



# Erkennung einer Emotion anhand von Sprache

## Projektarbeit

des Studiengangs Angewandte Informatik

an der Dualen Hochschule Baden-Württemberg Stuttgart

von

**Mesut Kuscu, Helge Brügner**

Januar 2016

| | |
|---|---|
| **Bearbeitungszeitraum** | 29.09.2015 - 07.01.2016 |
| **Matrikelnummer, Kurs** | 7429032, 3256960, TINF13A |
| **Ausbildungsfirma** | Hewlett Packard Enterprise GmbH, Böblingen |
| **Gutachter** | Prof. Dr. Dirk Reichardt |

# Erklärung

Ich erkläre hiermit ehrenwörtlich:

1. dass ich meine Projektarbeit mit dem Thema *Erkennung einer Emotion anhand von Sprache* ohne fremde Hilfe angefertigt habe;
2. dass ich die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe;
3. dass ich meine Projektarbeit bei keiner anderen Prüfung vorgelegt habe;
4. dass die eingereichte elektronische Fassung exakt mit der eingereichten schriftlichen Fassung übereinstimmt.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Stuttgart, Januar 2016

---

Mesut Kuscu, Helge Brügner

# Inhaltsverzeichnis

# Tabellenverzeichnis

# Listings

# 1 Einleitung

Die Voraussetzungen für die Anwendung der Evidenztheorie ist gegegeben, da folgendes gilt:

1. Jegliche Alternativmengen sind gegeben. Das bedeutet, dass davon ausgegangen wird, dass eine der Grundemotionen stets auf den Benutzer zu trifft und alle Grundemotionen vom Programm erkannt werden können.

2. Jegliche Alternativen schließen sich gegenseitig aus: Falls eine Grundemotion *erkannt* wurde, schließt diese andere aus; d.h. eine Tonaufnahme eines Benutzers beinhaltet immer nur eine der Grundemotionen.

Der Aufbau dieser Arbeit umfasst vorerst das Programmdesign. Hierbei werden die Ausprägungen der Emotionen konkretisiert, fehlende Werte für Konfidenzen hinzugefügt und mögliche Emotionsklassen mit den zugehörigen Merkmalen tabellarisiert. Daraufhin folgt die Dokumentation der Implementation und eine Beschreibung

# 2 Abbildung der Evidenzen

Dieses Kapitel befasst sich mit der Zuordnung jeder Emotion und deren Attribute. Vorerst wird diese Zuordnung der Aufgabenstellung gemäß dargestellt. Daraufhin werden die Werte der Beispieldaten näher betrachtet und die zuvor vorgestellte Zuordnung der Emotionen konkretisiert.

## 2.1 Emotionen

Die Tabelle 2.1 stellt die Einstufung der Emotionen aus der Aufgabenstellung dar. Hierbei fällt auf, dass die Schallstärke nicht ausschlaggebend für die Emotion *Angst* ist und die Sprechgeschwindigkeit keinen Einfluss auf *Traurigkeit* hat. Außerdem fällt auf, dass keine Mittelstufe in den Ausprägungen der Merkmale existiert.

| Emotion | Kürzel | Sprechgeschwindigkeit | Tonlage | Schallstärke |
|---|---|---|---|---|
| Angst | A | Schnell | Hoch | - |
| Überraschung | U | Schnell | Hoch | Hoch |
| Wut | W | Schnell | Hoch | Hoch |
| Freude | F | Schnell / Langsam | Hoch | Hoch |
| Ekel | E | Langsam | Tief | Schwach |
| Traurigkeit | T | - | Tief | Schwach |

Tabelle 2.1: Abbildung der Emotionen nach Aufgabenstellung

## 2.2 Nutzung der Beispieldaten

Das Projekt benutzt die Beispielwerte aus den Dateien `E_004.csv` und `E_004b.csv`. In beiden Dateien sind jeweils 50 Takte angegeben, die zu analysieren sind.

In jedem Takt entstehen jeweils drei Evidenzen. In der Aufgabenstellung geht zwar hervor, dass jeweils immer nur 3 Levels (wenig, normal, viel) von Ausprägungen der jeweiligen Evidenz erkannt werden muss (so wie oben in der Tabelle gekennzeichnet), jedoch sind in den Beispieldaten in der Tonlage und in der Schallstärke weitere Ausprägungen zu finden

(sehr wenig, sehr viel). Die folgende Liste enthält alle Ausprägungen der Merkmale, die in den Beispieldaten zu finden sind.

