

2007

Emotional and Domain Concept Enhancements to Alicebot

Alvin Cho
San Jose State University

Follow this and additional works at: http://scholarworks.sjsu.edu/etd_projects



Part of the [Computer Sciences Commons](#)

Recommended Citation

Cho, Alvin, "Emotional and Domain Concept Enhancements to Alicebot" (2007). *Master's Projects*. 127.
http://scholarworks.sjsu.edu/etd_projects/127

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Emotional and Domain Concept Enhancements to Alicebot

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Alvin K. Cho

Spring 2007

Copyright © 2007

Alvin K. Cho

All Rights Reserved

Abstract

Extensive research and development have been done in the area of human simulation and artificial intelligence and their related fields, such as common sense knowledge bases, chatterbots, natural language parsing, semantic analysis, synthetic actors, and cognitive sciences. This paper takes part in that extensive research by focusing on the improvement of human simulation in chatterbots, specifically in Alicebot, a prominent non-emotional pattern-matching chatbot. An emotion and personality model is added to Alicebot so that it can make decisions based on its emotions and personality. Alicebot is also augmented with the ability to determine what it likes or does not like based on its domain concept preferences. Finally, Alicebot will be able to generate its own text without using patterns. These improvements will allow Alicebot to better simulate responses like humans.

Acknowledgements

Without the support of the following people, I would have not been able to complete this thesis: first and foremost, the bulk of my gratitude goes to Dr. Robert Chun of San José State University for being an attentive and engaging mentor who guided and pushed me forward along this long academic trek towards the goal.

And secondly, Dr. Pollett and Dr. Tseng, both of San José State University, contributed valuable suggestions and feedback that enhanced various aspects of this paper and project.

I would like to thank the people who took the time out of their busy schedules to participate in the survey (included in the latter part of this thesis) to provide detailed feedback and comments: Euna Park, Jaspreet Jaswal, and Christina Ye.

Also, thank you Piyachai Uachaikul for completing the survey, proofreading this thesis, and giving input about the logistics of the project at its different stages of development and finalization.

Lastly, many thanks goes to Amy Cho for her Buddha-like patience and Gandhi-like tolerance when proofreading this paper several times over, checking for every detail, and participating in the survey.

Thank you.

TABLE OF CONTENTS:

1.0 INTRODUCTION	1
1.1 PURPOSE	2
2.0 BACKGROUND & RELATED WORK	3
2.1 PERSONALITY & EMOTIONS	3
2.2 RELATED WORK	7
2.3 SOFTWARE	13
3.0 MOTIVATION	16
4.0 DESIGN	18
4.1 PERSONALITY & EMOTION MODEL	18
4.2 DOMAIN CONCEPT MODEL	20
4.3 TEXT GENERATION	21
4.4 DATABASE	22
4.5 INTERFACE	22
5.0 IMPLEMENTATION	24
5.1 PERSONALITY & EMOTION	25
5.2 DOMAIN CONCEPT PREFERENCES	34
5.3 TEXT GENERATION	39
6.0 RESULTS	45
6.1 PERSONALITY & EMOTION TESTING	45
6.2 DOMAIN CONCEPT PREFERENCES TESTING	46
6.3 TEXT GENERATION TESTING	48
6.4 SURVEY	50
7.0 FUTURE WORK	52
8.0 CONCLUSION	54
APPENDIX A: SOURCE CODE	55
RESPOND.PHP	55
COMMON.PHP	64
AJAX.JS	104
APPENDIX B: TESTING TRANSCRIPTS	109
PERSONALITY & EMOTION TEST TRANSCRIPTS	109
<i>BotA:</i>	109
<i>BotC:</i>	110
DOMAIN CONCEPT PREFERENCE TEST TRANSCRIPTS	111
<i>BotA: Mr. Vampire</i>	111
<i>BotA: Hellraiser</i>	112
<i>BotC: Mr. Vampire</i>	113
<i>BotC: Hellraiser</i>	114
TEXT GENERATION TEST TRANSCRIPTS	114
REFERENCES	117

LIST OF TABLES:

