technische universität
dortmund

Bachelorarbeit

Erweiterung von AMUSE zur Verarbeitung mehrerer Modalitäten

Clara Pingel

26. März 2024

überarbeitete Version vom 27. August 2024

Betreuer:
Prof. Dr. Günter Rudolph
Dr. Igor Vatolkin

Fakultät für Informatik
Algorithm Engineering (LS 11)
Technische Universität Dortmund
http://ls11-www.cs.tu-dortmund.de

Inhaltsverzeichnis

Kapitel 1

Einleitung

Mit einem Musikstück sind viele verschiedene Eigenschaften assoziiert, von denen manche von Menschen intuitiv benannt werden können. Der Forschungsbereich „Music Information Retrieval“ (MIR) [2] befasst sich mit der automatischen Analyse von Musik. Dabei werden Merkmale entwickelt und verwendet, die Repräsentationen von Musik in Form verschiedener Dateiformate benutzen, um Charakteristika von Musik zu ermitteln und darzustellen. Merkmale bewegen sich im Kontext verschiedener Modalitäten, verfolgen, abhängig von der Modalität, unterschiedliche Ansätze und können sich unter anderem in ihrem Abstraktionsgrad, in ihrem Fokus auf einen bestimmten Aspekt von Musik und bezüglich des Datentyps, auf den sie angewendet werden, unterscheiden.

1.1 Motivation

Die Extraktion von Merkmalen im AMUSE-Framework [28] erfolgt bislang ausschließlich aus der Modalität „Audio“. Dagegen ist jede Repräsentation von Musik durch bestimmte modalitäts- und formatsspezifische Faktoren, wie beispielsweise die Aufnahmequalität von Audiodateien und auch das Format an sich, in seinem Informationsgehalt limitiert. So können möglicherweise bestimmte Eigenschaften eines Musikstücks oder einer Kategorie von Musikstücken aus einem bestimmten Dateityp nicht oder nicht optimal ermittelt werden. Beispielsweise ist die harmonische Struktur eines Musikstücks auch in einer Audiodatei, aber in einer MIDI-Datei desselben Stücks in deutlich präziserer Form enthalten.

Im Allgemeinen bietet die Verwendung von verschiedenen Modalitäten und Dateiformaten bei der Extraktion von Merkmalen, dem Training von Modellen und der Klassifikation von Musik das Potenzial, Eigenschaften von Musik umfassender, präziser und detaillierter bestimmen zu können, indem durch die Auswahl von Merkmalen verschiedener Modalitäten formats- und modalitätsspezifische Vorteile effektiv genutzt werden. Im Kontext von Genre-Klassifikation kann die Auswahl von Merkmalen und den zugrundeliegenden Modalitäten zu einer Annäherung an genre-spezifische Besonderheiten beitragen.

In der Konsequenz könnte die Extraktion aus anderen Quellen für die Klassifikation von Musik auch im Rahmen von AMUSE vorteilhaft sein und möglicherweise die Informationen aus Audiodateien ergänzen.

1.2 Verwandte Arbeiten

Diese Arbeit steht im Kontext verwandter Arbeiten, die sich mit der multimodalen Klassifikation von Musik befassen. Dabei wurde die Verwendung von Daten auf Basis mehrerer Modalitäten zur Klassifikation verschiedener Kategoriesysteme, zum Beispiel die Klassifikation von Musik in Genre- oder Stimmungskategorien, und damit im Rahmen verschiedener Problemstellungen, untersucht. Die modalitätsspezifischen Repräsentationen für jedes Musikstück, die dazu verwendet wurden, stammen teilweise aus derselben Quelle, wurden mit verschiedenen Verfahren aus dem Internet extrahiert und sind mit Matching-Verfahren einander zugeordnet oder nur auf Basis des Musikstücks miteinander assoziiert. Die verwendeten Daten unterscheiden sich zudem auch in der Anzahl der verwendeten Modalitäten sowie in der Anzahl verwendeter Musikstücke. Ausführungen zu multimodalen Datensätzen finden sich in Abschnitt 4.2. Die Experimente unterscheiden sich weiterhin in der Wahl des Klassifikators, der verwendeten Merkmalsmenge und hinsichtlich der Verwendung von „Feature-Selecting“-Verfahren.

Ein Beispiel für ein Experiment, welches Audiodateien und symbolische Daten zur Genre-Klassifikation benutzt und vergleicht, ist die Arbeit von G. Tzanetakis et al. [25]. In dieser wurde gezeigt, dass die Bestimmung von Tonhöhen aus Audiodateien durch geeignete Algorithmen im Kontext von Tonhöhen-Histogrammen Fehlern unterliegen kann, die bei der Berechnung von Tonhöhen-Histogrammen aus symbolischen Daten nicht entstehen. Auch wenn die jeweilige Verwendung von Merkmalen beider Datentypen gute Klassifikationsergebnisse bewirkt, können Merkmale aus symbolischen Daten durchaus ein Zugewinn hinsichtlich der Präzision der extrahierten Merkmale und damit möglicherweise auch der Klassifikationsergebnisse bedeuten.

Auch die Arbeit von Z. Cataltepe et al. [3] verwendet Audiodaten gemeinsam mit symbolischen Daten und erzielt mit der Klassifikation auf Basis einer Mehrheitsentscheidung unter Verwendung mehrerer Klassifikationsalgorithmen, Abtastraten und Liedlängen gute Ergebnisse. Dabei ist zu beachten, dass die in diesem Versuch verwendeten Audiodateien aus den MIDI-Dateien generiert wurden und dass Klassifikationsalgorithmen zunächst nur auf jeweils einer Modalität operieren. Berücksichtigt werden beide Modalitäten erst im Zuge der Mehrheitsentscheidung. Dabei wurde versucht, sich den sehr guten Ergebnissen von C. McKay und I. Fujinaga [17] anzunähern, die diese bei der ausschließlichen Verwendung von symbolischen Merkmalen erzielt haben.

Andere Arbeiten verwenden dagegen eine größere Anzahl an Modalitäten. Beispielsweise wurden in der Arbeit von C. McKay und I. Fujinaga [18] Merkmale aus Audiodaten,

aus symbolischen und kulturellen Daten zur Genre-Klassifikation verwendet und dabei der Einfluss der drei Modalitäten für jede mögliche Kombination aus den drei Merkmalsmengen getestet. Die Klassifikation auf Basis aller drei Merkmalsgruppen stellte sich dabei im Vergleich zur Klassifikation mit nur einer Merkmalsgruppe als vorteilhaft heraus. Auch die Arbeit von B. Wilkes et al. [31] nutzt Musikdaten aus drei Modalitäten: Audiodaten, Bilddaten und Textdaten. Im Rahmen dieses Experiments erwies sich die Kombination zweier Modalitäten gegenüber der Kombination aller drei Modalitäten insgesamt häufiger als gewinnbringend.

Die Arbeit von I. Vatolkin und C. McKay [27] untersucht die Verwendung von Merkmalen aus verschiedenen Modalitäten zur Genre-Klassifikation mithilfe eines Datensatzes, der Musikdaten aus sechs verschiedenen Modalitäten enthält. Es wurde ein evolutionärer Ansatz gewählt, der die Optimierung der Merkmalsmenge unter Berücksichtigung mehrerer Kriterien zum Ziel hat. Obwohl neben der Minimierung des balancierten relativen Klassifikationsfehlers auch hinsichtlich der Homogenität der verwendeten Modalitäten zur Berechnung der Merkmale optimiert wurde, basierte die Merkmalsauswahl, die für die Klassifikation eines jeweiligen Genres am besten bewertet wurde, durchweg auf mehr als einer Modalität. Unter der Vorraussetzung einer geeigneten Merkmalsauswahl betont dies den Einfluss, den die Verwendung von Merkmalen auf Basis verschiedener Modalitäten auf die Klassifikationsergebnisse haben kann. Die Arbeit gibt außerdem Hinweise darauf, welche Merkmalstypen ausschlaggebend für die Zuordnung spezifischer Genres sein können.

Ein im Kontext multimodaler Klassifikation häufig getestetes Szenario ist auch die Kombination von Audiodaten und Liedtexten. Zu nennen sind dabei unter anderem die Arbeiten von R. Neumayer und A. Rauber [19] sowie die Arbeit von C. Laurier et al. [13].

1.3 Ziele

Die Arbeit soll sich daher mit der multimodalen Erweiterung von AMUSE befassen. Mit AMUSE sollen Musikdaten nicht nur, wie bislang aus Audiodateien, sondern entsprechende Merkmale auch aus anderen Datentypen extrahiert werden können.

Denkbar sind dabei unter anderem symbolische Merkmale, beispielsweise aus MIDI- MusicXML- oder ABC-Dateien als auch Merkmale, die aus Liedtexten, kulturellen Daten oder Audiocovern extrahiert werden. Der Fokus der Arbeit soll darauf liegen, die Extraktion von symbolischen Daten in AMUSE zu ermöglichen, um die extrahierten Merkmale, zusätzlich zu extrahierten Merkmalen aus Audiodateien, zur Klassifikation von Musikstücken verwenden zu können.

Vorbereitend auf die Extraktion von weiteren Datentypen soll die Klassenstruktur des AMUSE-Frameworks an die zusätzliche Modalität angepasst werden. Die Anpassungen in der Klassenstruktur sollen die Funktionalitäten der verschiedenen Modalitäten trennen

und bieten die Möglichkeit, perspektivisch weitere Modalitäten in AMUSE zu integrieren. Bezüglich der Extraktion von Merkmalen, soll außerdem die korrekte Zuordnung von Dateiformaten zu Tools durch entsprechende Fehlerbehandlung sichergestellt werden.

Um Merkmale aus symbolischen Daten in AMUSE extrahieren zu können, soll zunächst eine Recherche zu geeigneten Tools erfolgen und anschließend jSymbolic sowie gegebenenfalls weitere Tools in AMUSE eingebunden werden. Dafür wird unter anderem die Klasse `jSymbolicAdapter` benötigt. jSymbolic enthält die konkrete Implementierung der Merkmalsextraktion. Die Merkmale, die mit jSymbolic extrahiert werden, werden in die grafische Oberfläche eingebaut, sodass diese für passende Eingabedateien zur Extraktion ausgewählt werden können.

Um die Merkmale aus symbolischen Daten bei der Klassifikation anzuwenden und zu testen, wird eine Studie durchgeführt. Der Einfluss von symbolischen Merkmalen auf die Klassifikationsgüte soll dabei in Experimenten, die nur Merkmale aus jeweils einer und aus beiden genannten Modalitäten benutzen, beurteilt werden. Die Dateien der beiden Modalitäten beziehen sich dabei jeweils auf dieselben Musikstücke. Dafür werden eine Auswahl an MIDI-Dateien aus dem Lakh-Datensatz (LMD-aligned) [22][1] und daran angepasste 30-sekündige MP3-Ausschnitte von 7digital.com verwendet. Dabei soll sich die Klassifikation auf das Genre des jeweiligen Musikstücks beziehen, wobei jedes Musikstück einem von fünf Genres angehört.

1.4 Aufbau

Diese Arbeit besteht aus fünf Kapiteln. In Kapitel 1 wurde eine Einleitung in den Inhalt gegeben, Motivation und Ziele erläutert und einen Überblick über verwandte Arbeiten gegeben. Kapitel 2 beschäftigt sich mit Grundlagen, auf die diese Arbeit aufbaut und soll den benötigten Kontext bezüglich dem AMUSE-Framework, MIDI-Dateien im Allgemeinen, relevanten Tools und Begriffen geben. In Kapitel 3 wird, gemäß den Zielen, eine Übersicht über relevante Teile der Klassenstruktur gegeben und die für die Extraktion mehrerer Modalitäten nötigen Änderungen dokumentiert. Darunter fallen Änderungen, die die hinzukommenden Modalitäten, die Oberfläche, den Ablauf von Extraktionsaufgaben und Konversionen, betreffen. Kapitel 4 enthält die zuvor genannte Studie, mit der die Anwendung und der Einfluss von MIDI-Dateien in einem Experiment veranschaulicht werden soll. Kapitel 5 gibt einen Ausblick hinsichtlich der weiteren Arbeit an AMUSE und weiterer möglicher Experimente im Anschluss an die erstellte Studie.

[1] `https://colinraffel.com/projects/lmd` (zuletzt geprüft am 19.03.2024 um 16:04 Uhr)

Kapitel 2

Grundlagen

Um den nötigen Kontext für die nachfolgenden Kapitel zu liefern, werden im Folgenden wichtige Begriffe wie *Modalität*, *Format* und *Merkmal* definiert und beschrieben. Außerdem wird das AMUSE-Framework, insbesondere der Ablauf von Extraktionsaufgaben, und, im Kontext von symbolischen Daten und Merkmalen, das MIDI-Format und verschiedene Tools zur Extraktion symbolischer Merkmale vorgestellt.

2.1 Modalität

Im Bereich Music Information Retrieval bezieht sich *Modalität* auf die grundsätzliche Form der musikalischen Informationen. Modalitäten bilden gleichzeitig eine Kategorie, in die *Repräsentationen* und *Dateiformate* fallen, die Musik im Sinne einer Modalität darstellen. Die Repräsentation beschreibt die Stuktur der Daten, während das Format diese in Form einer spezifischen, von einem Rechner interpretierbaren Darstellungsweise konkretisiert.

Daten einer Modalität können in mehrfacher Hinsicht mit einem Musikstück in Verbindung stehen. Beispielsweise kann es von einem Musikstück Videomaterial mehrerer *Kategorien*, wie einen Konzertmitschnitt oder ein klassisches Musikvideo, geben. Die Tabelle 2.1 enthält einen Vorschlag, wie Kategorien von musikbezogenen Daten Modalitäten zugeordnet werden, und diese strukturell und in Form eines Formats repräsentiert sein können.

Dabei ist zu beachten, dass die Modalität nicht durch die Repräsentation oder das Format definiert wird, diese sich also doppeln können. Beispielsweise enthält die Modalität „Kulturelle Daten“ Kategorien musikalischer Daten, die eine Darstellungsform einer anderen Modalität besitzen, dessen Informationen aber ihren Ursprung in einer „kulturellen“ oder „sozialen Handlung“, wie dem Erstellen einer Playlist, haben.

Merkmale werden, wie bislang in AMUSE, aus Audiodateien, aus symbolischen Musikrepräsentationen [33], beispielsweise durch die in Abschnitt 2.5 aufgelisteten Tools, aus visuellen Daten, wie Albumcovern oder aus Liedtexten [8,14,31] extrahiert. Auch Merkma-

Modalität	**Kategorien**	**Repräsentationen**	**Formate**
Audiosignal („Audio“)	Studioaufnahme, Cover, Konzertmitschnitt	Zeitreihe, 2D-Vektor (Spektrum)	AIFF, AIFC, AU, FLAC, MP3, SND, WAVE
Symbolische Daten („Symbolisch“)	Händische Transkription, Automatische Transkription	Matrixdarstellung, Graphendarstellung, Sequenzdarstellung	ABC, MEI, MIDI, MUSICXML
Bild	Albumcover, Foto des Künstlers	3D-Array	JPEG, PNG
Text	Liedtext	Zeichenkette	TXT
Video	Musikvideo, Konzertmitschnitt	4D-Array	MOV, MP4
Kulturelle Daten („Kulturell“)	Vorkommen in Playlists, Suchmaschinen-Ergebnisse, Soziale Tags	Graphendarstellung, Schlüssel/Wert-Paare, Zeichenkette	JSON, TXT

Tabelle 2.1: Auflistung von Modalitäten mit zugehörigen Kategorien, Repräsentationen und Formaten (beispielhaft) im Kontext Musik.

le, die aus kulturellen Informationen zu Musikstücken, wie Tags [12] oder das Vorkommen in Playlists [26, 32], entstehen, sind möglich und werden im Kontext von „Music Information Retrieval“ erforscht und getestet. Wie alle anderen genannten Modalitäten können auch audiovisuelle Daten wertvolle Informationen zu einem Musikstück liefern.

Für die Modalitäten „Audio“ und „Symbolisch“ sind bekannte Formate in Tabelle 2.1 aufgelistet. Die aufgelisteten Formate dieser Modalitäten werden im Zuge der Anpassungen in AMUSE berücksichtigt.

2.2 Merkmale

Wie in Kapitel 1 beschrieben, beinhaltet der MIR-Forschungsbereich die Entwicklung und Anwendung von Merkmalen auf Musikdaten. Diese berechnen Informationen zu verschiedenen Eigenschaften von Musik. Merkmale können in verschiedenen Kontexten zur direkten Analyse und durch Machine-Learning-Verfahren weiterverwendet werden. Grundsätzlich ist jede Form von digitaler Musikrepräsentation oder -information als Grundlage für Merkmale denkbar. In Abschnitt 2.1 sind bekannte Modalitäten aufgelistet, aus denen Merkmale extrahiert werden können.

Digitale Musikrepräsentationen oder -informationen können aus verschiedenen Quellen stammen. Zum Beispiel können Audiodateien von Musikstücken aus Schallwellen durch

Umwandlung in ein digitales Signal oder durch die Weiterverarbeitung vorhandener digitaler Signale entstehen. Dazu können geeignete Programme verwendet werden, um entsprechende Dateien manuell oder automatisch zu generieren. Im Allgemeinen verwenden Audiomerkmale ein digitales Signal, das heißt, die Abtastung von Schallwellen oder Funktionen, um Daten in numerischer Form zu erzeugen.

Daten können auch aus der Konvertierung aus anderen Formaten oder teilweise auch Modalitäten stammen. Dabei sind eine große Anzahl von Kombinationen denkbar. Zum Beispiel können aus MIDI-Dateien Audiodateien entstehen oder andersherum, aus Audiodateien MIDI-Dateien berechnet werden. Aus Audiodateien können mittels Spracherkennungsverfahren Liedtexte oder auch die Sprache der Texte ermittelt werden. AMUSE unterstützt die Konvertierung von MP3- zu WAVE-Dateien und soll zukünftig auch weitere Konvertierungen beinhalten. Dies ist in Abschnitt 3.2 näher beschrieben. Eine weitere Quelle für Musikdaten stellen Angaben oder erzeugte Daten von Experten oder Allgemein von Musikhörern dar, darunter das Vorkommen von Musikstücken in, von Hörern erstellten, Playlists.

In den überwiegenden Fällen bewegen sich die zugrundeliegenden Daten und die Art des Merkmals in derselben Modalität. Bei Hinzunahme von weiteren Tools können aber auch Informationen weiterer Modalitäten aus einer Datei berechnet werden. Ein gutes Beispiel für ein modalitätsübergreifendes Merkmal ist das music21-Merkmal „Composer-Popularity“, welches die Anzahl der Treffer bestimmt, die eine Google-Suche mit dem Interpreten eines Musikstücks hat. Dabei wird der Interpret des Musikstücks einer symbolischen Datei entnommen.

Die digitalen Informationen, die eine Datei eines Formats über ein Musikstück enthält, werden im Zuge der Merkmalsberechnung durch verschiedene Verfahren und Tools verarbeitet. Dabei entstehen Informationen, die sich entweder auf *Extraktionszeitfenster*, im Folgenden auch *Zeitfenster* gennannt, also einer Reihe von Abtastwerten, oder ganze Stücke beziehen. Audiomerkmale werden vorwiegend, so auch in AMUSE, auf Zeitfenstern berechnet. Dafür ist es notwendig, dass die zugrundeliegende Datenstruktur prinzipiell in der zeitlichen Dimension interpretierbar ist. Merkmale, wie die Anzahl der Google-Treffer, können demnach nicht sinnvoll auf Extraktionszeitfenstern berechnet werden. Das Ergebnis der Berechnung ist eine Matrix, die für jedes Zeitfenster alle dem Merkmal zugehörigen Dimensionen enthält. Die Größe des Zeitfensters und die Anzahl der Dimensionen hängen dabei von der Art des Merkmals und der Implementierung ab. Extrahierte Merkmale können dann, wie auch durch AMUSE, weiterverarbeitet und modifiziert werden, wie in Abschnitt 2.3 beschrieben. Die verarbeiteten Zeitfenster werden auch *Klassifikationsfenster* genannt.

Eine weitere Eigenschaft von Merkmalen ist ihr Abstraktionsgrad. Merkmale lassen sich prinzipiell in „low-level“, „mid-level“ und „high-level“-Merkmale einteilen. „low-level“-Merkmale, wie zum Beispiel das Merkmal „Zero-crossing rate“, interpretieren nahe am

Signal. „high-level“-Merkmale beschreiben dagegen musikalische Eigenschaften [11] auf einem symbolischen Level, wie zum Beispiel das Vorkommen von Akkorden oder Instrumenten. Allgemein haben Merkmale, die aus symbolischen Daten berechnet werden, einen höheren Abstraktionsgrad, da sie, im Gegensatz zu Audiodateien, kein Signal enthalten, aus denen „low-level“-Merkmale extrahierbar wären. Dasselbe gilt für Merkmale, die auf Modalitäten wie beispielsweise Liedtexten oder kulturellen Daten basieren.

Merkmale, die aus Audiodateien berechnet werden, nutzen oft die Fouriertransformation [30, S. 123-130]. Diese berechnet den Anteil von Frequenzen an einem Audiosignal. Diese Merkmale interpretieren den *Frequenzbereich* eines Signals. Merkmale, die direkt aus dem Audiosignal berechnet werden, bewegen sich auf dessen *Zeitbereich*. Ein Merkmal, welches in den Frequenzbereich fällt, ist das Merkmal „Spectral Centroid“, welches den Schwerpunkt des Frequenzspektrums angibt. Ein Beispiel für ein Merkmal im Zeitbereich ist das Merkmal „Zero-crossing rate“, welches misst, wie oft das Signal die x-Achse durchläuft.

2.3 AMUSE-Framework

Zur Berechnung von Merkmalen kann das AMUSE-Framework genutzt werden. AMUSE ist ein Java-Framework, das am Lehrstuhl 11 der TU Dortmund entwickelt wird [28]. Es bietet einen Rahmen für die Extraktion von Merkmalen und die Weiterverarbeitung entlang einer Klassifikationspipeline, also die Verarbeitung der Merkmale, das Training von Modellen, die Klassifikation mithilfe dieser trainierten Modelle, die Validierung von Klassifikationsergebnissen und die Optimierung von Parametern. Diese ist in Abbildung 2.1 skizziert. Jeder der in Abbildung 2.1 enthaltenen Schritte, wird von AMUSE als ein *Task* behandelt, der seperat vom Nutzer definiert und von AMUSE bearbeitet wird. Die Beschreibung und Ausführung von Tasks kann zum einen durch die in AMUSE implementierte grafische Oberfläche sowie auch per Übergabe einer Task-Konfiguration in Form einer ARFF-Datei[1] geschehen.

Dabei nutzt AMUSE verschiedene Tools, zum einen für die Extraktion von Merkmalen und zum anderen für die Verwendung der daraus entstandenen Daten zur Generierung von Klassifikationsmodellen durch geeignete Lernverfahren. Diese Tools sind entweder direkt oder in Form von Plugins in AMUSE integriert. Tools, die für die Merkmalsextraktion genutzt werden, werden im Folgenden auch *Extraktionstools* oder *Extraktoren* genannt.

