Erkennung von Sequenzen mimischer Schmerzausdrücke durch genetische Programmierung*

Christoph Stocker[1], **Michael Siebers**[2] **und Ute Schmid**[3]
Kognitive Systeme, Fakultät Wirtschaftsinformatik und Angewandte Informatik
Otto-Friedrich-Universität Bamberg
[1]c.stocker.mail@googlemail.com, [2]michael.siebers@uni-bamberg.de, [3]ute.schmid@uni-bamberg.de

Abstract

Aus empirisch erhobenen Videos von Gesichtern, die Schmerzmimik zeigen, werden über Abfolgen von Action Units (AUs) reguläre Ausdrücke generalisiert, um für Schmerz charakteristische Sequenzen von Bewegungen im Gesicht zu identifizieren. Für die Generierung regulärer Ausdrücke über AU-Sequenzen wurde ein genetischer Algorithmus implementiert. Als Datenstruktur werden Syntaxbäume regulärer Ausdrücke verwendet. Die Individuen sind Grammatiken, die die in den Trainingsdaten gegebenen Sequenzen erkennen können. Die resultierende Präzision der Grammatiken bewegt sich zwischen ca. 80–100%. Problem bei der Optimierung des Verfahrens ist die Justierung der zahlreichen Mutationsparameter. Zudem erschwert das Fehlen von negativen Trainingsdaten eine Evaluation der gelernten Ausdrücke.

1 Einleitung

Angesichts einer kontinuierlich alternden Gesellschaft wird die Betreuung pflegebedürftiger Menschen immer wichtiger. Die Entwicklung von computergestützten Systemen, die das Pflegepersonal bei dieser Aufgabe unterstützen, ist daher ein interessantes Forschungsgebiet. Ein möglicher Teilaspekt dabei ist die Kommunikation mit Menschen, die aufgrund einer Erkrankung nicht oder nur erschwert zu sprachlichen Äußerungen oder Gestik in der Lage sind. In vielen Fällen sind jedoch reaktive Bewegungen, wie sie etwa von Schmerzreizen ausgelöst werden, voll funktionsfähig. Es können daher Systeme entwickelt werden, die diese Bewegungen erkennen, um die nonverbale Kommunikation mit dem Patienten zu ermöglichen.

Verfahren zur Analyse von Gesichtsausdrücken lassen sich in solche unterscheiden, die direkt Klassifikatoren über metrische Bildmerkmale generieren und solche, die in einem Zwischenschritt sogenannte *Action Units* (AUs) in Bilder identifizieren [Fasel and Luettin, 2003]. Die Nutzung von AUs basiert auf den psychologischen Arbeiten von Ekman und Friesen [Ekman and Friesen, 1978], die gezeigt haben, dass sich Emotionen kulturunabhängig und stabil durch die Bewegung von Teilmengen der 43 Gesichtsmuskeln charakterisieren lassen. Das sogenannte *Facial Action Coding System* (FACS) beschreibt die Menge der möglichen AUs und die Zuordnung von Teilmengen dieser AUs zu Emotionen. Das FACS wurde auch für andere, nicht-emotionale mimische Reaktionen – beispielsweise für Schmerz – erweitert [Lucey *et al.*, 2011]. Üblicherweise erfolgt die Analyse von mimischen Ausdrücken über die *Menge* der in einem Zeitintervall aufgetretenen AUs. Es besteht jedoch die Möglichkeit, dass die *sequentielle Abfolge* von AUs zusätzlich diagnostisch relevante Information enthalten kann [Schmid *et al.*, 2012].

Im Folgenden werden zunächst die Datengrundlage sowie Vorarbeiten zum Lernen von charakteristischen AU-Sequenzen dargestellt. Danach wird der genetische Algorithmus zum Lernen regulärer Ausdrücke beschrieben. Es folgt eine Evaluation des Algorithmus' an zwei Datensätzen. Schließlich werden die Ergebnisse bewertet und ein Ausblick auf weitere Arbeitsschritte gegeben.

