Integration von Wissenskomponenten in ein Digitales Archiv zur Schreiberidentifikation von historischen Musikhandschriften

Ilvio Bruder, Temenushka Ignatova & Lars Milewski
Institut für Informatik, Universität Rostock
E-Mail: {ilr,temi,mile}@informatik.uni-rostock.de

Die Verwaltung domänenspezifischer Daten benötigt spezielle Modellierungs- und Speicherungstechniken. Basierend darauf können spezielle Zugriffs- und Recherchemethoden angeboten werden. Am Beispiel eines digitalen Archivs für Notenhandschriften soll insbesondere gezeigt werden, wie mit Hilfe von integriertem Expertenwissen eine Schreibererkennung implementiert wird.

1 Einführung

Ein wichtiger Schritt für die Verbesserung der Nutzbarkeit von digitalen Bibliotheken ist das Anbieten spezialisierter Services. Beispiele für derartige Bedürfnisse finden sich in vielen Bereichen. In unserer Arbeit gehen wir auf die Bedürfnisse von Musikwissenschaftlern bzgl. der Verwaltung von historischen Notenhandschriften und der Erkennung der Notenschreiber ein. An der Universität Rostock lagern mehr als 5000 Notenhandschriften des 17. und 18. Jahrhunderts. Diese zu digitalisieren, zu analysieren und Mechanismen für die Schreibererkennung zu entwickeln, ist Aufgabe des eNoteHistory-Projektes [2]. Wir haben dazu das Wissen der Musikexperten genutzt, um die Handschriften mit 80 Features aus 13 Featuregruppen zu beschreiben. Die momentane Umsetzung beinhaltet eine manuelle Bestimmung der Schreibercharakteristiken.

In diesem Artikel soll die Realisierung eines solchen digitalen Archivs für historische Musikschriften und die Integration spezieller domänenspezifischer Daten und Methoden für eine Schreiberidentifizierung beschrieben werden. Dabei werden insbesondere Formalismen für die Repräsentation einer Handschrift, Implementierungsaspekte und Tests vorgestellt.

2 Feature Base

Das Expertenwissen der Musikwissenschaftler haben wir formalisiert und in einer Feature Base implementiert. Diese beinhaltet die Strukturen und Wertebereiche der Features sowie Wissen zu Vergleichsmöglichkeiten. Jedes Feature wird mittels einer baumartigen Hierarchie dargestellt. Um mögliche Werte dieser Features zu einer Notenhandschrift zu bestimmen, wird in den Bäumen navigiert und der ähnlichste Wert bestimmt.

Feature Base Die Informationen der Feature Base werden in einem gerichteten, azyklischen, knotenmarkierten Graphen (bzw. Baum) dargestellt: $FB = (V, E, \mu, \nu, \tau)$, wobei $V = \{v_1, ..., v_n\}$ eine Menge von Knoten ist und $E \subseteq V \times V$ eine Menge von gerichteten Kanten, die diese Knoten verbinden. Durch Definition der Feature Base in Form eines einzigen Baumes, wird eine gemeinsame Wurzel für alle Feature-Gruppen definiert. Jedem Knoten wird eine Bezeichnung $\mu : V \rightarrow B$ aus einer Menge B von Knotenbezeichnern zugeordnet. Außerdem bekommt jeder Knoten eine laufende Nummer, die ihn von seinen Geschwistern unterscheidet: $\nu : V \rightarrow \mathbb{N}$. Für den Knoten '4.1.2. Achtelfähnchen aufwärts kaudiert' (x) z.B. sind $\mu(x)$ ='aufwärts kaudiert' und $\nu(x) = 2$. Die Funktion τ weist jedem Knoten einen Typ aus der Menge Type = {'prefix',