Die Buttons der grafischen Oberfläche des „Experiment Configurators“ (s. Abbildung 2.2) entsprechen den einzelnen Teilschritten der Klassifikationspipeline. Die Ansichten, die mit den Buttons erreicht und mit denen die verschiedenen Tasks konfiguriert werden können, sind ebenfalls nach diesen benannt. Beispielsweise nennt sich die Ansicht,

[1] `https://waikato.github.io/weka-wiki/formats_and_processing/arff_stable/` (zuletzt geprüft am 22.08.2024 um 17:10 Uhr)

mit der Extraktions-Tasks erstellt werden können, „Extraction Configurator“. Die übrigen Konfigurationsansichten sind analog nach der Art des Tasks benannt.

Im ersten Schritt der Klassifikationspipeline erfolgt die Extraktion von Merkmalen, die in Abschnitt 2.2 beschrieben ist. Dabei ist jedem Merkmal in AMUSE eine Merkmals-ID zugeordnet, die zusammen mit der ID des Extraktionstools und weiteren Attributen in einer ARFF-Datei mit dem Namen `featureTable.arff` gespeichert ist. Extrahierte Merkmale können entweder direkt für unterschiedliche Zwecke verwendet oder, im Zuge der Merkmalsverarbeitung, durch AMUSE weiterverarbeitet werden. Dazu werden die Merkmalswerte aller ausgewählten Merkmale zu einer Matrix harmonisiert [30, S. 372-373]. Für jede Merkmalsdimension und jedes Zeitfenster enthält diese einen Merkmalswert oder einen NaN-Eintrag, also einen fehlenden Wert. Zeitfenster können dabei eine feste, vom Musikstück unabhängige Größe und Überlappung haben oder sich an den Ereignissen („Onsets“) des Musikstücks orientieren. Diese müssen in AMUSE mit einem weiteren Merkmal berechnet werden. Im Zuge der Merkmalsverarbeitung können die Daten außerdem weiter modifiziert werden, indem beispielsweise fehlende Werte ersetzt oder der Wertebereich normalisiert werden.

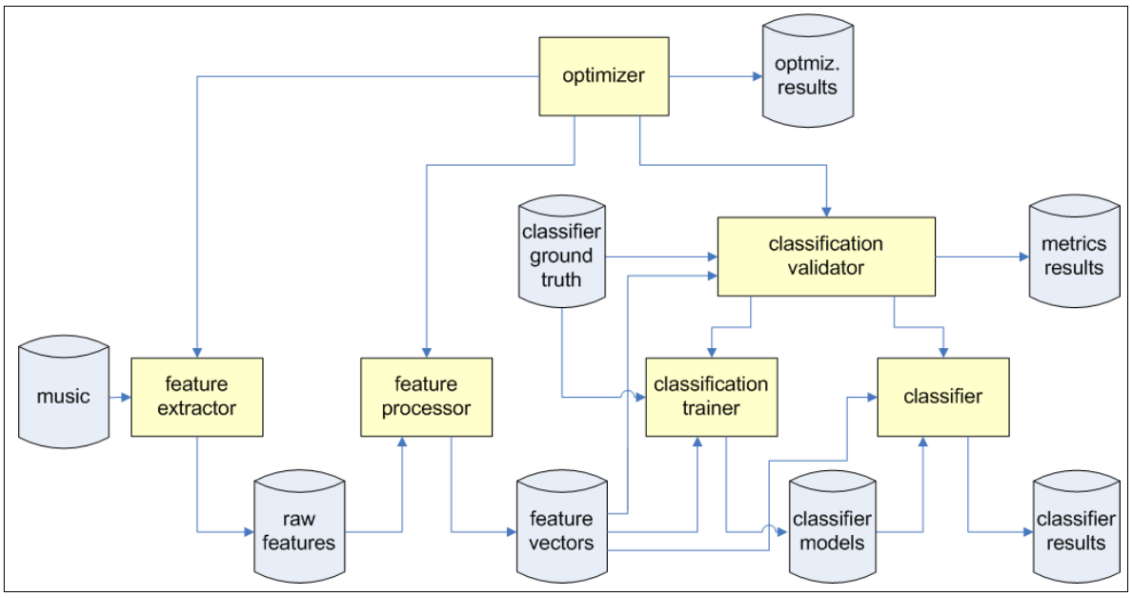

Abbildung 2.1: Schema der AMUSE-Klassifikationspipeline [28]. AMUSE-Tasks und damit Teilschritte der Klassifikationspipeline sind gelb dargestellt. Ein- und Ausgabedaten der jeweiligen Teilschritte sind grau dargestellt.

Im nächsten Schritt können Modelle des jeweiligen Lernverfahrens mit den verarbeiteten Merkmalen trainiert werden, um Muster in den Daten zu lernen und eine Vorhersagefunktion zu generiern, um neue, nicht gesehene Daten zu klassifizieren. Auch die Klassifikation neuer Daten und die Validierung der Klassifikationsergebnisse mittels Bewertungskriterien kann mit AMUSE durchgeführt werden. Dabei können Bewertungskriterien bezogen

auf ganze Stücke sowie bezogen auf Klassifikationsfenster berechnet werden. Die Objekte, hier Musikstücke, die klassifiziert werden, werden im Folgenden auch *Instanzen* genannt. Instanzen, die der zu klassifizierenden Kategorie angehören, gehören zur *positiven Klasse*. Diejenigen Musikstücke, die der Kategorie nicht angehören, gehören zur *negativen Klasse*. Die entsprechenden Musikstücke werden analog als positiv oder negativ bezeichnet. Um die Klassifikationsergebnisse zu verbessern, können Änderungen am Modell, an den Parametern des Lernverfahrens oder an den zugrundeliegenden Daten vorgenommen werden.

Für das Training und die Validierung der Klassifikation wird die *Ground Truth* verwendet. Diese enthält die tatsächliche Zugehörigkeit von Instanzen zu den Kategorien eines Klassifikationsproblems. Im Kontext der Genre-Klassifikation enthält die „Ground Truth“ für jede im Training oder in der Validierung benutzte Instanz die Information, ob diese dem Genre zugerechnet wird.

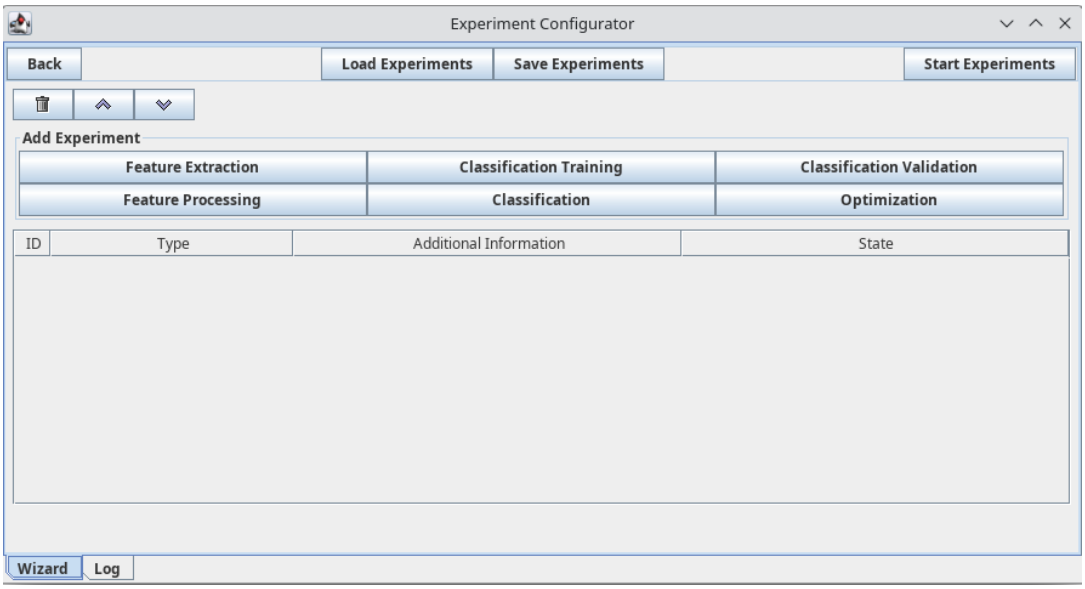

Abbildung 2.2: Grafische Oberfläche des „Experiment Configurators“.

In Abbildung 2.3 ist der grobe Datenfluss beim Ablauf der zuvor beschriebenen Tasks dargestellt. Zum Starten eines Tasks wird der `Scheduler`, entweder durch die grafische Oberfläche oder direkt durch die Kommandozeile, gestartet. Diesem wird eine oder mehrere `TaskConfiguration`s übergeben, welche vom `Scheduler` nacheinander dem richtigen `TaskStarter` übergeben werden. Für jeden AMUSE-Task besteht jeweils eine Klasse, die von `TaskStarter` erbt. Falls beispielsweise eine, dem `Scheduler` übergebene `TaskConfiguration` ein `ExtractionConfiguration`-Objekt ist, wird sie an den `FeatureExtractionStarter` weitergegeben. Die `ExtractionConfiguration` enthält zum einen ein `FeatureTable`-Objekt, welches die zu extrahierenden Merkmale beschreibt und zum anderen ein `FileTable`-Objekt, das die Musikstücke auflistet, aus denen alle Merk-

male der `FeatureTable` extrahiert werden sollen. Für jedes Extraktionstool besteht eine Adapterklasse, die das Interface `ExtractorInterface` implementiert.

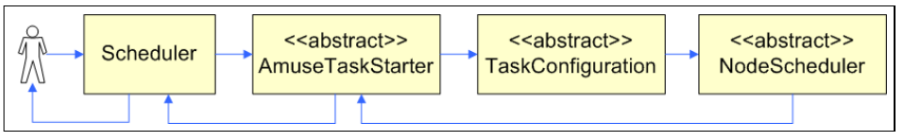

Abbildung 2.3: Datenfluss beim Ablauf von AMUSE-Tasks [28].

Im `FeatureExtractionStarter` wird die `ExtractionConfiguration` ausgelesen und für jedes Extraktionstool, für das die `FeatureTable` mindestens ein Merkmal enthält, eine Datei, meist eine XML-Datei, erstellt, die die Information enthält, welche Merkmale ein Extraktionstool beim späteren Aufruf extrahieren soll. Außerdem wird an dieser Stelle für jede Datei, die sich in der `FileList` befindet, eine neue `ExtractionConfiguration`, die die gesamte Merkmalsliste, aber jeweils nur eine Datei enthält, generiert. Jede `ExtractionConfiguration`, deren `FileTable` nur ein Musikstück enthält, wird dann nacheinander mit einem Aufruf von `ExtractorNodeScheduler` in einem *Job* bearbeitet. Der `ExtractorNodeScheduler` benutzt die Adapterklassen der Extraktionstools, um diese zu starten. Extrahierte Merkmale werden nach Ablauf der Extraktionsaufgabe in der „Feature Database“ gespeichert.

Der Ablauf innerhalb des `ExtractorNodeScheduler` läuft nach dem in Abbildung 2.4 dargestellten Schema ab. Beim Starten eines Extraktions-Jobs, also der Extraktion aller angegebenen Merkmale für ein Musikstück, werden zunächst relevante Einstellungen und Plugins geladen. Da die grafische Oberfläche nur WAVE- und MP3-Eingaben erlaubt, befindet sich die Eingabe nach der eventuellen Konvertierung im WAVE-Format und wird im Anschluss in Bezug auf die Abtastrate bearbeitet. Dann werden die benötigten Extraktoren für alle konfigurierten Merkmale gesucht. Falls mindestens ein Extraktor gefunden und geladen werden konnte, wird die Merkmalsextraktion pro Extraktor gestartet. Falls kein Extraktor gefunden werden konnte, wird ein Fehler ausgegeben. Der `ExtractorNodeScheduler` nutzt für jeden Task einen Ordner, der die Eingabedatei und gegebenenfalls weitere benötige temporäre Dateien enthält, und löscht diese im nächsten Schritt. Die Merkmale, die von den Extraktoren erstellt wurden, werden anschließend in die „Feature Database“ verschoben, in der alle durch AMUSE extrahierten Merkmale gespeichert werden. Der Ordner, den AMUSE als „Feature Database“ benutzt, befindet sich im „AMUSE-Workspace“, ein weiterer Ordner, welcher vom Nutzer selbst beim ersten Start der AMUSE-Oberfläche festgelegt wird. Jedes Merkmal wird dort wie folgt gespeichert:

```
/amuse-workspace/Features/absoluter-pfad-der-eingabe/merkmal_id.arff
```

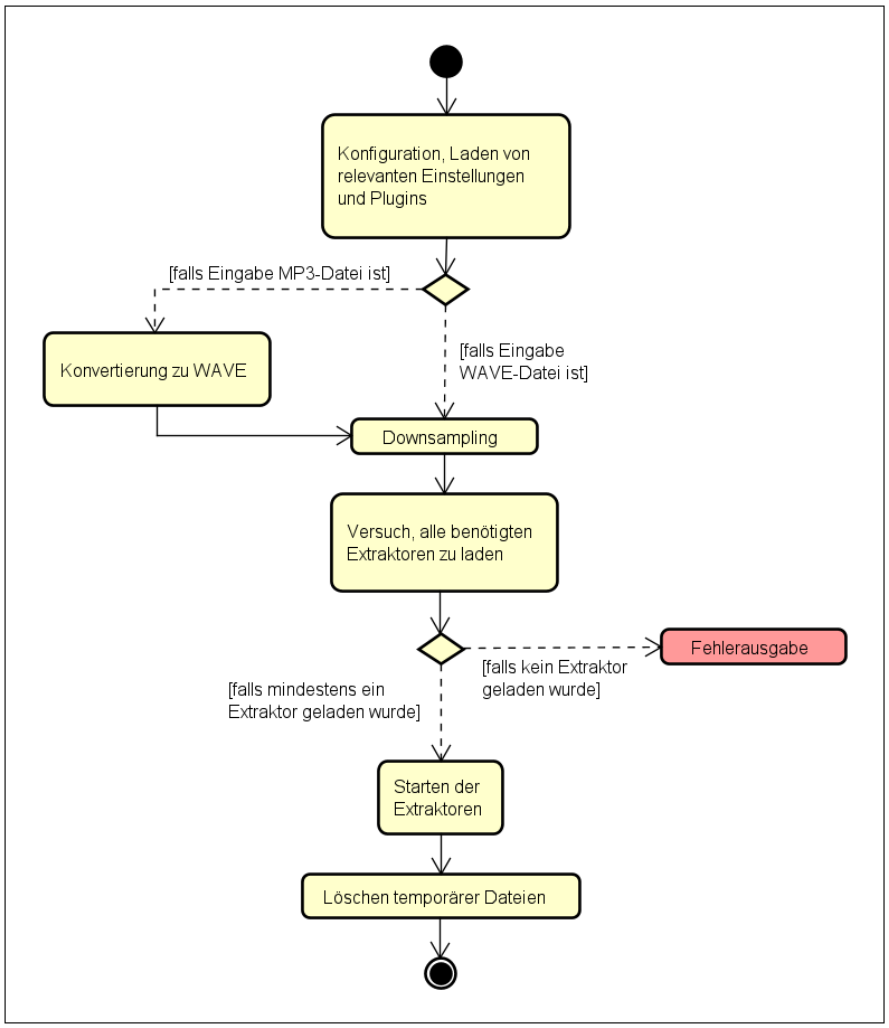

Abbildung 2.4: Schematischer Ablauf von Extraktionsaufgaben im `ExtractorNodeScheduler`.

2.4 MIDI

Die Bezeichnung „MIDI“ ist ein Akronym für „Musical Instrument Digital Interface“. MIDI stellt demnach eine Schnittstelle für digitale Musikinstrumente und andere MIDI-fähige Geräte und Programme dar und definiert das MIDI-Protokoll[2]. Gemäß seines ursprünglichen Zwecks wird dieses benutzt, um die Kommunikation digitaler Instrumente, wie beispielsweise eines Synthesizers, durch den Austausch von Steuerinformationen zu regeln und zu standardisieren. Diese Verbindung besteht traditionell über ein MIDI-Kabel. Die entsprechenden Informationen können auch, beispielsweise zur späteren Wiedergabe, in einer Datei gespeichert und mit geeigneten Programmen verändert werden. Dabei werden die „MIDI-messages“, die im MIDI-Protokoll definiert sind, mittels Zeitstempeln in einen

[2] `https://midi.org/specs` (zuletzt geprüft am 22.08.2024 um 20:01 Uhr)

zeitlichen Zusammenhang gebracht. MIDI-Dateien können außerdem als Basisinformationen für die Berechnung weiterer Merkmale dienen.

Die Struktur von MIDI-Dateien unterscheidet sich maßgeblich von der von Audiodateien, da MIDI-Dateien selbst keine Sounds enthalten. Sie sind strukturell vergleichbar mit einem Notenblatt und enthalten unter anderem Informationen darüber, wann eine oder mehrere Noten gespielt werden, welche Noten dies sind und darauf bezogen, mit welcher Dynamik ("Velocity") sie gespielt werden.

Darüber hinaus können MIDI-Dateien auch weitere Informationen enthalten wie beispielsweise das Tempo, Informationen, die das „Pitchbending“, also die Beugung eines Tons in Richtung eines anderen, betreffen oder definieren, welchen Instrumentensound („Patch“) ein Gerät nutzen soll. Dabei liegt die Klangerzeugung weiterhin aufseiten des empfangenden Geräts und wird je nach Gerät ein unterschiedliches Ergebnis produzieren, abhängig von den verwendeten „Samples“ zur Erzeugung eines Instrumentensounds. Seltener enthalten die Dateien auch Songtexte oder Informationen zur Tonart.

MIDI-Dateien mit der Dateiendung .mid, .midi oder .smf werden als „Standard MIDI Files“ (SMF) bezeichnet und gehören in den überwiegenden Fällen 2 Formaten, SMF0 oder SMF1, an. Dateien des Formats SMF0 arbeiten mit nur einer Spur, die alle „MIDI-Events“ enthält, dagegen können SMF1-Dateien theoretisch unbegrenzt viele Spuren nutzen.

Darüber hinaus gibt es eine weitere Standardisierung, die MIDI-fähige digitale Musikinstrumente betrifft, die sich Merkmale bei der Analyse zur Nutze machen. Das „General-MIDI-System“ definiert neben weiteren Anforderungen an MIDI-fähige Geräte die Zuordnung von „Patch“-Nummern zu Instrumenten. Die „Patch“-Nummer wird von einer „MIDI-message“ verwendet, um den Instrumentensound festzulegen, welcher von einem klangerzeugenden Gerät zur Wiedergabe der entsprechenden Töne verwendet werden soll. Die Standardisierung der laut „General-MIDI-System“ 128 „Patches“ erlaubt es, durch MIDI-Dateien auch das Vorkommen einzelner Instrumente oder Gruppen von Instrumenten zu analysieren. Die „Patches“ sind in acht Instrumentengruppen eingeteilt und decken einen Großteil in westlicher Musik üblicher Instrumente ab.

Insbesondere werden Merkmale aber aus den Tonhöhen und ihrer Beziehung zueinander berechnet. Dafür werden die von MIDI berücksichtigten 128 Tonhöhen, angefangen beim Ton „C-2“ bis hin zum Ton „G8“, die gespielt die Frequenzen 8,176 und 12543,9 Hertz besitzen, verwendet. Die MIDI-Tonhöhen berücksichtigen die enharmonische Verwechselung [24, S.11,12] von Tönen in dem Sinne, dass sie enharmonisch verwechselte Töne, also Töne, die dieselbe Frequenz bezeichnen, aber aufgrund ihres harmonischen Kontextes, wie der Tonart eines Stücks, unterschiedliche Bezeichnungen haben, zu einem Ton zusammenfassen.

Die Ausführungen in diesem Abschnitt beziehen sich auf G. Mazzola et al. [15].

2.5 Tools für symbolische Merkmale

2.5.1 jSymbolic

jSymbolic ist eine von C. McKay entwickelte, 2006 veröffentlichte und mehrmals erweiterte Java-Anwendung [16] zur Berechnung von symbolischen Merkmalen aus MIDI- und MEI-Dateien[3], welche zur Analyse von Musik und für das Training von Modellen genutzt werden können. Obwohl jSymbolic MEI-Dateien akzeptiert, wird die MEI-Eingabe für die überwiegende Mehrheit der Merkmale in das MIDI-Format konvertiert. Dabei können die Merkmale durch die grafische Oberfläche, per Kommandozeile oder durch Nutzung der API-Schnittstelle extrahiert werden.

In jSymbolic sind, seit der Veröffentlichung von jSymbolic2.2, 246 Merkmale enthalten, die sich auf die musikalische Struktur, genauer auf Rythmus, Textur, Harmonie, Melodie, Dynamik und Instrumentierung beziehen oder Tonhöhen-Statistiken berechnen. Damit ist jSymbolic das wahrscheinlich umfangreichste Extraktionstool für Merkmale aus symbolischen Musikdaten.

Die Programmstruktur ist darauf ausgelegt, die Implementierung weiterer Merkmale möglichst zu vereinfachen. Unter den Merkmale befinden sich zum einen allgemeinere Merkmale, wie z.B., das „Basic Pitch Histogram“ und das „Pitch Class Histogram“ sowie Merkmale, die sich auf musikalische Eigenschaften im Speziellen beziehen, wie beispielsweise die Merkmale „Saxophone Prevalence“ oder „Diminished and Augmented Triads“. Dabei kann die Extraktion von Merkmalen entweder über ganze Stücke oder über Extraktionszeitfenster mit optionaler Überlappung geschehen.

Viele der in jSymbolic enthaltenen Merkmale aus dem Bereich Tonhöhen-Statistiken können zum einen, bezogen auf absolute Tonhöhen und zum anderen, bezogen auf Klassen von Tonhöhen, extrahiert werden. Für die Berechnung eines „Pitch Histogramms“ kann beispielsweise entweder das „Basic Pitch Histogram“, bezogen auf absolute Tonhöhen oder das „Pitch Class Histogram“, bezogen auch Tonhöhen-Klassen, berechnet werden. Dabei ist die Zugehörigkeit eines Tons zu einer Ton-Klasse dann gegeben, wenn ein Ton aus der einfachen oder vielfachen Oktavierung aller anderen Töne dieser Tonhöhen-Klasse entsteht. Dabei ist zu beachten, dass die MIDI-Notenrepräsentation Töne, die aus ihrer enharmonischen Verwechslung entstehen, als einen einzelnen Ton behandelt, wie in Abschnitt 2.4 beschrieben. Aus 128 MIDI-Tonhöhen entstehen demnach 12 Tonhöhen-Klassen.

Das „Basic Pitch Histogram“ sowie das „Pitch Class Histogram“ wurden in dieser Form erstmals von G. Tzanetakis und P. Cook definiert [25] und werden darauf basierend in jSymbolic aufgegriffen und angewendet. Das Merkmal „Basic Pitch Histogram“ berechnet einen 128-dimensionalen Merkmalsvektor, der die normalisierten Häufigkeiten

[3] `https://music-encoding.org/resources/schemas.html` (zuletzt geprüft am 22.08.2024 um 18:09 Uhr)

der 128 MIDI-Tonhöhen darstellt. Mit dem Merkmal „Pitch Class Histogram“ werden die normalisierten Häufigkeiten der zuvor beschriebenen Tonhöhen-Klassen berechnet.

