In dieser Arbeit werden Amplituden mit Werten zwischen -6 und 6 als Stillstände betrachtet. Diese Werte wurden ermittelt. Um berücksichtigt zu werden, sollte der Stillstand mindestens eine Sekunde anhalten. Als Pause werden Stillstände von mindestens drei Sekunden betrachtet.

6.5 Detection of Interesting Scene Changes

In den vorhergehenden Abschnitten wurde beschrieben, wie audiovisuelle Features extrahiert werden. Dabei wurde bei visuellen Features der Wechsel von Schlüsselbildern als Anfang einer neuen Szene betrachtet. Der Ansatz der Audio-Feature-Extraktion diente dazu, Pausen im Video zu erkennen. In diesem Schritt werden diese Features kombiniert, um interessante Szeneänderungen zu erkennen. Dazu werden die zeitlichen Feature-Vektoren untersucht. Feature-Positionen mit gleichzeitigen Änderungen in Audio- und visuellen Features werden als interessant betrachtet. Diese Positionen erhalten eine höhe-

re Priorität. Es wurde mehrfach beobachtet, dass der Professor vor einem Themawechsel eine Frage stellt, daraufhin auf die Antwort wartet und danach die PowerPoint-Folie wechselt oder zuerst die PowerPoint-Folie wechselt, und dann die Frage stellt und auf die Antwort wartet. Auch solche Ereignisse im Video werden als interessante Änderungen betrachtet. Außerdem wird eine konfigurierbare minimale Dauer von zwei Minuten für interessante Szenen definiert. Interessante Szenen dürfen keine Fehlerszenen sein, die durch Aufnahmefehler entstehen. In Anbetracht dessen können interessante Szenen aus einer Sequenz von Szenen bestehen.

6.6 Scene Classification and Cutting the Videos

Nach der Erkennung von interessanten Szenen werden diese klassifiziert. Hierfür werden fünf Klassen definiert:

- Slide: Szene mit PowerPoint-Folie im Vollbild
- Whiteboard: Szene mit Skizze, Handschiff
- Professor: Szenen mit dem Professor im Vollbild
- Fault: Aufnahmefehler (schwarzes Bild)
- Other: Externe Programme, Text und Sonstiges

Abbildung 6.3 veranschaulicht die verschiedenen Szenenklassen.

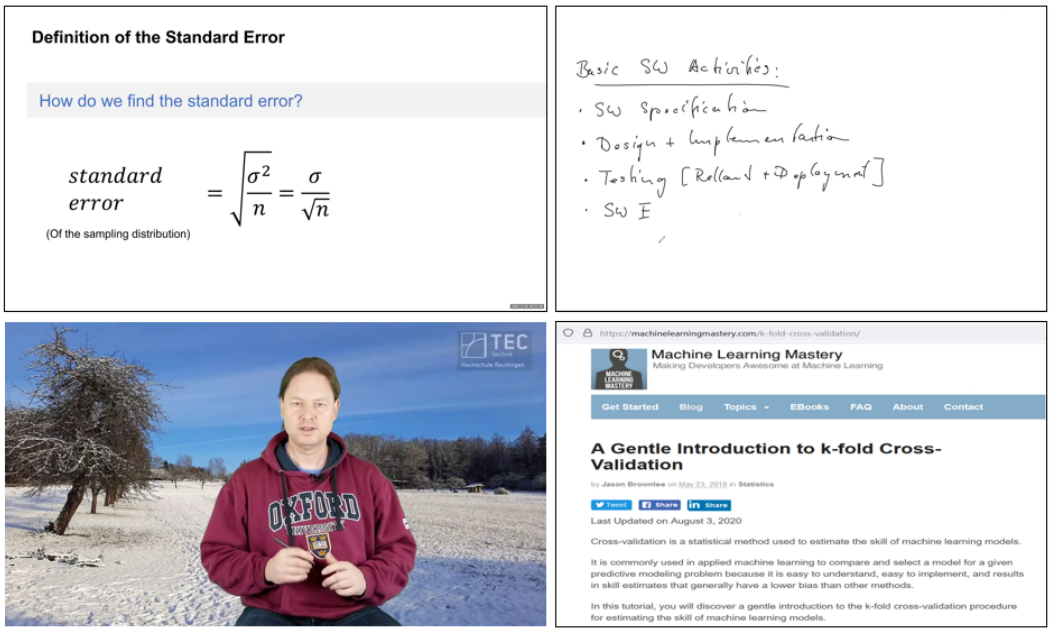


Abbildung 6.3: Unterschiedliche Arten von Szenen einer Vorlesung: Slide, Whiteboard, Professor, Other

Als Eingang für den Klassifikator werden pro Szene drei Bilder übergeben: das Start-, das End- und das Mittelbild. Anhand eines trainierten Modells weist der Klassifikator den Bildern eine Klasse zu. Um Fehler bei der Klassenerkennung zu vermeiden, wird die am häufigsten zugewiesene Klasse der drei Bilder als Szeneklasse verwendet. Das zu verwendende vortrainierte Modell besteht aus verschiedenen Vorlesungsbildern der genannten Klassen. Dieses Modell, das der CNN- Klassifikator verwendet, stammt aus einer Vorarbeit zum vorliegenden Thema. Bei CNN handelt es sich um ein künstliches neuronales Netzwerk, das für die Bilderkennung und -verarbeitung geeignet ist [35].

Das Ergebnis der Klassifizierung wird verwendet, um die Liste der interessanten Szenen zu verbessern, indem Fault-Szenen (Fault) gefiltert werden. Des Weiteren werden die Szeneklasseninformationen bei der Generierung der LMS-Daten verwendet, um die Vielfalt der generierte Daten zu gewährleisten. Am Ende der Szeneklassifizierung wird eine CSV-Datei erstellt, in der interessante Szenen, ihre Start- und Endpositionen, ihre Dauer und ihre Szeneklasse dokumentiert werden. Diese Datei wird im weiteren Verlauf Vorlesungsskript genannt. Des Weiteren wird anhand der Start- und der Endpositionen jeder interessanten Szene, das Eingangsvideo mit dem Tool FFmpeg geschnitten. Auf Basis der Ergebnisse des Vergleichs PySceneDetect und Google Cloud API in [9], ist FFmpeg für die vorliegende Thesis als am geeignetsten für das Schneiden von Videos ge-

sehen. Die Auswahlkriterien waren die Zuverlässigkeit und Opensource Lizenz, die den freien Einsatz der Software ermöglicht.

6.7 LMS Interaction Data

Wie erwähnt, bilden die Interaktionsdaten des LMS die Schnittstelle zwischen der automatisierten Zerlegung der Aufzeichnungen und dem Empfehlungssystem für Lerneinheiten. Die aus der Zerlegung resultierenden Lernabschnitte werden verwendet, um LMS-Interaktionsdaten zu generieren, die für die Implementierung und die Evaluierung des Empfehlungssystems benötigt werden. In Kapitel 5 wurden Studierende als Endbenutzer des Empfehlungssystems für Lerneinheiten beschrieben. Zunächst werden die Studentenprofile definiert.

- learningDuration in %: Hierbei handelt es sich um die Nutzungsdauer der Lerneinheit durch den Studierenden. Es wird angenommen, dass die Studierenden mit interessanten Lernabschnitten mehr Zeit verbringen. Diese Dauer wird in Prozent angegeben. Wenn ein Studierender einen Abschnitt vollständig angeschaut hat, dann beträgt die Dauer der Nutzung 100 %.
- numberOfPosts: Dieser Wert gibt die Anzahl der Nachrichten wieder, die der Studierende während der Lerneinheit im Forum des LMS abgeschickt hat. Dabei wird angenommen, dass eine Lerneinheit bei mehr Nachrichten interessanter ist.
- userId: Hierbei handelt es sich um eine eindeutige Nummer, die den Studierenden identifiziert.
- sceneId: Hierbei handelt es sich um eine eindeutige Nummer, die die Lerneinheit identifiziert.

Anhand des Profils wird das Vorlesungsskript herangezogen, um die Benutzer und ihre Profildaten zu generieren. Hierbei werden vier Benutzergruppen generiert: faule, mittelfaule, mittel-gute und gute Studierende. Der Unterschied zwischen diesen Gruppen wird durch den Grad der Interaktion mit den Lerneinheiten bestimmt. So interagieren gute Studierende z. B. mehr mit Lerneinheiten als faule Studierende. Außerdem variieren diese Profildaten je nach Art (Klasse) der Lerneinheit. Am meisten werden Whiteboards in Vorlesungen für Übungen und Demonstrationen verwendet. Faule Studierende interessieren sich nicht für Übungen.

6.8 Collaborative Filtering and Recommendation of Learning Units

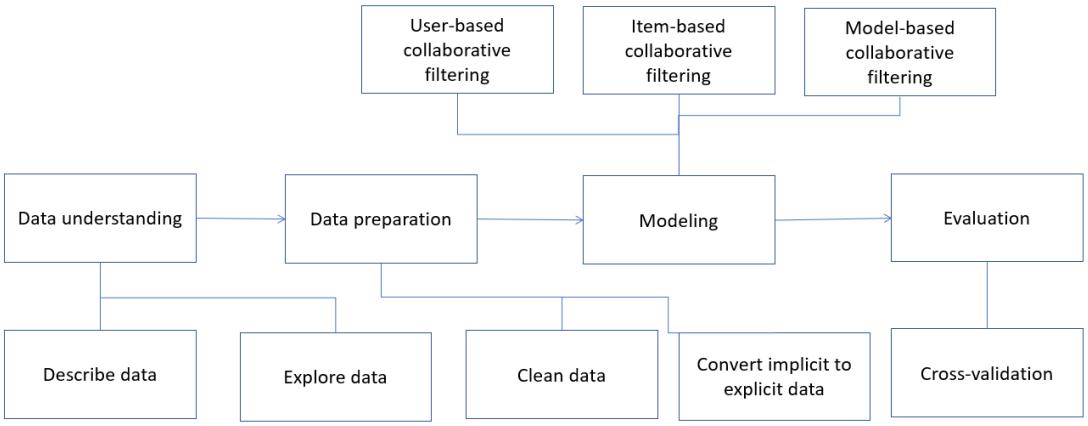


Abbildung 6.4: Lösungsprozess des Empfehlungssystems basiert auf CRISP-DM

Die Implementierung des Empfehlungssystems orientiert sich an den klassischen Lösungsschritten für ein Problem aus dem Bereich des maschinellen Lernens. Abbildung 6.4 veranschaulicht den Lösungsprozess, der vom CRISP-Modell abgeleitet ist. Im Folgenden werden die einzelnen Komponenten des Prozesses beschrieben.

6.8.1 Data Understanding and Preparation

In diesem Schritt werden die Interaktionsdaten des LMS und des Vorlesungsskripts analysiert. Danach werden die Interaktionsdaten bereinigt, indem mögliche redundante Daten entfernt werden. Des Weiteren werden die impliziten Bewertungsdaten in explizite umgewandelt, um so klassische kollaborative Algorithmen auf den Datensatz anwenden zu können. Diese Algorithmen erwarten Daten in Form eines Tupels (*user*, *object*, *rating*). Die Umwandlung erfolgt durch die Normierung und Gewichtung der Features *learningDuration in %* und *numberOfPosts*. Nachdem beide Features in explizite Bewertungen umgewandelt worden sind, wird bei der Modellierung das Feature *learningDuration in %* als Rating verwendet. Während der Evaluation wird es mit dem Feature *numberOfPosts* kombiniert, um den Einfluss von neuen Features auf das Empfehlungssystem zu bewerten. Eine Möglichkeit zu einer solchen Kombination bildet die Addition.

6.8.2 Modeling

Bei der Modellierung werden kollaborative Algorithmen auf den vorbereiteten Datensatz angewendet. In Kapitel 4 wurden das memorybasierte und das modellbasierte kollaborative Filtern beschrieben. Bei dem modellbasierten Verfahren wurde die Matrix-Faktorisierung mit SVD als das populärste Modell vorgestellt. Beim memorybasierten Verfahren wurde zwischen einem benutzerbasierten und einem objektbasierten Ansatz unterschieden. Im Rahmen dieser Arbeit wird mit den genannten Verfahren experimentiert, um schließlich das für das geplante System geeignetste Verfahren auszuwählen. Im Folgenden wird auf konkrete Beispiele für memorybasierte Ansätze eingegangen.