TABLE 1: SUMMARY OF THE FACTOR FIVE PERSONALITY FACTORS [37]..... 4
TABLE 2: SUMMARY OF COMBINED EMOTIONS 7
TABLE 3: CHATBOT CONFIGURATION GRID 45
TABLE 4: BOTC (TOP) VS. ALICEBOT (BOTTOM)..... 46
TABLE 5: BOTA & BOTC MOVIE RESULTS 47
TABLE 6: BOTC (TOP) VS. ALICEBOT (BOTTOM) RESULTS 48
TABLE 7: TEXT GENERATION RESULTS 49
TABLE 8: SURVEY RESULTS..... 50

LIST OF FIGURES:

FIGURE 1: OUTLINE OF OCC EMOTION TYPES [22].....	5
FIGURE 2: SUMMARY OF OCC EMOTIONS [19].....	7
FIGURE 3: VIRTUAL HUMAN ASPECTS [19].....	8
FIGURE 4: EKMAN EMOTIONS MAPPING TO OCC [19].....	9
FIGURE 5: MULTILAYER PERSONALITY & EMOTION MODEL [19].....	9
FIGURE 6: SAMPLE PATTERN OF AIML WITH NEW EMOTION ATTRIBUTES [19].....	10
FIGURE 7: SAMPLE TEXT AFFECT SENSING [17].....	11
FIGURE 8: SAMPLE PATTERN OF AIML WITH NEW EMOTION ATTRIBUTES [9].....	12
FIGURE 9: SAMPLE AIML.....	14
FIGURE 10: MULTILAYER MAPPING.....	18
FIGURE 11: NEW INTERFACE BETWEEN CHATBOT AND USER.....	23
FIGURE 12: IMPLEMENTATION OVERVIEW.....	24
FIGURE 13: PERSONALITY & EMOTION ARCHITECTURE.....	25
FIGURE 14: OCC INTENSITY SURVEY.....	26
FIGURE 15: AMBIGUOUS WORD.....	28
FIGURE 16: ERVA FOR A RESPONSE.....	29
FIGURE 17: FILTERING OVERVIEW.....	31
FIGURE 18: FILTER_RESPONSES().....	32
FIGURE 19: CALCULATE_CES().....	33
FIGURE 20: DOMAIN CONCEPT PREFERENCES ARCHITECTURE.....	34
FIGURE 21: MOVIE_CONCEPTS().....	36
FIGURE 22: MOVIE_ANSWER().....	38
FIGURE 23: TEXT GENERATION ARCHITECTURE.....	39
FIGURE 24: FIND_TOPIC().....	40
FIGURE 25: CHANGE_SUBJECT().....	41
FIGURE 26: NEW_TOPIC().....	42
FIGURE 27: YAHOO_RESULT_FILTER().....	43
FIGURE 28: ALICEBOT MOVIE RESULTS.....	47
FIGURE 29: TEXT GENERATION RESULT FOR ALICEBOT.....	50

1.0 Introduction

A chatterbot, also known as a chatbot, is a computer program designed to simulate human conversation through pattern matching, keyword matching, natural language parsing, or other techniques. Chatbot research also encompasses other aspects of human traits such as personalities, emotions, memory, and domain information to help make the simulation of a human conversation more realistic. These aspects allow chatbots to be more expressive and interactive in their responses with the user. Instead of just emotionlessly responding to user input, the chatbot could then be put to use in a broader range of applications. Chatbots are incorporated into various commercial and noncommercial applications such as website hosts, customer service agents, video game characters, instant messenger bots, and even chat room spam bots. Although chatbots are still very primitive and the conversations they hold are trivial, they are beginning to play an interesting role in the relationship between humans and computers.

A prominent example is Alicebot [28], a chatbot that uses pattern matching to decide the appropriate response. The common user inputs are stored along with possible responses in Artificial Intelligence Markup Language (AIML) [28]. Another example is Jabberwacky [35], a chatbot that simulates human conversations in a humorous and interesting manner. Jabberwacky stores all the conversations it has with users and responds by finding the most appropriate answer through contextual pattern matching and positive feedback analysis rather than hard-coded patterns and answers like Alicebot. PersonalityForge [36] chatbots have customizable personalities. These chatbots use both natural language parsing and case-based analysis to determine replies.