'feature', 'value'} zu: $\tau : V \rightarrow Type$. Ein Knoten x mit $\tau(x)$ ='feature' heißt Feature. Der darunter liegende Teilbaum Λ_x ist der Wertebereich dieses Features. Jeder Knoten dort hat den Typ 'value', weil er einen möglichen Wert für dieses Feature darstellt. Alle Knoten, die über x in der Hierarchie liegen, haben den Typ 'prefix'. Damit können nun auch die Definitionen für die Menge aller Features F und für den Wertebereich W_i angegeben werden: $F = \{f | f \epsilon V \wedge \tau(f) =$'feature'$\}$ und $W_i = \Lambda_{f_i}$, für alle $f_i \epsilon F$. In der Realität wird jedem Knoten auch noch ein Bild zugeordnet, das die manuelle Bestimmung erleichtert. Der Pfad im Baum dient als eindeutiger Bezeichner für einen Knoten: $PATH : V \rightarrow P$. P ist eine Menge von Pfadausdrücken.

Distanzfunktion Das Ziel des Schreibererkennungssystems ist es, zwei Handschriften zu vergleichen, indem die sie beschreibenden Feature-Vektoren γ_a und γ_b verglichen werden. Gesucht ist dafür eine Distanzfunktion der Form: $d_\Gamma : \Gamma \times \Gamma \rightarrow [0..1]$. Dies kann z.B. die normalisierte Hamming-Distanz [1] sein. Danach gilt für $\gamma_a = (v_1^a, ..., v_n^a)^T$ und $\gamma_b = (v_1^b, ..., v_n^b)^T$, dass $d_\Gamma = \frac{d_{f_1} + ... + d_{f_n}}{n}$, wobei d_{f_i} die Differenz (oder anders: Distanz) von v_i^a und v_i^b ist. Die Funktion d_{f_i} dient dazu, die einzelnen Attribute jeweils paarweise zu vergleichen, z.B. den G-Schlüssel-Wert des einen Vektors mit dem des anderen. Es wird also zusätzlich eine Distanzfunktion für jedes einzelne Feature f_i benötigt: $d_{f_i} : W_i \times W_i \rightarrow [0..1]$.

Es stellt sich aber noch immer die Frage, wie die Ähnlichkeit gemessen, interpretiert und dargestellt werden kann. Dafür gibt es drei Möglichkeiten, die sich in Aufwand und Komplexität unterscheiden. Die einfachste Lösung ist eine Boolsche Logik, bei der nur unterschieden wird, ob die Werte gleich oder verschieden sind. In den meisten Fällen wird dieses einfache Abstandsmaß jedoch nicht ausreichen, denn es impliziert, dass die kleinste Änderung im Schriftbild eines Kopisten genauso wie ein maximaler Unterschied behandelt wird. Es wird daher ein Maß benötigt, welches eine Abbildung auf den gesamten Bereich $[0, 1]$ erlaubt. Die Idee dabei ist, die Struktur des Feature-Base-Baumes zur Bestimmung der Distanzen auszunutzen. Dazu seien die Codes $x_P = PATH(x)$ und $y_P = PATH(y)$ von zwei Werten x und y gegeben. Unter einer Wertanalyse ist das Zerlegen von x_P und y_P in $(x_P^1, .., x_P^n)$ und $(y_P^1, .., y_P^n)$ zu verstehen. Die Werte x_P^i und y_P^i geben jeweils den Knoten an, der in der i. Ebene gewählt wurde. Zwei Werte, deren Pfade sich nur in einer Ebene unterscheiden, wären damit sehr ähnlich. Diese Distanzfunktion erreicht die Genauigkeit $\frac{1}{n}$, wobei n die Tiefe des Wertebereichbaumes ist. Außerdem basiert sie immer noch auf einer Distanzfunktion mit Boolscher Logik. Darum werden Ähnlichkeiten von Werten einer Ebene nicht erkannt. Dazu werden zusätzliche Informationen benötigt. Bei vielen Merkmalen kann die Distanz nicht aus der Baumstruktur abgeleitet werden. In solchen Fällen kann eine Distanzmatrix darüber Auskunft geben, welche Distanz jeweils zwei Werte haben.

