Ernesto Elsäßer (8107052), Sophie Eichhorn (6676666) – TINF11B – Wissensbasierte Systeme (Prof. Dr. Dirk Reichardt)



Programmentwurf „Emotionserkennung mittels Evidenztheorie/Dempster’s Regel“

Inhaltsverzeichnis

Einleitung 3

Vorüberlegungen 4

Zu Grunde liegende Daten 4

Algorithmus 4

Rechenbeispiel 5

Entwurf & Umsetzung 6

Struktur 6

Implementierung des Dempster-Shafer-Algorithmus’ 8

Benutzerinteraktion 10

Auswertung 11

# Einleitung

Es soll ein lauffähiges Programm geschrieben werden, welches Emotionen anhand von Features erkennt, beziehungsweise deren Plausibilität angibt. Diese Features werden vorab von einer anderen Software aus einer Aufzeichnung extrahiert und pro Frame in CSV Dateien erfasst.

Die Erkennung und Bewertung der erkannten Emotionen, soll mittels der Evidenztheorie (Dempster’s Regel) vorgenommen werden.

Da der spätere Code ausschließlich englisch ist, werden in der Dokumentation selbst zwar die deutschen Bezeichnungen verwendet, in allen Codebeispielen, Berechnungen, Listings und Diagrammen wird der Nachvollziehbarkeit halber die englische Bezeichnung verwendet.

# Vorüberlegungen

## Zu Grunde liegende Daten

Wie bereits der Aufgabenstellung zu entnehmen ist, liegen Beispieldaten in CSV Dateien vor. Diese haben eine Überschriftenzeile, welche die folgenden Parameter beschreibt und im Folgenden beliebig viele Zeilen mit je einem kommaseparierten Parameter für die fortlaufende Takt-ID, Geschwindigkeit, Tonlage und Schallstärke, wobei die Takt-ID ein ganze Zahl, die Geschwindigkeit eine Fließkommazahl und Tonlage und Schallstärke als String angegeben werden. Diese Strings sind jedoch nicht beliebig, sondern eine Auswahl aus *sehr niedrig, niedrig, normal, hoch* und *sehr hoch.*

Beim Einlesen dieser Dateien ist also zu beachten, dass die erste Zeile ignoriert werden kann und die anderen Zeilen als ein Messwert einer Messung interpretiert werden.

## Algorithmus[[1]](#footnote-1)

Für die Anwendung des Dempster-Shafer-Algorithmus’ auf eine Menge mit Objekten werden zunächst die Eigenschaften der Objekte, sowie deren Gewichtung benötigt. Zur Menge gehören in diesem Fall die sechs Basis Emotionen *Angst, Überraschung, Wut, Freude, Ekel* und *Traurigkeit*. Als Eigenschaften werden die Features aus den Beispieldaten gewählt, so dass für jeden einzelnen Takt genau drei Evidenzen zur Bewertung zur Verfügung stehen.

Da aus der Aufgabenstellung heraus keine Gewichtung der einzelnen Features vorgegeben ist, wird diese hier wie folgt definiert:

G(Geschwindigkeit) = 0.8

G(Tonlage) = 0.6

G(Schallstärke) = 0.7

## Rechenbeispiel

Um den Ablauf des Algorithmus während der Programmierungsphase anschaulicher vor Augen zu haben und am Ende die Richtigkeit der Implementierung zu Überprüfen, wird vorab eine Berechnung zu einem Messwert (E\_020.csv Frame 1) durchgeführt und die Plausibilität zweier Emotionen überprüft. Um dabei eine gewisse Toleranz gegenüber Abweichungen zu gewährleisten, wird für jeden Messwert eine Einheit Abweichung in jede Richtung toleriert.

|  |  |
| --- | --- |
| *speed=3.9* *pitch=niedrig* *intensity= normal* |  |

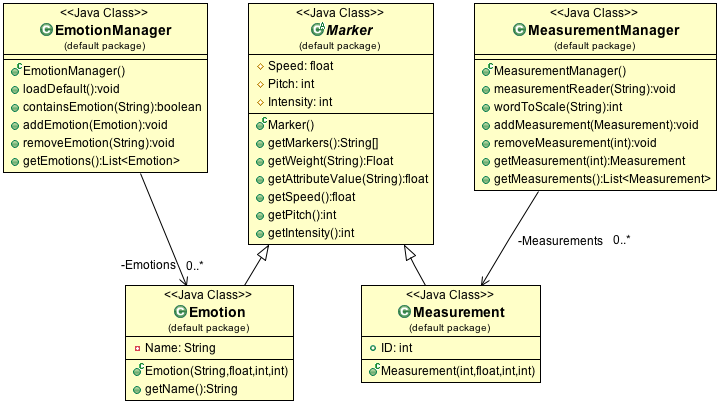
# Entwurf & Umsetzung

Auf Grund persönlicher Präferenzen ist das Programm in Java umgesetzt, womit auch die Entscheidung für Objektorientierung gefällt wurde.