2 Lernen von Grammatiken aus Sequenzen von Action Units

Im Rahmen eines psychologischen Experiments [Kunz *et al.*, 2007] wurden Patienten mit einer dementiellen Erkrankung ($n = 42$), ältere gesunde Personen ($n = 54$) sowie Studierende ($n = 28$) kurzen Episoden von Druckschmerz ausgesetzt. Dabei wurden die mimischen Bewegungen im Gesicht aufgezeichnet. Die Videos wurden von einer ausgebildeten FACS-Coderin mit AUs annotiert. Da bei mimischen Reaktionen die onset-Zeiten von AUs klar identifizierbar sind, die offset-Zeiten jedoch nicht, da mimische Reaktionen üblicherweise langsam „zerfallen", wurden aus den FACS-Kodierungen Sequenzen von AUs bezüglich der onset-Zeiten extrahiert. Zudem wurden nur diejenigen AUs in die Sequenz aufgenommen, bei denen die Intensität der verabreichten Schmerzreize – gemessen auf einer Skala von 1 bis 5 – mindestens 4 betrug, da bei schwächeren Reizen häufig keine mimische Reaktion gezeigt wurde. Haben zwei oder mehr AUs die gleiche onset-Zeit wurden diese AU-compounds in die Sequenz übernommen. Das den Sequenzen zugrundeliegene Alphabet besteht aus insgesamt 76 AUs und AU-compounds.

In einer Vorgängerarbeit [Schmid *et al.*, 2012] wird die Datengrundlage detailliert beschrieben. In dieser Arbeit wurde ein Verfahren zum Lernen regulärer Grammatiken (*Alignment Based Learning* ABL [van Zaanen, 2002]) zur Generalisierung einer Schmerzgrammatik verwendet. Dabei wurden bereits vielversprechende Ergebnisse erzielt. Allerdings waren die resultierenden Grammatiken zur Charakterisierung von Schmerzmimik sehr komplex. Nun sollte geprüft werden, ob mittels genetischer Algorithmen, bei denen die Komplexität der Grammatik als Kriterium in die Fitness-Funktion aufgenommen wird, kompaktere Beschreibungen gelernt werden können.

*Ergebnisse eines studentischen Projekts im Master-Studiengang *Computing in the Humanities* an der Universität Bamberg

3 Konzeption eines genetischen Algorithmus

3.1 Datenstrukturen

Ein regulärer Ausdruck ist eine intensionale Beschreibung von Mengen von Zeichenketten. Der Ausdruck selbst ist dabei eine syntaktische Beschreibung, welche die in der Menge enthaltenen Sequenzen erfüllen müssen. Durch reguläre Ausdrücke lassen sich reguläre Mengen beschreiben, d.h. Mengen, die durch die Operationen der Vereinigung, Konkatenation und Sternbildung entstehen. Im Gegensatz zu einer Darstellung durch Produktionsregeln können allerdings nur reguläre Grammatiken dargestellt werden. Dieser Nachteil wurde angesichts der entfallenden Konsistenzprüfung in Kauf genommen.

Um reguläre Ausdrücke adäquat darstellen zu können, wurde entschieden, diese als Syntaxbäume zu implementieren, wie dies im Bereich der genetischen Programmierung üblich ist [Mitchell, 1997, S. 262]. Jeder Knoten im Baum entspricht dabei einem Operator eines regulären Ausdrucks. Aus Sicht der genetischen Programmierung entspricht jeder Operator einer Funktion sowie jede AU einem Terminal. Es wurden folgende Operatoren implementiert: Verknüpfung ('Und-Knoten'), Alternation ('Oder-Knoten') sowie die Quantoren '*','+' und '?'.

3.2 Algorithmus

Die Konzeption des genetischen Algorithmus orientiert sich an den in diesem Bereich üblichen Konzepten der Selektion bezüglich einer Fitness-Funktion [Mitchell, 1997, S. 251].