Damit gilt für die Distanzfunktion: $d_{f_i}^{Inf}(x, y) = d_{x,y}$, wobei $d_{x,y}$ aus der Distanzmatrix an der Stelle (x,y) stammt. Im praktischen Einsatz gilt nun, dass die rein Boolsche Distanzfunktion den beiden anderen Funktionen aufgrund ihrer geringen Komplexität vorzuziehen ist, sofern sie Ergebnisse mit ausreichend guter Qualität liefert. Das Gleiche gilt für die pfadbasierte gegenüber der Distanzmatrix-basierten Distanzfunktion. Trotzdem muss die Distanzmatrix-basierte in den meisten Fällen aufgrund der höheren Flexibilität eingesetzt werden.

Spezielle Feature-Werte Wenn ein Schreiber eine Note auf zwei verschiedene Weisen schreibt, so dass der Nutzer sich nicht eindeutig für eine entscheiden kann, muss er die Möglichkeit haben, beide Werte anzugeben. Das heißt, im allgemeinen Fall beinhaltet der Feature-Vektor wiederum Mengen von Werten. Da es nun Feature-Vektoren mit komplexen Werten gibt, muss auch eine angepasste komplexe Feature-Distanzfunktion definiert werden: $d_{f_i}^K : \mathfrak{P}(W_i) \times \mathfrak{P}(W_i) \rightarrow [0..1]$ ($\mathfrak{P}(W_i)$: Potenzmenge von W_i). Die Frage, wie die Distanz zwischen zwei komplexen Werten berechnet werden kann, wurde durch umfangreiche Tests bestimmt (siehe Abschnitt 4). Im Fall des Schreibererkennungssystems existieren weiterhin Nullwerte mit zwei unterschiedlichen Bedeutungen: eine Non-Information-Null (?) bedeutet, dass über diesen Wert keine Informationen vorliegen. Das kann z.B. der Fall sein, wenn der Schreiber in dem vorliegenden Werk keinen

C-Schlüssel benutzt. In diesem Fall würde das C-Schlüssel-Feature mit dem Nullwert '?' belegt. Eine No-Applicable-Null ($\top$) bedeutet dagegen, dass nie ein Wert existieren wird. Das kann bspw. beim C-Schlüssel auftreten. Der C-Schlüssel besteht aus mehreren Elementen, die jeweils durch ein Attribut beschrieben werden. Es ist typisch für einen Schreiber, welche dieser Elemente er schreibt und welche nicht. Wenn er kein linkes Element schreibt, bekommt das Feature '1.2.1. Linkes Element' den Nullwert '$\top$'. Im Gegensatz zu dem ersten Nullwert (?), bei dem nicht bekannt ist, wie ein Kopist ein Symbol schreibt, ist bei '$\top$' klar, dass er es nicht schreibt.

3 Implementation und Beispiel

Der implementierte Prototyp[1] basierend auf IBM DB2 beinhaltet eine Reihe von Datenbankschemata für Metadaten und Bilder der Notenhandschriften, für die Feature Base, für die konkreten Featurewerte, für die einzelnen Schreiber sowie Funktionen für die Distanzberechnung und Clustering/Klassification. Der Prototyp hat Schnittstellen für die Suche und Navigation in den Metadaten und Bildern. Die Schnittstelle für die Schreibererkennung basiert auf ein Tool zur Beschreibung von Handschrift-Features. Bevor wir Daten und Methoden des Prototyp beschreiben, sollen einige existierende Tools betrachtet werden. Durch unsere Anwendung sind bestimmte Anforderungen zu beachten: (1) Support für instanzbasierte Klassifikation, (2) Beliebige Distanzfunktionen, (3) Anpassung von Ähnlichkeitsparameter, (4) Wahl von Distanzmatrizen für jedes Attribut einzeln und (5) Unterstützung von Nullwerten und komplexen Werten. Drei der untersuchten Data Mining Tools unterstützen das erste Kriterium: Darwin [4], MLC++ [3], and Weka [5]. Alle drei benutzen ein instanzbasiertes Klassifikationsverfahren basierend auf dem k-Nearest-Neighbor-Algorithmus. Einzelne Parameter, Gewichte, Einstellungen sind modifizierbar. MLC++ und Weka könnten nach größeren Modifikationen benutzt werden. Dies würde allerdings einer Reimplementierung gleich kommen. Aufgrund der zusätzlichen Bedingung diese Tools in die Datenbankumgebung zu integrieren, haben wir entschieden, die benötigte Funktionalität selbst zu implementieren.