## Struktur

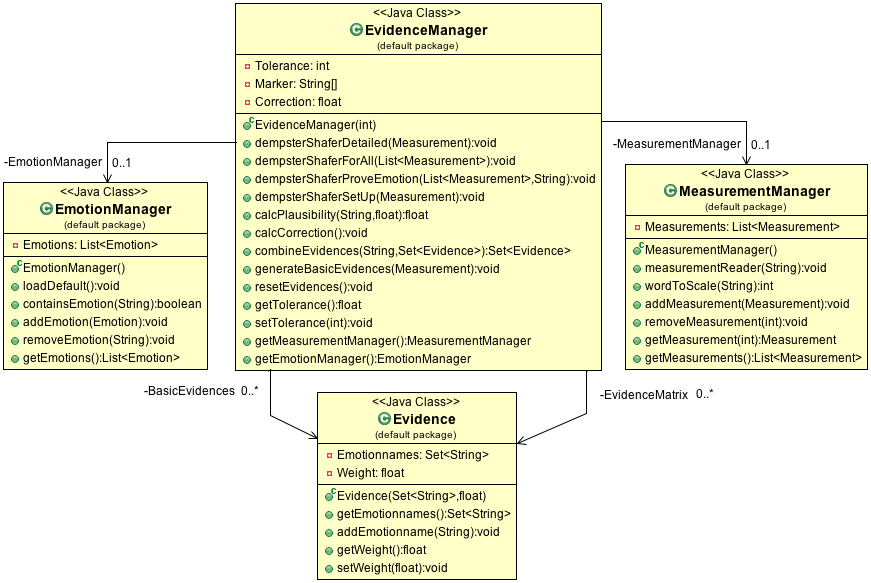
Unabhängig von der späteren Funktionalität gilt es also in diesem Zusammenhang zunächst einmal die notwendigen Klassen zu identifizieren. Die beiden Offensichtlichsten dürften wohl die *Emotionen* und die *Messungen* sein. Während die Struktur ersterer klar sein dürfte - ein Name und Werte für die drei Features - so muss bereits für die Messungen überlegt werden, wie diese zu gestalten sind. Die Frage, die dabei im Raum steht ist, ob Instanzen dieser Klasse einzelne Messwerte oder die ganze Messung repräsentieren sollen. Betrachtet man wiederum die Struktur eines einzelnen Messwertes, so fällt auf, dass dieser einer Emotion sehr ähnelt: Er besitzt ebenso einen Wert für jedes Feature, nur statt eines Namens hat er eine ID, die den gemessenen Frame bezeichnet. Es bietet sich also an, eine Abstrakte Klasse für die Features zu definieren, die dann jeweils von der Klasse für die Emotionen oder Messwerte implementiert wird.

Um die einzelnen Emotionen, beziehungsweise Messwerte zueinander in Bezug bringen zu können, gibt es jeweils eine eigene Manager-Klasse, die die administrativen Aufgaben (hinzufügen, löschen, etc.) beinhaltet.



Um nun den Dempster-Shafer-Algorithmus darauf anwenden zu können, muss eine weitere Klasse für Evidenzen eingeführt werden. Je Evidenz muss es eine Liste von Emotionsnamen und deren Gewichtung geben. Auch für diese wird wieder eine Manager-Klasse benutzt, da im späteren Verlauf eine Übersicht über alle Evidenz Instanzen notwendig werden wird. Da eine Evidenz jeweils mit Instanzen von Messwerten und Emotionen arbeitet, benutzt sie Instanzen der anderen Manager-Klassen.

In dieser Evidenz-Manager-Klasse wird dann letztendlich auch der eigentliche Dempster-Shafer-Algorithmus implementiert.



Den Abschluss bildet die ausführende Klasse, der *EmotionRecognizer*, der auch die *main* Methode enthalten wird. Von dieser Klasse aus werden die Benutzerinteraktion und der Programmablauf gesteuert werden.

## Implementierung des Dempster-Shafer-Algorithmus’

Wenn man die Funktionsweise des Algorithmus auf das Gröbste herunterbricht, so kann man seinen Ablauf in drei Schritte gliedern:

1. Erstellen der Basis Evidenzen
2. Kombinieren aller Basis Evidenzen zu einer Gesamtevidenz
3. Bestimmen der Plausibilität für eine bestimmte Teilmenge, der zur Verfügung stehenden Menge

Bevor jedoch diese Schritte implementiert werden können, muss sich vor Augen gehalten werden, welche Arten von Plausibilitätsprüfungen überhaupt vorgenommen werden sollen. Diese Folgenden ergeben sich aus der Aufgabenstellung, wobei auch durchaus andere Fälle denkbar sind:

* Es soll eine Beurteilung vorgenommen werden, welche Emotion für eine komplette Messung (alle Messwerte eines CSV) am plausibelsten ist
* Es soll nur ein Messwert detailliert betrachtet werden und für diesen Plausibilität aller Emotionen angegeben werden
* Es soll die Plausibilität einer konkreten (erkannten) Emotion für eine komplette Messung bestimmt werden.

