**Modifizierung des Sprachdialogsystem 'ELIZA' und Konzeption der Anbindung an das 'WASABI'-Emotionsmoduls**

**Programmentwurf**

**1 Einleitung**

Dieser Programmentwurf befasst sich mit dem Sprachdialogsystem “ELIZA”, welches auf der Basis der Programmiersprache Python 1966 von Joseph Weizenbaum entwickelt wurde. Dieses Sprachdialogsystem soll mit einem Emotionssystem erweitert werden. Hierzu wird die WASABI (Affect Simulation for Agents With Believable Interactivity) von Christian Becker-Asano als Referenz genutzt.

**2 Rahmenbedingungen**

Folgende Rahmenbedingungen wurden bei dem Programmentwurf eingehalten:

* Python 3 Code Standards <http://google.github.io/styleguide/pyguide.html>
* Verwendung von Github <https://github.com/>

**3 Umsetzung und Erfahrungen**

**3.1 Umwandlung der Regeln auf Deutsch**

*"Stellen Sie die vorhandenen Regeln (siehe doctor.txt) von English auf Deutsch um, indem Sie eine Datei “doctor\_de.txt” anlegen, modifizieren und einbinden. Das Systemverhalten bei Unterhaltungen auf Deutsch soll dem auf Englisch so ähnlich wie möglich sein. Dokumentieren Sie ihr Vorgehen und die gemachten Erfahrungen."*

Folgende Erfahrungen wurden gemacht:

* Trennzeichen ‘%%’ wurden genutzt statt ‘ , da zusammengehörige Passagen Leerzeichen enthalten könnten
* Wörtliche Übersetzung manchmal nicht möglich, darum Umformulierung
* Teilweise gar keine deutsche Übersetzung möglich (Me, myself & I), darum weglassen von Formen
* Andere Possessivpronomen im Deutschen ( *your my -> post: 'du bist meine' 'du bist mein' )*
* Höflichkeitsformen / Im Englischen nur "you": Der Doktor siezt, will aber geduzt werden.
* Rechtschreibfehler abhandeln: Andere Möglichkeit im Englischen als im Deutschen
* Programmatische Keywörter wurden englisch belassen

**3.2 Erweiterung des Regelsets**

*"Erweitern Sie das Regelset in der Art, dass der Computer fragen zu seinem Gemüt (also seinem Emotionszustand) beantworten kann. Ein Beispieldialog könnte folgender Maßen aussehen:*

*1> User: Wie fuehlst Du Dich gerade?   
 2> Computer: Ich bin zur Zeit <oneOf:glücklich|wütend|traurig|...>.   
 3> User: Bist Du <emotion>?   
 4> Computer: Ja, ich bin <emotion>.   
 5> User: Bist Du ein Computer?   
 6> Computer: Was denkst Du über Maschinen?*

*Hinweise:*

*1. Die Antwort in Zeile zwei kann zufällig aus einem Satz von möglichen Emotionszuständen erzeugt werden.   
 2. Die Antwort in Zeile 4 sollte immer “ja” lauten und die gefragte Emotion wiederholen, aber nur, wenn die Variable auch wirklich einen Wert hat, der eine Emotion bezeichnet!   
 3. Die Zeilen 5 und 6 sind bereits ohne Modifikation durch das Original-Regelset “doctor.txt” realisiert.*

*Sie sollten weiterhin so funktionieren wie vorher.*

*Begründen Sie Ihre Design-Entscheidungen und beschreiben Sie Ihre “Lessons learned”. Dokumentieren Sie ihren Python-Code."*

Vorgehen:

* Neues Emotion-Synonym, das genutzt wird um zu erkennen ob der Bot eine Frage nach einer vorhandenen Emotion erhält. (z.B. “Bist du traurig ?”) Dadurch ist auch eine Unterscheidung zu Fragen wie “Bist du ein Computer?” möglich.
* Fragen wie “Wie geht es dir?” oder “Wie fühlst du dich?” werden beantwortet. Zunächst existiert nur ein Platzhalter, der dann nach Aufgabe 2 zufällig durch eine Emotion ersetzt wird (nach Aufgabe 4 durch die Emotion die von der EmotionDynamics-Klasse kommt)

**3.3 Konzipierung der technische Anbindung des WASABI-Emotionsmoduls**

*"Konzipieren Sie eine technische Anbindung des WASABI-Emotionsmoduls, so dass das ELIZA-System um eine eigene Emotionssimulation erweitert wird. Ihre Konzeption sollte folgende Fragen beantworten:*

*1. Was könnte es dem Nutzer des ELIZA-Systems nützen, wenn dieses über eine eigene Emotionalität als Reaktion auf seine Texteingaben verfügt?*

Es könnte von Vorteil sein, dass der Nutzer das Gefühl hat ernst genommen zu werden und nicht mit einem empathielosen Computer zu reden.

Der Doktor würde mit eigenen Emotionen eine stärkere Persönlichkeit vermitteln und somit glaubwürdiger erscheinen.