**Sprechgeschwindigkeit in Silben/Sekunden** langsam, normal, schnell

**Durchnschnittliche Tonlage** sehr niedrig, niedrig, normal, hoch, sehr hoch

**Schallstärke** sehr niedrig, niedrig, normal, hoch, sehr hoch

Da die Anzahl der Ausprägungen in den Beispieldaten größer ist als in der Aufgabenstellung, befasst sich der verbleibende Teil dieses Kapitels mit der Nutzung und Einteilung der Beispieldaten.

## 2.3 Normierung der Ausprägungen der Merkmale

Jegliche Normierungen sind über eine externe Einstellungsdatei statisch festgelegt. Diese können, unabhängig vom Programmcode, vor Programmstart angepasst werden. Der Programmaufbau wird im nächsten Kapitel näher betrachtet.

### 2.3.1 Sprechgeschwindigkeit

Nach einer Analyse der Werte der Sprechgeschwindigkeit ist festgestellt worden, dass die Werte in beiden Dateien zwischen 2 und 7 variieren. Die folgende Tabelle 2.2 stellt die Analyse dar.

| Dateiname | Minimum | Maximum | Mittelwert |
|---|---|---|---|
| `E_004.csv` | 2.6 | 5.9 | 4 |
| `E_004b.csv` | 2.9 | 6.5 | 4.442 |

Tabelle 2.2: Ausprägung der Sprechgeschwindigkeit

Dadurch wurde folgende Norm für die Ausprägung der Sprechgeschwindigkeit aufgestellt, welche für beide Beispieldaten gilt:

**langsam** Der Wert ist kleiner als 2.5

**normal** Der Wert liegt im Intervall: 2.6 - 5.5

**schnell** Der Wert ist größer als 5.6

Die ausgewählten Werte orientieren sich von den Durchschnittswerten aus der Analyse der Beispieldatei. Die Analysefunktion (`printAverageSpeed()`) findet sich auch im Programmcode um spätere Analysen für andere Daten zu gewährleisten.

### 2.3.2 Durchschnittliche Tonlage und Schallstärke

Die Ausprägung für Schallstärke und durchschnittliche Tonlage wird wie folgt zusammengefasst:

**niedrig** falls sehr niedrig oder niedrig

**normal** falls normal

**hoch** falls hoch oder sehr hoch

Durch diese Normierung werden die Beispieldaten an die Aufgabenstellung angepasst: Es sind nur noch drei Ausprägungen zu analysieren.

## 2.4 Konkretisierung der Emotionen

Da nun weitere Ausprägungen durch die Beispieldaten dazugekommen sind, ist es notwendig, die Zuordnung der Emotionen zu konkretisieren. Die folgende Tabelle 2.3 stellt die finale Zuordnung der Emotionen dar. Unterschiede zu der vorherigen Tabelle 2.1 sind mit fetter Schrift makiert. In der Tabelle ist zu erkennen, dass die Ausprägung *Normal* hinzugefügt worden ist. Diese Ausprägung steht in der Meinung des Entwicklerteams im starken Zusammenhang mit der Emotion *Freude*.

| Emotion | Sprechgeschwindigkeit | Tonlage | Schallstärke |
|---|---|---|---|
| Angst A | Schnell | Hoch | - |
| Überraschung U | Schnell | Hoch | Hoch |
| Wut W | **Normal**/Schnell | Hoch | Hoch |
| Freude F | Langsam/**Normal**/Schnell | **Normal**/Hoch | **Normal**/Hoch |
| Ekel E | Langsam | Tief | Schwach/**Normal** |
| Traurigkeit T | - | Tief | Schwach |

Tabelle 2.3: Konkretisierung der Emotionen

## 2.5 Konfidenz der Messungen

Da keine Zuverlässigkeit der Messung in der Aufgabenstellung angegeben ist, müssen diese vom Entwicklerteam fiktiv geschätzt werden. Diese Werte können statisch vor Programmstart leicht in der Einstellungsdatei verändert werden. Die folgende Liste stellt die ausgesuchten Werte der Konfidenz der jeweiligen Messungen dar:

**Sprechgeschwindigkeit** liegt bei 0.6

**Tonlage** liegt bei 0.8

**Schallstärke** liegt bei 0.7

## 2.6 Ergebnisauswertung

Durch Akkumulation der Evidenzen führt das Programm zu einer Schlussfolgerung. Es wird die Emotion ausgewählt, dessen Plausabilität am höchsten ist. Da beide Dateien jeweils 50 Takte beinhalten, sollten im Idealfall insgesamt die 50 Emotionen ausgegeben werden, bei denen im jeweiligen Takt die Plausibilität am höchsten ist.

# 3 Implementierung

Das Programm ist komplett in C++ geschrieben und kann dabei in vier Teile eingeteilt werden.