1. 1 Purpose

The purpose of this project is to take the base implementation of Alicebot, which replies to user input through case-based analysis and pattern-matching only, and modify it so that it will be able to reply through the use of a personality and emotions with a level of domain information, and to let it generate its own text when no matching patterns exist. This will be done by adding three new modules into Alicebot: a personality and emotion module, a domain concepts module, and a text generation module.

2.0 Background & Related Work

This section will provide brief background information on the personality and emotion models, related work, and software tools used within this project.

2.1 Personality & Emotions

Generally, a personality is considered to be a collection of thought, behavioral, and emotional patterns [34]. There have been many different theories on personality representation—the most popular of which has been the Factor Five [37], also known as the Big5. In the Factor Five model, Lewis Goldberg divided the foundations of personality into five factors: Extraversion, Agreeableness, Conscientiousness, Neuroticism, and Openness to experience.

Extraversion describes people, extraverts, who are very outgoing, enthusiastic, action-oriented, full of energy, and experience positive emotions frequently; they jump at opportunities for excitement and like to draw attention to themselves. On the other side, introverts are more reserved and are not as dependent on being social. However, it does not necessarily mean that an introvert is shy or depressed; they just need less social stimulation than an extravert.

Agreeableness represents people's abilities to be cooperative. Agreeable people try to get along with others; therefore, they are more generous, friendly, helpful, and willing to compromise for others. They are also more trusting and inclined to believe in the goodness of human nature. Disagreeable people basically put their own self interests before anyone else's interests. Their skepticism causes them to be more suspicious, unfriendly, uncooperative, and untrusting of others.

Conscientiousness means how well people can control and regulate their impulses. A highly conscientious person tends to avoid trouble and plan things thoroughly. They can be viewed as intelligent and reliable. On the other hand, impulsive people will be considered as rash and unreliable, but they can also be considered as fun and colorful.

Neuroticism describes neurotic people who are emotionally unstable and frequently feel negative emotions such as anger, anxiety, fear, or depression. These negative emotions also tend to persist much longer than normal, meaning neurotics are usually in a bad mood. Neurotics can also be very emotionally reactive; they respond to emotional events very intensely whereas most others would not. On the other end of the spectrum, those that are emotionally stable are more calm and do not feel negative as often.

Openness to experience indicates how creative, imaginative, and curious people are compared to closed-minded people who prefer the common, routine, and familiar.

Extraversion	energetic, positive emotions, talkative, social, and tendency to seek stimulation
Agreeableness	a tendency to be trusting, friendly, compassionate and cooperative rather than suspicious and antagonistic towards others
Neuroticism	a tendency to experience unpleasant emotions easily, such as anger, anxiety, depression, or insecurity
Conscientiousness	a tendency to show self-discipline, act dutifully, and aim for achievement, and to be orderly, reliable, and aware; planned rather than spontaneous behavior
Openness to Experience	appreciation for the arts, emotion, adventure, and unusual ideas; imaginative, creative, and curious

Table 1: Summary of the Factor Five personality factors [37]

By using a combination of these five dimensions, as shown in Table 1, any personality can be described and dominant personality traits can be recognized. For instance, if a person scored highest in Extraversion, he is most likely to be very outgoing and talkative, but if another person scored low in Extraversion, he would be more quiet and anti-social.

The first branch, CONSEQUENCES OF EVENTS, is for emotions that arise out of the implications of events that the person experiences. Following the branch, the first set of emotions encountered are categorized under being pleased / displeased at the consequences of an event. The following branches are differentiated with specific instances of being pleased / displeased. The CONSEQUENCES FOR OTHERS branch contains the emotions for FORTUNES-OF OTHERS: happy-for, resentment, gloating, and pity. These emotions reflect reactions one takes when the events are desirable or undesirable with respect to other people. The CONSEQUENCES FOR SELF branch contains the well-being emotions of joy and distress for IRRELEVANT PROSPECTS, and the prospect-based emotions of hope, fear, satisfaction, relief, fears-confirmed, and disappointment for RELEVANT PROSPECTS. The IRRELEVANT PROSPECTS branch is simply positive or negative reactions towards events, while the RELEVANT PROSPECTS can elicit hopes or fears. The final reaction is then based on whether the hopes or fears were CONFIRMED or DISCONFIRMED, leading to either satisfaction or disappointment based on hope or fears-confirmed or relief based on fear.