6.8.2.1 Benutzerbasiertes kollaboratives Filtern

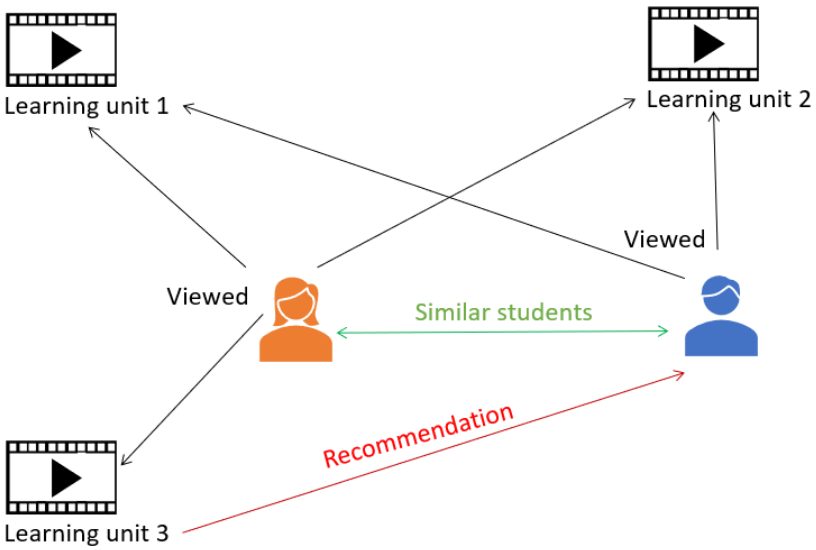


Abbildung 6.5: Beispiel benutzerbasiertes kollaboratives Filtern

Abbildung 6.5 zeigt ein Beispiel für den benutzerbasierten Ansatz. Wie in dieser Abbildung dargestellt, hat die Studentin die Lerneinheiten 1, 2 und 3 angeschaut, während der Student die Lerneinheiten 1 und 2 betrachtet hat. Da der Student die Lerneinheit 3 noch nicht angeschaut hat, wird das System ihm dies empfehlen, sofern er ähnliche Interessen wie die Studentin besitzt.

6.8.2.2 Objektbasiertes kollaboratives Filtern

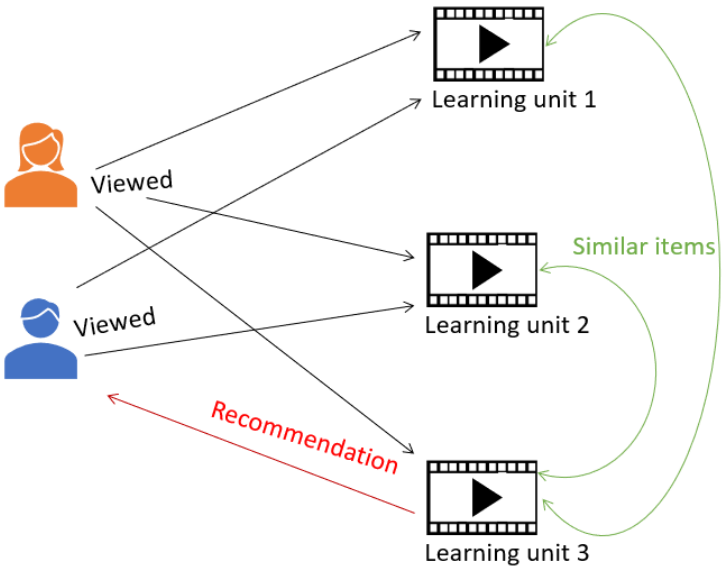


Abbildung 6.6: Beispiel objektbasiertes kollaboratives Filtern

Abbildung 6.6 zeigt ein Beispiel für die objektbasierte Methode. In diesem Beispiel hat die Studentin alle Lerneinheiten angeschaut, während der Student bislang nur die Lerneinheiten 1 und 2 betrachtet hat. Da dieser Student die Lerneinheit 3 noch nicht angeschaut hat, wird ihm dies vom System empfohlen, wenn mindestens eine der von ihm angeschauten Lerneinheiten (1, 2) Ähnlichkeiten zu Lerneinheit 3 aufweist.

6.8.3 Evaluation

In Kapitel 4 wurde die Offline-Evaluierung als geeignetes Verfahren für die Forschung und das Benchmarking beschrieben. Da es in dieser Arbeit um die Durchführung von Experimenten geht, wird diese Methode verwendet. Dabei werden die entwickelten kollaborativen Algorithmen auf die generierten Datensätze angewendet, um auf diese Weise ihre Genauigkeit zu überprüfen. Dazu wird das Framework k-fold Kreuzvalidierung (k-fold Cross Validation) verwendet. Hierbei handelt es sich um eine statistische Methode zur Einschätzung der Fähigkeiten von Modellen des maschinellen Lernens, die auch bei Empfehlungssystemen angewandt wird. Dieses Framework teilt den Datensatz in k gleich große Partitionen. Jede einzelne Partition wird als eigenständiger Datensatz für die Evaluation verwendet. Die Modelle des Verfahrens werden auf die restlichen k−1 Partitionen

trainiert, um daraufhin mit der Testpartition die Vorhersage für die Benutzer zu evaluieren [36]. In dieser Arbeit wird das beschriebene Verfahren mit dem Parameter $k = 5$ evaluiert.

7 Implementierung

In diesem Kapitel werden die Implementierungsdetails zum beschriebenen Konzept in Kapitel 6 veranschaulicht. Für die Implementierung werden Vorlesungsaufzeichnungen vom Forschungsinstitut Herman-Hollerith-Zentrum zur Verfügung gestellt, die sich aus verschiedenen Studienfächern der Bachelor- und Masterprogramme zusammensetzten. Diese Online-Vorlesungen wurden von verschiedenen Dozenten gehalten und mit dem Videokommunikationsprogramm Zoom aufgezeichnet. Für alle Programmierungsaktivitäten wird Python verwendet. Python ist ein High-Level und Allzweckprogrammiersprache, die aufgrund ihrer hochentwickelten Datenstrukturen im Bereich des maschinellen Lernens weit verbreitet ist [37]. Ausgehend von den verwendeten Technologien werden im Folgenden alle wesentlichen Aktivitäten beschrieben.

7.1 Technologien

Aufgrund der explorativen Datenanalyse findet die Entwicklung in zwei unterschiedlichen Umgebungen statt. Die Entwicklung des Empfehlungssystems für Lerneinheiten erfolgt in Jupiter Notebook. Jupiter Notebook ist eine Open-Source-Webanwendung, die die Erstellung von Dokumenten ermöglicht, die Python-Code, Visualisierungen und Kommentare enthalten [38]. Die Entwicklungsaktivitäten für die automatisierte Zerlegung der Aufzeichnungen finden in Pycharm statt. Pycharm ist eine integrierte Entwicklungsumgebung, die die Entwicklung von Data-Science-Anwendungen mit Anaconda unterstützt [39]. Verschiedene Frameworks werden für die Implementierung des Systems eingesetzt. Abbildung 7.1 veranschaulicht diese sowie auch die Teilprozesse, in denen sie eingesetzt werden.

Framework	Beschreibung	Teilprozess
OpenCV	Open Computer Vision (OpenCV) ist eine Open-Source-Software-Bibliothek für die Bild- und Videoanalyse, die in C/C++ implementiert ist [40].	Visual Features Extraction
Surprise	Surprise ist eine Open-Source-Python-Bibliothek für die Entwicklung von Empfehlungssystemen mit expliziten Bewertungsdaten. Das Framework bietet vorgefertigte Vorhersage-Algorithmen wie KNN, SVD und NMF [41].	Collaborative Filtering - Modeling and Evaluation
Pandas	Pandas ist eine Python-Bibliothek, die High-Level-Datenstrukturen für Daten-bezogene Aufgaben zur Verfügung stellt [42].	Collaborative Filtering - Data Understanding and Preparation
NumPy	NumPy ist eine Python-Bibliothek für wissenschaftliche Berechnungen. Die Bibliothek bietet Implementierungen von mehrdimensionalen Arrays und Matrizen [43].	Collaborative Filtering - Data Understanding and Preparation Audiovisual Features Extraction
SciPy	SciPy ist eine Sammlung mathematischer Algorithmen und in Wissenschaft und Technik gängiger numerischer Routinen [44].	Visual Features Extraction
Matplotlib	Matplotlib ist eine Python-Bibliothek zur Erstellung statischer, animierter und interaktiver Visualisierungen [45].	Audiovisual Features Extraction
Moviepy	Moviepy ist ein Python-Modul für die Videobearbeitung, das auch grundlegende Operationen für die Audio-Extraktion anbietet [46].	Audio Features Extraction
CSV	CSV ist ein Python-Modul zum Lesen und Erzeugen von tabellarischen Daten im CSV-Format [47].	LMS Interaction Daten, Cutting the videos
Keras	Keras ist eine Open-Source-Bibliothek, die eine Python-Schnittstelle für künstliche neuronale Netze bietet [48].	Scene Classification
FFmpeg	FFmpeg ist ein Modul, das aus Bibliotheken und Programmen zur Verarbeitung von Video- und Audiodaten besteht [49].	Cutting the Videos

Tabelle 7.1: Frameworks

7.2 Klassen und Notebooks

In Abbildung 7.1 werden die verwendeten Klassen und Notebooks sowie deren Funktionen aufgeführt.

Klasse/Notebook	Beschreibung
main.py	Hierbei handelt es sich um das Startprogramm, das die nötigen Klassen aufruft, um die Aufzeichnungen zu zerlegen.
sceneDetector.py	Diese Klasse ist für die Szenendetektion verantwortlich. Sie verwendet die Hilfsklassen audioFeaturesUtils.py und visualFeaturesUtils.py.
audioFeaturesUtils.py	Diese Klasse stellt Methoden für die Audio-Feature-Extraktion und -analyse zur Verfügung.
visualFeaturesUtils.py	Diese Klasse stellt Methoden für die visuelle Feature-Extraktion und -Analyse zur Verfügung.
sceneClassificator.py	Diese Klasse wird verwendet, um erkannte Szenen zu klassifizieren.
cutVideo.py	Diese Klasse ist für das Schneiden der Videos zuständig.
generateLMSinteractiondata.py	Diese Klasse wird für die Generierung der Interaktionsdaten des LMS verwendet.
utils.py	Diese Klasse bietet allgemeine Hilfsfunktionen für das Programm.
recommenderSystem_data_Preparation.ipynb	Dieses Notebook wird für die Vorbereitung des Datensatzes verwendet.
recommenderSystem_item-based-collaborative-filtering.ipynb	Dieses Notebook wird für die Umsetzung des objektbasierten kollaborativen Filterns verwendet.
recommenderSystem_user-based-collaborative-filtering.ipynb	Dieses Notebook wird für die Umsetzung des benutzerbasierten kollaborativen Filterns angewandt.
recommenderSystem_model-based-collaborative-filtering.ipynb	Dieses Notebook bietet eine Implementierung für das modellbasierte kollaborative Filtern.

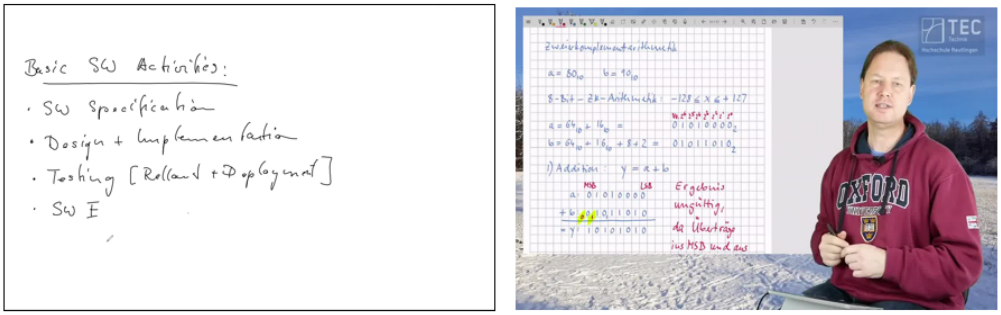
Abbildung 7.1: Struktur der Implementierung

7.3 Visuelle Feature-Extraktion

Nachdem alle erforderlichen Bibliotheken in PyCharm importiert worden sind, werden im ersten Schritt des Prozesses der *automatisierten Zerlegung der Vorlesungsaufzeichnungen* die visuellen Features extrahiert. Dazu wird das Video-Verzeichnis aus den Programmargumenten extrahiert. In diesem Verzeichnis werden MP4-Dateien durchsucht und einzeln prozessiert. Die Datei wird mit der Methode *videoCapture* der OpenCV-Bibliothek ausgelesen und in einem Objekt *videoCap* gespeichert. Von diesem Objekt werden die Video-Eigenschaften wie die Bildfrequenz (*fps*) und die Gesamtzahl der Bilder des Videos (*totalFrames*) ausgelesen. Diese Eigenschaften werden während der gesamten Videoanalyse gebraucht.