Um diesen unterschiedlichen Anfragen gerecht zu werden, müssen zum einen die ersten beiden Schritte des Dempster-Shafer-Algorithmus besonders modular implementiert werden, damit sie von allen Plausibilitätsprüfungen wiederverwendet werden können, zum anderen braucht aber auch jede Überprüfung ihren eigenen dritten Schritt.

In dieser Implementierung werden deshalb die Basis Evidenzen je Messwert erstellt, indem zu jedem einzelnen Feature eine Menge von Emotionen gebildet wird, die zu dem gemessenen Wert des Features „passen“. Diese Beurteilung erfolgt zum einen anhand der Normwerte der Emotionen für die gegebenen Features, zum anderen aber auch an der gewählten Toleranz, die beim Erstellen eines Evidenz-Manager Objektes dem Constructor übergeben werden muss. Mit diesen beiden Werten kann ein Wertebereich bestimmt werden, mittels dessen kategorisiert wird, ob ein Messwert zu einer Evidenzmenge gehört, oder nicht. Die kombinierte Gesamtevidenz wird dann als Liste von Evidenzen behandelt, auf der die jeweilige Überprüfung angewendet wird.

Darauf aufbauend gibt es drei Programm Optionen:

1. Emotionserkennung für alle Messwerte: Das Programm überprüft für jeden Messwert der Messung alle Emotionen und gibt je Messwert die Emotion, bzw. bei Gleichheit die Emotionen, mit der höchsten Plausibilität aus. Während der einzelnen Berechnungen werden auch die Werte verrechnet, sodass am Ende noch eine Statistik über das absolute und relative Vorkommen jeder Emotion ausgegeben werden kann. Diese Option ist vor allem für einen groben Überblick gedacht, während die folgende Option eher dazu geeignet ist Einzelheiten zu betrachten
2. Emotionserkennung für 1 konkreten Messwert: Der Benutzer muss dem Programm eine konkrete Frame ID mitgeben und bekommt im Anschluss die Plausibilität für alle Emotionen, sowie die Korrektur für diesen Messwert ausgegeben. Ein typischer Anwendungsfall für diese Option ist beispielsweise sich Ausreißer anzusehen, die in der Betrachtung aller Messwerte aufgefallen sind, um beispielsweise zu sehen, wie knapp die Klassifizierung der Emotion(en) war.
3. Überprüfung einer erkannten Emotion: Der Benutzer muss dem Programm eine erkannte Emotion mitteilen, dieses gibt dann je Frame die Plausibilität der Emotion an, sowie die durchschnittliche Plausibilität über alle Frames.

## Benutzerinteraktion

Um dem Benutzer das Programm auch zugänglich zu machen, wird für die *main* Methode ein Menü implementiert, welches Output auf der Konsole erzeugt, und die Eingaben des Benutzers auswertet:

------------------------------------------------------------------

| Menu: |

| (1) Load CSV |

| (2) Set Tolerance (Current: 1,0) |

| (3) Emotion Recognition for all Frames |

| (4) Emotion Recognition for one Frame (with Details) |

| (5) Prove Plausibility for a Single Emotion |

| |

| INFO: (a) will abort all submenus and bring you back to this one.|

------------------------------------------------------------------

Jede der Menüoptionen leitet den User zu der damit verknüpften Funktion weiter und fordert ihn ggf. zu weiteren Eingaben auf.

# Auswertung

Zur Ergebnisbewertung muss hauptsächlich die Korrektheit der Ergebnisse überprüft werden. Im Zuge dessen, wurde die Plausibilität einiger Emotionen für einige Messwerte per Hand berechnet und mit dem Programmergebnis verglichen. Da diese Tests alle ausnahmslos fehlerfreie Ergebnisse offenbarten, wird nun angenommen, dass der Algorithmus an sich korrekt implementiert ist.

Des Weiteren sind alle direkt aus der Aufgabenstellung entnommenen Anforderungen, sowie alle impliziten Benutzerfunktionen (CSV-Loader, Menü, etc.) umgesetzt. Außerdem wurde im Backend darauf geachtet, möglichst modular zu arbeiten, um eventuelle Erweiterungen leicht einbringen zu können.

1. Es wird nicht der Dempster-Shafer-Algorithmus an sich behandelt, sondern dessen Anwendung, bzw. Umsetzung in diesem konkreten Fall [↑](#footnote-ref-1)