Eine Gruppe von Individuen verbessert sich im Laufe mehrerer Lebenszyklen, indem starke Individuen ihr Erbgut durch Fortpflanzung weitergeben, während schwache Individuen im Lauf der Zeit verschwinden. Die Fortpflanzung besteht aus Generierung neuer Individuen, welche eine zufällige Kombination des Erbguts der Eltern enthalten. Zusätzliche spontane Mutationen verhindern dabei, dass das Verfahren in lokalen Maxima zum Erliegen kommt. Unter einem Individuum wird hierbei im Sinne dieser Arbeit eine einzelne Grammatik verstanden. Das Erbgut eines Individuums sind demzufolge die Regeln, aus denen die Grammatik besteht.

In einem Selektionsschritt wird bestimmt, welche Individuen zur Fortpflanzung geeignet sind. Dies wird entschieden, indem für jedes Individum geprüft wird, wie gut es die gestellte Aufgabe erfüllt. Im Falle dieser Arbeit bedeutet dies, dass eine Grammatik dann geeignet ist, wenn sie möglichst viele Sequenzen effektiv erkennen kann.

Die wichtigsten Schritte des Verfahrens sind demnach die Selektion, die Mutation sowie das Crossover. Zudem ist die Umsetzung der Fitnessfunktion ein weiteres Schlüsselelement.

Selektion

Für die Selektion wurde eine probabilistische Variante verwendet. Die Selektion eines Individuums h_i erfolgt nach einer Gewichtung gemäß der Fitness, bezogen auf die aufsummierte Fitness der gesamten Population:

$$Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{p} Fitness(h_j)}$$

Fitness

Die Fitnessfunktion bewertet die Effizienz eines Individuums. Es ist daher naheliegend die Präzision einer Grammatik als Maß der Fitness in Betracht zu ziehen. Ein Individuum ist folglich umso fitter, je mehr Sequenzen der Trainingsdaten es erkennen kann. Aus der Umsetzung der Grammatiken sowie einigen technischen Aspekten ergeben sich allerdings noch weitere Kriterien welche für die Bewertung einer Grammatik in Betracht gezogen werden können.

Zum einen erhöhen große Grammatiken signifikant die Laufzeit des Algorithmus. Der Grund hierfür ist neben der steigenden Anzahl an Knoten im Allgemeinen die steigende Komplexität, welche sich durch Oder-Knoten sowie Quantoren ergibt. Enthält eine Grammatik viele Oder-Knoten, so müssen beim Matching der regulären Ausdrücke für jede Veroderung alternative Möglichkeiten in Betracht gezogen werden. Dies kann durchaus dazu führen, dass die Laufzeit des Algorithmus nicht mehr im akzeptablen Bereich liegt. Demzufolge ist es sinnvoll, Grammatiken niedriger zu bewerten, welche viele Oder-Knoten enthalten.

Zum anderen lässt sich über die Fitnessfunktion das Generalisierungsverhalten des Algorithmus steuern. Grundsätzlich lässt sich festhalten, dass durch die Verwendung von regulären Ausdrücken als Repräsentationsgrundlage sowie durch das Fehlen negativer Trainingsbeispiele im Prinzip jede Sequenz durch einen einfachen regulären Ausdruck erkennen lässt:

$$(au1|au2|au3|au4|\ldots|auN)^*$$

Dieser Ausdruck liefert zwar für jede beliebige Menge von Sequenzen eine Präzision von 100%, ist jedoch nicht sehr aussagekräftig, da er Sequenzen, welche Schmerz anzeigen, nicht von anderen Sequenzen unterscheiden kann. Eine Klassifikation ist somit nicht mehr möglich.