Als Nächstes findet die visuelle Feature-Extraktion in einer While-Schleife statt. Das Videoobjekt wird Bild für Bild mit der Methode read der OpenCV-Bibliothek ausgelesen. Um die Dauer der Analyse zu reduzieren, wird nach dem Lesen das Bild zu 50 % verkleinert. Im weiteren Verlauf wird zwischen zwei Arten von Vorlesungsvideos unterschieden. Die erste Art von Video enthält Whiteboard-Szenen und Aufnahmen des Professors, wobei beide im Vollbild zu sehen sind.

Für diese Art von Videos wird eine Vorprozessierung durchgeführt, wobei die Bildregion mit dem Region-of-Interest (ROI) extrahiert wird. Der restliche Teil des Bildes wird ignoriert. Das Ziel dieser Vorprozessierung ist es, störende Informationen aus dem Bild zu entfernen. Zur Extraktion des ROI wird die Methode *getRegionOfInterest* verwendet, die die Methode findContours der OpenCV-Bibliothek verwendet. Des Weiteren werden von der extrahierten Region die Hintergrundinformationen subtrahiert. Dafür wird die Methode *preprocess* verwendet, die die Algorithmen GaussianBlur, Canny, dilate und erode der OpenCV-Bibliothek auf das Bild anwendet. Die zweite Art von Vorlesungsvideo ist ein Video mit Whiteboard-Szenen, in denen kein Professor im Vollbild zu sehen ist. Für diese Art von Video wird keine Vorprozessierung durchgeführt. Abbildung 7.2 veranschaulicht die beiden Arten von Videos.



Whiteboard ohne Professor im Bild | Whiteboard mit Professor im Vollbild

Abbildung 7.2: Unterschiedliche Arten von Videos

Im nächsten Schritt wird das Farbhistogramm des Bildes berechnet. Dafür wird die Methode *calcHist* der OpenCV-Bibliothek genutzt. Anhand des Farbhistogramms wird der euklidische Abstand zwischen dem aktuellen Bild und den vorherigen Bildern berechnet. Dazu werden die Farbmomente dieser Bilder berechnet. Dazu gehören der Farbdurchschnitt und die Standardabweichung. Die euklidische Distanz ergibt sich aus der Summe der Differenzen der Farbmomente des aktuellen Bildes und des vorherigen Bildes. Hierzu wird die Methode *getEuclideanDistance(currentColorMoments, previousColorMoments)* verwendet.

Neben dem euklidischen Abstand werden auch die Entropie und die dominante Farbe des aktuellen Bildes erfasst. Die Entropie repräsentiert die durchschnittliche Information des Bildes. Hierzu kommt die logarithmische Funktion der Math-Bibliothek zum Einsatz. Auf Basis der Entropie wird die Entropie-Differenz zwischen dem aktuellen und dem vorherigen Bild berechnet. Die oben erwähnte While-Schleife wird verlassen, sobald alle Bilder prozessiert worden sind. Die Abbildung veranschaulicht das Ergebnis dieses Vorgangs für ein kurzes Video von ca. zwei Minuten.

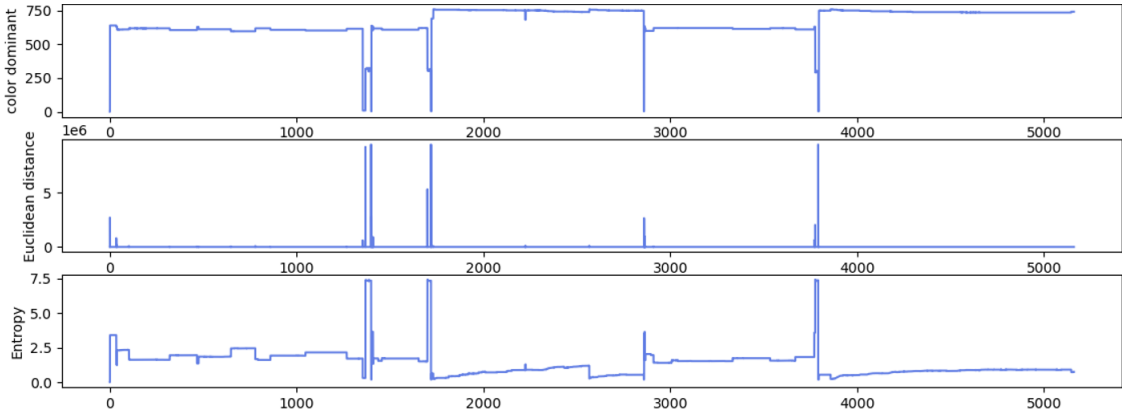


Abbildung 7.3: Ergebnis der Extraktion der visuellen Features

7.4 Audio-Feature-Extraktion

Als erster Schritt der Audio-Feature-Extraktion wird die Audiospur vom Eingangsvideo getrennt. Dazu wird das Video mit der Methode *VideoFileClip* der Moviepy-Bibliothek ausgelesen, um daraufhin seine Audio-Spur mit der Methode *write_audiofile* in einer WAV-Datei zu speichern. Im nächsten Schritt wird diese WAV-Datei mit der Methode *wavfile.read* der Scipy-Bibliothek ausgelesen und als numpy-Array von Audio-Frames in einer neuen Variable gespeichert. Die Audio-Frames repräsentieren die Amplituden des Audio-Signals. Abbildung 7.4 veranschaulicht die gelesenen Amplituden für das in Abschnitt 7.3 zuvor erwähnte Video.

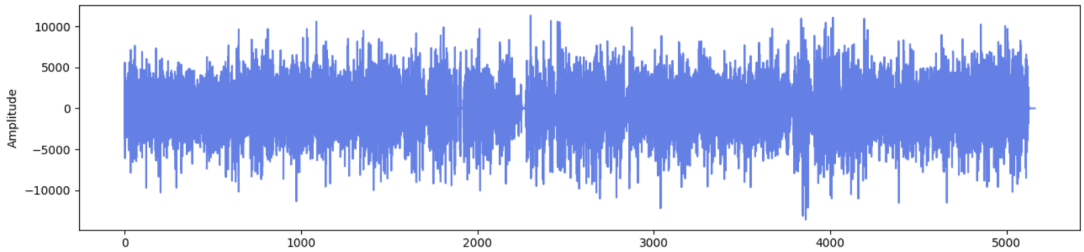


Abbildung 7.4: Amplituden des Audio-Signals

Im nächsten Schritt werden anhand dieser Amplitude-Daten Ruhezeiten im Video ermittelt. Vorher wird versucht Audio- und Videoframes zu synchronisieren, indem bei Audio-Frames nur jeder k Frame mit k=*audio_rate/video_rate* betrachtet wird. Schließlich werden die Ruhezeiten in der Audioframeliste durchsucht. Wie in der Konzeption

erwähnt, werden Ruhezeiten als Bereiche mit einer Amplitude zwischen −6 und +6 betrachtet. Das Ergebnis dieses Teilprozesses ist eine zeitliche Liste mit 0 und 1, wobei 0 an Positionen ohne Ruhezeiten und 1 an Positionen mit Ruhezeiten steht.

7.5 Erkennung von interessanten Szeneänderungen

7.5.1 Erkennung von Szeneänderungen

Nachdem die audiovisuellen Features extrahiert worden sind, werden auf deren Basis Szeneänderungen ermittelt. Als Szene wird hierbei eine sequenzielle Abfolge von gleichmäßigen Bildern bezeichnet. Bevor die Szenen erfasst werden, wird die durchschnittliche euklidische Distanz (*meanEuclideanDistance*) zwischen allen Videobildern berechnet. Der berechnete Wert wird als Schwellenwert für die Erkennung der Bildänderung verwendet. Dafür wird die Liste der euklidischen Distanzen sekundenweise durchgegangen. Innerhalb jeder Sekunde werden die Abstände mit dem genannten Schwellenwert verglichen. Sollte der Abstand den Schwellenwert übersteigen, wird die Distanz des entsprechenden Bildes zu dem nächsten Bild verglichen. Sollte diese den definierten Schwellenwert von 1 übersteigen, wird das Bild als Schlüsselbild betrachtet, wobei die entsprechende Position im Eingangsvideo gespeichert wird. Abbildung 7.5 stellt ein Beispiel für Positionen von Schlüsselbildern im Eingangsvideo dar. Die Features Color Dominant und Entropy werden hier nicht weiter betrachtet, da sie dasselbe Ergebnis wie die euklidischen Distanzen liefern.

```
[938.0, 1177.0, 2803.0, 2852.0, 3023.0]
```

Abbildung 7.5: Indexes von Schlüsselbildern

7.5.2 Interessante Szeneänderungen

Bislang wurden auf Basis audiovisueller Aspekte Szeneänderungen und Pausen im Video ermittelt. Nun werden diese Features kombiniert, um auf diese Weise interessante Szeneänderungen zu erfassen. Dazu werden die beiden zeitlichen Feature-Vektoren durchgegangen (Abbildung 7.6). Wie im Konzept erklärt, werden Positionen, an denen Ereignisse in beiden Features gleichzeitig oder nahezu gleichzeitig passieren, in einer neuen Liste gespeichert. In diesem Vektor erhalten Positionen mit Ereignissen den Wert 1. Ebenso

wird die Dauer der einzelnen Szenen überprüft. Sollte eine Szene kürzer als zwei Minuten sein, wird sie nicht als einzelne Szene betrachtet.

```
TimeLine:              0  1  2  3  4  5  6  7  8  9  10 ...  5000
BreakTimeLine:         0  1  0  0  0  0  0  0  0  1  0  ... 0
SceneChangeTimeLine:   0  1  0  0  0  0  0  0  0  0  1  ... 0
```

Abbildung 7.6: Feature-Vektoren

7.6 Szenenklassifizierung

In diesem Schritt werden die im vorigen Abschnitt ermittelten Szenen klassifiziert. Dabei werden aus jeder Szene die drei genannten Bilder extrahiert, um diese dann dem CNN-Klassifikator zu übergeben. Mit der Keras-Bibliothek werden die Bilder geladen und dann zum Bild-Array konvertiert. Auf Basis dieses Array vergleicht der Klassifikator das Bild mit dem in Kapitel 6.6 erwähnten Modell. Schließlich wird dem Bild mit der Vorhersage-Funktion des CNN-Algorithmus eine Klasse zugeordnet. Da pro Szene drei Bilder übergeben wurden, wird die am häufigsten zugeordneten Klasse als finale Szeneklasse beibehalten. Abbildung 7.7 zeigt ein Beispielergebnis dieses Klassifizierungsprozesses. Auf Basis der Ergebnisse der Klassifizierung wird die Liste der interessanten Szenen verbessert, indem Szenen der Klasse Fault entfernt werden.

```
{1: 'Slide', 2: 'Slide', 3: 'Professor', 4: 'Fault',
5: 'Whiteboard',6: 'Whiteboard'}
```

Abbildung 7.7: Ergebnis der Klassifikation

7.7 Schneiden der Videos

In diesem Teilprozess wird eine Liste interessanter Szeneänderungen verwendet, die aus Indexen der ersten Bilder jeder interessanten Szene besteht. Als Nächstes werden diese Bilderindexe dann in Sekunden umgerechnet. Um die Videos zu schneiden, wird das Kommandozeilenprogramm *FFmpeg* verwendet. Dem Programm werden die Start- und die Endpositionen in Sekunden für jede Szene übergeben. Nach Ausführung des Programms werden die Videos in einem Distributionsverzeichnis erstellt, das vom Videoverzeichnis abgeleitet wird. Ebenso wird das beschriebene Vorlesungsskript erstellt. Abbildung 7.8 stellt ein Beispiel des Outputs dieses Teilprozesses dar.