Neben diesen beiden Merkmalen berechnet jSymbolic eine Reihe von weiteren Merkmalen aus dem Bereich „Pitch Statistics“, von denen viele auf dem „Basic Pitch Histogram“, beziehungsweise auf dem „Pitch Class Histogram“ basieren. Beispielsweise können verschiedene Merkmale zum Vorkommen von absoluten Tonhöhen oder Tonhöhenklassen, der Tonumfang eines Stückes, der Anteil von Tonhöhen einer Tonhöhenklasse oder eines Bereichs von Tonhöhen, die Variabilität von Tonhöhen oder Merkmale zu bestimmten musikalischen Besonderheiten, wie dem Vorkommen von Vibratos oder Glissandos, berechnet werden.

Außerdem berechnet jSymbolic Merkmale, die sich erstens mit der Beziehung von Tönen im horizontalen, also zeitlichen, und zweitens mit Tönen im vertikalen, also tonalen Zusammenhang, befassen. Unter die Analyse der horizontalen Beziehung von Tönen, also der melodischen Struktur eines Stücks, fallen beispielsweise die Merkmale „Most Common Melodic Interval“ oder „Repeated Notes“. Die vertikale Beziehung von Tönen beschreibt die Struktur und das Vorkommen von Intervallen und Akkorden, beispielsweise mit dem Merkmal „Seventh Chords“, welches den Anteil von Septakkorden bestimmt.

Neben Merkmalen, die Informationen zum Rhythmus eines Stücks berechnen, wie das Merkmal „Variability of Note Durations“, welches die Standardabweichung der Tonlängen bestimmt, können symbolische Daten, anders als Audiodateien Informationen zur Instrumentierung, wie unter anderem in Abschnitt 2.4 beschrieben, beeinhalten. Diesen Umstand macht sich jSymbolic bei der Berechnung von Merkmalen wie „Electric Instrument Prevalence“ oder „Orchestral Strings Prevalence“ zur Nutze.

2.5.2 music21

Im Gegensatz zu jSymbolic verfolgt music21 einen eher multifunktionalen Ansatz [4]. Das Python-Tool enthält eine Reihe von Funktionen und Anwendungsmöglichkeiten, darunter die Komposition und Visualisierung von Musik und eine Vielzahl von Analysewerkzeugen. music21 arbeitet entsprechend der zugrundeliegenden Programmiersprache mit Objekten, die verschiedene Aspekte und Bestandteile von Musik repräsentieren. Diese Objekte können transformiert, visualisiert und analysiert werden. Es können entsprechende Objekte entweder direkt durch music21 oder durch das Parsen von Daten verschiedenster Formate konstruiert werden. Eine zentrale Struktur für die weitere Analyse wie der Berechnung von Merkmalen sind Stream-Objekte, die wiederum Stream-Objekte oder verschiedene Elemente, wie Noten oder Akkorde, enthalten können.

music21 enthält symbolische Merkmale aus potentiell drei Quellen [5]. Das Programm enthält 57 ausgewählte jSymbolic-Merkmale im Paket `music21.features.jSymbolic` sowie 20 eigene Merkmale im Paket `music21.features.native`. Durch die kompakte und

universelle Darstellung von symbolischen Inhalten können eigens kreierte Merkmale in music21, wie auch in jSymbolic, leicht integriert werden.

Von den in music21 enthaltenen Merkmalen beziehet sich ein Großteil auf das Vorkommen von Intervallen. Darunter sind die Merkmale „DiminishedSeventhSimultaneityPrevalence“, „DiminishedTriadSimultaneityPrevalence“, „DominantSeventhSimultaneityPrevalence“, „MajorTriadSimultaneityPrevalence“, „MinorTriadSimultaneityPrevalence“ und „TriadSimultaneityPrevalence“ zu nennen. Außerdem enthält music21 vier Merkmale, „ChordBassMotionFeature“, „LandiniCadence“, „QualityFeature“ und „TonalCertainty“, die grob in den Kontext „harmonische Struktur“ eingeordnet werden können, drei Merkmale aus dem Bereich „Pitch Class Statistics“ und zwei Merkmale, die Informationen zum Rhythmus beziehungsweise zur Tonlänge berechnen. Besonders zu beachten ist das Merkmal „LanguageFeature“, mit dem die Sprache der möglicherweise in den symbolischen Daten enthaltenen Liedtexte identifiziert werden kann. Auch das Merkmal „ComposerPopularity“, welches in music21 enthalten war, aus der aktuellen Version aber entfernt wurde, zeigt die Möglichkeiten auf, die Merkmale aus symbolischen Daten haben können.

Für AMUSE bieten die in music21 integrierten jSymbolic-Merkmale vor allem in dem Sinne einen Mehrwert, dass durch die potentielle Nutzung von music21 als Extraktionstool der entsprechenden jSymbolic-Features die Eingabe nicht nur aus MIDI-Dateien bestehen kann, sondern die in music21 enthaltenen jSymbolic-Merkmale auch aus MusicXML, ABC oder aus einer Reihe von weiteren Formaten, dem „Humdrum format“, „Capella“, „MuseData“, „Noteworthy“, „NoteworthyBinary“, „RomanText“, „TinyNotation“ und „Volpiano“ extrahiert werden könnten.

Durch music21 können die Eingabedateien der genannten Formate nicht nur eingelesen und daraus Merkmale berechnet werden, music21 kann, wegen der großen Zahl der Formate, die es bearbeitet, auch zur Konvertierung genutzt werden. Mögliche Ausgabeformate sind dabei „Braille“, „Lilypond“, „MIDI“, „MusicXML“, „RomanText“, „Scala“, „Vexflow“, „Volpiano“ oder eine Repräsentation in Form einer einfachen Textdatei.

2.5.3 MIDIToolbox

MIDIToolox ist ein weiteres, auf matlab basierendes, im Jahre 2004 veröffentlichtes Tool zur Extraktion von symbolischen Merkmalen [6, 7]. Die ursprünglich von T. Eerola und P. Toiviainen entwickelte und durch E. Large und K. Schutte erweiterte Anwendung ermöglicht die Analyse und Visualisierung von MIDI-Dateien. MIDIToolbox liest diese ein und parst sie zu einer Noten-Matrix oder generiert diese aus Referenzstücken, welche Einträge für die Onset-Time, den MIDI-Channel, die Tonhöhe, „Velocity„ und Spieldauer für jede Note enthält. Diese Noten-Matrix dient als Basis für verschiedene in MIDIToolbox enthaltene Analysewerkzeuge.

Kapitel 3

Erweiterung von AMUSE

AMUSE akzeptiert bislang zwei Arten von Dateien als Eingabe für die Merkmalsextraktion: WAVE-Dateien und MP3-Dateien. Beim Start eines Extraktions-Tasks werden alle MP3-Dateien in WAVE-Dateien umgewandelt, die Samplingrate von WAVE-Dateien sowie umgewandelten MP3-Dateien verringert und, falls es sich um eine Stereo-Datei handelt, also zwei Audiokanäle vorhanden sind und dies in AMUSE so eingestellt wurde, diese in einen Kanal gemischt.

Da zuvor also nur mit Audiodateien eines Formats gearbeitet wurde, muss die Klassenstruktur von AMUSE für die Verwendung mehrerer Modalitäten und Verarbeitung unterschiedlicher Formate an relevanten Stellen entsprechend angepasst werden. Jedes Extraktionstool definiert Anforderungen an die Modalität und das Format von Musikstücken, die bei dem Erstellen von Tasks durch AMUSE-Nutzer und der Verarbeitung berücksichtigt werden sollen.

3.1 Eingaben verschiedener Modalitäten

Um zukünftig die richtige Zuordnung von Eingaben verschiedener Modalitäten und Formate zu Tools zu gewährleisten, wird ein Paket benötigt, dessen Klassen eine Hierarchie von Eingaben definieren. Ein vollständiges Klassendiagramm des Pakets findet sich in Abbildung 3.1.

Objekte der Klassen, die das Interface `Modality` implementieren, werden in den Adapterklassen der Extraktionstools in einem Attribut aufgelistet und definieren, welche Modalitäten und Dateiformate diese als Eingabe akzeptieren. Das hat den Vorteil, dass ein Extraktionstool Merkmale aus mehreren Modalitäten extrahieren kann.

Für jede Modalität, aus der AMUSE Merkmale extrahiert, existiert eine Klasse: aktuell `AudioModality` für Audiodateien sowie `SymbolicModality` für symbolische Daten. Diese implementieren jeweils das Interface `Modality` und damit die dort definierten Me-

thoden `getFormats()`, `matchesRequirements(File file)` und `getModalityEnum()`, wie im Klassendiagramm in der Abbildung 3.1 zu sehen.

Jedes Objekt vom Typ `Modality` definiert Anforderungen, die ein Tool bezüglich dieser Modalität an eine Eingabedatei hat und führt dazu eine Liste `List<Format> formats` mit Formaten, die ein Extraktionstool als Eingabe akzeptiert. Diese kann durch die Methode `getFormats()` abgerufen werden.

Auf `Modality`-Objekte ist außerdem die Methode `matchesRequirements(File file)` anwendbar, welche bei der Merkmalsextraktion zur Überprüfung der richtigen Eingaben benutzt wird. Die übergebene Datei wird mit den Formaten des Objekts hinsichtlich der Dateiendung und diese gegebenfalls mit dem Inhalt der Datei durch die, auf jedes Format anwendbare, Methode `boolean confirmFormat(File file)` abgeglichen. `Format`-Objekte, für deren Format noch keine passende Bibliothek zu dessen Validierung gefunden wurde, geben beim Aufruf dieser Methode stets `true` zurück. `Modality`-Objekte implementieren außerdem die Methode `getModalityEnum()`, die pro Modalität ein spezielles `ModalityEnum`-Objekt zurückgibt.

Das Interface definiert zudem zwei weitere statische Methoden: `getAllFormats()` und `getFormat(File file)`. Beim Aufruf von `getAllFormats()` wird eine Liste mit allen AMUSE bekannten Formaten, also den Enum-Objekten aller Klassen, die das Interface `Format` implementieren, zurückgegeben. Die Methode `getFormat(File file)` prüft unabhängig der Modalität die Dateiendung der übergebenen Datei und gibt das entsprechende Enum-Objekt zurück. Dabei bleibt der tatsächliche Inhalt der Datei unberücksichtigt.

Die Klasse `SymbolicModality` sowie die Klasse `AudioModality` implementieren die statische Methode `getEndings()`, die eine Liste aller möglichen Dateiendungen der jeweiligen Modalität zurückgibt.

Das `Modality`-Interface enthält eine innere Klasse, die die möglichen Modalitäten als Enum-Objekte auflistet und weitere statische Methoden enthält, die sich auf die Modalitäten im Allgemeinen beziehen. Diese Objekte werden dazu benutzt, die Modalität in einfacher und vergleichbarer Form speichern zu können, wenn die zugehörigen Formate keine Relevanz haben. Diese Enum-Objekte werden vor allem von Klassen aus dem Paket `amuse.scheduler.gui` genutzt, um beispielsweise die zugehörige Modalität einer Eingabe-Datei an relevante Klassen weiterzugeben.

Alle `Modality`-Klassen enthalten eine innere Enum-Klasse, die unter anderem die Formate, die mit der jeweiligen Modalität assoziiert sind, auflistet. Dabei sind alle in Tabelle 2.1 aufgelisteten Formate berücksichtigt. Jedes Objekt einer `Modality`-Klasse hält eine Liste dieser Formate, die dem Konstruktor übergeben wird. So können für beliebige Extraktionstools mögliche Eingaben definiert werden. Die Klasse `AudioModality` enthält beispielsweise die Klasse `AudioFormat`, welche die Enum-Objekte `MP3`, `WAVE`, `AIFF`, `AIFC`, `FLAC`, `AU` und `SND` auflistet.

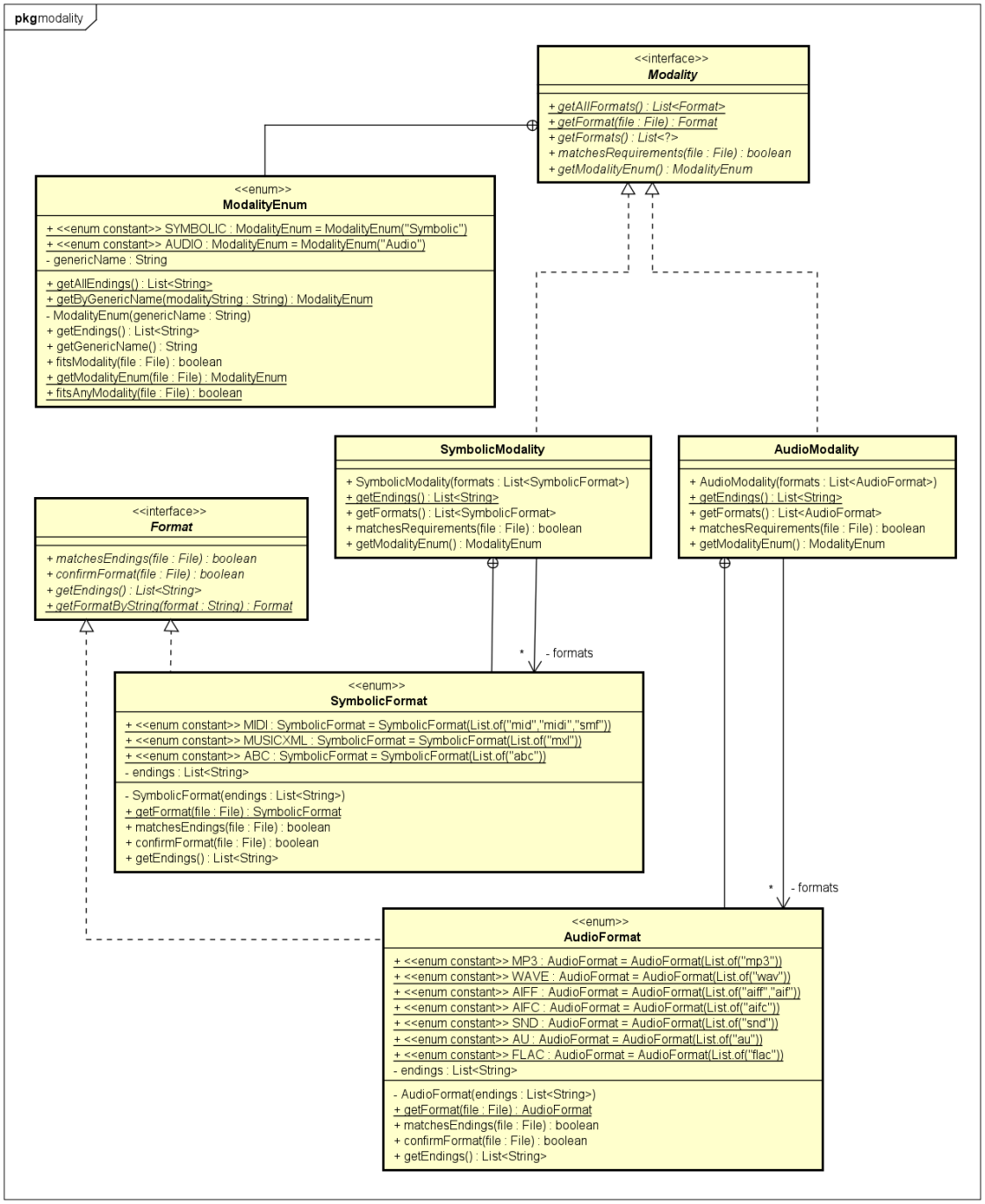

Abbildung 3.1: Klassendiagramm des Pakets `amuse.data.modality` mit allen zugehörigen Klassen.

Die Enum-Klassen, die die modalitätspezifischen Formate definieren, implementieren das Interface `Format` und besitzen somit die Methoden `matchesEndings(File file)`, `getEndings()` und `confirmFormat(File file)`. An dieser Stelle werden auch die, dem Format zugehörigen, Dateiendungen definiert und dem Enum-Objekt in Form einer Liste

von Strings als Attribut angefügt. Die Enum-Klasse `SymbolicFormat` enthält beispielsweise das Objekt

```
MIDI (List.of("mid", "midi"))
```

mit einer Auflistung von möglichen Dateiendungen. Bei der Extraktion von Merkmalen werden Dateien mithilfe der Methode `matchesEndings(File file)`, die von der Methode `matchesRequirements()` benutzt wird, daraufhin getestet, ob sie eine zu den Anforderungen des extrahierenden Tools passende Dateiendung besitzen. Die Methode `confirmFormat(File file)` kann, wie zuvor beschrieben, optional dazu benutzt werden, das tatsächliche Format der Datei zu überprüfen. Dafür werden die Pakete `javax.sound.sampled` und `javax.sound.midi` verwendet.

3.2 Konvertierung

Da alle bislang integrierten Extraktionstools WAVE-Dateien akzeptieren und durch die grafische Oberfläche in AMUSE nur MP3- und WAVE-Dateien hinzugefügt werden können, verläuft die Verarbeitung von Musikstücken bislang wie in Abschnitt 2.3 beschrieben. Da AMUSE Extraktionstools, die aus verschiedenen Modalitäten extrahieren, integrieren soll und diese nur mit bestimmten Formaten arbeiten, soll die Klassenstruktur so geändert werden, dass, neben der Konvertierung von MP3- in WAVE-Dateien, weitere Konvertierungen in AMUSE eingebunden werden können.

Für jede Konvertierung wird eine Klasse erstellt, die das Interface `ConverterInterface` implementiert. Diese Klassen fungieren, falls sie externe Tools zur Konvertierung benutzen, als Adapter-Klassen nach E. Gamma et al. [9, S.171] zwischen dem AMUSE-Framework und den genannten Tools, können aber auch Java-Bibliotheken zur Konvertierung nutzen.

Für die Konvertierung von MP3- in WAVE-Dateien wird die Java-Bibliothek JLayer [1] verwendet und die Samplingrate anschließend mit dem Paket `java.sound.sampled` geändert. `ConverterInterface` definiert die Methoden `convert(File file, File outputFolder)` und `getEnding()`, wobei die Methode `convert(File file, File outputFolder)` die nötigen Schritte zur Konvertierung enthält und `String getEnding()` die Dateiendung des Zielformats zurückgibt. Ein Klassendiagramm des Pakets `amuse.util.converters`, welches das `ConverterInterface` und alle implementierenden Klassen enthält, findet sich in Abbildung 3.2.

Die Klasse `ConverterInterface` enthält zwei statische Methoden, `public static List<Format> conversionsAvailable(Format format)` und `public static ConverterInterface getConversionClass(Format sourceFormat, Format targetFormat)`, die vom `ExtractorNodeScheduler` benutzt werden, um passende Konvertierungen für ein Musikstück zu finden. Beide Methoden lesen eine neue ARFF-Datei mit

[1] `https://github.com/umjammer/jlayer` (zuletzt geprüft am 19.03.2024 um 16:06 Uhr)

dem Dateinamen `conversionTable.arff` aus, welche die Identifikationsnummer, Quellformat, Zielformat und Konversionsklasse für jede in AMUSE integrierte Konversion enthält. Abbildung 3.3 enthält den Inhalt der ARFF-Datei mit allen aktuell in AMUSE verfügbaren Konversionen. Die Methode `public static List<Format> conversionsAvailable(Format format)` lädt die Tabelle und sucht Tabelleneinträge, die das übergebene Format als Quellformat angegeben hat. Die Zielformate dieser Tabelleneinträge werden einer Liste hinzugefügt, welche die Methode zurückgibt.

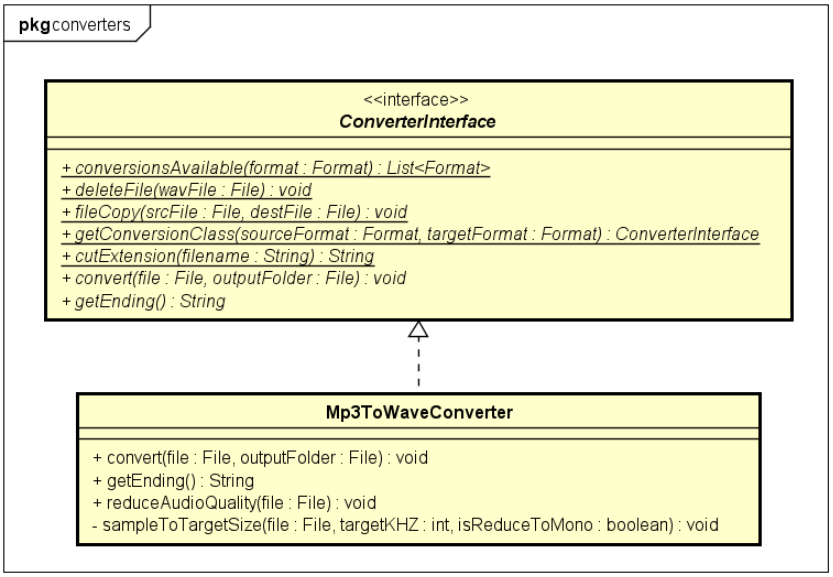

Abbildung 3.2: Klassendiagramm des Pakets `amuse.util.converters`.

Wie in Abschnitt 3.4 beschrieben, kann im `ExtractorNodeScheduler`, falls das Format der Eingabedatei nicht zu einem Merkmal der Extraktionsaufgabe passt, mit dieser Methode geprüft werden, welche Konversionen aus diesem Format verfügbar sind. In Anschluss wird in dieser Liste ein Format gesucht, welches die Anforderungen des Extraktionstools erfüllt. Falls ein passendes Format gefunden wird, kann mit der Methode `public static ConverterInterface getConversionClass(Format sourceFormat, Format targetFormat)` ein Objekt der Klasse, welche das Ursprungsformat in das zum Extraktionstool passende Format umwandelt, erzeugt werden und die Datei anschließend konvertiert werden.

3.3 Einbindung von jSymbolic

Für die Einbindung von jSymbolic wird die Klasse `JSymbolicAdapter` erstellt. Die neue Klasse implementiert, wie alle anderen in AMUSE integrierten Extraktionstools, das Interface `ExtractorInterface`. Dieses definiert alle Methoden, die für die Extraktion von Merkmalen mithilfe dieses Tools benötigt werden (s. Abbildung 3.4).

```
% Conversion table
@RELATION conversions

% Unique Conversion ID
@ATTRIBUTE Id NUMERIC
% Source format
@ATTRIBUTE Source STRING
% Target format
@ATTRIBUTE Target STRING
% Java class which performs conversion
@ATTRIBUTE ConversionClass STRING

@DATA
0, "MP3", "WAVE", "amuse.util.converters.Mp3ToWaveConverter"
```

Abbildung 3.3: ARFF-Tabelle `conversionTable.arff` mit verfügbaren Konversionen.

Seit der Einführung mehrerer Modalitäten definiert `ExtractorInterface` auch die Methode `List<Modality> getModalities()`. jSymbolic hält, wie alle anderen Adapterklassen, ein Attribut vom Typ `List<Modality>`, welches gültige Eingaben für das jeweilige Extraktionstool beschreibt und welches von dieser Methode zurückgegeben wird. Das entsprechende Attribut in der Klasse `JSymbolicAdapter` ist:

```
private static final List<Modality> modalities
  = List.of(new SymbolicModality(List.of(SymbolicFormat.MIDI)));
```

jSymbolic akzeptiert demnach symbolische Daten des Formats MIDI.

Die Methode `setFilenames` soll dem entsprechenden Tool, die durch AMUSE verarbeiteten Eingabedateien, aus denen Merkmale extrahiert werden sollen, bereitstellen. Da jSymbolic mit einer Textdatei aufgerufen wird, die alle nötigen Parameter für die Extraktion enthält, wird der Pfad zur aktuellen Eingabedatei sowie alle weiteren benötigten Informationen beim Aufruf der Methode in die Datei `jSymbolicConfig.txt` geschrieben. Diese befindet sich im jSymbolic-Toolordner `/AMUSE/amuse/tools/jSymbolic`.