Implementation in der Datenbankumgebung Die Feature Base (siehe vorherigen Abschnitt) wurde als Datenbankschema in IBM DB2 implementiert. Die von den Musikwissenschaftlern spezifizierten Feature-Bäume und Distanzmatrizen haben wir dann in das Datenbankschema importiert und die benötigten Relationen zu den Metadaten der Notenhandschriften hergestellt. IBM DB2 unterstützt als integrierbare Methoden/Funktionen User-Defined Functions (UDF) und Stored Procedures. Es gibt drei verschiedene UDFs: skalare Funktionen für einzelne Werte, Spaltenfunktionen für Tabellenspalten in der Datenbank und Tabellenfunktionen zur Verarbeitung ganzer Tabellen. UDFs werden innerhalb von SQL-Anweisungen verwendet. Stored Procedures sind nur über einen speziellen Aufruf in der Regel durch die Applikation aufrufbar. Wir haben zwei Hauptfunktionen A_{FV} und A_S implementiert. A_{FV} gibt die k nächsten Feature Vektoren zu einem gegebenen Anfragevektor γ_q zurück: $A_{FV} = \{\gamma \epsilon \Gamma | d_\Gamma(\gamma, \gamma_q) \leq t\}$ bei gegebener Distanzfunktion d_Γ. Der Schwellwert t bestimmt die Menge an Resultaten. Eine Liste von relevanten Schreibern gibt die A_S-Funktion basierend auf A_{FV} zurück. Für beide Funktionen benötigen wir jeweils eine Tabellen-UDF. Beide Funktionen benutzen einige weitere Interfaces zu den Features und Distanzen in der Datenbank. Die Ergebnistabelle beinhaltet einen Feature-Vektor oder Schreibernamen mit einem Ähnlichkeitsmaß.

Beispiel Der Vorgang vom Erkennen der Features bis zur Schreiberidentifikation soll durch das folgende Beispiel gezeigt werden. Abbildung 1 zeigt dazu die Erkennung eines G-Schlüssels.

Zuerst müssen wir eine charakteristische Notation des G-Schlüssels im Original Notenblatt finden. Anschließend navigieren wir durch den Feature-Baum des G-Schlüssels, um die ähn-

[1] Der Prototyp ist unter http://www.enotehistory.de verfügbar

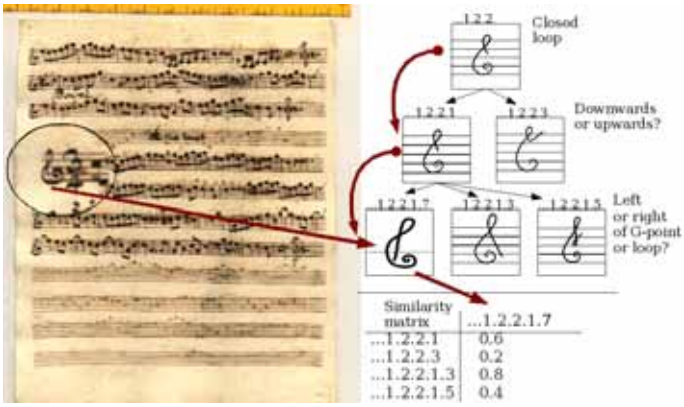

Abbildung 1: Beispiel der manuellen Feature-Extraktion

Nr	Ähnlichkeit	Qualität	Schreiber
1.	0,018	0,534	AN305_a
2.	0,091	0,564	AN305_b
3.	0,126	0,548	Reichardt I_a
4.	0,192	0,394	J.M. Baldauff_a
5.	0,214	0,434	Büchler IV_a