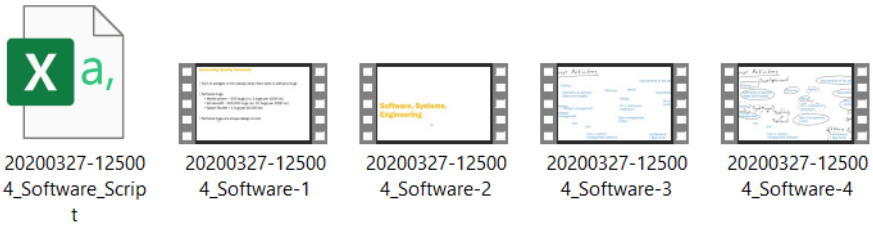


Abbildung 7.8: Ergebnisse der Zerlegung

7.8 Interaktionsdaten vom LMS

Die zu generierenden Daten wurden in Kapitel 6.9 beschrieben. Die Herausforderung bei diesem Teilprozess besteht darin, ähnliche Benutzer zu generieren, um so das Ergebnis der benutzerbasierten kollaborativen Filter überprüfen zu können. Dazu wurden vier Benutzerprofile definiert. Für jedes Profil werden arbiträre Wertebereiche definiert, von denen jeder aus einer Unter- und einer Obergrenze besteht (Abbildung 7.9).

```
very_lazy_students = [[[80, 100], [0, 0], [0, 10]], [0, 0], [5]]
middle_lazy_students = [[[60, 80], [0, 10], [10, 30]], [0, 1], [5]]
medium_students = [[[40, 60], [50, 60], [50, 80]], [1, 2], [5]]
good_students = [[[10, 20], [0, 100], [20, 100]], [1, 2], [5]]
```

Abbildung 7.9: Unterschiedliche Benutzergruppen

Das erste Listenelement ist wieder eine Liste von Wertebereichen für l*earningDuration in %*, da der Wertebereich für dieses Profil von der Szeneklasse abhängt. Das erste Element dieser Liste steht für den Wertbereich der Szenen der Art *Professor*, das zweite für *Whiteboard* und das dritte für *Slide*. Zum Beispiel schauen Studierende des Profils mittlere Studierende (*medium_students*) 40 bis 60 % der Szenen der Klasse Professor, 50 bis 60 % der Szenen der Klasse *Whiteboard* und 50 bis 80 % der Szenen der Klasse *Slide*. Das zweite Listenelement gibt den Wertebereiche für die Profilinformation *numberOfPosts* an, während das letzte die Anzahl der zu generierenden Benutzer für diese Gruppe enthält. Nachdem die Wertebereiche für Daten definiert worden sind, werden für jede Gruppe Zufallsdaten erzeugt. Um sicherzustellen, dass die Benutzer innerhalb der einzelnen Profile stark miteinander korrelieren bzw. starke Ähnlichkeiten aufweisen, wird pro Profil ein Benutzer generiert, um dann auf Basis dessen die restlichen Benutzer als eine Art Klone zu generieren.

7.9 Kollaboratives Filtern

Dieser Abschnitt veranschaulicht die Implementierungsdetails zum kollaborativen Empfehlungssystem, das in Kapitel 5 ausführlich beschrieben wurde. Nach dem Import der nötigen Bibliotheken in das Jupiter Notebook wird im ersten Schritt dieses Prozesses versucht die Daten zu verstehen.

7.9.1 Data Unterstanding

Die LMS-Interaktionsdaten aus Abschnitt 7.8 und das Vorlesungsskript aus Abschnitt 7.7 werden in Jupiter Notebook geladen und in einzelnen panda-Dataframes gespeichert. Unkomplizierte Datenanalysemethoden werden durchgeführt, um nützliche Informationen über die Daten zu erhalten. Als Erstes werden die Spalten der Datensätze untersucht. Tabelle 7.10 bietet einen Überblick über die Spalten des Vorlesungsskripts.

Spalte	Beschreibung
sceneId	Die sceneId ist eine eindeutige ID, anhand derer sich die Szene identifizieren lässt; eine Szene repräsentiert hierbei eine Lerneinheit.
start frame	Index des ersten Bildes der Szene im Originalvideo.
end frame	Index des letzten Bildes der Szene im Originalvideo.
duration in s	Dauer der Szene in Sekunden
scene class	Die bei der Klassifikation zugeordnete Szeneklasse.

Abbildung 7.10: Spalten des Vorlesungsskript

Um die ersten fünf Daten dieser Tabelle anzuzeigen (Abbildung 7.11) wird die Methode head der Panda-Bibliothek verwendet.

	sceneId	start frame	end frame	duration in s	scene class
0	1	0.0	350.0	350.08	Slides
1	2	350.0	950.0	600.12	Professor
2	3	950.0	1830.0	880.32	Slides
3	4	1830.0	2863.0	1032.96	Slides
4	5	2863.0	3178.0	314.52	Slides

Abbildung 7.11: Erste fünf Einträge des Vorlesungsskripts

Mit der Methode *describe* der genannten Bibliothek wird die beschreibende Statistik des Datensatzes ausgegeben (Abbildung 7.12).

	sceneId	title	start frame	end frame	duration in s	scene type
count	12.000000	12	12.000000	12.000000	12.000000	12
unique	NaN	12	NaN	NaN	NaN	2
top	NaN	movie_2	NaN	NaN	NaN	Slides
freq	NaN	1	NaN	NaN	NaN	10
mean	6.500000	NaN	3193.433333	3648.280000	454.776667	NaN
std	3.605551	NaN	1976.001373	1793.854356	368.932838	NaN
min	1.000000	NaN	0.000000	350.000000	6.600000	NaN
25%	3.750000	NaN	1610.000000	2604.750000	172.900000	NaN
50%	6.500000	NaN	3737.500000	4373.920000	332.300000	NaN
75%	9.250000	NaN	4788.400000	5012.290000	670.170000	NaN
max	12.000000	NaN	5450.720000	5457.320000	1119.560000	NaN

Abbildung 7.12: Deskriptive Statistik des Vorlesungsskripts

Der Statistik ist zu entnehmen, dass der Datensatz aus zwölf Szenen besteht. Die durchschnittliche Dauer einer Szene beträgt dabei 7,5 Minuten. Die längste Szene dauert 19 Minuten. Die Mehrheit der Szenen sind der Klasse Slide zugeordnet. Nachdem ein Überblick über die Szenen bzw. Lerneinheiten des Systems gewonnen wurde, werden die Interaktionsdaten vom LMS analysiert. Die Spalten dieses Datensatzes wurden bereits in Kapitel 6.7 beschrieben. Abbildung 7.13 stellt die ersten fünf Daten dieses Datensatzes dar.

Mit der beschreibenden Statistik werden mehr Informationen über den Datensatz gesammelt (Abbildung 7.14).

	userId	sceneId	learningDuration in %	numberOfPosts
0	1	1	55	1
1	1	2	39	2
2	1	3	23	2
3	1	4	40	1
4	1	5	42	2

Abbildung 7.13: LMS Interaktionsdaten

Abbildung 7.14: Beschreibende Statistik der LMS Interaktionsdaten

	userId	sceneId	learningDuration in %	numberOfPosts
count	240.000000	240.000000	240.000000	240.000000
mean	10.500000	6.500000	38.316667	1.337500
std	5.778332	3.459267	26.332913	1.022063
min	1.000000	1.000000	0.000000	0.000000
25%	5.750000	3.750000	17.000000	1.000000
50%	10.500000	6.500000	33.000000	1.000000
75%	15.250000	9.250000	57.250000	2.000000
max	20.000000	12.000000	100.000000	4.000000

Wie der Statistik zu entnehmen ist, betrachten die Studierenden im Durchschnitt 38 % aller Szenen. Eine Vielzahl von Szenen wurden noch nicht angeschaut.

7.9.2 Data Preparation

Nachdem die Datensätze verstanden wurden, wird der Datensatz mit LMS-Interaktionsdaten vorbereitet. Dabei wird zunächst überprüft, ob der Datensatz Duplikate enthält. Diese Prüfung ergibt keine Duplikate. Des Weiteren wird durch einen Normierungsprozess implizites in explizites Feedback umgewandelt. Die betroffenen Features sind *learningDuration in %* und *numberOfPosts*. Bei der Normierung werden beide Features in den Wertebereich [0,1] gebracht. Da die Informationen für *learningDuration in %* in Prozent vorliegen, werden diese durch 100 geteilt, um zum genannten Wertebereich zu gelangen. Dadurch erhält jede vollständig angeschaute Szene einen Bewertungspunkt. Wenn z. B. 10 % einer Szene angeschaut wurden, erhält diese 0,1 Bewertungspunkte. Was das

Feature *numberOfPosts* angeht, wird dieses mit dem Logarithmus-basierten Ansatz normiert. Das erfolgt durch die Anwendung der untenstehenden Gleichung, die in [50] für die Normierung von impliziten Häufigkeiten in die expliziten Bewertungen erfolgreich eingesetzt wurde.

$$log(\frac{1 + numberOfPosts}{1 + MAX(numberOfPosts)}) \tag{7.1}$$

In dieser Gleichung ist $MAX(numberOfPosts)$ die maximale Anzahl an Posts im Datensatz. Als Nächstes werden die genannten Features mit ihren Gewichten multipliziert. Da das Feature *learningDuration in %* eine bessere Aussage über die Präferenz des Benutzers ermöglicht, wird diesem das Gewicht 5 zugewiesen, während dem Feature *numberOfPosts* das Gewicht 2 zugewiesen wird. Somit liegt die Bewertung mit *learningDuration in %* im Wertebereich [0,5], während diejenige von *numberOfPosts* im Bereich [0,2] liegt. Nach der Umwandlung wird – wie in der Konzeption erwähnt – für die Modellierung das Feature learningDuration in % als Benutzerbewertung verwendet (rating = *learningDuration in %*). Weiterhin werden während der Evaluation beide Features durch Addition kombiniert (rating = *learningDuration in %* + *numberOfPosts*). Schließlich werden *learningDuration in %* und *numberOfPosts* gelöscht, da sie bereits in das neue Feature rating umgewandelt wurden. Abbildung 7.15 präsentiert das Ergebnis der Vorbereitung in der Form (*user, item rating*). Dieses Ergebnis wird in einer separaten CSV-Datei gespeichert.

	userId	sceneId	rating
0	1	1	2
1	1	2	1
2	1	3	1
3	1	4	2
4	1	5	2

Abbildung 7.15: Ergebnis der Konvertierung von impliziten in expliziten Daten

Abbildung 7.16 veranschaulicht die Verteilung der Bewertungen im vorbereiteten Datensatz.

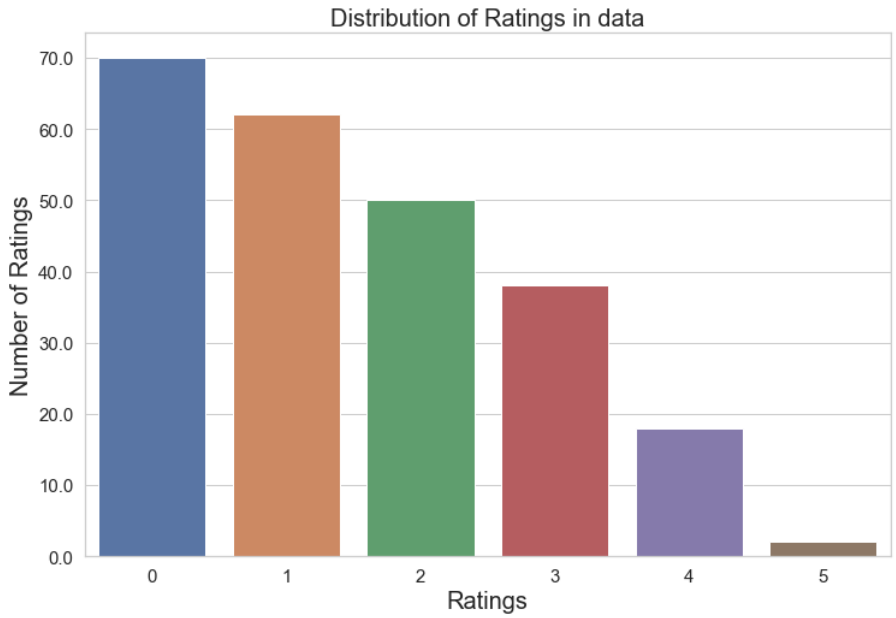


Abbildung 7.16: Explizite Bewertungsdaten

7.9.3 Modeling

In diesem Teilprozess findet die tatsächliche Umsetzung des kollaborativen Filterns statt. Nachfolgend wird auf die Implementierungsdetails eingegangen. Zur Umsetzung des modellbasierten und des objektbasierten Ansatzes wird die Surprise-Bibliothek verwendet, die hierfür entsprechende zuverlässige Algorithmen bereitstellt [41]. Für den benutzerbasierten Ansatz wird der Algorithmus gemäß dem geschilderten Ablauf Schritt für Schritt umgesetzt. Im Rahmen dieser Arbeit wird die Methode Ranking Version of Problem als geeignet angesehen. Nachdem also die Bewertungen der unbewerteten Objekte vorhergesagt worden sind, werden die geschätzten Top-N-Objekte als Ergebnis der Empfehlung genutzt.

7.9.3.1 Benutzerbasiertes kollaboratives Filtern

Abbildung 7.17 stellt die konkrete Implementierung des Algorithmus dar.