Aufgrund des Mangels an negativen Trainingsdaten lässt sich dieses Phänomen nicht vollständig verhindern. Allerdings kann das Generalisierungsverhalten des Algorithmus auf ein Mindestmaß eingeschränkt werden, indem extrem generalisierende Ausdrücke vermieden werden. Der oben genannte Ausdruck lässt sich beispielsweise vermeiden, indem Grammatiken niedrig bewertet werden, welche viele Veroderungen enthalten. Hierbei ist jedoch anzumerken, dass dies das Voranschreiten der Evolution negativ beeinflusst, da potenziell wertvolle Oder-Knoten ebenfalls seltener auftreten.

Des Weiteren macht es gemäß des Sparsamkeitsprinzips durchaus Sinn, die Anzahl der Knoten eines Syntaxbaums ebenfalls in die Bewertung einfließen zu lassen. Von zwei Grammatiken, welche die gleiche Präzision erzielen, wird demzufolge diejenige bevorzugt, welche weniger Knoten enthält und damit tendenziell eher spezifischer ist.

In die Implementierung der Fitnessfunktion wurden alle genannten Aspekte miteinbezogen. Die endgültige Fitness einer Grammatik h besteht somit aus der gewichteten Summe aller Teilaspekte. Hierbei bezeichnet a_i einen Teilaspekt, sowie w_i das dazugehörige Gewicht:

$$Fitness(h) = \frac{\sum a_i w_i}{\sum w_i}$$

Die Präzision einer Grammatik wird direkt anhand der Trainingsdaten berechnet. Der Aspekt der vorhandenen Oder-Knoten kann einfach quantifiziert werden, indem man die Anzahl der Oder-Knoten einer Grammatik der Gesamtanzahl an Knoten gegenüberstellt, für den Aspekt der verwendeten Quantoren wurde ebenso verfahren. Die Länge einer Grammatik berechnet sich aus $\frac{1}{1+n}$, wobei n die Gesamtanzahl der Knoten einer Grammatik bezeichnet. Wie genau die Einstellung der Gewichte vorzunehmen ist,

hängt von der konkreten Implementierung sowie deren Einsatzgebiet ab.

Mutation
Eine Mutation wird für jede in der Population vorhandene Grammatik mit einer vorgegebenen Wahrscheinlichkeit angewendet. Da eine Grammatik als ein Syntaxbaum repräsentiert wird, ist die naheliegendste Form der Mutation, einen beliebigen Knoten im Syntaxbaum durch einen neuen zu ersetzen.

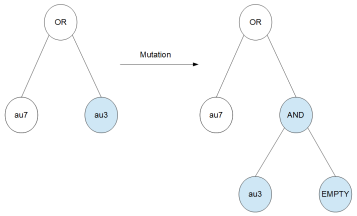

Abbildung 1: Beispiel für eine Mutation

Ein Beispiel für dieses Vorgehen ist in Abbildung 1 dargestellt. Die Action Unit *au3* wird durch einen Und-Knoten ersetzt. Hier zeigt sich bereits das breite Spektrum an Entscheidungen auf, welche bei der Implementierung der Mutation auftreten. Wird wie im genannten Beispiel eine Action Unit durch einen Und-Knoten ersetzt, so muss entschieden werden, welche Kindknoten im Und-Knoten eingefügt werden. Eine mögliche Option wäre es, die alte Action Unit sowie einen leeren Knoten einzufügen. Die resultierende Grammatik hätte demnach die selbe Aussagekraft wie die ursprüngliche, hätte jedoch durch den zusätzlichen Leerknoten weiteres Potenzial durch zukünftige Mutation eine höhere Präzision zu erreichen. Eine andere Option wäre es zufällig generierte Knoten als Kindknoten einzusetzen. Dies bewirkt einen größeren Sprung im Suchraum, erzeugt jedoch mit einer größeren Wahrscheinlichkeit ineffiziente Grammatiken.