Tabelle 1: Ergebnismenge mit Ähnlichkeit, Qualitätsmaß (N_D siehe folgende Evaluationsmaße) und dem Schreibernamen (_a und _b definieren verschiedene Perioden eines Schreibers)

lichste Repräsentation dieses G-Schlüssels zu finden (in Abb. 1 ist es ein G-Schlüssel mit einer geschlossenen und nach unten gerichteten Schleife links vom G-Punkt). Diese Repräsentation ist ein möglicher Wert des G-Schlüssel. Der numerische Wert (...1.2.2.1.7) wird für die folgende Ähnlichkeitsberechnung benötigt. Zur Bestimmung der Ähnlichkeit zwischen zwei verschiedenen Notationen eines Features benötigen wir die Ähnlichkeitsmaße und die Ähnlichkeitsmatrizen. Wenn wir den G-Schlüssel aus diesem Beispiel mit einem anderen G-Schlüssel, z.B. ...1.2.2.1, vergleichen, bekommen wir eine Ähnlichkeit von 0.6. Tabelle 1 zeigt eine Ergebnismenge bzgl. einer Anfrage. Das Ergebnis beinhaltet die Ähnlichkeit, ein Qualitätsmaß und die Schreibernamen. Die Schreibernamen sind fiktiv, da oft der richtige Name nicht bekannt ist.

4 Tests

Um unseren Ansatz zu testen, haben wir einige Maße zur Evaluierung der Distanzfunktion, deren Parameter, die Feature-Gewichte und die Anfrageergebnisse definiert. Das Ziel ist, die optimale Konfiguration für das System zu finden.

Bewertungsmaße Wir benutzen eine Scoring Function (SF) zur Evaluierung der instanzbasierten Clustering-Methoden. SF ist ein Maß für die Dichte/Enge eines Clusters. Die Distanz zwischen Instanzen desselben Clusters soll so klein wie möglich sein, wogegen die Distanz zwischen Instanzen verschiedener Cluster möglichst groß sein soll. Mit den momentanen Daten ist es nicht nötig, ein Clustering durchzuführen, da wir die jeweiligen Schreiber für alle Features kennen. Deshalb können wir mit diesem Maß die Distanzfunktion und deren Parameter direkt evaluieren. Die Evaluierungsmaße Precision und Recall, im Information Retrieval Bereich etabliert, bewerten das entwickelte System anhand der Anfragen bei einer Schreiberanalyse. Precision ist ein Maß für die Qualität und Recall ist ein Maß für die Quantität der Anfrageergebnisse. Das Maß K wird zur Evaluierung der Anfrageergebnisse nach der Interpretation verwendet. Im Gegensatz zu Precision/Recall evaluiert K die Schreiber-Ergebnislisten. K = $\frac{X}{N}$, wobei X die Anzahl der Ergebnislisten mit korrekt erkannten Schreiber und N die Anzahl aller Ergebnislisten ist. Für die Bewertung des Einflusses von Nullwerten haben wir zwei Maße eingeführt. Das

erste Maß repräsentiert die Beziehung zwischen den Gewichten der Nullwerte bzgl. der Gewichte aller Feature-Werte: $N_F(\gamma) = \frac{\sum_{\forall i: v_i = null} w_i}{\sum_{\forall i} w_i}$. Das zweite Maß definiert das Verhältnis der Features, die aufgrund eines Nullwertes bei der Distanzberechnung nicht in das Ergebnis einfließen: $N_D(d(\gamma_a, \gamma_b)) = \frac{\sum_{\forall i: v_i^a = null \vee v_i^b = null} w_i}{\sum_{\forall i} w_i}$.