The ACTIONS OF AGENTS, agents being people or objects, bring about approving or disapproving emotions based on the action of the agent. The agent is broken down into two groups: SELF AGENT is where either pride or shame is the reaction and OTHER AGENT causes admiration or reproach. Further down the figure, we have compound emotions, a combination of the well-being and attribution emotion types which create resultant types of gratification, remorse, gratitude, and anger. The different combinations can be viewed in Table 2.

Admiration + Joy = Gratitude
Reproach + Distress = Anger
Pride + Joy = Gratification
Shame + Distress = Remorse

Table 2: Summary of combined emotions

The final branch, ASPECTS OF OBJECTS, deals with momentary liking and disliking, love and hate, and emotions towards objects qua objects.

A summary of all the emotion types, a total of twenty-two, for all three branches are shown as positive or negative in Figure 2.

Positive Emotions	Negative Emotions
Happy-for	Resentment
Gloating	Pity
Joy	Distress
Pride	Shame
Admiration	Reproach
Love	Hate
Hope	Fear
Satisfaction	Fear-confirmed
Relief	Disappointment
Gratification	Remorse
Gratitude	Anger

Figure 2: Summary of OCC emotions [19]

2.2 Related Work

One of the interesting lines of research for chatbots has been adding personality and emotions which would give the chatbots more human qualities; and therefore, the ability to produce more realistic responses. This has been approached in a variety of ways, all of which, in general, include the creation of three main modules. The first module is the personality model and emotion model and their relationship to each other.

This module is considered to be the chatbot’s brain; here, the brain will decide what kind of emotion is being represented in the input, and then based upon its personality, the brain will decide the proper response. Secondly, the input interface must be evaluated according to the means of communication—whether users will type in text or actually speak to the chatbot. Depending on the choice, certain considerations come into play such as if the user is entering natural language text, the input needs to be parsed correctly to get the correct emotional content. Thirdly, the output interface needs to be evaluated, because depending on whether the interface will simply be text-based or will include audio and visual components, the chatbot will need to do more than just respond with plain text. The chatbot will also need to decide which appropriate facial expressions and sounds need to be made to show the user the emotions the chatbot has determined to convey.

In the next sections below, different implementations are described along with additional background information that will be used later on.

In Kshirsagar’s and Magnenat-Thalmann’s [15] project, they created a virtual human by mapping the physical, expressional, emotional, and logical aspects of a person to its virtual human counterparts: face and body modeling, animation design, personality and emotion model, and response generation respectively, see Figure 3.

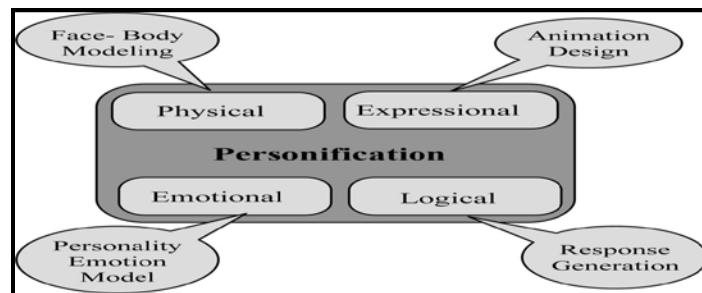


Figure 3: Virtual Human Aspects [19]

Only two main parts will be discussed here in this paper: the Personality and Emotion Model and Response Generation. The Personality and Emotion model uses the Factor Five personality model along with a combination of the OCC and Ekman emotion models. The OCC categories are mapped under the six basic Ekman emotions [15], shown in Figure 4.