Im ersten Schritt wird der vorbereitete Datensatz im neuen Jupiter Notebook geladen und dann in einem *Panda*-Dataframe gespeichert. Danach wird die Benutzer-Objekt-Matrix gebildet (Abbildung 7.18)

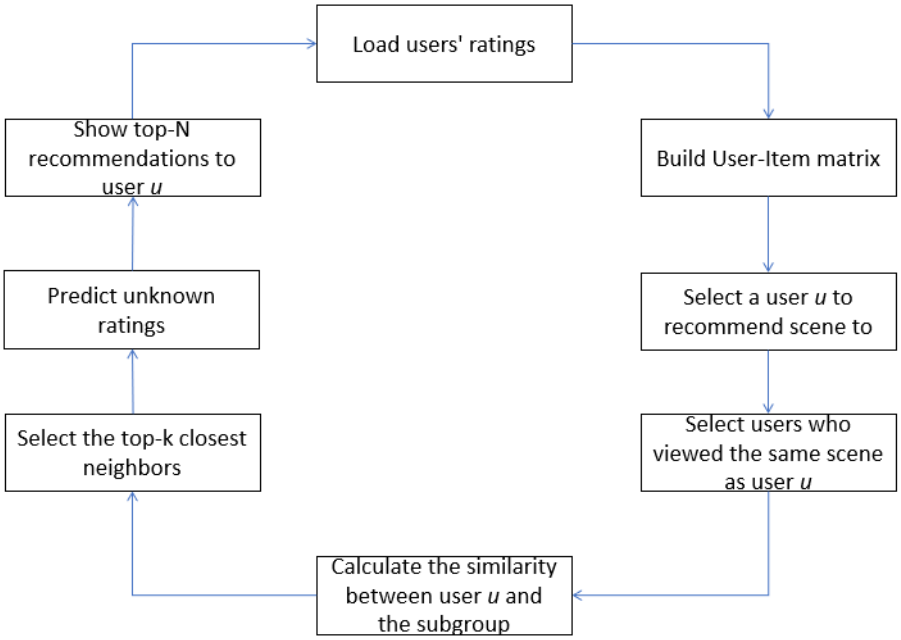


Abbildung 7.17: Algorithmus zum benutzerbasierten kollaborativen Filtern

sceneId	1	2	3	4	5	6	7	8	9	10	11	12
userId												
1	2	1	1	2	2	1	2	2	3	4	4	4
2	3	1	0	1	2	0	3	1	3	4	5	3
3	3	1	1	1	2	1	2	2	3	3	4	4
4	2	2	1	1	1	0	2	2	4	4	4	5
5	2	2	1	1	1	1	2	2	3	4	4	4

Abbildung 7.18: Benutzer-Objekt-Matrix

In dieser Matrix repräsentieren Szenen mit dem Bewertungswert 0 für den Benutzer unbekannte Szenen, deren Bewertungswerte vorhergesagt werden sollen. Als Erstes wird ein aktiver Benutzer u ausgewählt, dessen Bewertungswerte vorhergesagt werden sollen. Um die Implementierung zu illustrieren, wird der Benutzer ($u = 2$) ausgewählt. In der Bewertungsliste dieses Benutzers finden sich noch zwei unbewertete Szenen. Dieser Benutzer gehört zusammen mit den Benutzern 1, 3, 4 und 5 zur Gruppe Gute Studieren-

de. Im nächsten Schritt werden alle Benutzer ausgewählt, die dieselben Szenen gesehen haben wie der Benutzer u. Dann wird für diese Sub-Gruppe von Benutzern eine Ähnlichkeitsberechnung durchgeführt, wobei die Pearson-Korrelation als Metrik verwendet wird. Das Ergebnis dieser Berechnung wird sortiert, um dann die Top-k der ähnlichen Benutzer auszugeben (Abbildung 7.19). In dieser Arbeit wurde $k = 5$ gewählt.

	similarityIndex	userId
1	1.000000	2
2	0.881358	3
0	0.860474	1
4	0.825340	5
3	0.785409	4

Abbildung 7.19: Top-5 der ähnlichsten Benutzer zu Benutzer u

Wie in dieser Abbildung zu sehen ist, sind die ähnlichsten Benutzer zu Benutzer u Benutzer mit dem gleichen Profil. Je näher der Korrelationswert (in Abbildung *similarityIndex*) an 1 ist, desto größer ist die Ähnlichkeit des Benutzers zu Benutzer u. In der gleichen Abbildung ist festzustellen, dass der aktive Benutzer u mit sich selbst zu 100 % korreliert. Im weiteren Verlauf wird dieser Benutzer aus der Liste ähnlicher Benutzer entfernt, um auf diese Weise einen Bias bei der Ergebnisempfehlung zu vermeiden.

Nachdem die k ähnlichsten Benutzer ermittelt worden sind, werden auf Basis ihrer Bewertungen die Bewertungen des Benutzers u vorhergesagt. Dafür werden zunächst die Bewertungen der k ähnlichsten Benutzer herangezogen, um diese dann mit dem jeweiligen Ähnlichkeitsmaß zu multiplizieren. Dadurch entsteht das Feature *weightedRating* (Abbildung 7.20).

Danach wird das *weightedRating* summiert, sobald diese pro Szene gruppiert ist. Die Vorhersage-Bewertungswerte des Benutzers u ergeben sich aus der Gleichung 7.2.

$$rating = weightedRating/sumimilarityIndex \tag{7.2}$$

Abbildung 7.21 veranschaulicht die Ergebnisse dieser Berechnung. Die Spalte Prediction Score enthält die vorhergesagten Bewertungswerte. In dieser Abbildung ist festzustellen, dass die bei der Gruppe beliebten Szenen an den ersten Stellen stehen. Außerdem nähern sich die Vorhersageergebnisse den echten Bewertungswerten des Benutzers u an.

	similarityIndex	userId	sceneId	rating	weightedRating
0	0.881358	3	1	3	2.644073
1	0.881358	3	2	1	0.881358
2	0.881358	3	3	1	0.881358
3	0.881358	3	4	1	0.881358
4	0.881358	3	5	2	1.762715

Abbildung 7.20: Feature *weightedRating*

prediction score	sceneId
4.234270	12
4.000000	11
3.737111	10
3.234270	9
2.262889	1

Abbildung 7.21: Vorhersage der Bewertungen des Benutzers u für alle Objekte im System

Da das Ergebnis der Empfehlung ausschließlich aus unbekannten Objekten bestehen soll, werden im nächsten Schritt unbekannte Objekte ermittelt, wobei deren Bewertung den Vorhersageergebnissen entnommen wird. Zur Ermittlung dieser Objekte wird der Datensatz durchgegangen, um dann alle Szenen des Benutzers u mit dem Bewertungswert 0 zurückzugeben. Die finale Liste mit den Top-N-Empfehlungen wird zurückgegeben (Abbildung 7.22). In dieser Arbeit wurde $N = 10$ ausgewählt. Da dem Benutzer u nur zwei unbekannte Objekte zuzuordnen sind, besteht das Ergebnis der Empfehlung aus diesen beiden Objekten.

prediction score	sceneId
1.00000	3
0.76573	6

Abbildung 7.22: Finales Ergebnis der Empfehlung

7.9.3.2 Objektbasiertes kollaboratives Filtern

Die Modellierung dieses Ansatzes findet in einem separaten Jupiter Notebook statt. Wie erwähnt, ähnelt der Ablauf des Algorithmus dem objektbasierten Ansatz. Hierbei besteht allerdings der Unterschied, dass die Berechnung der Ähnlichkeiten auf einem objektbasierten Verfahren beruht. Die Implementierung dieses Ansatzes erfolgt mit der Surprise-Bibliothek. Im ersten Schritt wird der vorbereitete Datensatz aus Abschnitt 7.9.2 in das Notebook geladen und in einem Panda-Dataframe gespeichert. Daraufhin wird dieses Dataframe in ein Surprise-Dataframe umgewandelt. Dafür wird der Wertebereich der Bewertungen des Datensatzes konfiguriert. Da die Bewertungen des Datensatzes für die Modellierung im Bereich [1,5] liegen, werden diese Werte konfiguriert. Für die Berechnung der Ähnlichkeit zwischen Objekten wird der KNN-Algorithmus der genannten Bibliothek verwendet. Bei der Instanziierung dieses Algorithmus wird der Parameter *sim_options* des Types *Dictionary* übergeben, der die Konfigurationsparameter *name* und *user_based* enthält. Beim Parameter *name* wird die zu verwendende Metrik für die Ähnlichkeitsberechnung konfiguriert. Hier wurde die Kosinusähnlichkeit ausgewählt. Der Parameter *user_based* legt fest, ob die Berechnung der Ähnlichkeit benutzerbasiert oder objektbasiert erfolgt. Dieser Parameter wird auf *False* gesetzt, um die Berechnung objektbasiert zu machen. Nach diesen Konfigurationen wird der KNN-Algorithmus auf dem Datensatz angelernt, nachdem dieser in ein Surprise-Trainset-Objekt umgewandelt wurde (Abbildung 7.23).

```
algo.fit(data_df.build_full_trainset())
```

```
Computing the cosine similarity matrix...
Done computing similarity matrix.

<surprise.prediction_algorithms.knns.KNNBasic at 0x1ff0de3c7f0>
```

Abbildung 7.23: Berechnung ähnlicher Objekte mit dem KNN-Algorithmus

Nachdem das Modell trainiert wurde, können die nächsten k ähnlichsten Objekte zu einem Ziel-Objekt mit der Methode *get_neighbors* des Algorithmus ausgegeben werden. Ein Beispiel hierfür bietet Abbildung 7.24. Dort werden für die Szene $i = 1$ die $k = 10$ ähnlichsten Szenen in sortierter Form ausgegeben.

Im nächsten Schritt werden die bislang unbewerteten Objekte durch den aktiven Benutzer ermittelt. Wie beim benutzerbasierten Ansatz wird für alle Objekte des aktiven Benutzers der Bewertungswert 0 zurückgegeben. Danach werden die Bewertungswerte

```
inner_id = algo.trainset.to_inner_iid(1)
neighbors = algo.get_neighbors(inner_id, k=10)
neighbors
```

```
[4, 7, 6, 8, 1, 3, 9, 2, 10, 11]
```

Abbildung 7.24: Ergebnis der Berechnung ähnlicher Objekte mit dem KNN-Algorithmus

des aktiven Benutzers für unbekannte Objekte mit der Methode *predict* des Algorithmus vorhergesagt. Schließlich werden die Ergebnisse der Vorhersage sortiert, um dann die Top-N-Elemente auszugeben. Die Abbildung stellt das Ergebnis der Empfehlung für den Benutzer $u = 2$ dar.

sceneId	prediction score
3	1.985293
6	1.938131

Abbildung 7.25: Ergebnis mit objektbasiertem Ansatz

7.9.3.3 Modelbasiertes kollaboratives Filtern

Ebenso wie beim objektbasierten Ansatz wird der vorbereitete Datensatz auch beim modellbasierten Ansatz in ein Surprise-Dataframe umgewandelt, um dann den zu verwendenden Algorithmus zu instanziieren. Hier wird der Algorithmus SVD der *Surprise*-Bibliothek verwendet, die eine Implementierung der Matrix-Faktorisierung bereitstellt. Als Nächstes wird das Modell auf dem Datensatz angelernt, um Muster in Bewertungen von Benutzern zu ermitteln (Abbildung 7.26).

```
algo.fit(data_df.build_full_trainset())
```

```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x1c8fde346d0>
```

Abbildung 7.26: Anlernen des Modells mit dem SVD-Algorithmus

Nach diesem Schritt werden die bislang unbewerteten Objekte durch den aktiven Benutzer ermittelt und deren Bewertungswerte vorhergesagt. Diese Vorhersage erfolgt mit der Methode *predict* des genannten Algorithmus, die als Parameter den Benutzer und das

vorherzusagende Objekt erwartet. Die Bewertungen aller unbekannten Objekte werden vorhergesagt. Die Ergebnisse werden in eine Liste eingetragen. Anschließend wird diese Liste nach dem Bewertungswert sortiert, wobei die Top-N-Elemente als Ergebnis der Empfehlung ausgegeben werden. Die Abbildung 7.27 zeigt das Ergebnis für den aktiven Benutzer $u = 2$. Die Evaluation der oben entwickelten Ansätze wird im nächsten Kapitel behandelt.

sceneId	prediction score
6	1.389786
3	1.283437

Abbildung 7.27: Ergebnis der Empfehlung mit dem SVD-Algorithmus

8 Evaluation

In diesem Kapitel wird das entwickelte System im Hinblick auf Performance und Genauigkeit evaluiert. Das Ziel der Evaluation ist es, Schwachstellen im System zu finden, um daraufhin mögliche Optimierungsmaßnahmen definieren zu können. Zunächst werden die verwendeten Testdaten und die genutzte Hardware beschrieben. Danach wird eine separate Evaluation der einzelnen Prozessschritte vorgenommen. Schließlich werden das Ergebnis wie auch mögliche Optimierungsmaßnahmen diskutiert.