1. Einlesen der Beispieldaten aus den CSV-Dateien,
2. Erstellen der Evidenzen für Sprechgeschwindigkeit ($m_1$), Tonlage ($m_2$) und Schallstärke ($m_3$),
3. Verarbeiten der Evidenzen (Bilden von $m_{123}$ und Berechnung der Plausibilität) und
4. Ergebnisausgabe (Schreiben in eine CSV-Datei).

Hierbei ist jegliches, relevante Codesegment im Quellcode kommentiert.

## 3.1 Dateistruktur

Das Programm ist in drei Dateien aufgeteilt:

**Main.cpp** Diese Datei verlinkt die anderen beiden Dateien und dient als Startpunkt.

**settings.h** In dieser Datei finden sich jegliche voreingestellten Makros, globale Initialisierungen, Konstanten und Variablen für das Programm. Die Headerdatei wird in Main.cpp und dempster.h eingebunden.

**dempster.h** Diese Datei beinhaltet die Dempsterfunktion, welche von der Main.cpp aufgerufen wird.

Der verbleibende Teil dieser Dokumentation fokussiert sich nacheinander auf jede, genannte einzelne Datei und bespricht die implementierten Funktionen.

## 3.2 settings.h

Die Voreinstellungen des Programmes können in drei Teile eingeteilt werden.

Der erste Teil beinhaltet die Pfade zu den Beispiel-Eingabedaten (`E_004.csv` und `E_004b.csv`) und den jeweiligen Ausgabedaten. Zusätzlich kann noch ein Index angegeben werden, an welchem die Daten in den Dateien starten (um beispielsweise den Header zu überspringen).

Im zweiten Teil können die Konfidenzwerte (die Gewichtung der Evidenzen) festgelegt werden, welche in Kapitel 2.5 besprochen wurden.

Im dritten Teil sind jegliche Werte der Ausprägungen festgelegt. Beispielsweise kann hier festgelegt werden, welcher Wert für den unteren Teil der Sprechgeschwindigkeit ausschlaggebend ist (Kapitel 2.3.1).

## 3.3 dempster.h

In dieser Datei sind alle unmittelbar mit der Evidenzberechnung zusammenhängenden Funktionen zu finden. Die *Dempster*-Funktion erwartet alle Evidenzen ($m_1$, $m_2$ und $m_3$) mit Gewichtung und akkumuliert sie zur finalen Evidenz $m_(123)$. Des Weiteren beinhaltet sie eine Funktion zur Berechnung der *Plausibilitäten*.

Zur Berechnung der Evidenzen wird das folgende Strukt benutzt:

```
1 struct Evidence {
2   Evidence() {}
3   set<string> emotions;
4   double value = 0.0;
5 };
```

Listing 3.1: Strukt zur Evidenzberechnung

Entscheidend sind hier das Set und der double-Wert. *Evidence.emotions* beinhaltet alle zu dieser Evidenz gehörenden Emotionen (beispielsweise *Wut* und *Freude*) und kann auch leer sein, und *Evidence.value* speichert die dazugehörige Konfidenz. Das entsprechende Omega wird mit dem gleichen Strukt gespeichert; *emotions* beinhaltet hierbei alle Emotionen und *value* ist $1 - (KonfidenzderEvidenz)$. Die Evidenz und das dazugehörige Omega werden in einem Vektor mit mindestens zwei Komponenten gespeichert (oder mehr, falls Resultat einer Akkumulation). Dies ist in Diagramm 3.1 erkennbar.