Um eine Eingabedatei von AMUSE korrekt an jSymbolic weiterzugeben, muss das Tool eine gültige Konfigurationsdatei übergeben bekommen. Ein Beispiel für eine solche Datei ist in Abbildung 3.6 angegeben. Jede gültige jSymbolic-Konfigurationsdatei enthält einen Textblock, der mit der Zeile `<input_files>` beginnt. Er enthält den Pfad zur Eingabedatei, die sich zur Extraktion im Ordner für diesen Task befindet. Der

`ExtractorNodeScheduler` bearbeitet Aufgaben für jedes Musikstück separat, prinzipiell können aber auch mehrere Eingabedateien an jSymbolic übergeben werden.

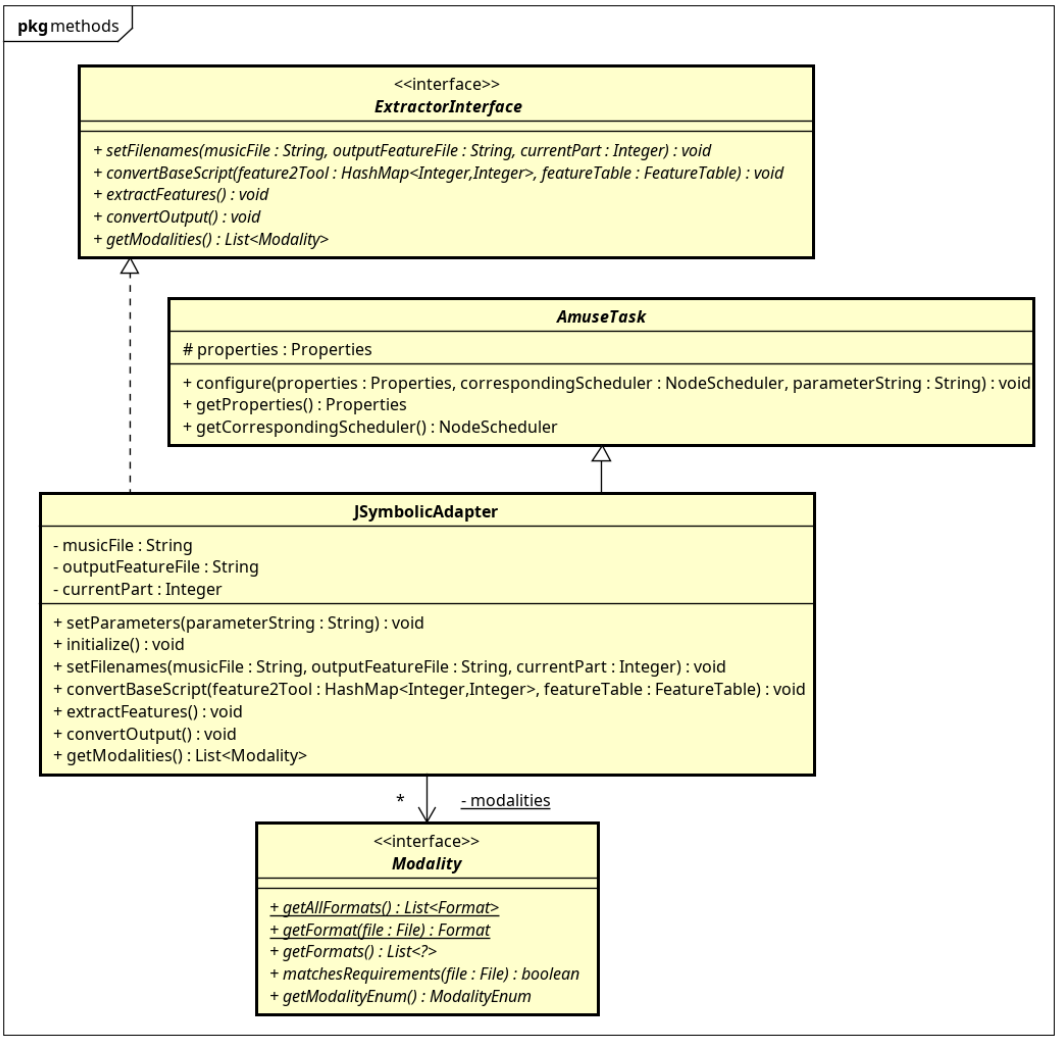

Abbildung 3.4: Klassendiagramm der Klasse `JSymbolicAdapter` mit relevanten Klassen.

Neben diesem Block werden für die erfolgreiche Extraktion weitere Textblöcke, die mit `<features_to_extract>`, `<output_files>` und `<jSymbolic_options>` eingeleitet werden, benötigt. Der Textblock `<features_to_extract>` enthält zeilenweise die Namen aller zu extrahierenden Merkmale.

jSymbolic produziert als Ergebnis der Extraktion zwei Dateien, deren Pfad im Block `<output_files>` festgelegt werden. An dieser Stelle müssen für diese zwei Dateien jeweils ein Dateipfad angegeben werden. Unter `feature_values_save_path` wird jSymbolic der Name und Pfad der Datei, die nach der Extraktion die Merkmalswerte enthalten soll, übergeben. Die Merkmalswerte werden in Form einer XML-Datei gespeichert, jSymbolic ermöglicht aber, durch die Verwendung von `convert_to_arff=true` oder

`convert_to_csv=true` in der Konfigurationsdatei, auch die Umwandlung in die Formate ARFF und CSV. `feature_definitions_save_path` speichert den Pfad und Dateinamen für die XML-Datei, die, analog zu den angegeben Merkmalen in der Konfigurationsdatei, Informationen zu allen extrahierten Merkmalen in Form von `feature`-Nodes enthält.

Das Merkmal „Mean Pitch Class“ würde in dieser XML-Datei durch das in Abbildung 3.5 dargestellte `feature`-Node beschrieben werden.

```
<feature>
  <name>Mean Pitch Class</name>
  <active>false</active>
  <description>
      Mean pitch class value, averaged across all pitched notes
      in the piece. A value of 0 corresponds to a mean pitch
      class of C, and pitches increase chromatically by semitone
      in integer units from there (e.g. a value of 2 would mean
      that D is the mean pitch class). Enharmonic equivalents
      are treated as a single pitch class.
  </description>
  <is_sequential>true</is_sequential>
  <parallel_dimensions>1</parallel_dimensions>
</feature>
```

Abbildung 3.5: Ausschnitt aus der XML-Datei, die `feature`-Nodes analog zu den Merkmalen in der Konfigurationsdatei speichert.

Jedes `feature`-Node enthält den Namen des Merkmals, eine Beschreibung, die Angabe, ob das Merkmal sequenzielle Informationen liefert und die Anzahl der Dimensionen. Neben den Eingabe- und Ausgabedateien werden in der Konfigurationsdatei auch weitere Optionen verarbeitet. Diese sind unabhängig von der Eingabe, sodass jede Konfigurationsdatei für die Berechnung von Merkmalen aus einer Eingabedatei einen Textblock enthält, der mit `<jSymbolic_options>` beginnt und die in Abbildung 3.6 dargestellten Optionen enthält.

Durch `window_size` und `window_overlap` werden die Größe des Extraktionszeitfensters und deren Überschneidung festgelegt. Dabei wird die Größe des Extraktionszeitfensters in Sekunden und die Überschneidung durch den Anteil am Extraktionszeitfenster angegeben. Die Angabe für den `window_overlap` liegt also im Intervall $[0, 1]$.
`save_features_for_each_window` und `save_overall_recording_features` speichern die Angaben, ob die Merkmalswerte pro Extraktionszeitfenster oder bezogen auf das ganze Stück berechnet werden sollen.

Die Methode `extractFeatures` startet jSymbolic mit einem externen Prozess, bei dem die zuvor erstellte Konfigurationsdatei übergeben wird. Nach dem Berechnen der Merkmale wird die Ausgabedatei, die die berechneten Merkmalswerte enthält, gelesen und in weitere ARFF-Dateien übersetzt, sodass diese den AMUSE-Konventionen entsprechen. Dafür wird die Methode `convertOutput` benutzt.

```
<features_to_extract>
Basic Pitch Histogram
Pitch Class Histogram
<input_files>
eingabedatei
<output_files>
feature_values_save_path=feature_values.xml
feature_definitions_save_path=feature_definitions.xml
<jSymbolic_options>
window_size=0.1
window_overlap=0.0
save_features_for_each_window=true
save_overall_recording_features=false
convert_to_arff=true
convert_to_csv=false
```

Abbildung 3.6: Inhalt einer jSymbolic-Konfigurationsdatei `jSymbolicConfig.txt` mit den in AMUSE verwendeten Einstellungen.

Die in AMUSE integrierte jSymbolic-Version wurde zudem leicht modifiziert, sodass sie Merkmalswerte innerhalb des Programms nicht in wissenschaftliche Notation umwandelt, um Fehler auf unterschiedlichen Systemen aufgrund der Zahlendarstellung zu vermeiden.

In Abschnitt A.1 sind alle in AMUSE integrierten jSymbolic-Merkmale nach Merkmalsgruppen aufgelistet.

3.4 Ablauf von Extraktionsaufgaben

Die Verarbeitung von Extraktionsaufgaben läuft bislang innerhalb des `ExtractorNodeScheduler` nach dem in Abbildung 2.4 beschriebenen Schema ab. Wichtig zu beachten ist dabei, dass für jede `TaskConfiguration` mit einer Liste von Musikstücken und einer Liste von zu extrahierenden Merkmalen zuvor je eine neue `TaskConfiguration` erstellt wird, die die vollständige Merkmalsliste, aber dessen `FileList` jeweils nur ein Musikstück enthält. Diese `TaskConfiguration` wird dann im `ExtractorNodeScheduler` in

der Methode `proceedTask(String nodeHome, long jobId, TaskConfiguration extractorConfiguration)` verarbeitet. Um nun Eingaben verschiedener Formate zu verarbeiten und Extraktionstools zuzuordnen, wurde der interne Ablauf dieser Methode, wie in Abbildung 3.7 dargestellt, verändert.

Wie zuvor werden zu Beginn einer Extraktionsaufgabe relevante Einstellungen, wie Pfade zu benötigten Ordnern und installierte Plugins, geladen. Im Anschluss wird versucht, alle benötigten Extraktoren zu laden. Dabei wird zunächst die benötigte Extraktionsklasse aus der `featureTable.arff` ausgelesen und dann versucht, diese zu instanziieren. Falls kein Extraktor geladen wurde, wird an dieser Stelle ein Fehler erzeugt und die Extraktionsaufgabe bricht ab. Falls mindestens ein Extraktor geladen wurde, wird die Eingabedatei, welche in der `TaskConfiguration` angegeben ist, geprüft. Dieser Schritt beinhaltet zunächst, dass nach Extraktoren gesucht wird, auf die die Eingabe nicht passt. Dazu wird eine Liste `List<Integer> extractorsNotFitting` geführt, die nach der Überprüfung aller Tools, mit denen Merkmale extrahiert werden sollen, alle Extractor-IDs enthält, die aus dem Format der Eingabedatei nicht extrahieren können. Um zu testen, ob die Datei den Anforderungen des Tools entspricht, wird auf die `Modality`-Objekte, die in der Adapterklasse des Tools gespeichert sind, die zuvor beschriebene Methode `matchesRequirements(File file)` angewandt.

Da Extraktionstools möglicherweise andere Eingabedateien benötigen, enthält die Klasse `ExtractorNodeScheduler` außerdem eine `HashMap<Integer,String> extractorToFilename`, die für jedes Extraktionstool eine passende Eingabedatei speichert. Die Eingabedateien befinden sich im Ordner des Tasks. Dateien, die zu einem passenden Format konvertiert werden können, werden auch in diesem Ordner gespeichert, sodass `extractorToFilename` nur den Namen der Datei speichern muss, nicht den absoluten Pfad. Falls die Eingabedatei schon auf das Tool passt, wird die Tool-Datei-Kombination in der `HashMap` gespeichert. Falls die Eingabedatei zu einem passenden Format konvertiert werden kann, wird die Konversion durchgeführt und die passende Datei zusammen mit dem Tool gespeichert. Im Anschluss wird die Extraktion für jeden Eintrag der `HashMap` gestartet. Für Tools, für die bis zu diesem Punkt keine Eingabedatei gespeichert wurde, gibt die `HashMap` für dieses `null` zurück und es wird folgende Fehlermeldung im AMUSE-Log erzeugt, ohne dass die Extraktion der Merkmale durch die übrigen Tools abbricht:

```
ERROR amuse.nodes.extractor.ExtractorNodeScheduler -
File does not match tool requirements and could not be converted:
eingabedatei
```

Nachdem alle Extraktoren gestartet wurden, werden die Dateien der generierten Merkmale in der „Feature Database“ gespeichert und temporäre Dateien, wie die möglicherweise mehreren Eingabedateien verschiedener Formate, aus dem Task-Ordner gelöscht.

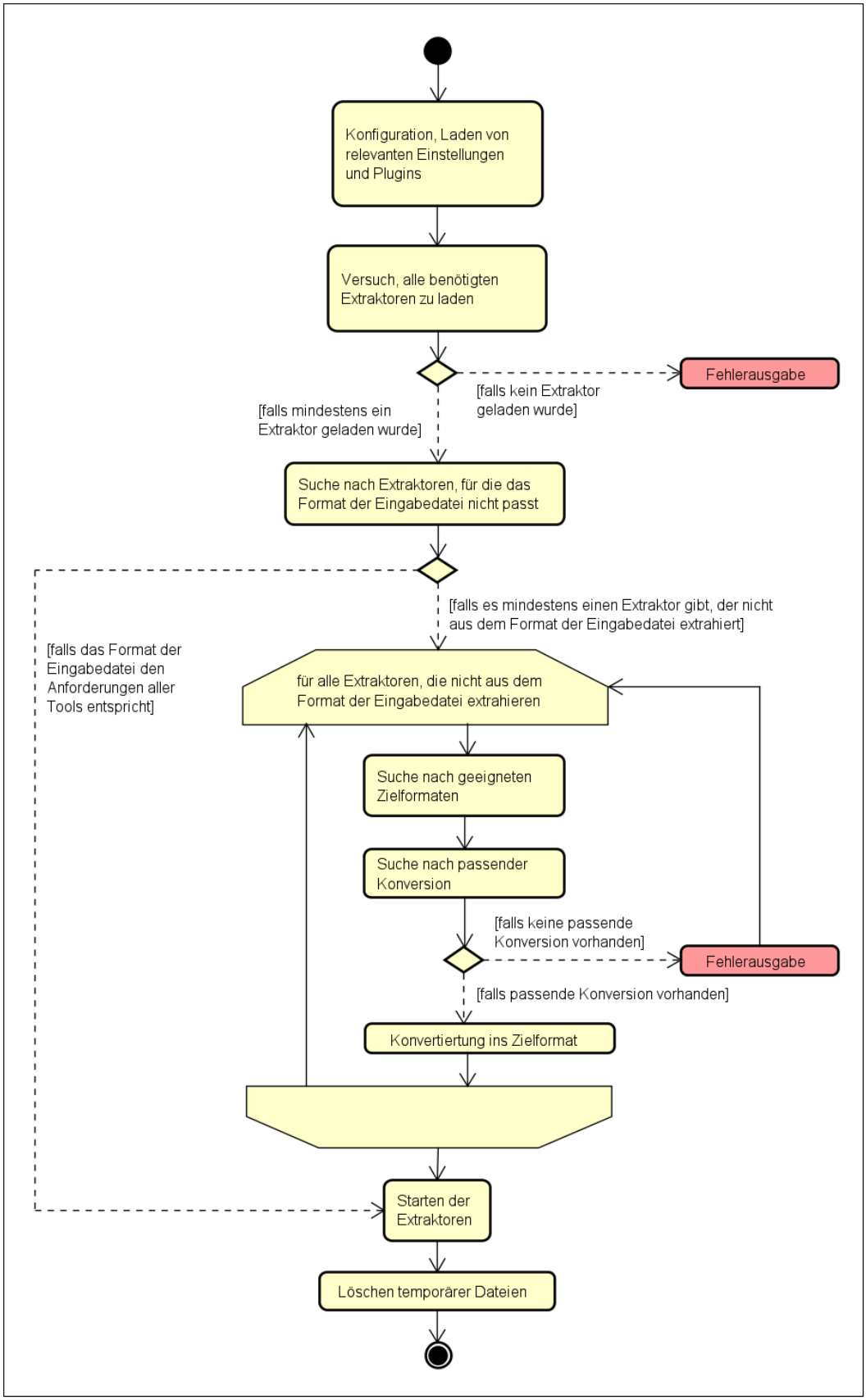

Abbildung 3.7: Schematischer Ablauf einer Extraktionsaufgabe im `ExtractorNodeScheduler` nach der Anpassung.

3.5 Grafische Oberfläche

Wie in Abbildung 3.8 zu sehen, kann im „Extraction Configurator“ eine Auswahl von Musikstücken und Merkmalen getroffen werden. Im linken Panel können verschiedene Buttons bedient werden, mit denen Musikstücke oder Listen von Musikstücken aus dem `FileTree` hinzugefügt oder entfernt werden können. Rechts können Merkmale ausgewählt und damit die `FeatureTable` bearbeitet werden. Es werden initial alle Merkmale aufgelistet, die in der `featureTable.arff` im Ordner `/AMUSE/amuse/config` gespeichert sind. Beim Starten des Extraktions-Tasks wird für jedes ausgewählte Musikstück jedes ausgewählte Merkmal berechnet.

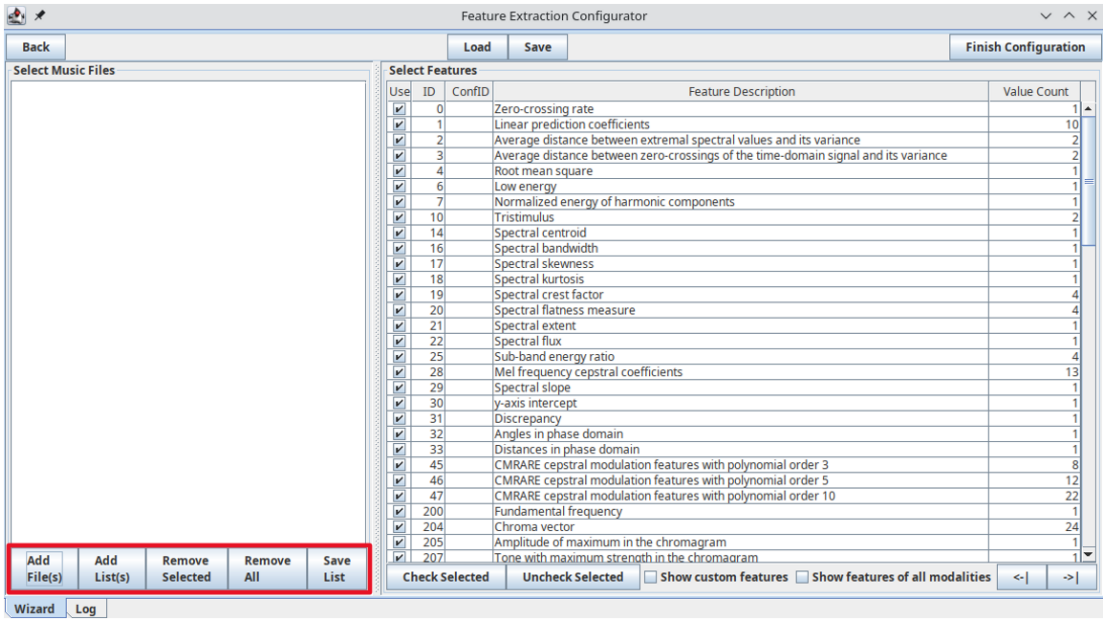

Abbildung 3.8: Grafische Oberfläche des „Extraction Configurators“. Die Buttons, mit denen der `FileTree` bearbeitet werden kann, sind rot umrandet.

Die Änderungen, die im Folgenden beschrieben sind, beziehen sich auf das AMUSE-Paket `amuse.scheduler.gui`.

3.5.1 Anzeige und Auswahl von Merkmalen

Die Merkmale, die im „Extraction Configurator“ angezeigt werden, werden in der Klasse `FeatureTableModel` verwaltet. Diese Klasse hält ein Attribut des Typs `FeatureTable`, welches wiederum das Attribut `List<Feature> features` enthält. Dieses enthält alle Merkmale und deren Eigenschaften, die später im „Extraction Configurator“ angezeigt werden. Um die Tabelle im „Extraction Configurator“ zu modifizieren, wird die `List<Feature> features` entsprechend geändert und Attribute von Merkmalen, die in der `JTable` im „Extraction Configurator“ angezeigt werden sollen, in einem zweidimensionalen Array gespeichert.

Um in der grafischen Oberfläche des „Extraction Configurators“ die Modalitäten von Merkmalen und Musikstücken sinnvoll miteinzubeziehen, wird die Oberfläche so angepasst, dass die `FeatureTable` Merkmale abhängig von den im `FileTree` vorkommenden Modalitäten anzeigt. In der `FeatureTable` werden nur Merkmale angezeigt, die aus allen Modalitäten der sich aktuell im `FileTree` befindenden Dateien extrahiert werden können. An dieser Stelle bleibt das Format der aktuell gespeicherten Dateien unberücksichtigt. Alle Aktionen, die die `FeatureTable` beinflussen, sind in Abbildung 3.9 dargestellt.

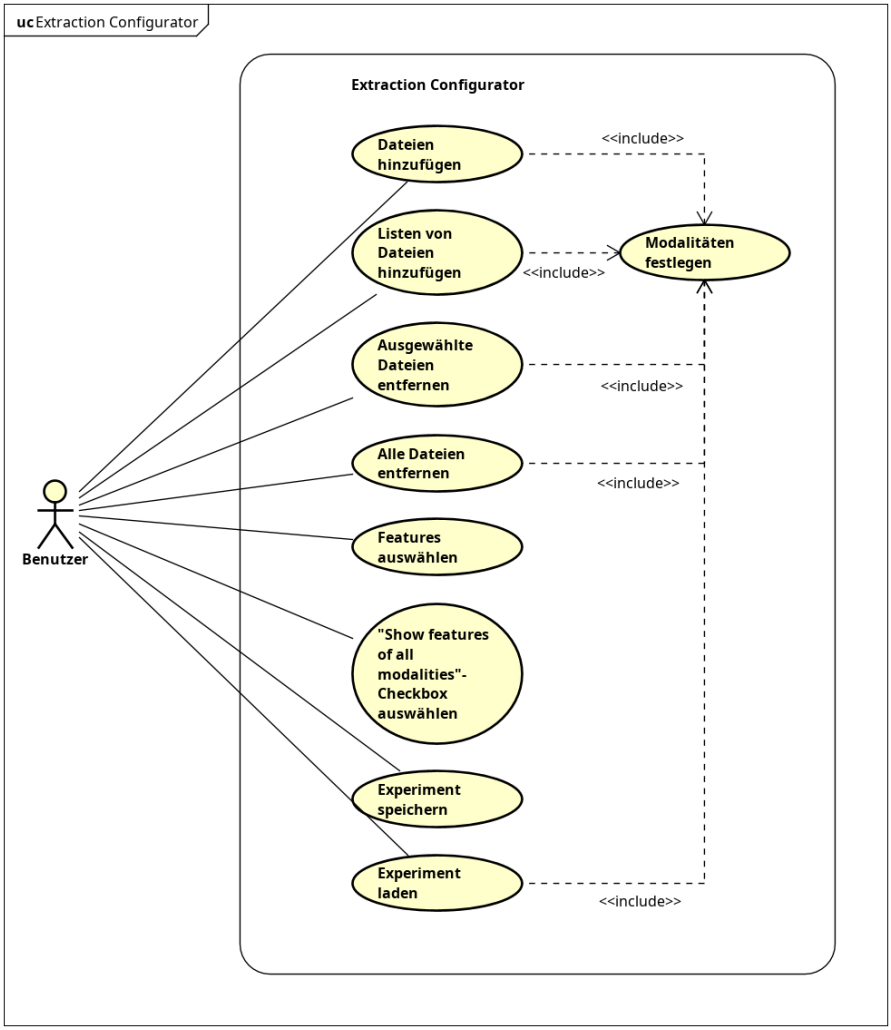

Abbildung 3.9: Aktionen im „Extraction Configurator“, die die angezeigten Merkmale in der Merkmalstabelle beeinflussen.