Testdaten

Die Tests werden mit Online-Vorlesungen aus verschiedenen Studienfächern durchgeführt. Da diese Vorlesungen durch verschiedene Professoren gehalten wurden, unterscheidet sich die jeweilige Vorlesungsstruktur. Tabelle 8.1 zeigt eine Übersicht über die verwendeten Online-Vorlesungen. Die ersten drei Vorlesungseinheiten wurden von der Hochschule zur Verfügung gestellt. Diese Vorlesungen wurden mit dem Videokommunikationsprogramm Zoom aufgezeichnet. Die vierte Einheit wurde mit einem anderen unbekannten Programm aufgezeichnet und stammt von der Plattform YouTube.

Vorlesung	Einheit	Professor
Software Engineering	GMT20200327-125004_Software-E_2736x1700	A
Computernetzwerke	GMT20200327-125004_Software-E_2736x1700	A
Artificial Intelligence	Artificial Intelligence - VL - W21-Statistics-3	B
Digitaltechnik	MEB09 Digitaltechnik Übung - 2021-01-18	C

Tabelle 8.1: Verwendete Online-Vorlesungen

Hardware

Das unterstehende Hardware wurde für die Implementierung und Evaluation verwendet.

- **Hardware**: Lenovo-Laptop
- **Betriebssystem**: Windows 10
- **Prozessor**:Intel Core i7-8750H 2.20 GHz
- **Arbeitsspeicher**: 8 GB

8.1 Automatisierte Zerlegung der Vorlesungsaufzeichnung

In diesem Abschnitt wird die Performance der einzelnen Prozesskomponenten gemessen. Des Weiteren wird die Erkennungsgenauigkeit der Vorlesungsszenen überprüft und das Ergebnis der Zerlegung thematisiert.

8.1.1 Performance

Die Performance bezieht sich auf den Datenspeicherverbrauch und die Laufzeit der Teilprozesse. Die Messungen werden in den Tabellen 8.2 bis 8.5 veranschaulicht. Dabei repräsentieren die temporären Artefakte die Zwischenergebnisse der einzelnen Teilprozesse. Die finalen Artefakte repräsentieren die einzelnen Endergebnisse.

Lecture: Computernetzwerke Professor: Prof A Audio fps: 44100 Video fps: 25 Duration: 1:30:57	Temporal Artifacts (MB)	Final Artifacts (MB)	Running Time (sec)
Audio Features Extraction	WAV (918)	-	66
Visual Features Extraction	-	-	2924
Detection of Interesting Scene Changes	JPG (22), TXT, CSV	-	56
Scene Classification	-	-	12
Cutting the Videos	-	CSV, MP4 (286)	2
LMS Interaction Data	-	CSV	<1
Total	940	286	3062
Legend: - = No artifact produced			

Tabelle 8.2: Performanzevaluation der Vorlesung Computernetzwerke

Vorlesung: Software Engineering Professor: Prof A Audio fps: 44100 Video fps: 25 Länge: 1:26:00	Temporal Artifacts (MB)	Final Artifacts (MB)	Running Time (sec)
Audio Feature Extraction	WAV (864)	-	67
Visuell Feature Extraction	-	-	2749
Detection of interesting Scene Changes	JPG (45,1), TXT	-	133
Scene Classification	-	-	26
Cutting the Video	-	CSV, MP4 (161)	3
LMS Interaktionsdaten	-	CSV	<1
Gesamt	891,1	161	2981
Legend: - = No artifact produced			

Tabelle 8.3: Performanzevaluation der Vorlesung Software Engineering

Lecture: Artificial Intelligence Professor: Prof B Audio fps: 44100 Video fps: 30 Länge: 00:31:41	Temporal Artifacts (MB)	Final Artifacts (MB)	Running Time (sec)
Audio Feature-Extraction	WAV (321)	-	30
Visual Feature-Extraction	-	-	923
Detection of Interesting Scene Changes	JPG (69,1), TXT	-	65
Scene Classification	-	-	125
Cutting the Videos	-	CSV, MP4 (69,6)	2
LMS Interaction Data	-	CSV	<1
Total	390,1	69,6	1200
Legend: - = No artifact produced			

Tabelle 8.4: Performanzevaluation der Vorlesung Artificial Intelligence

Lecture: Digitaltechnik Professor: Prof B Audio fps: 44100 Video fps: 30 Duration: 01:12:24	Temporal Artifacts (MB)	Final Artifacts (MB)	Running Time (sec)
Audio Feature Extraction	WAV (730)	-	54
Visual Feature Extraction	-	-	949
Detection of Interesting Scene Changes	JPG (222), TXT	-	63
Scene Classification	-	-	124
Cutting the Videos	-	CSV, MP4 (702)	4
LMS Interaction Data	-	CSV	<1
Total	952	702	1196
Legend: - = No artifact produced			

Tabelle 8.5: Performanzevaluation der Vorlesung Digitaltechnik

Datenspeicher

Aus der Messung ist zu entnehmen, dass der meiste Speicherplatz vom Teilprozess *Audio Feature Extraction* belegt wird. Die erzeugte Audio-Datei (WAV in temporäre Artefakte) ist bis zu dreimal größer als das Eingangsvideo. Dieser Unterschied ist auf das genutzte Kodierungsformat zurückzuführen. Der nächstgrößte Speicherplatzverbraucher hängt vom gewählten Aufzeichnungsprogramm bei. Bei Zoom-Aufzeichnungen ist dies der Teilprozess *Recognitions of interesting Scene Changes* durch die Erzeugung von Schlüsselbildern (JPG temporäre Artefakte). Dem folgt der Teilprozess Cutting the Videos zur Erstellung von Lerneinheiten (MP4 in finale Artefakte). Die Größe der Bilderdateien (JPG in temporäre Artefakte) ist abhängig von der Anzahl der ermittelten Schlüsselbilder. Was die Gesamtgröße der Lerneinheiten betrifft, ist diese fast genauso groß wie das Eingangsvideo, was zu erwarten war. Im Gegensatz zu den Zoom-Vorlesungen ist bei der Vorlesung mit dem unbekannten Aufzeichnungsprogramm der Teilprozess *Cutting the Videos* als der nächstgrößte Speicherplatzverbraucher zu identifizieren. Dieser Unterschied liegt wahrscheinlich darin begründet, dass das Programm FFmpeg eine andere Kodierungstechnik als das Eingangsvideo verwendet. Die Größe der CSV- und der TXT-Artefakte liegt im Byte- bis Kilobyte-Bereich und kann daher bei der Speicherplatz-Belegung vernachlässigt werden.

Laufzeit

Auch hier ist festzustellen, dass die Gesamtlaufzeit des Prozesses der *automatisierten Zerlegung der Aufzeichnungen* vom verwendeten Aufzeichnungsprogramm abhängt. Für eine Zoom-Vorlesung von einer Stunde und 30 Minuten beträgt diese Zeit 51 Minuten. Dies entspricht 57 % der Dauer des Eingangsvideos. Bei der Vorlesung Digitaltechnik mit einem anderen Aufzeichnungsprogramm fällt diese Laufzeit wesentlich kürzer aus. So beträgt sie hier nur 27 % der Dauer des Eingangsvideos. Insgesamt wird die meiste Zeit für den Teilprozess *Visual Feature Extraction* verbraucht. Die für diesen Prozess benötigte Zeit beläuft sich auf mehr als 90 % der Gesamtlaufzeit des Prozesses. Dies ist auf den Umstand zurückzuführen, dass das Eingangsvideo bei der Feature-Extraktion Bild für Bild ausgelesen wird. Was die Szeneklassifikation angeht, ist die Laufzeit abhängig von der Anzahl an einzelnen Vorlesungsszenen. Außerdem ist zu erwähnen, dass die Videoauflösung keinen Einfluss auf die Gesamtlaufzeit hat. Dieser Vorteil wurde durch die Redimensionierung der Bilder im Schritt der Vorverarbeitung gewonnen.

8.1.2 Erkennungsgenauigkeit

In diesem Evaluierungsschritt wird überprüft, mit welcher Genauigkeit das System die einzelnen Szenen, Pausen und interessanten Szenen in der Vorlesungsaufzeichnung erkennt. Dazu werden die Aufzeichnungen manuell auf die Anzahl an vorkommenden Szenen, Pausen und interessanten Szenen analysiert, um das Ergebnis dann mit dem Systemergebnis zu vergleichen. Für die manuelle Analyse wurde das VLC Media Player verwendet. Die Auswertung wird in Tabelle 8.7 dargestellt. Als Nächstes werden die Einträge dieser Tabelle in Tabelle 8.6 beschrieben.

Metriken	Beschreibung
Single Scenes	Einzelne Szenen im Eingangsvideo. Für eine Szene der Klasse PowerPoint handelt es sich um den Videoteil mit derselben PowerPoint-Folie. Für eine Szene der Klasse Whiteboard ist es der Videoteil zwischen zwei leeren Whiteboards.
Breaks	Pausen im Eingangsvideo. Als Pause werden Ruhezeiten von mindestens 3 s betrachtet.
Interesting Scenes	Lerneinheiten im Eingangsvideo. Hierbei handelt es sich um eine Sequenz von Szenen.
Recognized Scenes	Vom System erkannte einzelne Szenen.
Recognized Breaks	Vom System erkannte Pausen.
Class of Recognized Scenes	Vom System zugeordnete Klasse der Szene.
Recognized Interesting Scenes	Vom System erkannte Lernabschnitte. Diese entstehen durch das Filtern von unnötigen Szeneänderungen.

Tabelle 8.6: Beschreibung der Metriken

Measurement \ Lecture	Computer-netzwerke	Software Engineering	Artificial Intelligence	Digitaltechnik
Scenes	16	21	13	6
Breaks	13	30	1	32
Interesting Scenes	14	10	3	4
Recognized Scenes	23	46	36	250
Recognized Breaks	13	30	1	32
Class of Recognized Scenes	P(3), W(7), S(10), F(3)	P(3), W(7), S(30), F(6)	P(2), W(7), S(32), F(2)	P(250)
Recognized Interesting Scenes	11	13	4	4
Legend: P = Professor, F = Fault, W = Whiteboard, S = Slide,				

Tabelle 8.7: Ergebnis der Szeneerkennung

Vorlesung Software Engineering

- Struktur: Diese Vorlesungseinheit besteht aus sieben Szenen, die unterschiedlichen Klassen zuzuordnen sind. Die Vorlesung beginnt und endet mit einer Szene der Klasse Slide. Während der Vorlesung wird zwischen Professor-Szenen und Whiteboard-Szenen oder auch zwischen Professor- und Slide-Szenen gewechselt. Die Slide-Szene besteht aus einzelnen Szenen (PowerPoint-Folien). Ebenso besteht die Whiteboard-Szene aus einzelnen Whiteboard-Szenen.
- Ergebnis der Erkennung: Alle Szenen und einzelnen Szenen werden vom System erkannt. Zusätzlich werden Fault-Szenen erfasst. Hierbei ist festzustellen, dass solche Fault-Szenen häufig durch die Transition von Szenen einer Klasse zu Szenen einer anderen Klasse verursacht werden. Es werden zwei einzelne Szenen der Klasse Slide und Whiteboard zweimal erkannt.
- Finales Ergebnis und Auswertung: Aus der Analyse resultieren 13 Lerneinheiten (in der Tabelle *Recognized Interesting Scenes*). Die Dauer dieser Lerneinheiten liegt im Bereich zwischen 2,5 und 24 Minuten.

Vorlesung Computernetzwerke

- Struktur: Diese Einheit besteht aus fünf Szenen mit unterschiedlicher Klasse. Die Vorlesung beginnt und endet mit einer Szene der Klasse Professor. Während der Vorlesung wird zwischen Professor- und Whiteboard-Szenen oder auch zwischen Professor- und Slide-Szenen hin und her gewechselt.
- Ergebnis der Erkennung: Alle Szenen und einzelnen Szenen werden vom System erkannt.
- Finales Ergebnis und Auswertung: Aus der Analyse ergeben sich elf Lerneinheiten. Die Dauer dieser Einheiten liegt zwischen zwei und 16 Minuten.

Vorlesung Artificial Intelligence

- Struktur: Ebenso wie die Vorlesung Computernetzwerke, beginnt und endet diese Vorlesung mit Professor-Szenen. Während der Vorlesung werden ausschließlich Powerpoint-Folien gezeigt. Diese Powerpoint-Folien enthalten viele Farben.
- Ergebnis der Erkennung: Alle Szenen und einzelne Szenen werden vom System erkannt. Es werden 3 einzelne Sildes-Szenen 2 Mal erkannt.

- Finales Ergebnis und Auswertung: Aus der Analyse resultieren 4 Lerneinheiten. Die Dauer dieser Einheiten liegt zwischen 4 und 11 Minuten.