Für jede implementierte Mutationsvariante muss festgelegt werden, mit welcher Wahrscheinlichkeit diese auftritt. Es hat sich erwiesen, dass Mutationen, welche neue Teilbäume erzeugen, nur sehr selten vorgenommen werden sollten (Größenordnung 1-2%). Werden zu häufig neue Teilbäume erstellt, so entstehen sehr schnell breite Syntaxbäume, was die Laufzeit signifikant erhöht. Insgesamt sind alle festgelegten Mutationswahrscheinlichlichkeiten als Parameter zu betrachten, da sie die Wirkungsweise des Algorithmus sowie die Laufzeit beeinflussen und keine allgemein gültige optimale Konfiguration möglich ist.

Crossover
Im Gegensatz zur Mutation gestaltet sich das Crossover bei der genutzten Repräsentation als Syntaxbäume relativ einfach. Ein Crossover wird durchgeführt, indem von zwei Grammatiken jeweils zufallsbasiert Teilbäume ausgetauscht werden. Welche Grammatiken durch Crossover verändert werden sollen wird zufallsbasiert, jedoch gewichtet nach der jeweiligen Fitness entschieden. Die dazu verwendete Berechnungsvorschrift entspricht der Formel aus Abschnitt 3.2.

4 Evaluation

Die Vollständigkeit des Algorithmus wird anhand der Präzision gemessen. Unter Präzision ist hierbei der prozentuale Anteil an Trainingsbeispielen zu verstehen, der von einer Grammatik erkannt werden kann.

	Datensatz 1	Datensatz 2
Anzahl AUs	30	76
Anzahl Sequenzen	59	347
Längste Sequenz	17	17
Kürzeste Sequenz	1	1
Mittlere Sequenzlänge	3.59	4.03

Tabelle 1: Eckdaten der Datensätze

Im Rahmen dieser Arbeit wurde mit zwei Datensätzen gearbeitet, die unterschiedliche Schwierigkeitsgrade aufgrund ihrer Komplexität aufweisen. Datensatz 2 umfasst FACS-kodierte Sequenzen aus AUs und AU-compounds aller im Experiment von Kunz et. al. betrachteten Personengruppen [Kunz *et al.*, 2007] – also von älternen Menschen mit dementieller Erkrankung, älteren gesunden Probanden und studentischen Untersuchungsteilnehmern. Datensatz 1 enthält nur die Sequenzen der Studentinnen, da diese Personengruppe die klarsten mimischen Reaktionen zeigte und die Sequenzen weniger variationsreich sind als bei den anderen Probandengruppen.

Die Eckdaten der beiden Datensätze sind in Tabelle 1 aufgelistet. Der Hauptunterschied der Datensätze besteht in der Anzahl der Sequenzen, sowie in der Anzahl der Action Units, die verwendet werden. Datensatz 1 ist aufgrund dieser Einteilung als der „einfache“ Datensatz zu verstehen, sowie Datensatz 2 als der „komplexe“ Datensatz.

Es ist anzumerken, dass dies nur eine grobe Unterteilung darstellt, die nicht effektiv auf die Komplexität der Daten hinweisen kann. Da es sich um eine rein statistische Einteilung handelt wird darin beispielsweise nicht berücksichtigt, ob die enthaltenen Sequenzen nur mit kontextsensitiven Regeln zu erkennen sind. Einteilungen nach Komplexität dieser Art müssten für genauere Untersuchungen berücksichtigt werden.

Um die Vollständigkeit des Algorithmus auszuwerten wurden auf jeden Datensatz 10 Iterationen ausgeführt. Im Allgemeinen lässt sich feststellen, dass die hier vorgestellte Umsetzung die beiden Datensätze recht gut erkennen kann. Für Datensatz 1 wurde der Zielwert von 95% Präzision bei jedem Durchgang erreicht. Hierbei waren, bis auf eine Ausnahme, nie mehr als 100 Iterationen notwendig.