Um die Menge der Modalitäten zu bestimmen, die ein Extraktionstool verarbeiten kann, wird das Attribut `modalities` in der jeweiligen Adapterklasse des Tools benutzt. Auf diese Weise müssen keine zusätzlichen Informationen in der `featureTable.arff` gespeichert werden und es können mehrere Modalitäten pro Extraktionstool angegeben werden.

Um die `FeatureTable` abhängig von Änderungen im `FileTree` zu modifizieren, muss `FeatureTableModel` von `FileTreeModel` über Änderungen benachrichtigt werden. Dazu wird der Listener `TreeModelModalityListener` benutzt. Das Interface wird von der Klasse `FeatureTableModel` implementiert. Die Methoden, die im Listener-Interface definiert sind, sind in Abbildung 3.10 dargestellt und decken alle bezüglich der Modalität relevanten Veränderungen im `FileTree` ab. Die Beziehung zwischen den Klassen `TreeModelModalityListener`, `FileTreeModel` und `FeatureTableModel` entspricht dem Entwurfsmuster des „Observer“ nach E. Gamma et al. [9, S.287].

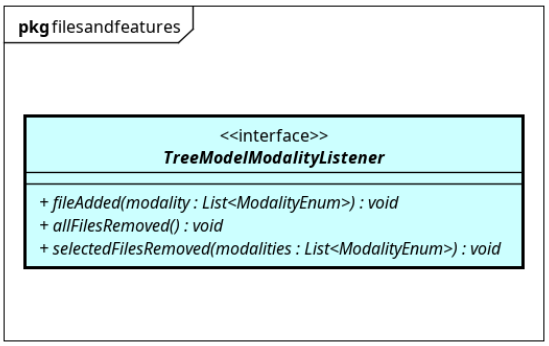

Abbildung 3.10: Klassendiagramm der Klasse `TreeModelModalityListener`.

Wie in Abbildung 3.11 zu sehen, ist die nächste Klasse, die sowohl auf `FileTreeModel` als auch auf `FeatureTableModel` direkten Zugriff hat, die Klasse `FilesAndFeaturesFacade`, in der `FeatureTableModel`, `FeaturTableView`, `FileTreeModel` und `FileTreeView` pro „Configurator“ beziehungsweise `Controller` initial erstellt werden. Die beiden `Controller`, die `FilesAndFeaturesFacade` nutzen, sind `ExtractionController` und `ProcessingController`. Darüber hinaus werden `FileTreeView`, `FileTreeController` und `FileTreeModel` vom „Classification Configurator“ benutzt. Da die Änderungen bezüglich der Modalität nur im „Extraction Configurator“ relevant sind, wird im `ExtractionController` das zugehörige `FeatureTableModel`-Objekt in die Liste `List<TreeModelModalityListener> listeners` des `FileTreeModel` aufgenommen.

Nach dem Hinzufügen einer oder mehrerer Dateien werden alle Listener mit `listener.fileAdded(List<ModalityEnum> modalities)` darüber benachrichtigt. Der Parameter der Methode enthält eine Liste von `ModalityEnum`-Objekten, die die Modalitäten der hinzugefügten Dateien beeinhaltet.

Falls über den Button „Remove Selected“ ausgewählte Dateien aus dem `FileTree` entfernt werden, werden die Listener ebenfalls durch den Aufruf von

`listener.selectedFilesRemoved(modalities)`

benachrichtigt, wobei die übergebene Liste in diesem Fall alle im `FileTree` verbliebenen Modalitäten enthält. An dieser Stelle wird überprüft, ob der `FileTree` nach dem Entfernen ausgewählter Dateien leer ist. Falls alle Dateien über den Button „Remove

All“ oder „Remove Selected“ aus dem `FileTree` entfernt wurden, wird für alle Listener `listener.allFilesRemoved()` aufgerufen.

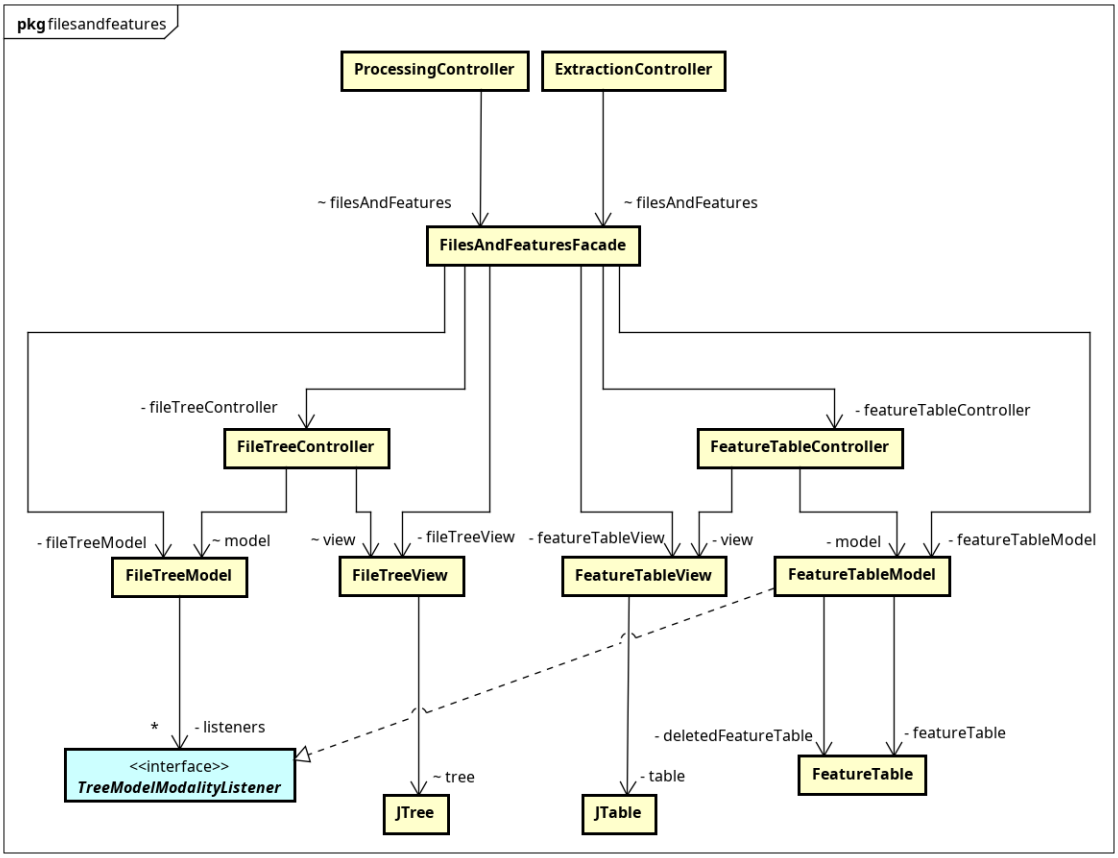

Abbildung 3.11: Struktur des Pakets `amuse.scheduler.gui.filesandfeatures`.

Mit der Checkbox `showFeaturesAllModalitiesCheckbox` können Merkmale angesehen werden, die nicht aus den Modalitäten aller aktuell gespeicherten Dateien im `FileTree` extrahieren können. Sie können nicht zur Extraktion ausgewählt werden, solange sie nicht zu den ausgewählten Musikdateien passen. Da sich die `Feature`-Objekte der Merkmale, die aktuell ausgegraut sind, mit den übrigen `Feature`-Objekten in derselben Liste befinden, wird der Klasse `Feature` ein neues Attribut `private boolean disabled` hinzugefügt.

Wenn dem `FileTree` beispielsweise eine WAVE-Datei hinzugefügt wird, werden alle Merkmale ausgegraut, für die sich in der `modalities`-Liste des zugehörigen Extraktionstools kein `AudioModality`-Objekt befindet. Abbildung 3.12 zeigt die entsprechende `FeatureTable`, falls dem `FileTree` eine WAVE-Datei hinzugefügt wird und über die Checkbox „Show features of all modalities“ ausgewählt ist.

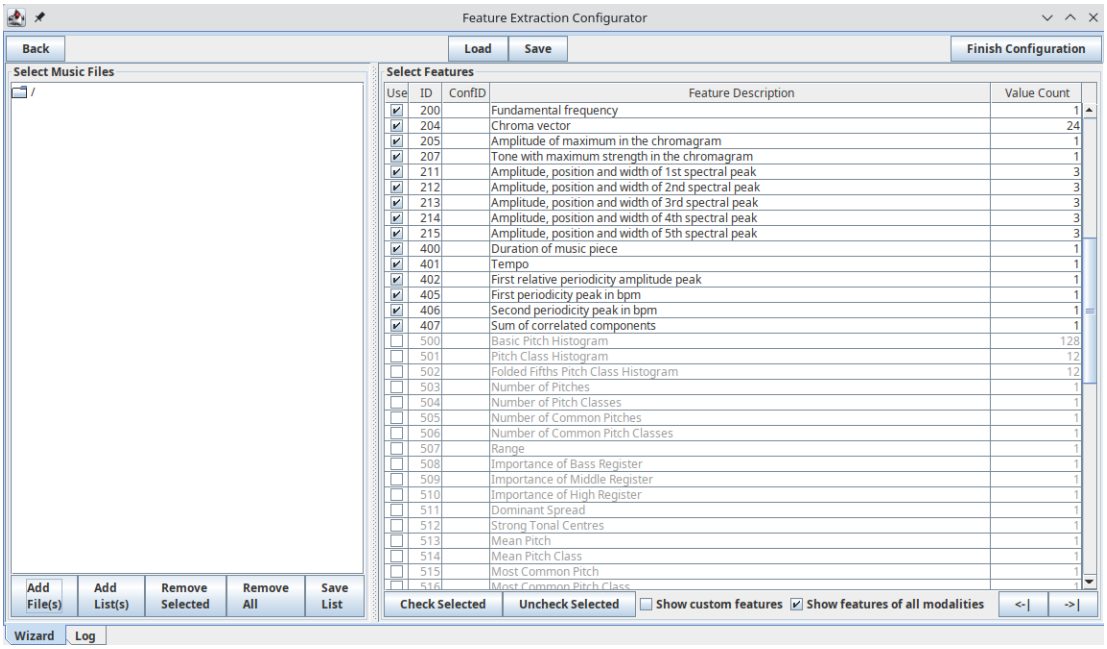

Abbildung 3.12: Grafische Oberfläche des „Extraction Configurators“, wenn sich Audiodateien im `FileTree` befinden und die Checkbox „Show features of all modalities“ ausgewählt ist.

Wie in Abbildung 3.11 zu sehen ist, verwaltet `FeatureTableModel` zwei `FeatureTable`-Objekte, die von den Methoden

`filterFeatureTable(List<ModalityEnum> modalities)`,
`reenableModalityFeatures(List<ModalityEnum> modalities)`,
`showAllModalities()` und `hideAllModalities()`

bearbeitet werden. `showAllModalities()` und `hideAllModalities()` werden aufgerufen, wenn die Checkbox „Show features of all modalities“ benutzt wird. Die `Feature`-Objekte, die nicht zu den Modalitäten der Dateien im `FileTree` passen, werden mit `hideAllModalities()` in die `FeatureTable deletedFeatures` verschoben. Anschließend wird die `JTable` aktualisiert, sodass die entsprechenden Merkmale im „Extraction Configurator“ nicht mehr angezeigt werden. Wenn die Checkbox ausgewählt wird, werden alle `Feature`-Objekte, die sich in der `FeatureTable deletedFeatures` befinden, in die `FeatureTable featureTable` verschoben und das Attribut `disabled` dieser `Feature`-Objekte auf `true` gesetzt, sodass nach dem Aktualisieren der `JTable` diese Merkmale in der `JTable` ausgegraut und nicht auswählbar sind.

Die Methoden `filterFeatureTable(List<ModalityEnum> modalities)` und `reenableModalityFeatures(List<ModalityEnum> modalities)`

laden die `extractorToolTable`, die sich im Ordner `/AMUSE/amuse/config` befindet. Alle `Feature`-Objekte, die mit den im Parameter übergebenen Modalitäten assoziiert sind, werden beim Aufruf von `filterFeatureTable(List<ModalityEnum> modalities)` deaktiviert oder falls die Checkbox „Show features of all modalities“ nicht ausgewählt ist, in

die `FeatureTable deletedFeatures` verschoben. Ähnlich verhält es sich mit der Methode `reenableModalityFeatures(List<ModalityEnum> modalities)`. Alle `Feature`-Objekte, die, je nach Status der Checkbox, aktuell deaktiviert sind oder sich in der `FeatureTable deletedFeatures` befinden, werden danach durchsucht, ob sie mit allen übergebenen Modalitäten assoziiert sind. Diese `Feature`-Objekte werden aktiviert und gegebenenfalls in die `FeatuteTable featureTable` verschoben. So müssen die Merkmale, die sich in der `featureTable.arff` befinden, nur einmal geladen werden.

Nach jeder Änderung der `FeatureTable featureTable` werden die `Feature`-Objekte primär danach, ob sie aktuell aktiviert sind und sekundär nach ihrer Merkmals-Id sortiert und anschließend in die `JTable` überführt.

3.5.2 Auswahl von Musikstücken

In Abbildung 3.8 sind alle Buttons dargestellt, mit denen im „Extraction Configurator“ der `FileTree` bearbeitet werden kann. Wird der Button „Add File(s)“ benutzt, kann im `FileChooser` eine Datei oder ein Ordner mit Dateien ausgewählt und hinzugefügt werden. Dabei wird die Methode `ModalityEnum.getAllEndings()` benutzt, die eine Liste mit allen in `Format`-Klassen vorkommenden Dateiendungen zurückgibt. Falls eine Datei keine der aufgelisteten Dateiendungen besitzt, wird sie weder im `FileChooser` angezeigt, noch dem `FileTree` hinzugefügt, falls sie sich im Ordner befindet, dessen Dateien hinzugefügt werden sollen.

Da sich Dateien verschiedener Modalitäten in den selben Verzeichnissen befinden können oder für die Verarbeitung sogar befinden müssen, braucht es einen Filter, der sich auf die Modalität der Eingabedateien bezieht. Es soll möglich sein, Dateien einer Modalität durch Auswählen der Datei oder eines darüberliegenden Ordners hinzuzufügen, auch wenn Dateien anderer Modalitäten im selben Verzeichnis vorhanden sind. Wie in Abbildung 3.13 zu sehen, wurde pro Modalität ein Filter erstellt. Daneben gibt es die Auswahl, die die Formate aller Modalitäten, die in AMUSE in entsprechenden `Modality`-Klassen angegeben sind, zulässt. Es ist also sowohl möglich, nur Dateien einer Modalität als auch Dateien aller Modalitäten zum Experiment hinzuzufügen.

Da der `FileFilter`, der dem `FileChooser` hinzugefügt wird, jeweils mit einer Modalität umgehen soll, wurde eine neue Klasse mit dem Namen `ModalityFileFilter` erstellt, welche von der Klasse `FileFilter` erbt. Diese hält ein `ModalityEnum`-Attribut, welches die akzeptierten Dateiformate und die im `FileChooser` angezeigte Filterbeschreibung festlegt. Ein Klassendiagramm der Klasse `ModalityFileFilter` findet sich in Abbildung 3.14.

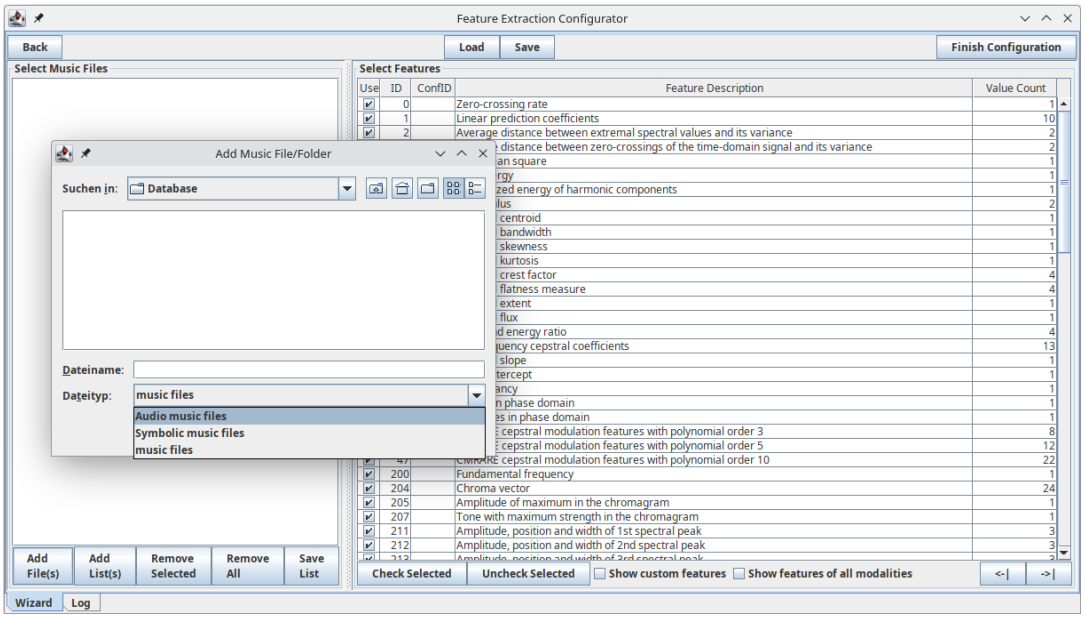

Abbildung 3.13: Grafische Oberfläche des modifizierten `JFileChoosers` im „Extraction Configurator“.

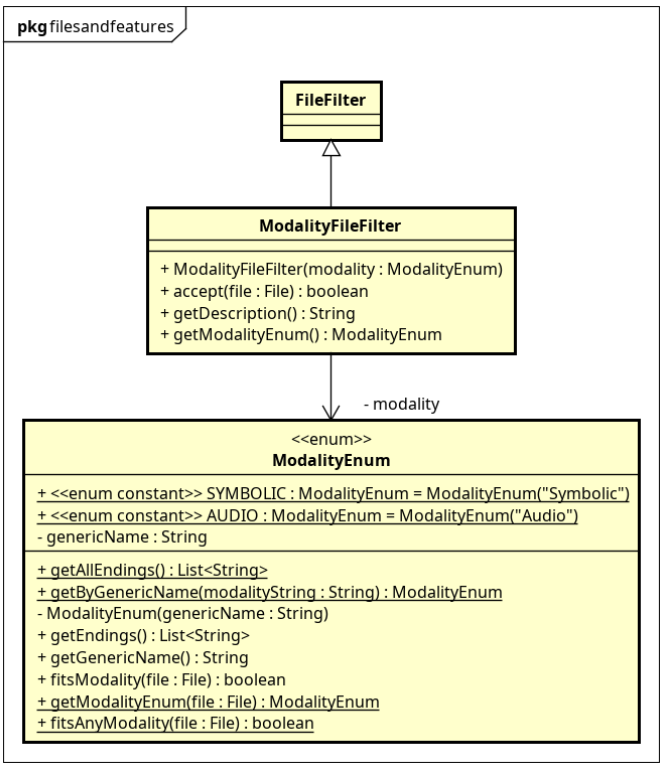

Abbildung 3.14: Klassendiagramm der Klasse `ModalityFileFilter`.

3.6 Verarbeitung von Merkmalen mehrerer Modalitäten

Wenn für ein Musikstück aus verschiedenen Modalitäten Merkmale berechnet und gegebenfalls Modelle trainiert werden sollen, müssen für dieses Stück verschiedene Dateien als Eingabe verarbeitet werden. Da möglicherweise manche Formate nicht von allen Extraktoren verarbeitet werden können, müssen gegebenenfalls mehrere Extraktionsexperimente erstellt und ausgeführt werden, um alle benötigten Merkmale zu extrahieren. Der Pfad jeder daraus resultierenden ARFF-Datei entspricht dem Ordner, in der sich das ursprüngliche Musikstück befindet, relativ zu dem Ordner, der in den Einstellungen als AMUSE-Workspace angegeben ist. Der Dateiname der ARFF-Datei setzt sich aus dem Namen des Musikstücks und der Merkmals-ID zusammen. Die Dateiendung des ursprünglichen Stücks ist demnach nicht im Namen der Merkmals-Datei enthalten. Wenn die Musikstücke der verschiedenen Modalitäten eines Musikstückes sich in einem Ordner befinden und abgesehen von der Dateiendung den gleichen Dateinamen besitzen, können die Merkmals-Dateien in AMUSE gemeinsam verarbeitet werden. Bei der Merkmalsverarbeitung werden alle gewünschten Merkmale ausgewählt und dem `FileTree` für alle verwendeteten Repräsentationen eines Musikstücks, die bei der Merkmalsextraktion verwendet wurden, je eine dieser Dateien hinzugefügt.

Kapitel 4

Studie

Um die Extraktion von Merkmalen auf Basis verschiedener Modalitäten und die Klassifikation von Genres mithilfe dieser Merkmale im Rahmen von AMUSE zu testen, wird eine Studie, gemäß der in Abschnitt 2.3 vorgestellten Klassifikationspipeline, durchgeführt. Dazu werden Merkmale aus Audiodateien und symbolischen Daten extrahiert.

Die Studie zielt darauf ab, den Einfluss von Merkmalen aus symbolischen Daten auf die Klassifikation von Musik in Genre-Kategorien zu beurteilen und vergleicht die Klassifikation von Musikstücken auf Basis von Audiodateien mit Klassifikation derselben Musikstücke unter Verwendung von Merkmalen beider genannten Modalitäten.

4.1 Versuchsaufbau

Dazu wird zunächst getestet, welche symbolischen Merkmale die Klassifikation in ein bestimmtes Genre unterstützen können. Die in jSymbolic integrierten Merkmale wurden im Rahmen der Arbeit von Vatolkin und McKay [27] in einer Merkmalsliste mit musikalischen Kategorien gruppiert, welche an dieser Stelle wiederverwendet werden.

Die Merkmalsgruppen und die Anzahl der im Experiment verwendeten Merkmale jeder dieser Gruppen sind in Tabelle 4.1 aufgelistet. Es wurden alle in jSymbolic2.2 enthaltenen Merkmale der gegebenen Merkmalsgruppen, mit Ausnahme des Merkmals „Duration in Seconds“, verwendet. Dieses Merkmal ist im Kontext dieses Experiments nicht hilfreich, da sich alle verwendeten Daten auf 30-sekündige Auschnitte beziehen.