Vorlesung Digitaltechnik

- Struktur: Diese Vorlesung beginnt mit einer Professor-Szene. Während der gesamten Vorlesung wird ausschließlich mit dem Whiteboard gearbeitet. Hierbei sind gelichzeitig der Professor und das Whiteboard im Vollbild zu sehen.
- Ergebnis der Erkennung: Die Professor-Szene wird richtig erkannt. Nicht alle Whiteboard-Szenen werden erkannt.
- Finales Ergebnis und Auswertung: Aus der Analyse resultieren vier Lerneinheiten mit einer Dauer von vier bis 35 Minuten. Die Nichterkennung einiger Whiteboard-Szenen führt zu einer längeren Dauer bestimmter Lerneinheiten.

8.2 Empfehlungssystem

In diesem Abschnitt werden die Vorhersagegenauigkeit der Benutzerbewertung und die Laufzeit des entwickelten kollaborativen Verfahrens überprüft. Am Ende wird das für den genutzten Datensatz geeignetste Modell ausgewählt. Die Evaluierung orientiert sich an Kapitel 6.8.3. Die Algorithmen werden unter verschiedenen Bedingungen getestet. Die Ansätze werden mit einem Datensatz getestet, bei dem die Anzahl der Benutzer wesentlich größer ist als die Anzahl an vorhandenen Objekten und umgekehrt. Für den Test werden die Lerneinheiten des vorigen Abschnitts verwendet, um LMS-Interaktionsdaten zu generieren. Zur Durchführung der Validierung wird das Kreuzvalidierung-Framework *cross_validate* der Surprise-Bibliothek verwendet, mit dem Parameter $kfolds = 5$.

8.2.1 Anzahl Benutzer größer als Anzahl Objekte

Zur Generierung des Datensatzes werden die Lerneinheiten der Vorlesung Computernetzwerk verwendet (elf Lerneinheiten). Insgesamt werden 1000 Benutzer aus den vier beschriebenen Profilen generiert. Die Abbildungen 8.1 bis 8.3 veranschaulichen das Ergebnis der Messung mit den Metriken MAE und RMSE.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	1.1839	1.1674	1.1691	1.1902	1.1830	1.1787	0.0089
MAE (testset)	0.9632	0.9317	0.9297	0.9365	0.9336	0.9389	0.0123
Fit time	0.97	1.01	0.92	0.96	0.94	0.96	0.03
Test time	1.05	0.92	0.86	0.92	0.88	0.93	0.07

10.77217149734497

Abbildung 8.1: Evaluation des benutzerbasierten kollaborativen Filterns

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.9630	0.9358	0.9732	0.9486	0.9424	0.9526	0.0137
MAE (testset)	0.7471	0.7323	0.7512	0.7306	0.7372	0.7397	0.0082
Fit time	0.02	0.02	0.02	0.02	0.01	0.01	0.00
Test time	0.04	0.03	0.04	0.06	0.03	0.04	0.01

0.38523316383361816

Abbildung 8.2: Evaluation des objektbasierten kollaborativen Filterns

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.7363	0.7482	0.7526	0.7567	0.7422	0.7472	0.0072
MAE (testset)	0.5859	0.5992	0.6040	0.6087	0.5870	0.5970	0.0091
Fit time	0.51	0.41	0.42	0.42	0.42	0.44	0.04
Test time	0.01	0.01	0.01	0.04	0.01	0.02	0.01

3.374509572982788

Abbildung 8.3: Evaluation des modellbasierten kollaborativen Filterns

Laufzeit

Der Messung ist zu entnehmen, dass der benutzerbasierte Ansatz viel Zeit in Anspruch nimmt. Dies lässt sich damit erklären, dass die Berechnung der Ähnlichkeit bei diesem Ansatz benutzerbasiert erfolgt, wobei das System 1000 Benutzer enthält. Des Weiteren ist festzustellen, dass der objektbasierte Ansatz wenig Zeit benötigt. Ein Grund dafür könnte sein, dass wenige Objekte (Lerneinheiten) im System vorhanden sind.

Genauigkeit

Aus der Messung ist zu entnehmen, dass der modellbasierte Ansatz eine bessere Vorhersagegenauigkeit als die anderen Ansätze liefert. Des Weiteren ist festzustellen, dass der objektbasierte Ansatz ein besseres Ergebnis als der benutzerbasierte Ansatz liefert. In Tabelle 8.8 wird das Ergebnis der Messung zusammengefasst.

Computernetzwerke, 1000 students, 11 learning units		MAE	RMSE	Running time(sec)
memory-based	User-based	0.93	1,17	10,7
	Item-based	0,73	0.95	0,3
model-based	SVD	0,59	0,74	3,3

Tabelle 8.8: Anzahl Benutzer größer als Anzahl Objekte

8.2.2 Anzahl Objekte größer als Anzahl Benutzer

Die Lerneinheiten aus der Zerlegung der beiden Vorlesungseinheiten von Professor A werden für die Generierung der Benutzer und der LMS-Daten verwendet. Der generierte Datensatz besteht aus zehn Benutzern und 24 Lerneinheiten. In den Abbildungen 8.4 bis 8.6 wird die Messung dargestellt.

```
                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.6259  1.3911  1.4799  1.6069  1.5770  1.5362  0.0882
MAE (testset)     1.4289  1.2181  1.2243  1.4536  1.3627  1.3375  0.0996
Fit time          0.00    0.00    0.00    0.00    0.00    0.00    0.00
Test time         0.00    0.00    0.00    0.00    0.02    0.00    0.01
0.3690311908721924
```

Abbildung 8.4: Evaluation des benutzerbasierten kollaborativen Filtern

```
                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9229  0.9367  0.8995  0.9344  0.9105  0.9208  0.0142
MAE (testset)     0.7666  0.7320  0.7446  0.7702  0.7576  0.7542  0.0142
Fit time          0.00    0.00    0.00    0.00    0.00    0.00    0.00
Test time         0.00    0.01    0.00    0.00    0.00    0.00    0.00
0.09914278984069824
```

Abbildung 8.5: Evaluation des objektbasierten kollaborativen Filtern

```
                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.8935  0.9214  0.9383  0.9078  0.9222  0.9167  0.0151
MAE (testset)     0.7155  0.7925  0.7650  0.7373  0.7881  0.7597  0.0295
Fit time          0.02    0.02    0.02    0.02    0.01    0.01    0.00
Test time         0.00    0.00    0.00    0.00    0.00    0.00    0.00
0.20740151405334473
```

Abbildung 8.6: Evaluation des modellbasierten kollaborativen Filtern

Laufzeit

Bezüglich der Laufzeit ist zu beobachten, dass diese beim benutzerbasierten Ansatz deutlich geringer ausfällt als beim vorherigen Test. Dies lässt sich damit erklären, dass im System nur wenige Benutzer vorhanden sind. Des Weiteren ist festzustellen, dass der objektbasierte Ansatz auch in diesem Test die beste Laufzeit aufweist.

Genauigkeit

Die Ergebnisse der Vorhersagegenauigkeit ähneln denen beim vorherigen Test. Des Weiteren ist diesbezüglich festzustellen, dass der objektbasierte Ansatz ein besseres Ergebnis erzielt als der vorherige Test. Sein Ergebnis ähnelt dem des modellbasierten Ansatzes, der auch hier durchgängig das beste Ergebnis liefert. Der Grund für das bessere Ergebnis des objektbasierten Ansatzes könnte darin bestehen, dass der Algorithmus mit mehr Daten besser arbeitet.

Das Ergebnis dieses Tests wird in Tabelle 8.9 zusammengefasst.

Computernetzwerke and Software Engineering, 15 students, 25 course units		MAE	RMSE	Running time(sec)
memory-based	User-based	1,33	1,53	0,3
	Item-based	0,75	0,92	0,09
model-based	SVD	0,75	0,91	0,2

Tabelle 8.9: Anzahl Objekte größer als Anzahl Benutzer

8.3 Diskussion

Die Evaluierung des Empfehlungssystems hat gezeigt, dass der benutzerbasierte Ansatz aufgrund der übermäßig langen Laufzeit und der unzureichenden Vorhersagegenauigkeit für den vorliegenden Datensatz nicht geeignet ist. Bezüglich der Laufzeit hat der objektbasierte Ansatz die besten Ergebnisse erzielt. Was die Vorhersagegenauigkeit betrifft, lieferte der modellbasierte Ansatz das beste Ergebnis. Für den Fall einer größeren Anzahl an Objekten als an Benutzern lieferte auch der objektbasierte Ansatz ein vergleichbares Ergebnis mit dem modellbasierten Ansatz. Werden die Performanz und die Vorhersagegenauigkeit betrachtet, ist der objektbasierte Ansatz besser für den verwen-

deten Datensatz geeignet. Außerdem sorgt dieser Ansatz für mehr Vielfalt hinsichtlich des Empfehlungsergebnisses.

Während der Evaluierung wurden auch die Features *learningDuration in %* und *numberOfPosts* kombiniert, um so den Einfluss eines neuen Features auf das Empfehlungssystem bewerten zu können. Diese Kombination führte zu einer Verschlechterung der Vorhersagegenauigkeit. Ein Grund dafür könnte sein, dass die Kombinierung dieser Features durch die Addition ungeeignet ist.

Bei der Evaluierung der automatisierten Zerlegung der Aufzeichnung wurden drei Engpässe ermittelt. Einige Szenen wurden nicht richtig erkannt, während andere zwei- oder dreimal erfasst wurden. Dies hatte zur Folge, dass einige der aus der Analyse resultierenden Lerneinheiten eine längere Übertragungsdauer aufweisen. Eine Möglichkeit zur Beseitigung dieses Problems könnte darin bestehen, die Erkennung von Szeneänderungen innerhalb einer Sequenz bei Whiteboard-Szenen mithilfe der Bildklassifikation durchzuführen. Dadurch könnte ein leeres Whiteboard erkannt werden und könnten die Videoteile zwischen zwei leeren Whiteboards als einzelne Szene betrachtet werden. Außerdem wurde bei der Performanzevaluierung festgestellt, dass die meiste Analysezeit auf den Teilprozess *Visual Feature Extraction* fällt. Dieses Problem ließe sich womöglich dadurch beseitigen, dass im Zuge des Leseprozesses nicht alle Videobilder berücksichtigt werden. Für die Bildauswahl könnte etwa ein bestimmtes zeitliches Intervall festgelegt werden. Obwohl das Empfehlungssystem auf Basis der generierten LMS-Interaktionsdaten evaluiert werden konnte, waren die generierten Daten nicht vielfältig. Zudem korrelierten einige Benutzer mit Benutzern anderer Profile. Dies ist der Tatsache geschuldet, dass die Korrelation vornehmlich auf dem Verhalten beruht. Dabei können aufgrund von Zufallsdaten Benutzer generiert werden, die dasselbe Verhalten wie Benutzer anderer Profile zeigen.

9 Fazit und Ausblick

Im Rahmen dieser Thesis wurde eine systematische Literaturrecherche zu den Forschungsgebieten der *Videozusammenfassung* und des *Empfehlungssystems* durchgeführt. Dabei wurde festgestellt, dass es in beiden Gebieten eine Vielzahl an Publikationen gibt, von denen jedoch nur wenige den Vorlesungs- oder Learning-Kontext adressieren. Ebenso wurden verwandte Arbeiten in genannten Forschungsfelder thematisiert. Dabei wurde festgestellt, dass bereits Ansätze existieren, die die definierte Forschungsfrage teilweise beantworten. Auf Grundlage dieser Recherche wurde eine Lösung konzipiert, die sich aus zwei Prozessen zusammensetzt.

Den ersten dieser Prozesse bildete die *automatisierte Zerlegung der Vorlesungsaufzeichnungen*. Für diesen Prozess wurden die Techniken der statischen Videozusammenfassung und der audiovisuellen Szeneanalyse eingesetzt, um Szeneänderungen zu erkennen. Die wesentlichen Vorteile dieser Lösung gegenüber bestehenden Ansätzen liegen darin, dass sie kein Wissen über die im Vorlesungsvideo verwendete Sprache voraussetzt.

Der zweite Prozess war das *Empfehlungssystem für Lerneinheiten*. Hierbei wurden unterschiedliche Ansätze des kollaborativen Filterns entwickelt. Als Methode zur Empfehlung von Lerneinheiten wurde in diesem Zusammenhang die *Top-N-Empfehlung* ausgewählt.

Als Schnittstelle zwischen beiden Prozessen wurden LMS-Interaktionsdaten generiert, wobei entsprechende Annahmen getroffen wurden. Das entwickelte System wurde in Kapitel 8 evaluiert. Dabei wurde die Performance der einzelnen Teilprozesse überprüft. Außerdem wurde beim ersten Prozess auf die Erkennungsgenauigkeit der Szenen eingegangen. Beim zweiten Prozess wurde die Vorhersagegenauigkeit der Benutzerbewertung der verschiedenen kollaborative Ansätze überprüft, um dann auf Basis des Ergebnisses den objektbasierten Ansatz als das geeignetste Modell auszuwählen.