*2. Welche negativen Auswirkungen auf die HCI könnte eine Anbindung von WASABI haben und wie könnten diese systemisch vermieden werden?*

Der Nutzer könnte sich einen Spaß daraus machen und nicht mehr ehrlich antworten, um zu schauen wie der Computer reagiert. Der Doktor könnte einem nicht mehr weiterhelfen, wenn er beleidigt ist. Es könnte Wertebereiche festgelegt werden, so dass der Doktor nicht so negative Emotionen bekommt. *3. Welche Ein-/Ausgabedaten zwischen dem ELIZA-System und der WASABI-Komponente müssen implementiert werden, wenn WASABI als eigenständiges C++-Programm parallel zum Python- Code ausgeführt würde?*Es müssen lediglich Impulse, die die Emotion beeinflussen können übertragen werden [-100;100]. Als Antwort muss Wasabi eine Emotion zurückliefern. Wann und wie oft diese Emotion abgefragt wird, wird vom Eliza-Programm kontrolliert.

*4. Wie werden im Python-Code sinnvoll Emotionsimpulse aus den Interaktionsdaten abgeleitet, die dann als Eingabedaten für die Emotionsdynamik an WASABI geschickt werden können?*Als sinnvolle Emotionsimpulse können die Antworten der Nutzer bewertet werden und die Positivität und Negativität ihrer Aussage mit einem Zahlensystem bewertet werden. Zusätzlich können bestimmte positiv oder negativ belegte “Trigger”-Wörter beobachtet werden. *5. Wie nutzt der Python-ELIZA-Code die von WASABI erzeugten Emotionen, um die Antworten systematisch und sinnvoll zu modifizieren? (Nennen Sie hier mindestens zwei Alternativen.)"*

Die triviale Methode ist, dass Eliza auf Nachfrage, nach ihrer derzeitigen Gemütslage, entsprechend antworten kann. Eine Zweite Methode wäre eine Veränderung der Anrede, durch die eine Emotion erkennbar wird (d.h. Eliza wird netter oder weniger freundlich). Eine dritte Methode wäre es, proaktiv Fragen zu stellen, warum der User Eliza z.B. traurig oder wütend macht (eher bei negativen Emotionen).

**3.4 Erstellung Python-Klasse 'EmotionDynamics'**

*"Schreiben Sie eine Python-Klasse “EmotionDynamics”, die die WASABI-Emotionsdynamik algorithmisch realisiert. Ziel ist es, den Kern der Berechnungen der Emotionsdynamik (ohne Parametriesierbarkeit aus Dateien) als Python-Klasse zur Verfügung zu haben. Dieser Kern ist hier in C++ programmiert worden. Als weitere Referenz stelle ich gerne die Dateien “Emotion.tz” und “EmotionalAgent.tz” zur Verfügung, in denen die Emotionsdynamik in der Interpretersprache Steve reimplementiert worden ist. Alternativ können Sie die hier verfügbare Standalone Version von WASABI (unter Windows) über Netzwerkkkommunikation an Ihr modifiziertes Python-Programm anbinden."*

Die Python-Klasse EmotionDynamics ist eine exakte Kopie der Implementierung in Steve. Es wurden lediglich alle notwendigen Änderungen zu korrektem Python-Code gemacht und unnötige Passagen wie Variablendefinitionen entfernt. Die Klasse wird importiert, eine Instanz erzeugt und mit der Funktion emoimpulse(<Wert>) werden Impulse an die Instanz geschickt, die diese beeinflussen. Mit get\_emotion() kann man sich eine Emotion von der Komponente holen. Impulse kommen durch Aussagen wie ja und nein, die positiv oder negativ wirken und Eliza damit mit beeinflussen oder durch eine Auskunft des Users, dass er sich selbst eher gut oder schlecht fühlt. Dazu werden an die möglichen Antworten in der Doktor\_de.txt einfach Werte mit dem vorangestellten Keyword “emoimpulseplaceholder” angehängt, der dann vor der Ausgabe entfernt wird. Der Zahlenwert geht als Impuls an das Emotionsmodul. Eine Ausgabe erfolgt in diesem Beispiel nur auf direkte Nachfrage wie es Eliza geht bzw. wie sie sich fühlt.

Alternativ wäre eine Anbindung an WASABI (Standalone Version unter Windows) möglich gewesen. Hier ist der Port unter dem Reiter “Network” individuell einstellbar. Als Default wird hier SenderPort ‘42424’ und als ReceiverPort ‘42425’ angegeben. Eine Anbindung über Sockets ist über Python möglich. Im Localhost-Netz könnten die Werte gesetzt werden: setXPos(var); setYPos(var); setZPos(var); setPValue(var); setAValue(var); setDValue(var); resetForces().

**3.5 Erweiterung der Python-Klasse um das Emotionsdynamik-Konzeptes**

*"Nutzen Sie Ihre in Punkt 4 erstellte Lösung, um den ELIZA-Computer mit einer eigenen Emotionsdynamik wie in Punkt 3 konzipiert auszustatten. Dazu muss diese parallel zum vorhandenen Python-Code laufen und in Abhängigkeit von Eingaben durch den Nutzer mit Emotionsimpulsen im Wertebereich −100,100 (!= 0) angestoßen werden (siehe EmotionenTeil1.pdf, Video auf Folie 27). Die nebenläufig erzeugten Emotionen sollen bei Nachfrage durch den Nutzer (siehe Punkt 2 oben) vom System sprachlich rückgemeldet werden können (wie in Punkt 2 vorbereitet)."*

Mit Hilfe von Multithreading läuft das Emotionsmodul dauerhaft parallel zum eigentlichen Python-Code von Eliza und kann jederzeit durch Impulse beeinflusst werden. Zusätzlich kann sich die Emotion nun von allein über Zeit ändern.

**4 Ergebnis**

Der finale Programmentwurf ist unter GitHub „<https://github.com/AlexMMic/eliza>“ aufrufbar.