Die Ergebnisse für Datensatz 2 sind ebenfalls als erfolgreich zu erachten. Die Präzision erreicht den erstrebten Zielwert von 95% in 5 der 10 durchgeführten Testläufe. Die restlichen Resultate bewegen sich in einem Bereich von ca. 78% bis 90%. Um diese Ergebnisse zu erreichen, waren bei Datensatz 2 jedoch deutlich mehr Iterationen erforderlich. In 5 von 10 Fällen erreichte der Algorithmus das festgelegte Maximum von 200 Iterationen. Bei diesen 5 Testläufen handelt es sich um diejenigen, die nicht den gewünschten Grenzwert erreichten. Für die erfolgreichen Durchläufe waren jedoch ebenfalls stets deutlich mehr als 100 Iterationen notwendig. Die dafür notwendige Laufzeit ist ebenfalls deutlich höher als bei Datensatz 1.

Um das generelle Verhalten der Umsetzung bezüglich Laufzeit und Präzision evaluieren zu können, wurde in jeder Testreihe die dabei erreichte Präzision festgehalten. Abbildung 2 und 3 stellen die durchschnittlich erreichte Präzision pro Iteration dar. Die Standardabweichung ist dabei als Y-Balken mit angegeben. Bei Betrachtung dieser beiden Abbildungen ist zu erkennen, dass unabhängig vom Schwierigkeitsgrad der Daten ca. die erste Hälfte der Iterationen eine stetige Verbesserung der Präzision erzielt,

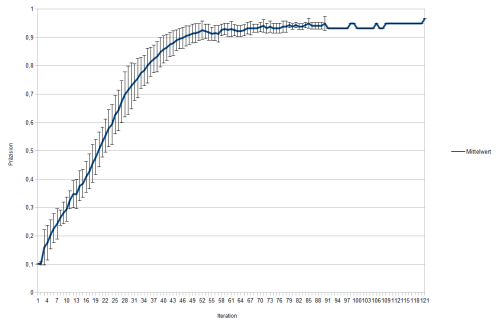

Abbildung 2: Mittlere Präzision, Datensatz 1

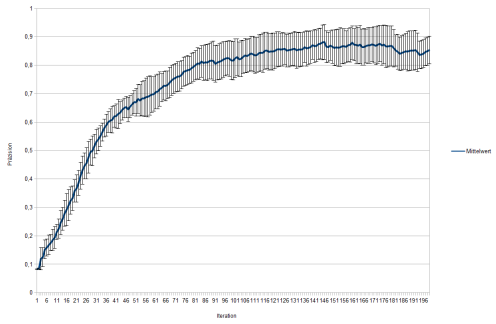

Abbildung 3: Mittlere Präzision, Datensatz 2

während die zweite Hälfte zur „Perfektion“ der gefundenen Grammatik dient. Betrachtet man die Standardabweichungen, so stellt man fest, dass die Testreihen nicht gleichmäßig verlaufen. Dies lässt sich durch lokale Maxima erklären, welche bei Datensatz 2 häufiger auftreten.

Eine Auswertung der Korrektheit gestaltet sich aufgrund des Mangels an Negativbeispielen sehr schwierig. Kontrollversuche wie etwa die Auswertung der Erkennungsrate von Zufallssequenzen sind nur bedingt sinnvoll, da für generierte Sequenzen nicht entschieden werden kann, ob diese Schmerz repräsentieren. Es lässt sich jedoch festhalten, dass durchaus eine Generalisierung der Trainingsdaten stattfindet, da durch die Verwendung von Quantoren und Alternation eine infinite Menge unterschiedlicher Sequenzen erkannt werden kann.

5 Ausblick

In dieser Arbeit wurde ein genetischer Algorithmus zur Induktion regulärer Ausdrücke vorgestellt, der in der Lage ist Sequenzen von Action Units zu erkennen, welche Schmerzausdrücke in der Mimik eines Probanden darstellen. Hierbei wurde ein genetischer Algorithmus implementiert. Grundsätzlich konnte der Klassifikator erfolgreich umgesetzt werden. Allerdings gibt es durchaus Möglichkeiten, den in dieser Arbeit vorgestellten Ansatz noch zu verbessern.