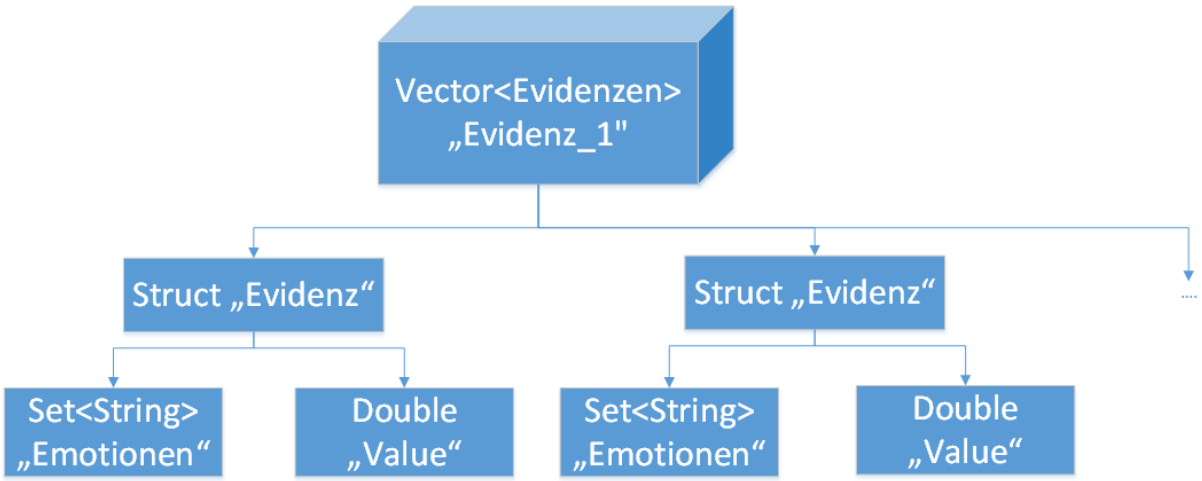


Abbildung 3.1: Struktur der Evidenzen

In der *Dempster*-Funktion wird zuerst aus $m_1$ und $m_2$ die Evidenz $m_(12)$ durch Vereinigung der Sets und Multiplikation der Konfidenzen gebildet, und es entstehen vier mögliche Kombinationen. Anschließend wird die Kombination aus Evidenz von $m_1$ und $m_2$ auf einen Konflikt untersucht; sollte ein Konflikt auftreten, wird $k$ gebildet, die entsprechende leere Menge aus $m_{12}$ entfernt und die übrigen Konfidenzen mit $\frac{1}{1-k}$ korrigiert. Durch Akkumulation von $m_{12}$ und $m_3$ entsteht auf gleichem Wege $m_{123}$, und die resultierenden Mengen werden auf Konflikte untersucht, diese gegebenenfalls gelöscht und der Rest korrigiert. Die Funktion wird einmal pro Takt aufgerufen.

Die Plausibilitätsberechnung geschieht in der Funktion `plausibility()`. Als Eingabeparameter wird hier die Evidenz $m_{123}$ erwartet. Durch Iterieren über alle Teilevidenzen werden für jede Emotion die Teilwerte aufsummiert und in einen neuen Vektor gespeichert. Anschließend wird die größte Plausibilität gesucht und zusätzlich abgespeichert.

## 3.4 Main.cpp

Die Main.cpp Datei beinhaltet die *Main-Funktion*, welche bei Programmstart aufgerufen wird. Zusätzlich zu der Main-Funktion sind hier weitere Funktionen zu finden. Der folgende Ausschnitt zeigt jegliche Funktionssignaturen der Main.cpp befindet sich im Abschnitt *Predefined Functions* dieser Datei:

```
1 double toDouble(string s);
2 void printFile(string filename);
3 vector<string> readFileInVector(string filename);
4 void writeResultsToFile(string filename,
5   vector<vector<Evidence>> evidences,
6   vector<map<string, double>> plausibilities);
```

```
7  void printAverageSpeed(vector<string> data);
8  set<string> getEmotionOfSpeed(double);
9  set<string> getEmotionOfPitch(string);
10 set<string> getEmotionOfIntensity(string);
11 void calculatePlausibilities(string input, string output);
```

Listing 3.2: Predefined classes/functions aus der Main.cpp

Die Funktionen `printFile()`, `readFileInVector` und `writeResultsToFile()` sind für Dateieingabe und -ausgabe zuständig. Mit `printAverageSpeed()` wurde die durchschnittliche Sprechgeschwindigkeit bestimmt. `getEmotionOfSpeed()`, `getEmotionOfPitch()` und `getEmotionOfIntensity()` liefern die Emotionen anhand der Eingabewerte entsprechend der Tabelle 2.3. In der Funktion `calculatePlausibilities()` befindet sich das Hauptprogramm bestehend aus Aufruf der Einlesefunktionen, einer Schleife, die für jeden Takt die *Dempster*-Funktion aufruft und dem Aufruf der Ausgabefunktion.

# 4 Starten der Applikation

Bei der Ausführung der Applikation ist zu beachten, dass die Eingabedateien sich mit dem korrekten Namen im korrekten Ordner befinden. Laut den Standardeinstellungen in der `settings.h`-Datei sollten sich die beiden Dateien im selben Ordner wie das Programm befinden und `E_004.csv` bzw. `E_004b.csv` genannt sein. Außerdem muss das Programm Schreibrechte auf die Ausgabedatei haben; laut Standardeinstellungen sind dies `resulta.csv` und `resultb.csv` im selben Ordner wie das Programm.

Es gibt zwei Möglichkeiten, um das Programm zu starten. Zum Einen kann die schon kompilierte .EXE-Datei auf einem Windows-PC ausgeführt werden, und zum Anderen kann auf Basis des Quellcodes das Programm noch einmal neu kompiliert werden. Wenn `Main.cpp` kompiliert wird, ist darauf zu achten, dass `settings.h` und `dempster.h` verlinkt werden. Alternativ zu einer Kompilierung "von Hand"kann mit Microsoft Visual Studio die Solution (befindlich im Unterordner SSolution") geöffnet werden.