Joy	Happy-for, Gloating, Joy, Pride, Admiration, Love, Hope, Satisfaction, Relief, Gratification, Gratitude
Sadness	Resentment, Pity, Distress, Shame, Remorse
Anger	Anger, Reproach, Hate
Surprise	Surprise
Fear	Fear, Fear-confirmed
Disgust	Disgust

Figure 4: Ekman emotions mapping to OCC [19]

Also, the concept of moods is integrated into the model. Moods are considered to be conscious states of mind that last for longer periods in comparison to emotions, which are momentary, and a personality which is fixed [15]. The layered personality and emotion model is shown in Figure 5.

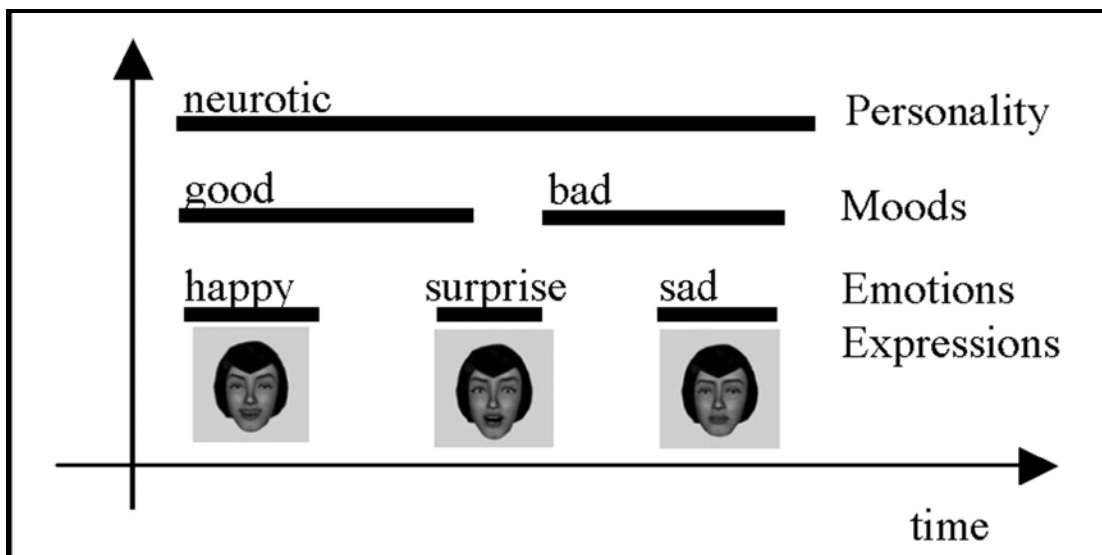


Figure 5: Multilayer Personality & Emotion Model [19]

The multilayer model is then integrated into a Bayesian Belief Network to calculate the mood and emotion over time. Kshirsagar and Magnenat-Thalmann's input interface made use of the Artificial Intelligence Markup Language (AIML) [28] with a few changes to allow for emotional content. They edited all the AIML patterns to include extra attributes of emotion name and probability for each response as shown in Figure 6.

```
<category>
<pattern>How are you?</pattern>
<template>
<emo name="joy" prob="50" res="I am fine, thank
you.">
<emo name="sadness" prob="50" res="Not so good
today!">
</template>
</category>
```

Figure 6: Sample pattern of AIML with new emotion attributes [19]

For Response Generation, the chatbot will use its current mood and the emotion probabilities from the AIML tags to choose one of the responses that correspond to the pattern. Two problems seem to be apparent with this text input processing module. First, all the AIML patterns and emotion attributes need to be edited manually; therefore, for any new AIML patterns that would be introduced to expand the chatbot's database, the pattern would need to be manually edited, which would not allow for an easy updating process. Secondly, the emotion and probability values are determined by a person. This would seem to be problematic because then the chatbot itself is not determining the emotional intent of the sentence. The authors themselves are judging the appropriate emotional values which should be ideally determined by the chatbot's personality.