Bewertung und Interpretation Durch die Verwendung der Bewertungsmaße sind wir zu folgenden Schlussfolgerungen gekommen: Wir haben verschiedene Distanzfunktionen getestet: Hamming-Distanz, Euklidische Distanz und Distanzmetriken höherer Potenz. Je höher die gewählte Potenz war, desto schlechter wurden die Ergebnisse. Die Hamming-Distanz zeigte die besten Werte für SF und K. Die Verteilung von Gewichten auf die einzelnen Features für den Einfluss der Bedeutung des einzelnen Features bei der Distanzberechnung wurde mit verschiedenen Verteilungen getestet. Eine erste Zuordnung der Gewichte haben wir durch Berechnung der Distanzen durch Weglassen des einzuordnenden Features bestimmt. Folgende Verteilungen haben wir anschließend getestet: (1) konstant, (2) linear gruppiert, (3) linear, (4) quadratisch gruppiert und (5) quadratisch. Dabei wurde SF von Verteilung (1) zu (5) besser und im Gegenzug wurde K von (5) zu (1) besser. Wir haben mit (3) einen Kompromiss zwischen beiden Bewertungsmaßen bestimmt. Die Ähnlichkeitsmaße in den Matrizen haben wir ebenfalls untersucht, indem verschiedene Veränderungen der Werte vorgenommen wurden: Verschiebung der Werte durch $x - 0.2$, $x - 0.1$, $x + 0.1$, $x + 0.2$; Skalierung durch $x/4$, $x/2$, $x * 2$ sowie Potenzieren/Wurzel ziehen durch x^2, x^3, $\sqrt{x}$, $\sqrt[3]{x}$. Die Funktionen, die die Werte verkleinerten, führten zu besseren Bewertungen. Schlechte Vergleiche können auch an zu vielen Nullwerten liegen. Mittels N_D und N_F ist festzustellen, dass die meisten falsch klassifizierten Instanzen auf zu viele Nullstellen und damit auf zu wenig Vergleichsmöglichkeiten beruhen. Bei den komplexen Werten sollte mittels Tests herausgefunden werden, welche Distanz zwischen zwei Mengen von Werten zu besten Ergebnissen führt. Die Tests haben ergeben, dass ein Distanzwert nahe dem Minimum der möglichen Einzeldistanzen die besten SF-Werte ergeben. Precision und Recall haben wir mit einer Menge von Anfragen und unter verschiedenen Schwellwerten von 0.01 bis 0.5 gemessen. Dazu haben wir jede Featuremenge (insgesamt 150) aus der Gesamtmenge herausgenommen und damit die Anfrage gestellt. Das beste Precision/Recall-Verhältnis liegt bei etwa 85% Precision und 75% Recall.

5 Zusammenfassung und Ausblick

Das Ziel des "eNoteHistory"-Projektes ist die Entwicklung eines System zur Unterstützung von Musikwissenschaftlern bei der Analyse historischer Musikhandschriften. Eine einfache Verwaltung der digitalen Dokumente und deren Metadaten ist nicht genug, die Anforderungen nach einer digitalen Hilfe für die spezielle Musikanalyse zu erfüllen. Dies führte zu dem hier vorgestellten spezialisierten Archivsystem mit Domänenwissen und speziellen Methoden für eine semiautomatische Erkennung von Musikhandschriften. Die Erkennung erreicht eine Rate von 90%. Die Evaluierung ist aufgrund der wenigen Testbeispiele (150 Features) nicht aussagekräftig genug, wird aber mit der zukünftigen Teilautomatisierung der Featureerkennung verbessert.

Literatur

[1] D. Aha and D. Kibler. Instance-based learning algorithms. *Machine Learning, vol.6*, 1991.

[2] I. Bruder, A. Finger, A. Heuer, and T. Ignatova. Towards a digital document archive for historical handwritten music scores. In *6th ICADL*, Kuala Lampur, Malaysia, 2003.

[3] R. Kohavi and D. Sommerfield. *MLC++ Machine Lerning Library in C++*, 1996.

[4] Oracle. *Darwin Installation and Administration. Release 3.7*, 2000.

[5] I. H. Witten and E. Frank. *Data Mining - Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, 2000.