Für jedes Genre und jede Merkmalsgruppe wird ein Experiment erstellt. Darin wird ein Modell mit den Merkmalen der Merkmalsgruppe trainiert und auf neue Daten angewendet, um den Einfluss dieser Merkmale auf die Klassifikation des entsprechenden Genres zu beurteilen.

Jedes Experiment bearbeitet ein binäres Klassifikationsproblem, also die Zuordnung zu einem, der in Abschnitt 4.3 genannten, Genres. Für das Training der Modelle werden, abgesehen von der Art und Anzahl der Merkmale, gleich verarbeitete Daten benutzt. Die

Größe der Klassifikationsfenster wurde, angelehnt an die Veröffentlichung von I. Vatolkin und G. Rudolph [29], auf 4s mit einer Überlappung von 2s festgelegt. Die Klassifikation erfolgt für alle nachfolgenden Experimente überwacht und mit dem Klassifikationsalgorithmus „Random Forest“ [10], wobei für alle Modelle eine Anzahl von 100 Bäumen verwendet wurde.

Kategorie	**Anzahl verwendeter Merkmale**	**Anzahl Dimensionen**
Pitch	39	61
Melodic	24	24
Chords	32	32
Rhythm	31	31
Tempo	28	28
Instruments	15	15
Texture	24	24
Dynamic	4	4

Tabelle 4.1: Auflistung der verwendeten Merkmalsgruppen mit der Gesamtzahl der in den Experimenten verwendeten Merkmale und der Gesamtzahl der verarbeiteten Dimensionen.

Im Anschluss wird für jedes Genre ein Modell mit den in Abschnitt A.2 aufgelisteten und in AMUSE enthaltenen Audiomerkmalen trainiert und dieses mit den symbolischen Modellen, die den kleinsten balancierten relativen Fehler, bezogen auf ganze Stücke und Klassifikationsfenster, erzeugt haben, verglichen. Für jedes Genre wird ein Modell mit den Merkmalsgruppen, die in den vorherigen Experimenten für die Klassifikation ganzer Musikstücke und Klassifikationsfenster den kleinsten balancierten relativen Fehler erzeugt haben, zusammen mit den genannten Audiomerkmalen trainiert, um zu bewerten, ob und wie stark sich der balancierte relative Fehler verkleinert.

4.2 Datensatz

Für die Experimente muss ein Datensatz gewählt werden. Es sind mehrere multimodale Datensätze verfügbar, darunter der Datensatz „Artificial Audio Multitracks“ (AAM) [20], der „SLAC“-Datensatz oder „The Lakh MIDI Dataset“ (LMD) [21].

Der multimodale Datensatz „AAM“ ist mit 3000 Stücken relativ umfangreich, es ist aber aktuell keine Annotation verfügbar und er hat weiterhin im Kontext dieser Arbeit den Nachteil, dass die Musikstücke durch Algorithmen komponiert und die Audiodateien künstlich mithilfe von Instrumenten-Samples analog zu den symbolischen Dateien generiert sind. Oft sind Audio-Repräsentationen von Musikstücken als Audio-Aufnahme vorhanden und möglicherweise nicht mit absolut präzisen MIDI-Dateien verfügbar. Auch wenn die Berechnung von MIDI-Dateien aus Audioaufnahmen möglich ist, kann diese vielen Fehler-

quellen unterliegen, die die Qualität der MIDI-Datei beeinflussen kann. Neben den Fehlerquellen, die Audiodateien im Vergleich mit symbolischen Daten im Allgemeinen haben, ist es beispielsweise möglich, dass zwei Dateien mit einem Musikstück assoziiert sind, aber die Daten der verschiedenen Modalitäten sich auf unterschiedliche Interpretationen eines Stücks beziehen. Die Verwendung dieses Datensatzes stellt demnach Fehler, die durch die Berechnung von Merkmalen aus Audiodateien im Vergleich zu den entsprechenden symbolischen Daten entstehen können, nicht angemessen dar.

Der Datensatz „SLAC“ enthält dagegen nur Daten zu 250 Musikstücken, für die aber jeweils Dateien aus vier Modalitäten verfügbar sind. Darunter findet sich für jedes Musikstück eine MP3- und eine MIDI-Datei, die aus verschiedenen Quellen stammen und nicht auseinander generiert wurden. Dieser Datensatz hat eine geringe Größe. Da in AMUSE nur zwei der vier Modalitäten verwendet werden können, kann der Datensatz nicht vollumfänglich genutzt werden.

Im Vergleich zu den beiden zuvor genannten Datensätzen enthält „The Lakh MIDI Dataset“ (LMD), mit einer Anzahl von insgesamt 176581 Dateien, Daten zu besonders vielen Musikstücken. Bei diesen Dateien handelt es sich um MIDI-Dateien, von denen 45129 Dateien mit Einträgen des „Million Song Datasets“ [1] abgestimmt sind. Das Audio-MIDI-Matching-Verfahren, welches dafür verwendet wurde, ist an dieser Stelle besonders hervorzuheben, denn die Paare von Audio- und MIDI-Dateien sind mittels lernbasierter Verfahren ausgewählt und wurden zusätzlich mit Hilfe von „Dynamic-Time-Warping“ [21] verglichen und bewertet (s. Abschnitt 4.3). Die 45129 Paare von Audio- und MIDI-Dateien, eine Teilmenge des gesamten Lakh-Datensatzes mit dem Namen „LMD-aligned“, beziehen sich genauer auf 30-sekündige Auschnitte der Musik-Plattform 7digital von im „Million Song Dataset“ enthaltenen Musikstücken.

4.3 Ground Truth

Als „Ground Truth“ wird eine ARFF-Filelist von I. Vatolkin und C. McCay [1] benutzt, die wiederum eine Teilmenge des LMD-aligned-Datensatzes beeinhaltet. Es wurden solche Musikstücke dafür ausgewählt, für die Daten aus allen sechs Modalitäten verfügbar sind. Für jedes der 1575 Musikstücke enthält sie eine Id-Nummer, den relativen Pfad des Stücks, den Namen des Interpreten, den Titel des Songs und die ihm zugeordneten Genres. Die Genre-Annotationen stammen von tagtraum industries incorporated und wurden für die Eintäge des Million Song Datasets erstellt [23], auf die sich die MIDI-Dateien des Lakh-Datensatzes beziehen. Die Annotation enthält Zuordnungen zu den Genres Pop, Rock, RnB, Rap, Country, Pop und Metal, wobei jedem Eintrag der Filelist entweder eines oder mehrere der genannten Genres zugeordnet sind. Um die Eindeutigkeit der Klassifikation zu erhöhen, wurden die Einträge, denen mehr als ein Genre zugeordnet sind, aus der Filelist

[1] `https://zenodo.org/records/5651429` (zuletzt geprüft am 19.03.2024 um 16:07 Uhr)

entfernt. Die Genres Metal und Rap sind ausschließlich in Kombination mit anderen Genres vorhanden und kommen durch die Modifikation nicht mehr vor. In Tabelle 4.2 sind die verbliebenen Genres und ihre Anzahl der Vorkommen in der Annotation aufgelistet. Die Annotation umfasst nach der Modifikation insgesamt 894 Einträge.

Genre	**Anzahl der Einträge**
Rock	192
RnB	163
Electronic	123
Country	264
Pop	152

Tabelle 4.2: Einträge pro Genre

Im Zuge des Matching-Verfahrens wurde für alle Kombinationen von MP3-Ausschnitten und MIDI-Dateien ein Konfidenz-Wert errechnet. Dieser liegt in den Grenzen $[0, 1]$ und gibt eine Aussage darüber, wie wahrscheinlich die Übereinstimmung der beiden Musikstücke ist. Dabei werden alle Paare, für die ein Konfidenzwert $C \geq 0.5$ berechnet wurde, als gültiges Paar definiert. Für jeden MP3-Ausschnitt sind demnach eine oder mehrere MIDI-Dateien vorhanden. Im Zuge dieser Studie wird pro MP3-Auschnitt nur die MIDI-Datei mit dem jeweils höchsten Konfidenzwert benutzt. Eine JSON-Datei, die für jede valide MP3-MIDI-Kombination den Konfidenzwert enthält, wurde der Webseite des Lakh-Datensatzes[2] entnommen. Die Datei mit dem höchsten Konfidenzwert wird jeweils zusammen mit dem MP3-Auschnitt in einen Ordner verschoben und die MIDI-Datei entsprechend diesem benannt, sodass beide, abgesehen von der Dateiendung denselben Dateinamen erhalten. So können die Dateien im nächsten Schritt, wie in Abschnitt 3.6 beschrieben, verarbeitet werden.

Jedes der folgenden Experimente benutzt eine Trainingsmenge und eine Testmenge. Das Genre mit der kleinsten Anzahl an Einträgen ist das Genre „Electronic“. Damit die Trainings- und Testmenge jedes Genres gleich groß ist, werden für jedes Genre soviele Musikstücke verwendet, wie der Kategorie „Electronic“ zugeordnet sind. 80% dieser Menge werden für den Trainingsdatensatz verwendet. Insgesamt dieselbe Menge an Musikstücken werden aus den übrigen Genres benutzt, sodass der Trainingsdatensatz zu gleichen Teilen aus positiven und negativen Stücken besteht und damit balanciert ist. Dabei sind alle 4 verbliebenen Genres zu gleichen Teilen vorhanden. Für den Testdatensatz werden die verbleibenden 20% der positiven Stücke benutzt und analog zum Trainingsdatensatz dieselbe Anzahl von negativen Stücken mit dem selben Verhältnis der verbliebenen Genres verwendet.

[2] `https://colinraffel.com/projects/lmd/` (zuletzt geprüft am 19.03.2024 um 16:08 Uhr)

4.4 Bewertungskriterien

Für die Bewertung der trainierten Modelle werden der *balancierte relative Fehler* und die Kriterien „Precision“ und „Recall“ berechnet. Die genannten Bewertungskriterien berechnen die *Klassifikationsgüte* des trainierten Modells auf Basis der *Konfusionsmatrix*. Diese bezeichnet die Zugehörigkeit von Instanzen zu einer Kategorie im Kontext eines binären Klassifikationsproblems mit folgenden Begriffen [30, S. 340,341]:

- Instanzen, die der positiven Klasse angehören und positiv klassifiziert wurden, werden als *wahre Positive* bezeichnet. *TP* bezeichnet die Anzahl dieser Instanzen.
- Instanzen, die der positiven Klasse angehören und negativ klassifiziert wurden, werden als *false Negative* bezeichnet. *FN* bezeichnet die Anzahl dieser Instanzen.
- Instanzen, die der negativen Klasse angehören und negativ klassifiziert wurden, werden als *wahre Negative* bezeichnet. *TN* bezeichnet die Anzahl dieser Instanzen.
- Instanzen, die der negativen Klasse angehören und positiv klassifiziert wurden, werden als *falsche Positive* bezeichnet. *FN* bezeichnet die Anzahl dieser Instanzen.

Der Bewertungskriterien *balancierter relativer Fehler*, „Precision“ und „Recall“ werden wie folgt berechnet [30, S.344]:

$$m_{BRE} = \frac{1}{2}\left(\frac{FN}{TP+FN} + \frac{FP}{TN+FP}\right) \tag{4.1}$$

$$m_{PREC} = \frac{TP}{TP+FP} \tag{4.2}$$

$$m_{REC} = \frac{TP}{TP+FN} \tag{4.3}$$

4.5 Ergebnisse

Die folgenden Histogramme enthalten die Werte des balancierten relativen Fehlers für die Modelle, die nur mit symbolischen Merkmalen trainiert wurden. Die Experimente sind zum ersten Teil der Studie zu zählen, in dem der Einfluss der in Tabelle 4.1 aufgelisteten Merkmalsgruppen auf jedes der in Tabelle 4.2 aufgelisteten Genres getestet wurde. Die Ergebnisse in Tabellenform inklusive dem „Precision“- und „Recall“-Wert sind in Abschnitt A.3 zu finden.

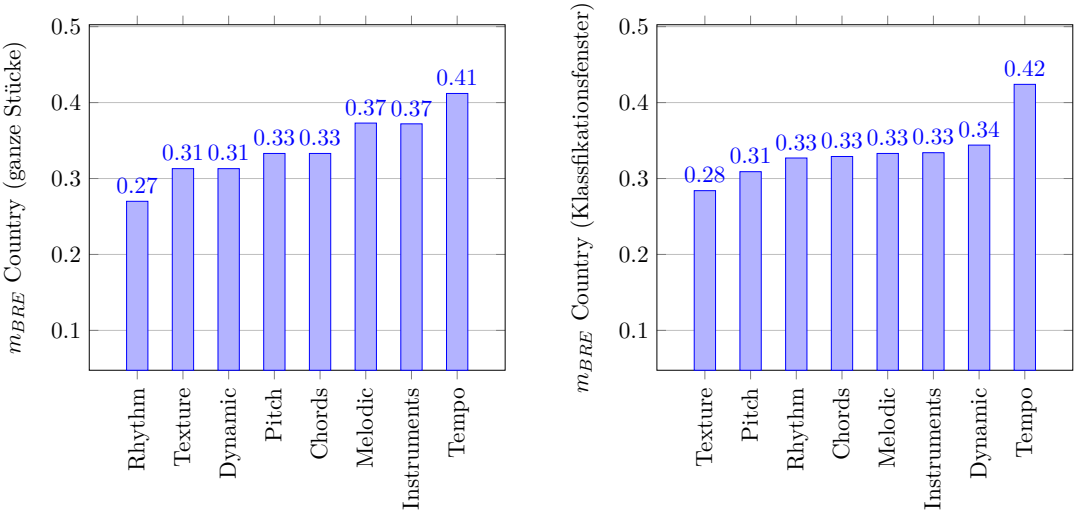

Abbildung 4.1: m_{BRE} der Klassifikation ganzer Stücke (links) und der Klassifikation von Klassifikationsfenstern (rechts) für das Genre „Country“ unter Verwendung der in Tabelle 4.1 aufgelisteten Merkmalsgruppen, aufsteigend sortiert nach m_{BRE}.

Tabelle 4.3 enthält das arithmetische Mittel jeder getesteten Merkmalsgruppe über alle Genres, berechnet sowohl für die Klassifikation von Klassifikationsfenstern als auch für die Klassifikation ganzer Stücke.

Gruppe	$\bar{x}$ des m_{BRE} für ganze Stücke	$\bar{x}$ des m_{BRE} für Klassifikationsfenster
Pitch	0.271	0.280
Melodic	0.310	0.312
Chords	0.298	0.310
Rhythm	0.326	0.348
Tempo	0.354	0.365
Texture	0.291	0.322
Instruments	0.326	0.331
Dynamic	0.260	0.290

Tabelle 4.3: Arithmetisches Mittel des m_{BRE} jeder Merkmalsgruppe über alle Genres, gerundet.

Tabelle 4.4 und Tabelle 4.5 enthalten die Klassifikationsergebnisse von Modellen, die allein mit Audiomerkmalen trainiert wurden.

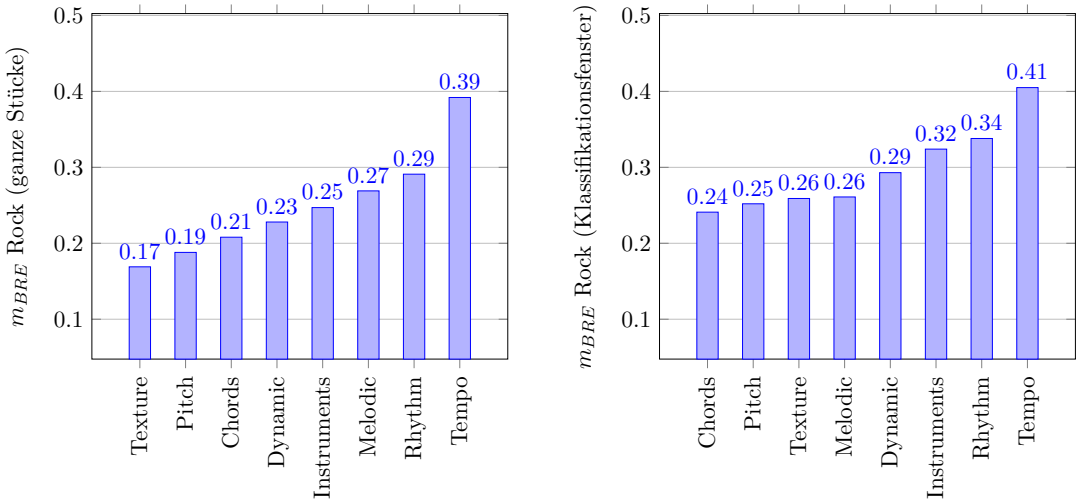

Abbildung 4.2: m_{BRE} der Klassifikation ganzer Stücke (links) und der Klassifikation von Klassifikationsfenstern (rechts) für das Genre „Rock“ unter Verwendung der in Tabelle 4.1 aufgelisteten Merkmalsgruppen, aufsteigend sortiert nach m_{BRE}.

Genre	m_{BRE}	m_{PREC}	m_{REC}
Country	0.248	0.710	0.880
Rock	0.287	0.704	0.760
RnB	0.268	0.700	0.840
Electronic	0.145	0.800	0.960
Pop	0.323	0.765	0.520

Tabelle 4.4: Ergebnisse der Klassifikation von ganzen Musikstücken unter Verwendung der in Abschnitt A.2 aufgelisteten Audiomerkmale, gerundet.

Genre	m_{BRE}	m_{PREC}	m_{REC}
Country	0.22	0.775	0.851
Rock	0.368	0.644	0.681
RnB	0.359	0.649	0.733
Electronic	0.208	0.816	0.837
Pop	0.378	0.679	0.515

Tabelle 4.5: Ergebnisse der Klassifikation von Klassifikationsfenstern unter Verwendung der in Abschnitt A.2 aufgelisteten Audiomerkmale, gerundet.

Für jedes Genre wird im Folgenden das Modell, welches nur mit Audiomerkmalen trainiert wurde, mit dem Modell verglichen, das in den vorherigen Tests auf Basis symbolischer Merkmale den kleinsten balancierten relativen Fehler erzeugt hat. Dabei wird die Klassifikation von ganzen Stücken und die Klassifikation von Klassifikationsfenstern

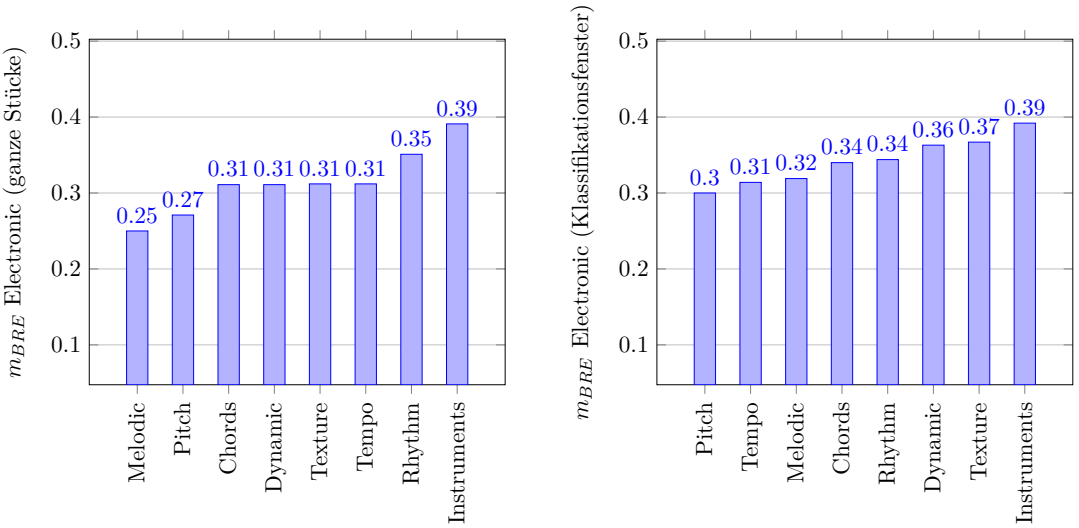

Abbildung 4.3: m_{BRE} der Klassifikation ganzer Stücke (links) und der Klassifikation von Klassifikationsfenstern (rechts) für das Genre „Electronic“ unter Verwendung der in Tabelle 4.1 aufgelisteten Merkmalsgruppen, aufsteigend sortiert nach m_{BRE}.

getrennt betrachtet. Die entsprechenden Diagramme, bezogen auf ganze Musikstücke und Klassifikationsfenster, finden sich in Abbildung 4.6. Die zugehörigen Merkmalsgruppen finden sich, zusammen mit dem relativen Fehler, in Tabelle 4.6 und Tabelle 4.7.

Genre	**Merkmalsgruppe**	m_{BRE}
Country	Rhythm	0.270
Rock	Texture	0.169
RnB	Pitch	0.292
Electronic	Melodic	0.250
Pop	Dynamic	0.139

Tabelle 4.6: Symbolische Merkmalsgruppen, die bei der Klassifizierung von ganzen Stücken, den kleinsten m_{BRE} erzeugt haben, gerundet.

Genre	**Merkmalsgruppe**	m_{BRE}
Country	Texture	0.284
Rock	Chords	0.241
RnB	Pitch	0.276
Electronic	Pitch	0.300
Pop	Dynamic	0.153

Tabelle 4.7: Symbolische Merkmalsgruppen, die bei der Klassifizierung von Klassifikationsfenstern, den kleinsten m_{BRE} erzeugt haben, gerundet.

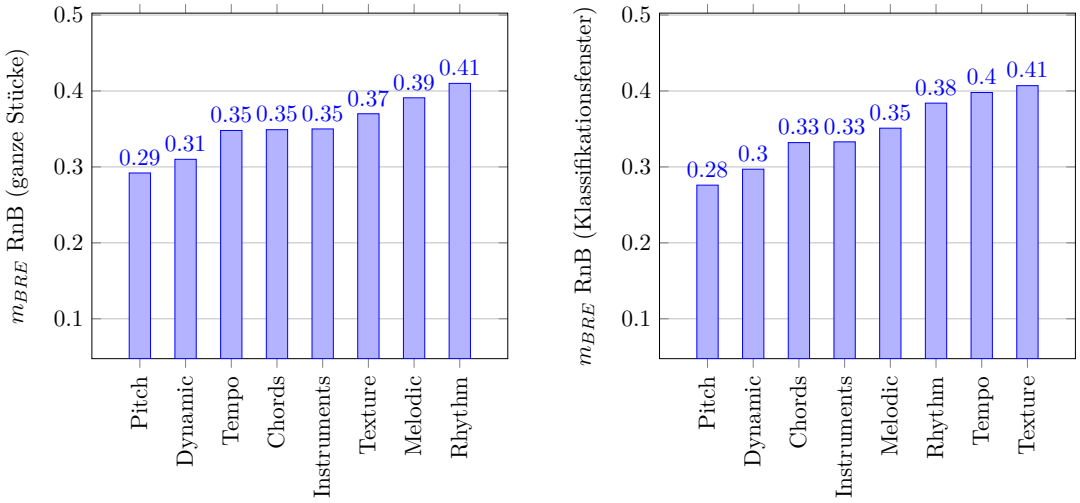

Abbildung 4.4: m_{BRE} der Klassifikation ganzer Stücke (links) und der Klassifikation von Klassifikationsfenstern (rechts) für das Genre „RnB“ unter Verwendung der in Tabelle 4.1 aufgelisteten Merkmalsgruppen, aufsteigend sortiert nach m_{BRE}.