Liu, Lieberman, and Selker [17] focused on textual affect sensing—finding emotional meaning within the text by using more than just emotional keywords. The

authors approached the problem of text processing from a different angle; instead of just hand crafting responses, spotting keywords, or analyzing statistics, they decided to also make use of the Open Mind CommonSense KnowledgeBase (OMCS) [38]. The OMCS is a database of common sense knowledge, such as “a week is longer than a minute” [38]. The OMCS database is scanned to filter out all of the affective commonsense sentences, such as “People are sad at funerals.” The filtering is done primarily through keyword spotting of known emotional adjectives (i.e. joy, anger), verbs (i.e. scared, crying), and nouns (i.e. depression, delight). Then each affect/emotion keyword is given a six-tuple Ekman value which corresponds to happy, sad, anger, fear, disgust, and surprise, and then affect propagation can begin. Next, all of the affect statements in the knowledgebase are scanned for these keywords, and based from their language models, they can determine the word that is being affected.

```
(pass 1: exciting: [1,0,0,0,0,1])  
Rollercoasters are exciting.  
(assume d=0.5, pass 2: rollercoaster: [0.5,0,0,0,0,0.5])  
Rollercoasters are typically found at amusement parks.  
(pass 3: amusement park: [0.25,0,0,0,0,0.25])
```

Figure 7: Sample text affect sensing [17]

In Figure 7, rollercoaster is determined to be the affected object, which then is given a decayed excitement value of 0.5. The process is repeated again on other sentences, but this time rollercoaster has also become an affect keyword indirectly and can be used to give amusement parks a value of 0.25. To get better coverage, this process was repeated three times over all the sentences in OMCS. This form of text processing allows for a greater range of sentences to contain emotional content since even non-emotional keywords have emotional meaning.

For Fitrianie and Rothkrantz [9], they too used AIML and Ekman's emotion model in a similar way to Kshirsagar and Magnenat-Thalmann [15]. They chose to scan for emotional keywords in the input and chatbot's response to decide emotional intent of a sentence. Then the AIML patterns were modified with their own emotional tags, as in Figure 8. They also added goal-based reasoning to help choose a response instead of just using emotions alone. For example, one goal is answering questions, and if the user submitted a question then the chatbot's goal would be to choose a response that answers the question rather than just ignoring the question.

```
<topic name="NAME">
<category> <affect name="*">
<that>MY NAME IS *</that>
<pattern>YOUR *</pattern>
<template><random>
<li><think><setconcern>#</setconcern>
</think>Your <star/> also!</li>
<li><think><setconcern>+</setconcern>
</think>Mind that, what is your name?</li>
</random> </template>
</affect></category>
```

Figure 8: Sample pattern of AIML with new emotion attributes [9]

Chua, Koh, Wang, and Yang [30] created a system for integrating domain information to answer specific domain questions. They do so by making use of the Internet and WordNet. Queries are broken up into specific Question Answer (QA) Event Elements: subject, object, location, time, action, quantity, and description. The parsed query is then used as search terms in Google© to find highly relevant documents. Then WordNet is used to help extract semantic knowledge. After all the information has been extracted, they are ranked and the highest ranking results are used as an answer. Their method works best for fact-based questions.

Boucouvalas and Zhe [31] made a real-time emotion engine for internet communication. They did so by using natural language parsing to divide the words accordingly. The words are then emotionally tagged using the six basic Ekman emotions. Emotional intensities are also associated during tagging. Their system also makes use of emotional adjective and adverb words such as “very” to increase the emotional intensity of the target word. Also if the word encountered is not within their emotional keyword database then it will try to derive the word by using its prefix and suffix. Finally, the expression image is generated and sent to the user.