Beispielsweise könnten komplexere Operatoren wie etwa Rückwärtsreferenzen implementiert werden, mit welchen mehr als nur reguläre Sprachen erzeugt werden können. Dies würde demnach die Ausdrucksmächtigkeit der Grammatiken erhöhen.

Zudem enthält die vorgestellte Umsetzung eine sehr große Anzahl an Parametern. Eine optimale Einstellung dieser Parameter konnten in Rahmen dieser Arbeit nicht ergründet werden.

Zur Validierung der Hypothese, dass die Abfolge von Action Units zusätzliche Information gegenüber des bloßen Vorhandenseins der Action Units gibt, um den mentalen Zustand zu erschließen, der einem mimischen Ausdruck zugrunde liegt, wurde ein erstes psychologisches Experiment durchgeführt. Hier sollte bei Gesichtsavataren beurteilt werden, ob diese Schmerz oder Ekel zeigen [Siebers *et al.*, submitted]. Es zeigt sich, dass die Erkennungsraten bei Abfolgen höher liegen als bei gleichzeitiger Umsetzung der Action Units. Darüber hinaus beurteilen Probanden die AU-Sequenzen als natürlicher. Entsprechend kann die Identifikation von Grammatiken über AUs sinnvoll verwendet werden, um natürlichere Animationen für die Interaktion mit Avataren und humanoiden Robotern zu realisieren.

Literatur

[Ekman and Friesen, 1978] Paul Ekman and Wallace V. Friesen. *The Facial Action Encoding System: A Technique for the Measurement of Facial Movement*. Consulting Psychologists Press, Palo Alto, CA, 1978.

[Fasel and Luettin, 2003] Beat Fasel and Juergen Luettin. Automatic facial expression analysis: a survey. *Pattern Recognition*, 36(1):259–275, 2003.

[Javier *et al.*, 2012] Cano Fco. Javier, Macias Fernando, and Milovec Martina. Learning grammars with swarm genetic approach. Master Project Cognitive Systems, SS 2012, 2012.

[Kunz *et al.*, 2007] Miriam Kunz, Siegfried Scharmann, Uli Hemmeter, Karsten Schepelmann, and Stefan Lautenbacher. The facial expression of pain in patients with dementia. *Pain*, 133:221–228, 2007.

[Lautenbacher *et al.*, 2007] S. Lautenbacher, M. Kunz, V. Mylius, S. Scharmann, U. Hemmeter, and K. Schepelmann. Mehrdimensionale schmerzmessung bei demenzpatienten. *Der Schmerz*, 21:529–538, 2007.

[Lucey *et al.*, 2011] Patrick Lucey, Jeffrey F. Cohn, Iain Matthews, Simon Lucey, Sridha Sridharan, Jessica Howlett, and Kenneth M. Prkachin. Automatically detecting pain in video through facial action units. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 41(3):664–674, 2011.

[Mitchell, 1997] T.M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[Schmid *et al.*, 2012] Ute Schmid, Michael Siebers, Dominik Seuss, Miriam Kunz, and Stefan Lautenbacher. Applying grammar inference to identify generalized patterns of facial expressions of pain. *Proceedings of the 11th International Conference on Grammatical Inference*, 21:1–6, 2012.

[Siebers *et al.*, submitted] Michael Siebers, Tamara Engelbrecht, and Ute Schmid. On the relevance of sequence information for decoding facial expressions of pain and disgust – An avatar study. In Dirk Reichardt, editor, *Proceedings of the 7th Workshop on Emotion and Computing - Current Research and Future Impact (KI 2013)*, submitted.

[van Zaanen, 2002] Menno van Zaanen. Bootstrapping structure into language: Alignment-based learning. *CoRR*, cs.LG/0205025, 2002.