Die Evaluierung hat gezeigt, dass das System bestimmten Einschränkungen unterliegt und gewisse Schwächen aufweist. Aufgrund der zeitlichen Rahmenbedingungen konnten die Qualität und der Mehrwert des Systems nicht in größerem Umfang überprüft werden. Hierfür würde sich eine von der Hochschule durchgeführte Online-Evaluierung des Ergebnisses des objektbasierten Ansatzes eignen, um so die Benutzerzufriedenheit

mit den präsentierten Empfehlungen zu messen. Diese Online-Evaluierung könnte die Form einer Studierendenbefragung annehmen. Des Weiteren könnten die Studierenden zur Gebrauchstauglichkeit der erstellten Lerneinheiten befragt werden. Um die Qualität der Zerlegung zu erhöhen, könnten leere Whiteboards in Vorlesungsvideos durch die Bildklassifikation erkannt werden. Schließlich könnte sich eine weiterführende Forschungsarbeit mit der Integration des entwickelten Empfehlungssystems in das LMS der Hochschule befassen.

Erklärung

Ich versichere, dass ich diese Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie alle wörtlich oder sinngemäß übernommenen Stellen in der Arbeit gekennzeichnet habe. Die Arbeit wurde noch keiner Kommission zur Prüfung vorgelegt und verletzt in keiner Weise Rechte Dritter

Waiblingen, den 21. Dezember 2021 [UNTERSCHRIFT]

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] Björn Bohnenkamp, Katja G. Marcus Burkhardt B. Marcus Burkhardt: Online-Lehre 2020 – Eine medienwissenschaftliche Perspektive. In: *Hochschulforum Digitalisierung*, 2020

[2] Romero, Cristobal ; Ventura, Sebastian: Educational data mining: A survey from 1995 to 2005. In: *Expert systems with applications* 33 (2007), Nr. 1, S. 135–146

[3] Brocke, Jan v. ; Simons, Alexander ; Niehaves, Bjoern ; Niehaves, Bjorn ; Reimer, Kai ; Plattfaut, Ralf ; Cleven, Anne: Reconstructing the giant: On the importance of rigour in documenting the literature search process. (2009)

[4] Fu, Dan ; Liu, Qingtang ; Zhang, Si ; Wang, Jianhu: The Undergraduate-Oriented Framework of MOOCs Recommender System. In: *2015 International Symposium on Educational Technology (ISET)*, 2015, S. 115–119

[5] Bousbahi, Fatiha ; Chorfi, Henda: MOOC-Rec: A Case Based Recommender System for MOOCs. In: *Procedia - Social and Behavioral Sciences* 195 (2015), 1813-1822. http://dx.doi.org/https://doi.org/10.1016/j.sbspro.2015.06.395. – DOI https://doi.org/10.1016/j.sbspro.2015.06.395. – ISSN 1877–0428. – World Conference on Technology, Innovation and Entrepreneurship

[6] Jain, Harshit ; Anika: Applying Data Mining Techniques for Generating MOOCs Recommendations on the Basis of Learners Online Activity. In: *2018 IEEE 6th International Conference on MOOCs, Innovation and Technology in Education (MITE)*, 2018, S. 6–13

[7] Romero, Cristóbal ; Ventura, Sebastián ; García, Enrique: Data mining in course management systems: Moodle case study and tutorial. In: *Computers & Education* 51 (2008), Nr. 1, S. 368–384

[8] Gulzar, Zameer ; Leema, A A. ; Deepak, Gerard: Pcrs: Personalized course recommender system based on hybrid approach. In: *Procedia Computer Science* 125 (2018), S. 518–524

[9] AlMousa, Mohannad ; Benlamri, Rachid ; Khoury, Richard: NLP-Enriched Automatic Video Segmentation. In: *2018 6th International Conference on Multimedia Computing and Systems (ICMCS)*, 2018, S. 1–6

[10] Kota, Bhargava U. ; Stone, Alexander ; Davila, Kenny ; Setlur, Srirangaraj ; Govindaraju, Venu: Automated Whiteboard Lecture Video Summarization by Content Region Detection and Representation. In: *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, S. 10704–10711

[11] Vimalaksha, Anusha ; Vinay, Siddarth ; Prekash, Abhijit ; Kumar, N. S.: Automated Summarization of Lecture Videos. In: *2018 IEEE Tenth International Conference on Technology for Education (T4E)*, 2018, S. 126–129

[12] Xu, Fei ; Davila, Kenny ; Setlur, Srirangaraj ; Govindaraju, Venu: Content Extraction from Lecture Video via Speaker Action Classification Based on Pose Information. In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*, 2019, S. 1047–1054

[13] Bora, Amit ; Sharma, Shanu: A Review on Video Summarization Approcahes : Recent Advances and Directions. In: *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, 2018, S. 601–606

[14] In: Ngo, Chong-Wah ; Wang, Feng: *Video Summarization*. Boston, MA : Springer US, 2009. – ISBN 978–0–387–39940–9, 3320-3324

[15] Kini M., Mahesh ; Pai, Karthik: A Survey on Video Summarization Techniques. In: *2019 Innovations in Power and Advanced Computing Technologies (i-PACT)* Bd. 1, 2019, S. 1–5

[16] Hou, Xiaodi ; Harel, Jonathan ; Koch, Christof: Image Signature: Highlighting Sparse Salient Regions. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34 (2012), Nr. 1, S. 194–201. http://dx.doi.org/10.1109/TPAMI.2011.146. – DOI 10.1109/TPAMI.2011.146

[17] Kühn, Benjamin: *Interessengetriebene audiovisuelle Szenenexploration*. Bd. 22. KIT Scientific Publishing, 2016

[18] Adomavicius, G. ; Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. In: *IEEE Transactions on Knowledge and Data Engineering* 17 (2005), Nr. 6, S. 734–749. http://dx.doi.org/10.1109/TKDE.2005.99. – DOI 10.1109/TKDE.2005.99

[19] Burke, Robin: Hybrid recommender systems: Survey and experiments. In: *User modeling and user-adapted interaction* 12 (2002), Nr. 4, S. 331–370

[20] Engelbert, Benedikt: *Maschinelle Generierung von Empfehlungen zur Lehr-/Lernunterstützung im Hochschulkontext*, Universität Osnabrück, Diss., 2016

[21] Aggarwal, Charu C. u. a.: *Recommender systems*. Bd. 1. Springer, 2016

[22] Herlocker, Jonathan L. ; Konstan, Joseph A. ; Terveen, Loren G. ; Riedl, John T.: Evaluating collaborative filtering recommender systems. In: *ACM Transactions on Information Systems (TOIS)* 22 (2004), Nr. 1, S. 5–53

[23] Guo, Yaqiong ; Huang, Mengxing ; Lou, Tao: A Collaborative Filtering Algorithm of Selecting Neighbors Based on User Profiles and Target Item. In: *2015 12th Web Information System and Application Conference (WISA)*, 2015, S. 9–14

[24] Ricci, Francesco ; Rokach, Lior ; Shapira, Bracha: Introduction to recommender systems handbook. In: *Recommender systems handbook*. Springer, 2011, S. 1–35

[25] Koren, Yehuda ; Bell, Robert ; Volinsky, Chris: Matrix Factorization Techniques for Recommender Systems. In: *Computer* 42 (2009), Nr. 8, S. 30–37. http://dx.doi.org/10.1109/MC.2009.263. – DOI 10.1109/MC.2009.263

[26] *How you can build simple recommender systems with Surprise*. https://towardsdatascience.com/how-you-can-build-simple-recommender-systems-with-surprise-b0d32a8e4802. – [Online; Abruf: 27. November 2021]

[27] Sicilia, Miguel Ángel ; García-Barriocanal, Elena ; Sánchez-Alonso, Salvador ; Cechinel, Cristian: Exploring user-based recommender results in large learning object repositories: the case of MERLOT. In: *Procedia Computer Science* 1 (2010), Nr. 2, 2859-2864. http://dx.doi.org/https://doi.org/10.1016/j.procs.2010.08.011. – DOI https://doi.org/10.1016/j.procs.2010.08.011. – ISSN 1877–0509. – Proceedings of the 1st Workshop on Recommender Systems for Technology Enhanced Learning (RecSysTEL 2010)

[28] Lu, Jie: A personalized e-learning material recommender system. In: *International Conference on Information Technology and Applications* Macquarie Scientific Publishing, 2004

[29] Oard, Douglas W. ; Kim, Jinmook u. a.: Implicit feedback for recommender systems. In: *Proceedings of the AAAI workshop on recommender systems* Bd. 83 WoUongong, 1998, S. 81–83

[30] Jawaheer, Gawesh ; Szomszor, Martin ; Kostkova, Patty: Comparison of implicit and explicit feedback from an online music recommendation service. In: *pro-*

ceedings of the 1st international workshop on information heterogeneity and fusion in recommender systems, 2010, S. 47–51

[31] Bligh, Donald A.: *What's the Use of Lectures?* Intellect books, 1998

[32] Schmidt, Henk G. ; Wagener, Stephanie L. ; Smeets, Guus A. ; Keemink, Lianne M. ; Molen, Henk T. d.: On the use and misuse of lectures in higher education. In: *Health Professions Education* 1 (2015), Nr. 1, S. 12–18

[33] Lauer, Tobias ; Trahasch, Stephan: Begriffsbesprechung: Vorlesungsaufzeichnung:. 4 (2005), Nr. 3, 61. http://dx.doi.org/doi:10.1524/icom.2005.4.3.61. – DOI doi:10.1524/icom.2005.4.3.61

[34] Jadon, Shruti ; Jasim, Mahmood: Video summarization using keyframe extraction and video skimming. In: *arXiv preprint arXiv:1910.04792* (2019)

[35] Matei, Alina ; Glavan, Andreea ; Talavera, Estefanía: Deep learning for scene recognition from visual data: a survey. In: *International Conference on Hybrid Artificial Intelligence Systems* Springer, 2020, S. 763–773

[36] *A Gentle Introduction to k-fold Cross-Validation*. https://machinelearningmastery.com/k-fold-cross-validation/. – [Online; Abruf: 30. Oktober 2021]

[37] Pedregosa, Fabian ; Varoquaux, Gaël ; Gramfort, Alexandre ; Michel, Vincent ; Thirion, Bertrand ; Grisel, Olivier ; Blondel, Mathieu ; Prettenhofer, Peter ; Weiss, Ron ; Dubourg, Vincent u. a.: Scikit-learn: Machine learning in Python. In: *the Journal of machine Learning research* 12 (2011), S. 2825–2830

[38] Pérez, Fernando ; Granger, Brian: *The Jupyter Notebook*. https://jupyter.org/. – [Online; Abruf: 13. Oktober 2021]

[39] Jetbrains: *The Python IDE for Professional Developers*. https://www.jetbrains.com/pycharm/. – [Online; Abruf: 13. September 2021]

[40] Culjak, Ivan ; Abram, David ; Pribanic, Tomislav ; Dzapo, Hrvoje ; Cifrek, Mario: A brief introduction to OpenCV. In: *2012 Proceedings of the 35th International Convention MIPRO*, 2012, S. 1725–1730

[41] Hug, Nicolas: Surprise: A Python library for recommender systems. In: *Journal of Open Source Software* 5 (2020), Nr. 52, 2174. http://dx.doi.org/10.21105/joss.02174. – DOI 10.21105/joss.02174

[42] Project, The P.: *Package overview*. https://pandas.pydata.org/pandas-docs/stable/getting_started/overview.html. – [Online; Abruf: 10. September 2021]

[43] DEVELOPERS, NumPy: *NumPy*. https://www.numpy.org/. – [Online; Abruf: 13. Oktober 2021]

[44] COMMUNITY, The S.: *SciPy community*. https://docs.scipy.org/doc/scipy/reference/tutorial/general.html. – [Online; Abruf: 13. November 2021]

[45] TEAM, Matplotlib development: *Matplotlib: Visualization with Python*. https://matplotlib.org/. – [Online; Abruf: 13. Oktober 2021]

[46] *User Guide*. https://zulko.github.io/moviepy/. – [Online; Abruf: 15. November 2021]

[47] *CSV File Reading and Writing*. https://docs.python.org/3/library/csv.html. – [Online; Abruf: 13. Oktober 2021]

[48] *Keras: simple, flexibel, powerful*. https://keras.io/. – [Online; Abruf: 12. Oktober 2021]

[49] *A complete, cross-platform solution to record, convert and stream audio and video*. https://www.ffmpeg.org/. – [Online; Abruf: 13. Oktober 2021]

[50] HU, Yifan ; KOREN, Yehuda ; VOLINSKY, Chris: Collaborative filtering for implicit feedback datasets. In: *2008 Eighth IEEE International Conference on Data Mining* Ieee, 2008, S. 263–272