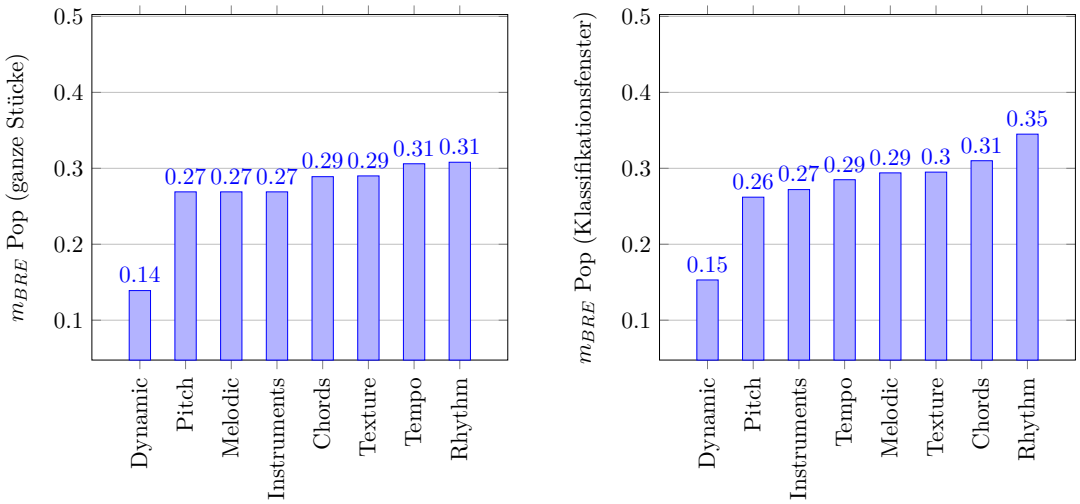

Abbildung 4.5: m_{BRE} der Klassifikation ganzer Stücke (links) und der Klassifikation von Klassifikationsfenstern (rechts) für das Genre „Pop“ unter Verwendung der in Tabelle 4.1 aufgelisteten Merkmalsgruppen, aufsteigend sortiert nach m_{BRE}.

Die Abbildung 4.7 enthält die Ergebnisse der Audiomodelle, der symbolischen Modelle, die bei der Klassifizierung von ganzen Stücken den kleinsten balancierten relativen Fehler erzeugt haben und der Modelle, die mit den Merkmalen beider Gruppen trainiert wurden. Die Abbildung 4.8 enthält analog die Ergebnisse für die genannten Audiomodelle, die symbolischen Modelle, die zuvor für die Klassifikation von Klassifikationsfenstern den kleinsten relativen Fehler erzeugt haben und Modelle, die mit beiden Merkmalsgruppen trainiert wurden.

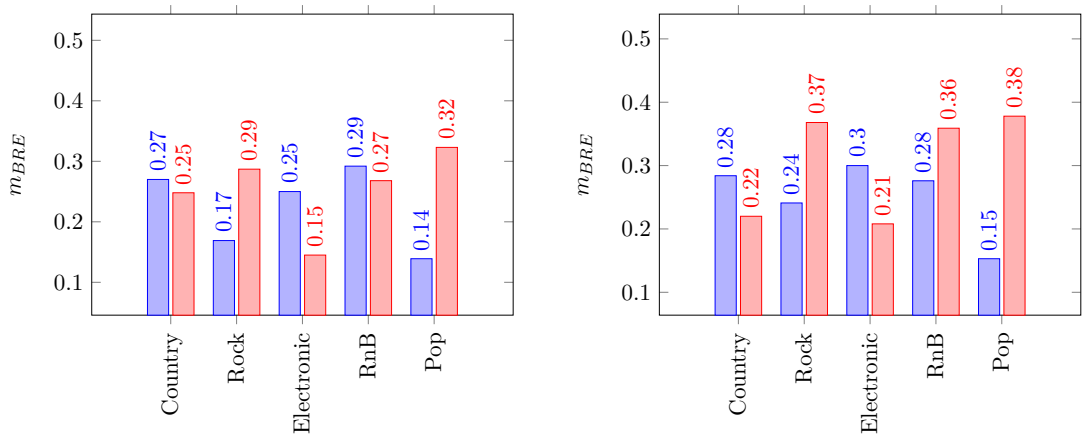

Abbildung 4.6: Vergleich der Audiomodelle (rot) mit jeweils dem Modell, was bei der Klassifikation desselben Genres auf Basis einer symbolischen Merkmalsgruppe den kleinsten balancierten relativen Fehler erzeugt hat (blau), berechnet für ganze Musikstücke (links) und Klassifikationsfenster (rechts), gerundet. Die betreffenden symbolischen Merkmalsgruppen sind in Tabelle 4.6, für ganze Musikstücke, und Tabelle 4.7, für Klassifikationsfenster, abzulesen.

Die symbolischen Merkmalsgruppen, die zuvor den kleinsten Fehler, berechnet für ganze Stücke und Klassifikationsfenster, erzeugt haben (s. Tabelle 4.6 und Tabelle 4.7), wurden zusammen mit den Audiomerkmalen trainiert. Die dabei entstandenen Modelle wurden bezüglich des balancierten relaiven Fehlers verglichen, der an dieser Stelle wieder für ganze Stücke und Klassifikationsfenster berechnet wurde. Das führt dazu, dass möglicherweise für ein Genre mehrere symbolische Merkmalsgruppen getestet werden, falls die Merkmalsgruppe mit dem kleinsten Fehler für ganze Musikstücke sich von der Merkmalsgruppe mit dem kleinsten Fehler für Klassifikationsfenster unterscheidet.

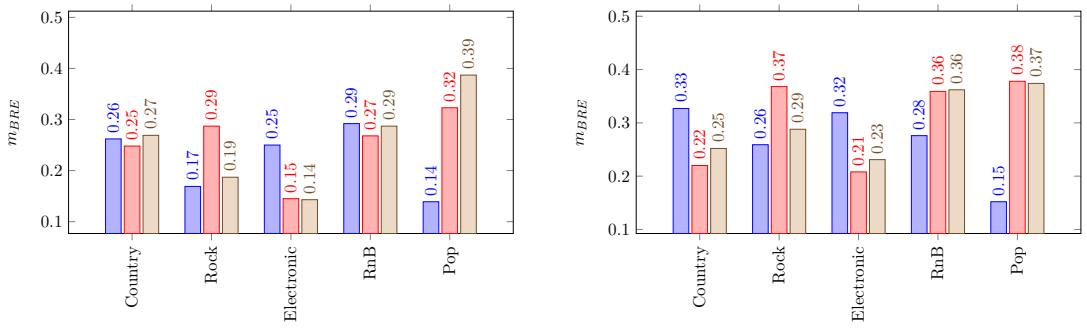

Abbildung 4.7: Vergleich der Audiomodelle (rot) mit jeweils dem Modell, was bei der Klassifikation desselben Genres auf Basis einer symbolischen Merkmalsgruppe den kleinsten balancierten relativen Fehler beim Klassifizieren von ganzen Musikstücken erzeugt hat (blau), mit dem mit beiden Merkmalsgruppen trainierten Modell (braun), berechnet für ganze Musikstücke (links) und Klassifikationsfenster (rechts), gerundet. Die betreffenden symbolischen Merkmalsgruppen für jedes Genre sind in Tabelle 4.6 abzulesen.

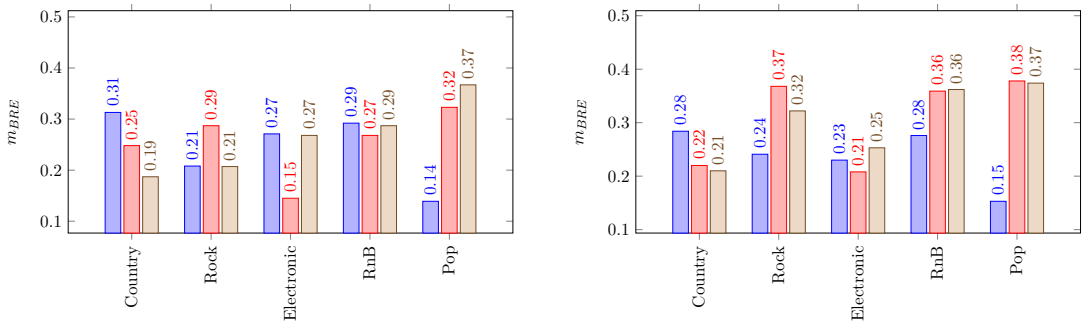

Abbildung 4.8: Vergleich der Audiomodelle (rot) mit jeweils dem Modell, was bei der Klassifikation desselben Genres auf Basis einer symbolischen Merkmalsgruppe den kleinsten balancierten relativen Fehler beim Klassifizieren von Klassifikationsfenstern erzeugt hat (blau), mit dem mit beiden Merkmalsgruppen trainierten Modell (braun), berechnet für ganze Musikstücke (links) und Klassifikationsfenster (rechts), gerundet. Die betreffenden symbolischen Merkmalsgruppen für jedes Genre sind in Tabelle 4.7 abzulesen.

Die Ergebnisse der Modelle, die mit jeweils zwei Modalitäten trainiert wurden, die in Abbildung 4.7 dargestellt sind, finden sich in Tabellenform, bezogen auf ganze Musikstücke, in Tabelle A.11 und, bezogen auf Klassifikationsfenster, in Tabelle A.12. Analog finden sich die Ergebnisse, die in Abbildung 4.8 dargestellt sind, in Tabelle A.13 und Tabelle A.14.

In Tabelle 4.8 und Tabelle 4.9 befinden sich die Modelle, die insgesamt für jedes Genre den kleinsten balancierten relativen Fehler erzeugt haben. Berücksichtigt sind dabei Modelle, die nur mit Audiomerkmalen trainiert wurden, Modelle, die nur mit symbolischen Merkmalen trainiert wurden und Modelle, die mit einer Kombination aus Audiomerkmalen und symbolischen Merkmalen trainiert wurden.

Genre	**Merkmalsgruppen**	m_{BRE}
Country	Audio, Texture	0.187
Rock	Texture	0.169
RnB	Audio	0.268
Electronic	Audio, Melodic	0.143
Pop	Dynamic	0.139

Tabelle 4.8: Modelle, die verglichen mit allen trainierten Modellen (auf Basis von Audiomerkmalen, symbolischen Merkmalen oder einer Kombination der beiden Merkmalsgruppen) bei der Klassifikation von ganzen Musikstücken eines Genres den kleinsten m_{BRE} erzeugt haben, zusammen mit den verwendeten Merkmalsgruppen, gerundet.

Genre	**Merkmalsgruppen**	m_{BRE}
Country	Audio, Texture	0.210
Rock	Chords	0.241
RnB	Pitch	0.276
Electronic	Audio	0.208
Pop	Dynamic	0.153

Tabelle 4.9: Modelle, die verglichen mit allen trainierten Modellen (auf Basis von Audiomerkmalen, symbolische Merkmalen oder einer Kombination der beiden Merkmalsgruppen) bei der Klassifikation von Klassifikationsfenstern eines Genres den kleinsten m_{BRE} erzeugt haben, zusammen mit den verwendeten Merkmalsgruppen, gerundet.

4.6 Interpretation

In Abschnitt 4.5 wurde zuerst der Einfluss von in Gruppen zusammengefassten, symbolischen Merkmalen auf den balancierten relativen Fehler bei der Genreklassifikation der in Tabelle 4.2 genannten Genres getestet. Dabei fällt auf, dass die Klassifikationsgüte der verwendeten Merkmalsgruppen für die unterschiedlichen Genres stark variiert. Für die Klassifikation von ganzen Musikstücken erreicht beispielsweise das Modell, welches mit der Merkmalsgruppe „Rhythm“ trainiert wurde, für das Genre „Country“ den kleinsten balancierten relativen Fehler, für das Genre „RnB“ dagegen den größten. Auch für die Klassifikation von ganzen Stücken und Klassfikationsfenstern unterscheidet sich die Leistung der mit denselben Merkmalsgruppen trainierten Modelle für jede Merkmalsgruppe teilweise. Für die Genres „RnB“ und „Rock“ sind die Modelle, die bezüglich des balancierten relativen Fehlers am besten abschneiden, mit derselben Merkmalsgruppe trainiert. Für die Genres „Country“, „Electronic“ und „Pop“ wurden die am besten abschneidenden Modelle, bezogen auf Klassifikationsfenster und ganze Musikstücke, jeweils mit verschiedenen Merkmalsgruppen trainiert.

Für die Klassifikation eines Genres unterscheidet sich der der relative Fehler der verschiedenen Merkmalsgruppen sowohl bezogen auf Klassfikationsfenster als auch bezogen auf ganze Stücke relativ deutlich. Die Merkmalsgruppe, die pro Genre am häufigsten den kleinsten Fehler generiert, ist für die Klassifikation von Klassifikationsfenstern die Merkmalsgruppe „Pitch“. Diese Merkmalsgruppe enthält, verglichen mit den restlichen Merkmalsgruppen, mit 39 Merkmalen und 61 Dimensionen die meisten Daten. Daraus ist nicht zu schließen, dass viele symbolische Merkmale zu einem kleinen balancierten relativen Fehler führen könnten, da auch die Merkmalsgruppe „Dynamic“ mit einer Anzahl von 4 Merkmalen und 4 Dimensionen beispielsweise für das Genre „Pop“ gute Ergebnisse erzielt. Die Merkmalsgruppen „Rhythm“, „Tempo“ und „Instruments“ schneiden, gemittelt über alle Genres, am schlechtesten, die Merkmalsgruppen „Pitch“ und „Dynamic“ am besten,

ab. Das arithmetische Mittel ist in diesem Zusammenhang nützlich zu betrachten, es sollte dabei aber beachtet werden, dass die Leistung der verschiedenen Merkmalsgruppen stark von dem zu klassifizierenden Genre abhängt.

Beim Vergleich der symbolischen Modelle, die für ein Genre den kleinsten Fehler erzeugt haben, mit den Fehlern der jeweiligen Audiomodelle, zeigt sich, dass die symbolischen Modell nicht immer besser sind als die Audiomodelle. Im Vergleich ganzer Stücke schneiden die Audiomodelle in drei von fünf Fällen besser ab als die für ein Genre jeweils besten symbolischen Modelle. Der balancierte relative Fehler fällt, bezogen auf Klassifikationsfenster, in zwei von fünf Fällen kleiner aus, wenn nur mit Audiomerkmalen trainiert wird. Besonders zu benennen ist dabei, dass der Fehler für die Klassifikation von ganzen Stücken und Klassifikationsfenstern des Genres „Pop“ für das symbolische Modell deutlich kleiner ausfällt als für das entsprechende Audiomodell. Auch für das Genre „Rock“ erzielt die Klassifikation mit symbolischen Modellen insgesamt einen erkennbar kleineren Fehler als die Klassifikation mit Audiomodellen. Daraus ist zu schließen, dass es unter Umständen vorteilhaft sein könnte, bei der Klassifikation bestimmter Genres auf Audiomerkmale ganz zu verzichten. Insgesamt ist festzustellen, dass auch hier die Ergebnisse stark von dem klassifizierten Genre abhängen.

Bei der Beurteilung der Modelle, die mit beiden Modalitäten trainiert und bei denen die symbolische Merkmalsgruppe mit dem kleinsten Fehler, bezogen auf ganze Stücke, verwendet wurde, hat die Kombination der Merkmale beider Modalitäten nur für das Genre „Electronic“ eine Verbesserung erwirkt. Der balancierte relative Fehler dieser Modelle schneidet, bezogen auf die Klassifikation von Klassifikationsfenstern, durchweg schlechter ab als mindestens eines der Modelle, die mit nur einer Modalität trainiert wurden. Interessant ist, dass die Modelle, die mit Audiomerkmalen und den symbolischen Merkmalen, die zuvor nach dem Fehler der Klassifikation von Klassfikationsfenstern ausgewählt wurden, bei der Klassifikation von ganzen Stücken der 2 von 5 Genres die Modelle, die mit Merkmalen einer Modalität trainiert wurden, übertreffen. Der Fehler der Klassifikation ganzer Stücke der Genres „Country“ und „Rock“ konnte durch die Verwendung mehrerer Modalitäten verbessert werden. Daraus ist zu schließen, dass die Verwendung verschiedener Kriterien für die Auswahl symbolischer Merkmale sinnvoll sein könnte.

Insgesamt ist zu beobachten, dass im Kontext dieses Versuchsaufbaus die Verwendung mehrerer Modalitäten für manche Genres bessere Klassifikationsergebnisse erzielt, diese aber für andere Genres besser ausfallen, wenn entweder nur Audiomerkmale oder nur symbolische Merkmale verwendet werden. Die jeweils besten Modelle aller Kategorien sind in Tabelle 4.8 und Tabelle 4.9 aufgelistet.

Kapitel 5

Ausblick

Durch die Anpassungen des AMUSE-Frameworks ist nun die Verarbeitung von symbolischen Merkmalen des Extraktionstools jSymbolic möglich. Die Struktur des Pakets `amuse.data.modality` ermöglicht es insgesamt, weitere Modalitäten und entsprechende Extraktionstools, die eine oder mehrere Modalitäten verarbeiten, zu integrieren. Für die Integration weiterer symbolischer Merkmale kommen unter anderem die in Abschnitt 2.5 aufgelisteten Extraktionstools in Frage. Auch Extraktionstools weiterer Modalitäten, darunter einige in Abschnitt 2.1 aufgezählt, bieten sich in diesem Zusammenhang an. Viele Extraktionstools, wie unter anderem jSymbolic und music21, sind mit dem Ziel entwickelt, eigens entworfene Merkmale in dieses zu integrieren. Auch solche Merkmale könnten zukünftig in AMUSE genutzt werden. Die grafische Oberläche ist funktional für mehrere Modalitäten verwendbar, könnte aber, insbesondere bezüglich der Auswahl von Merkmalen, optimiert werden. Denkbar sind dabei zum Beispiel die Verwendung verschiedener Reiter im „Extraction Experiment“ (s. Abbildung 3.8), mit denen Merkmale verschiedener Modalitäten ausgewählt und angesehen werden können.

Ein weiteres spannendes Aufgabenfeld ist die Integration von Konversionen in AMUSE. Diese können dazu beitragen, weitere Merkmale nutzen zu können, wenn die zugrundeliegenden Daten nicht in dem zum Extraktionstool passenden Format vorhanden sind. Das Paket `amuse.util.converters` ermöglicht die Integration weiterer Konversionen, analog zur Konversion von MP3- in WAVE-Dateien. Insbesondere das Tool music21 bietet die Möglichkeit, verschiedenste Formate ineinander zu überführen (s. Unterabschnitt 2.5.2).

Die in Kapitel 4 beschriebene Studie wirft weiterhin einige Anschlussfragen bezüglich multimodaler Klassifikation auf Basis von Audiomerkmalen und symbolischen Merkmalen auf. Insgesamt ist weiter zu testen, welche symbolischen Merkmale sich für die Klassifikation bestimmter Genres besonders eignen, mit dem Ziel das zugrundeliegende Format im Kontext des Genres und der Wahl der Merkmale möglichst gut zu nutzen. Die Ergebnisse der durchgeführten Studie legen die Verwendung von „Feature Selection“-Verfahren nahe, da auch Modelle, die mit wenigen Merkmalen trainiert wurden, gute Ergebnisse erziehlt

haben. Auch bezüglich der zugrundeliegenden Daten sind Einflüsse auf die Klassifikationsergebnisse möglich, die bei Tests mit weiteren Datensätzen beurteilt werden könnten. Die gewählten Daten sind außerdem auf die fünf verwendeten Genres beschränkt. Da diese das musikalische Spektrum aber nicht in Gänze darstellen können, wäre die Verwendung weiterer Genres angebracht. Auch der Einfluss der Verarbeitung, insbesondere die Größe der Klassifikationsfenster, ist in weiteren Experimenten mit möglicherweise weiteren Fenstergrößen zu beurteilen. Eine weitere Limitation der erstellten Studie ist die Wahl des Klassifikationsalgorithmus. Es wurde für die Studie dieser Arbeit „Random Forest“ für die Klassifikation verwendet. Weitere in AMUSE enthaltene Klassifikationsalgorithmen bieten sich für die multimodale Klassifikation und die Bewertung der Studienergebnisse an. Die Arbeit von Z. Cataltepe et al. [3] gibt außerdem einen Hinweis darauf, dass auch die Wahl von Klassifikationsalgorithmen und deren Parametern Gegenstand der Optimierung in Bezug auf die zugrundeliegende Modalität sein kann.

Anhang A

Anhang

A.1 jSymbolic-Merkmale

Die folgenden Tabellen listen alle in AMUSE integrierten jSymbolic-Merkmale nach, im jSymbolic-Manual[1] definierten, Merkmalsgruppen.

A.1.1 Tonhöhenstatistiken

Tabelle A.1: Auflistung von den in AMUSE integrierten jSymbolic-Features der Merkmalsgruppe „Pitch Statistics“.

AMUSE-Id	Name des Merkmals	Dimensionen
500	Basic Pitch Histogram	128
501	Pitch Class Histogram	12
502	Folded Fifths Pitch Class Histogram	12
503	Number of Pitches	1
504	Number of Pitch Classes	1
505	Number of Common Pitches	1
506	Number of Common Pitch Classes	1
507	Range	1
508	Importance of Bass Register	1
509	Importance of Middle Register	1
510	Importance of High Register	1
511	Dominant Spread	1
512	Strong Tonal Centres	1
513	Mean Pitch	1

[1] `https://jmir.sourceforge.net/manuals/jSymbolic_manual/featureexplanations_files/featureexplanations.html` (zuletzt geprüft am 25.03.2024 um 12:44 Uhr)

514	Mean Pitch Class	1
515	Most Common Pitch	1
516	Most Common Pitch Class	1
517	Prevalence of Most Common Pitch	1
518	Prevalence of Most Common Pitch Class	1
519	Relative Prevalence of Top Pitches	1
520	Relative Prevalence of Top Pitch Classes	1
521	Interval Between Most Prevalent Pitches	1
522	Interval Between Most Prevalent Pitch Classes	1
523	Pitch Variability	1
524	Pitch Class Variability	1
525	Pitch Class Variability After Folding	1
526	Pitch Skewness	1
527	Pitch Class Skewness	1
528	Pitch Class Skewness After Folding	1
529	Pitch Kurtosis	1
530	Pitch Class Kurtosis	1
531	Pitch Class Kurtosis After Folding	1
532	Major or Minor	1
533	First Pitch	1
534	First Pitch Class	1
535	Last Pitch	1
536	Last Pitch Class	1
537	Glissando Prevalence	1
538	Average Range of Glissandos	1
539	Vibrato Prevalence	1
540	Microtone Prevalence	1

A.1.2 Melodische Intervalle

Tabelle A.2: Auflistung von den in AMUSE integrierten jSymbolic-Features der Merkmalsgruppe „Melodic Intervals“.

AMUSE-Id	**Name des Merkmals**	**Dimensionen**
541	Most Common Melodic Interval	1
542	Mean Melodic Interval	1
543	Number of Common Melodic Intervals	1

544	Distance Between Most Prevalent Melodic Intervals	1
545	Prevalence of Most Common Melodic Interval	1
546	Relative Prevalence of Most Common Melodic Intervals	1
547	Amount of Arpeggiation	1
548	Repeated Notes	1
549	Chromatic Motion	1
550	Stepwise Motion	1
551	Melodic Thirds	1
552	Melodic Perfect Fourths	1
553	Melodic Tritones	1
554	Melodic Perfect Fifths	1
555	Melodic Sixths	1
556	Melodic Sevenths	1
557	Melodic Octaves	1
558	Melodic Large Intervals	1
559	Minor Major Melodic Third Ratio	1
560	Melodic Embellishments	1
561	Direction of Melodic Motion	1
562	Average Length of Melodic Arcs	1
563	Average Interval Spanned by Melodic Arcs	1
564	Melodic Pitch Variety	1

A.1.3 Akkorde und Vertikale Intervalle

Tabelle A.3: Auflistung von den in AMUSE integrierten jSymbolic-Features der Merkmalsgruppe „Chords and Vertical Intervals“.

AMUSE-Id	**Name des Merkmals**	**Dimensionen**
565	Average Number of Simultaneous Pitch Classes	1
566	Variability of Number of Simultaneous Pitch Classes	1
567	Average Number of Simultaneous Pitches	1
568	Variability of Number of Simultaneous Pitches	1
569	Most Common Vertical Interval	1
570	Second Most Common Vertical Interval	1

571	Distance Between Two Most Common Vertical Intervals	1
572	Prevalence of Most Common Vertical Interval	1
573	Prevalence of Second Most Common Vertical Interval	1
574	Prevalence Ratio of Two Most Common Vertical Intervals	1
575	Vertical Unisons	1
576	Vertical Minor Seconds	1
577	Vertical Thirds	1
578	Vertical Tritones	1
579	Vertical Perfect Fourths	1
580	Vertical Perfect Fifths	1
581	Vertical Sixths	1
582	Vertical Sevenths	1
583	Vertical Octaves	1
584	Perfect Vertical Intervals	1
585	Vertical Dissonance Ratio	1
586	Vertical Minor Third Prevalence	1
587	Vertical Major Third Prevalence	1
588	Chord Duration	1
589	Partial Chords	1
590	Standard Triads	1
591	Diminished and Augmented Triads	1
592	Dominant Seventh Chords	1
593	Seventh Chords	1
594	Non-Standard Chords	1
595	Complex Chords	1
596	Minor Major Triad Ratio	1

A.1.4 Rhythmus

Tabelle A.4: Auflistung von den in AMUSE integrierten jSymbolic-Features der Merkmalsgruppe „Rhythm“.

AMUSE-Id	**Name des Merkmals**	**Dimensionen**
597	Simple Initial Meter	1

598	Compound Initial Meter	1
599	Complex Initial Meter	1
600	Duple Initial Meter	1
601	Triple Initial Meter	1
602	Quadruple Initial Meter	1
603	Metrical Diversity	1
604	Total Number of Notes	1
605	Note Density per Quarter Note	1
606	Note Density per Quarter Note per Voice	1
607	Note Density per Quarter Note Variability	1
608	Range of Rhythmic Values	1
609	Number of Different Rhythmic Values Present	1
610	Number of Common Rhythmic Values Present	1
611	Prevalence of Very Short Rhythmic Values	1
612	Prevalence of Short Rhythmic Values	1
613	Prevalence of Medium Rhythmic Values	1
614	Prevalence of Long Rhythmic Values	1
615	Prevalence of Very Long Rhythmic Values	1
616	Prevalence of Dotted Notes	1
617	Shortest Rhythmic Value	1
618	Longest Rhythmic Value	1
619	Mean Rhythmic Value	1
620	Most Common Rhythmic Value	1
621	Prevalence of Most Common Rhythmic Value	1
622	Relative Prevalence of Most Common Rhythmic Values	1
623	Difference Between Most Common Rhythmic Values	1
624	Rhythmic Value Variability	1
625	Rhythmic Value Skewness	1
626	Rhythmic Value Kurtosis	1
627	Mean Rhythmic Value Run Length	1
628	Median Rhythmic Value Run Length	1
629	Variability in Rhythmic Value Run Lengths	1
630	Mean Rhythmic Value Offset	1
631	Median Rhythmic Value Offset	1
632	Variability of Rhythmic Value Offsets	1

633	Complete Rests Fraction	1
634	Partial Rests Fraction	1
635	Average Rest Fraction Across Voices	1
636	Longest Complete Rest	1
637	Longest Partial Rest	1
638	Mean Complete Rest Duration	1
639	Mean Partial Rest Duration	1
640	Median Complete Rest Duration	1
641	Median Partial Rest Duration	1
642	Variability of Complete Rest Durations	1
643	Variability of Partial Rest Durations	1
644	Variability Across Voices of Combined Rests	1
645	Number of Strong Rhythmic Pulses - Tempo Standardized	1
646	Number of Moderate Rhythmic Pulses - Tempo Standardized	1
647	Number of Relatively Strong Rhythmic Pulses - Tempo Standardized	1
648	Strongest Rhythmic Pulse - Tempo Standardized	1
649	Second Strongest Rhythmic Pulse - Tempo Standardized	1
650	Harmonicity of Two Strongest Rhythmic Pulses - Tempo Standardized	1
651	Strength of Strongest Rhythmic Pulse - Tempo Standardized	1
652	Strength of Second Strongest Rhythmic Pulse - Tempo Standardized	1
653	Strength Ratio of Two Strongest Rhythmic Pulses - Tempo Standardized	1
654	Combined Strength of Two Strongest Rhythmic Pulses - Tempo Standardized	1
655	Rhythmic Variability - Tempo Standardized	1
656	Rhythmic Looseness - Tempo Standardized	1
657	Polyrhythms - Tempo Standardized	1
658	Initial Tempo	1
659	Mean Tempo	1
660	Tempo Variability	1

661	Duration in Seconds	1
662	Note Density	1
663	Note Density Variability	1
664	Average Time Between Attacks	1
665	Average Time Between Attacks for Each Voice	1
666	Variability of Time Between Attacks	1
667	Average Variability of Time Between Attacks for Each Voice	1
668	Minimum Note Duration	1
669	Maximum Note Duration	1
670	Average Note Duration	1
671	Variability of Note Durations	1
672	Amount of Staccato	1
673	Number of Strong Rhythmic Pulses	1
674	Number of Moderate Rhythmic Pulses	1
675	Number of Relatively Strong Rhythmic Pulses	1
676	Strongest Rhythmic Pulse	1
677	Second Strongest Rhythmic Pulse	1
678	Harmonicity of Two Strongest Rhythmic Pulses	1
679	Strength of Strongest Rhythmic Pulse	1
680	Strength of Second Strongest Rhythmic Pulse	1
681	Strength Ratio of Two Strongest Rhythmic Pulses	1
682	Combined Strength of Two Strongest Rhythmic Pulses	1
683	Rhythmic Variability	1
684	Rhythmic Looseness	1
685	Polyrhythms	1

A.1.5 Instumentierung

Tabelle A.5: Auflistung von den in AMUSE integrierten jSymbolic-Features der Merkmalsgruppe „Instrumentation“.

AMUSE-Id	**Name des Merkmals**	**Dimensionen**
686	Variability of Note Prevalence of Pitched Instruments	1
687	Variability of Note Prevalence of Unpitched Instruments	1

688	Number of Pitched Instruments	1
689	Number of Unpitched Instruments	1
690	Unpitched Percussion Instrument Prevalence	1
691	String Keyboard Prevalence	1
692	Acoustic Guitar Prevalence	1
693	Electric Guitar Prevalence	1
694	Violin Prevalence	1
695	Saxophone Prevalence	1
696	Brass Prevalence	1
697	Woodwinds Prevalence	1
698	Orchestral Strings Prevalence	1
699	String Ensemble Prevalence	1
700	Electric Instrument Prevalence	1

A.1.6 Musikalische Textur

Tabelle A.6: Auflistung von den in AMUSE integrierten jSymbolic-Features der Merkmalsgruppe „Musical Texture“.

AMUSE-Id	**Name des Merkmals**	**Dimensionen**
701	Maximum Number of Independent Voices	1
702	Average Number of Independent Voices	1
703	Variability of Number of Independent Voices	1
704	Voice Equality - Number of Notes	1
705	Voice Equality - Note Duration	1
706	Voice Equality - Dynamics	1
707	Voice Equality - Melodic Leaps	1
708	Voice Equality - Range	1
709	Importance of Loudest Voice	1
710	Relative Range of Loudest Voice	1
711	Relative Range Isolation of Loudest Voice	1
712	Relative Range of Highest Line	1
713	Relative Note Density of Highest Line	1
714	Relative Note Durations of Lowest Line	1
715	Relative Size of Melodic Intervals in Lowest Line	1
716	Voice Overlap	1
717	Voice Separation	1
718	Variability of Voice Separation	1

719	Parallel Motion	1
720	Similar Motion	1
721	Contrary Motion	1
722	Oblique Motion	1
723	Parallel Fifths	1
724	Parallel Octaves	1

A.1.7 Dynamik

Tabelle A.7: Auflistung von den in AMUSE integrierten jSymbolic-Features der Merkmalsgruppe „Dynamic“.

AMUSE-Id	**Name des Merkmals**	**Dimensionen**
725	Dynamic Range	1
726	Variation of Dynamics	1
727	Variation of Dynamics In Each Voice	1
728	Average Note to Note Change in Dynamics	1

A.2 Audiomerkmale

Tabelle A.8: Auflistung von verwendeten, in AMUSE integrierten, Audiomerkmalen.

AMUSE-Id	**Name des Merkmals**	**Dimensionen**
15	Spectral centroid	1
16	Spectral bandwidth	1
17	Spectral skewness	1
18	Spectral kurtosis	1
19	Spectral crest factor	1
20	Spectral flatness measure	4
21	Spectral extent	1
22	Spectral flux	1
28	Mel frequency cepstral coefficients	13
204	Chroma vector	24

A.3 Ergebnisse Tabellenform

Tabelle A.9: Ergebnisse der Klassifikation ganzer Musikstücke mit „Random Forest“ für jede Merkmalsgruppe und jedes Genre, gerundet.

Genre	m_{BRE}	m_{PREC}	m_{REC}
Pitch			
Country	0.333	0.610	1.000
Rock	0.188	0.735	1.000
RnB	0.292	0.641	1.000
Electronic	0.271	0.658	1.000
Pop	0.269	0.676	0.920
Melodic			
Country	0.373	0.590	0.920
Rock	0.269	0.676	0.920
RnB	0.391	0.594	0.760
Electronic	0.250	0.676	1.000
Pop	0.269	0.676	0.920
Chords			
Country	0.333	0.615	0.960
Rock	0.208	0.727	0.960
RnB	0.349	0.633	0.760
Electronic	0.311	0.639	0.920
Pop	0.289	0.667	0.880
Rhythm			
Country	0.270	0.667	0.960
Rock	0.291	0.649	0.960
RnB	0.410	0.586	0.680
Electronic	0.351	0.618	0.840
Pop	0.308	0.667	0.800
Tempo			
Country	0.412	0.576	0.760
Rock	0.392	0.588	0.800
RnB	0.348	0.643	0.720
Electronic	0.312	0.632	0.960
Pop	0.306	0.708	0.680
Texture			
Country	0.313	0.625	1.000
Rock	0.169	0.774	0.960
RnB	0.370	0.613	0.760

Electronic	0.312	0.632	0.960
Pop	0.290	0.657	0.920
Dynamic			
Country	0.313	0.625	1.000
Rock	0.228	0.719	0.920
RnB	0.310	0.647	0.880
Electronic	0.311	0.639	0.920
Pop	0.139	0.908	0.806
Instruments			
Country	0.372	0.600	0.840
Rock	0.247	0.724	0.840
RnB	0.350	0.625	0.800
Electronic	0.391	0.594	0.760
Pop	0.269	0.676	0.920

Tabelle A.10: Ergebnisse der Klassifikation von Klassifikationsfenstern mit „Random Forest“ für jede Merkmalsgruppe und jedes Genre, gerundet.

Genre	m_{BRE}	m_{PREC}	m_{REC}
Pitch			
Country	0.309	0.636	0.926
Rock	0.252	0.715	0.974
RnB	0.276	0.687	0.977
Electronic	0.300	0.684	0.979
Pop	0.262	0.707	0.916
Melodic			
Country	0.333	0.624	0.882
Rock	0.261	0.716	0.934
RnB	0.351	0.666	0.738
Electronic	0.319	0.679	0.912
Pop	0.294	0.695	0.839
Chords			
Country	0.329	0.626	0.889
Rock	0.241	0.733	0.940
RnB	0.332	0.6834	0.751
Electronic	0.340	0.665	0.901
Pop	0.310	0.685	0.811

Rhythm			
Country	0.327	0.621	0.873
Rock	0.338	0.657	0.832
RnB	0.384	0.642	0.666
Electronic	0.344	0.672	0.830
Pop	0.345	0.679	0.695
Tempo			
Country	0.424	0.552	0.761
Rock	0.405	0.609	0.763
RnB	0.398	0.642	0.585
Electronic	0.314	0.686	0.902
Pop	0.285	0.738	0.734
Texture			
Country	0.284	0.651	0.963
Rock	0.259	0.730	0.887
RnB	0.407	0.628	0.630
Electronic	0.367	0.648	0.867
Pop	0.295	0.684	0.882
Dynamic			
Country	0.344	0.619	0.855
Rock	0.293	0.709	0.836
RnB	0.297	0.699	0.834
Electronic	0.363	0.659	0.811
Pop	0.153	0.924	0.757
Instruments			
Country	0.334	0.626	0.862
Rock	0.324	0.690	0.787
RnB	0.333	0.673	0.791
Electronic	0.392	0.641	0.755
Pop	0.272	0.699	0.911

Tabelle A.11: Ergebnisse der Klassifikation mit denen in Abschnitt A.2 aufgelisteten Audiomerkmalen und den, bezogen auf die Klassifikation von ganzen Musikstücken, besten symbolischen Merkmalsgruppen, hinsichtlich dem kleinsten m_{BRE}, von ganzen Musikstücken mit „Random Forest“, gerundet.

Genre	**Symbolische Merkmalsgruppe**	m_{BRE}	m_{PREC}	m_{REC}

Country	Rhythm	0.269	0.676	0.920
Rock	Texture	0.187	0.750	0.960
RnB	Pitch	0.287	0.704	0.760
Electronic	Melodic	0.143	0.846	0.880
Pop	Dynamic	0.387	0.636	0.560

Tabelle A.12: Ergebnisse der Klassifikation mit denen in Abschnitt A.2 aufgelisteten Audiomerkmalen und den, bezogen auf die Klassifikation von ganzen Musikstücken, besten symbolischen Merkmalsgruppen, hinsichtlich dem kleinsten m_{BRE}, von Klassifikationsfenstern mit „Random Forest“, gerundet.

Genre	**Symbolische Merkmalsgruppe**	m_{BRE}	m_{PREC}	m_{REC}
Country	Rhythm	0.252	0.725	0.897
Rock	Texture	0.288	0.687	0.856
RnB	Pitch	0.362	0.650	0.711
Electronic	Melodic	0.231	0.799	0.811
Pop	Dynamic	0.374	0.668	0.558

Tabelle A.13: Ergebnisse der Klassifikation mit denen in Abschnitt A.2 aufgelisteten Audiomerkmalen und den, bezogen auf die Klassifikation von Klassifikationsfenstern, besten symbolischen Merkmalsgruppen, hinsichtlich dem kleinsten m_{BRE}, von ganzen Musikstücken mit „Random Forest“, gerundet.

Genre	**Symbolische Merkmalsgruppe**	m_{BRE}	m_{PREC}	m_{REC}
Country	Texture	0.187	0.750	0.960
Rock	Chords	0.207	0.742	0.920
RnB	Pitch	0.287	0.704	0.760
Electronic	Pitch	0.268	0.700	0.84
Pop	Dynamic	0.387	0.636	0.560

Tabelle A.14: Ergebnisse der Klassifikation mit denen in Abschnitt A.2 aufgelisteten Audiomerkmalen und den, bezogen auf die Klassifikation von Klassifikationsfenstern, besten symbolischen Merkmalsgruppen, hinsichtlich dem kleinsten m_{BRE}, von Klassifikationsfenstern mit „Random Forest“, gerundet.

Genre	**Symbolische Merkmalsgruppe**	m_{BRE}	m_{PREC}	m_{REC}
Country	Texture	0.210	0.759	0.925
Rock	Chords	0.322	0.660	0.828
RnB	Pitch	0.362	0.650	0.711

Electronic	Pitch	0.253	0.769	0.826
Pop	Dynamic	0.374	0.668	0.558

Abbildungsverzeichnis

Algorithmenverzeichnis

Literaturverzeichnis

[1] Bertin-Mahieux, T., D. P. W. Ellis, B. Whitman und P. Lamere: *The million song dataset.* In: *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, Seiten 591–596, 2011.

[2] Burgoyne, J. A., I. Fujinaga und J. S. Downie: *Music information retrieval.* In: *A New Companion to Digital Humanities*, Seiten 213–228. Wiley, 2015.

[3] Cataltepe, Z., Y. Yaslan und A. Sonmez: *Music genre classification using MIDI and audio features.* EURASIP Journal on Advances in Signal Processing, 2007:1–8, 2007.

[4] Cuthbert, M. S. und C. Ariza: *music21: A toolkit for computer-aided musicology and symbolic music data.* In: *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, Seiten 637–642, 2010.

[5] Cuthbert, M. S., C. Ariza und L. Friedland: *Feature Extraction and Machine Learning on Symbolic Music using the music21 Toolkit.* In: *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, Seiten 387–392, 2011.

[6] Eerola, T. und P. Toiviainen: *https://github.com/miditoolbox/*. zuletzt abgerufen am 21.03.2024 um 10:37 Uhr.

[7] Eerola, T. und P. Toiviainen: *MIR In Matlab: The MIDI Toolbox.* In: *Proceedings of the 5th International Society for Music Information Retrieval Conference (ISMIR)*, Seiten 22–27, 2004.

[8] Fell, M. und C. Sporleder: *Lyrics-based analysis and classification of music.* In: *Proceedings of the 25th International Conference on Computational Linguistics (COLING)*, Seiten 620–631, 2014.

[9] Gamma, E., R. Helm, R. Johnson und J. Vlissides: *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software.* Pearson Deutschland GmbH, 2004.

[10] Ho, T. K.: *Random decision forests.* In: *Proceedings of 3rd international conference on document analysis and recognition*, Seiten 278–282, 1995.

[11] Kitahara, T.: *Mid-level representations of musical audio signals for music information retrieval.* In: *Advances in Music Information Retrieval*, Seiten 65–91. Springer, 2010.

[12] Lamere, P.: *Social tagging and music information retrieval.* Journal of new music research, 37(2):101–114, 2008.

[13] Laurier, C., J. Grivolla und P. Herrera: *Multimodal music mood classification using audio and lyrics.* In: *Proceedings of the 7th international conference on machine learning and applications (ICMLA)*, Seiten 688–693, 2008.

[14] Logan, B., A. Kositsky und P. Moreno: *Semantic analysis of song lyrics.* In: *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, Band 2, Seiten 827–830, 2004.

[15] Mazzola, G., Y. Pang, W. Heinze, K. Gkoudina, G. Afrisando, J. G. Pujakusuma, Z. Chen, T. Hu und Y. Ma: *Basic Music Technology.* Springer, 2018.

[16] McKay, C., J. Cumming und I. Fujinaga: *JSYMBOLIC 2.2: Extracting Features from Symbolic Music for use in Musicological and MIR Research.* In: *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Seiten 348–354, 2018.

[17] McKay, C. und I. Fujinaga: *Automatic Genre Classification Using Large High-Level Musical Feature Sets.* In: *Proceedings of the 5th International Society for Music Information Retrieval Conference (ISMIR)*, Seiten 525–530, 2004.

[18] McKay, C. und I. Fujinaga: *Combining Features Extracted from Audio, Symbolic and Cultural Sources.* In: *Proceedings of the 9th International Society for Music Information Retrieval Conference (ISMIR)*, Seiten 597–602, 2008.

[19] Neumayer, R. und A. Rauber: *Integration of text and audio features for genre classification in music information retrieval.* In: *Proceedings of the European Conference on Information Retrieval*, Seiten 724–727, 2007.

[20] Ostermann, F., I. Vatolkin und M. Ebeling: *AAM: a dataset of Artificial Audio Multitracks for diverse music information retrieval tasks.* EURASIP Journal on Audio, Speech, and Music Processing, 2023(1):13, 2023.

[21] Raffel, C.: *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching.* Doktorarbeit, Columbia University, 2016.

[22] Raffel, C. und Daniel PW Ellis: *Optimizing DTW-based audio-to-MIDI alignment and matching.* In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Seiten 81–85, 2016.

[23] Schreiber, H.: *Improving Genre Annotations for the Million Song Dataset.* In: *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, Seiten 241–247, 2015.

[24] Sikora, F.: *Neue Jazz-Harmonielehre: Verstehen-Hören-Spielen.* Schott Music, 2022.

[25] Tzanetakis, G., A. Ermolinskyi und P. Cook: *Pitch histograms in audio and symbolic music information retrieval.* Journal of New Music Research, 32(2):143–152, 2003.

[26] Vatolkin, I., G. Bonnin und D. Jannach: *Comparing audio features and playlist statistics for music classification.* In: *Analysis of Large and Complex Data - Second European Conference on Data Analysis (ECDA)*, Seiten 437–447, 2016.

[27] Vatolkin, I. und C. McKay: *Multi-objective investigation of six feature source types for multi-modal music classification.* Transactions of the International Society for Music Information Retrieval, 5(1), 2022.

[28] Vatolkin, I., W. Theimer und M. Botteck: *AMUSE (Advanced MUSic Explorer)-A Multitool Framework for Music Data Analysis.* In: *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, Seiten 33–38, 2010.

[29] Vatolkin, I., W. M. Theimer und G. Rudolph: *Design and comparison of different evolution strategies for feature selection and consolidation in music classification.* In: *Proceedings of the IEEE Congress on Evolutionary Computation*, Seiten 174–181, 2009.

[30] Weihs, C., D. Jannach, I. Vatolkin und G. Rudolph: *Music data analysis: Foundations and applications.* Chapman and Hall/CRC, 2016.

[31] Wilkes, B., I. Vatolkin und H. Müller: *Statistical and Visual Analysis of Audio, Text, and Image Features for Multi-Modal Music Genre Recognition.* Entropy, 23(11):1502, 2021.

[32] Zangerle, E., M. Tschuggnall, S. Wurzinger und G. Specht: *Alf-200k: Towards extensive multimodal analyses of music tracks and playlists.* In: *Proceedings of the 40th European Conference on IR Research (ECIR)*, Seiten 584–590, 2018.

[33] Zhang, H., E. Karystinaios, S. Dixon, G. Widmer und C. E. Cancino-Chacón: *Symbolic Music Representations for Classification Tasks: A Systematic Evaluation.* In: *Proceedings of the 24th International Society for Music Information Retrieval Conference (ISMIR)*, Seiten 848–858, 2023.

Eidesstattliche Versicherung

Pingel, Clara	192493
Name, Vorname	Matr.-nr.

Ich versichere hiermit an Eides statt, dass ich die vorliegende Bachelorarbeit mit dem Titel

Erweiterung von AMUSE zur Verarbeitung mehrerer Modalitäten

selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Dortmund, den 27. August 2024	
Ort, Datum	Unterschrift

Belehrung:

Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/ die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -)

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird gfls. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin“) zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Dortmund, den 27. August 2024	
Ort, Datum	Unterschrift