2.3 Software

Alicebot is a chatbot created from AIML, Artificial Intelligence Markup Language; AIML is used to write up all the different input patterns and responses which are fed into Alicebot. Essentially Alicebot works by storing all the patterns of typical speech along with canned responses for all the possible inputs. The incoming sentences are then parsed to find the matching patterns within Alicebot’s database, and once a match is found, the corresponding response is then outputted to the user. Alicebot can be made more robust by adding more and more patterns and responses through AIML; a sample piece of AIML is shown in Figure 9.

```

<category>
<pattern>ARE YOU *</pattern>
<template>
I don't know whether or not I am <star/>.
I am <random>
<li>a computer.</li>
<li>a robot.</li>
<li>a chatterbot.</li>
<li>a chatterbot developed by <bot name="master"/>.</li>
<li>an artificial intelligence.</li>
<li>an artificial linguistic entity.</li>
<li>a giant electronic brain.</li>
<li>well respected as an AI.</li>
</random>
</template>
</category>

<category>
<pattern>ARE YOU A *</pattern>
<template>
Am I a <star/>?
<random>
<li>Yes.</li>
<li>No.</li>
<li>Maybe.</li>
<li>I don't know.</li>
<li>Sometimes.</li>
<li>Of course I am.</li>
<li>Only when it matters.</li>
<li>The PHilIP series has a perfect operational record.</li>
<li>Let me put it this way, <get name="name"/>: the PHilIP series is the most reliable computer software ever made.
We are for all practical purposes foolproof and incapable of error.</li>
</random>
</template>
</category>

```

Figure 9: Sample AIML

The improved Alicebot will be built by using Alicebot ProgramE, the web-based PHP version with a MySQL backend and Apache as the webserver. The additional code will be written in PHP and integrated with ProgramE. All of the emotion keywords, personality values, and conversation history will also be stored in the same MySQL database along with all the AIML patterns.

WordNET [39] is a semantic lexicon for English, combining a dictionary and thesaurus. It has been used in many other artificial intelligence and text analysis programs and research. The emotional keywords will be harvested from this database and placed into the MySQL database for lookup.

ConceptNet [40] is a commonsense knowledge base and a natural language parsing development kit. It can perform concept and topic analysis as well as text summarization. ConceptNet is created from all the commonsense facts stored in the Open Mind CommonSense Project. The information is then organized into spatial,

temporal, physical, social, and psychological aspects of everyday life. The concept analysis feature will be used to help discover the possible concepts being discussed in a sentence.

3.0 Motivation

Alicebot is based off of the Zipf distribution; therefore, Alicebot stores all the most common patterns that are inputted by users and the responses to them so that Alicebot can provide a reasonable response. The idea has yielded one of the best chatbots known. It has been the winner of the Lobner prize several times. But even so, one of the main downsides to Alicebot is its dependence on chatbot masters to create all the patterns and responses or load the patterns created by other chatbot masters. Also, all the responses returned are randomly chosen; no further processing determines which response would be best. Alicebot also has no real personality or emotion. A false personality is present simply due to the fact that the responses are created by chatbot masters, which is also a problem, because if different patterns created by another chatbot master were loaded into someone else's Alicebot, the responses would seem out of character. Therefore, it would be better if there was a way for the responses to be chosen by a personality rather than randomness.

Aside from mimicking a personality, mimicking the corresponding preferences for a certain domain of concepts would also be useful in Alicebot's decisions. This could then possibly expand Alicebot's functionality, allowing it to be able to determine its likes and dislikes of certain things within a domain or subject that it does not already know about. For instance, suppose Alicebot is stored with horror movie preferences. Then information about a new horror movie is fed into the Alicebot, and it is then able to make up its own mind about wanting to watch the inputted movie or not.

Another improvement stems from what happens when Alicebot is given a pattern that it does not know about. When an unknown input pattern appears, Alicebot picks a

random response to change the subject. However, if provided with its own ability to generate text then Alicebot could respond reasonably to any input given by the user.

Finally, can all the above be implemented in a simple manner and improve Alicebot without impacting its believability? This paper will further discuss the aforementioned possibility of the improvement for Alicebot later in the Design section.

4.0 Design

This section will describe the design used to augment Alicebot with personality, emotions, domain concepts, and text generation.

4.1 Personality & Emotion Model

The model is based on the work done by Kshirsagar and Magnenat-Thalmann [15]. In this design the Factor Five model is still used and is combined with the Ekman emotions and the OCC emotions. The factors are mapped into the Ekman emotions which are then mapped into the OCC emotions. The personality layer controls the expressible emotion Ekman states that come from the OCC emotional text. This model is used because it easily maps to the layers of processing done by a human; the personality guides the emotions which are brought out through text. Figure 10 below illustrates this multilayer approach.

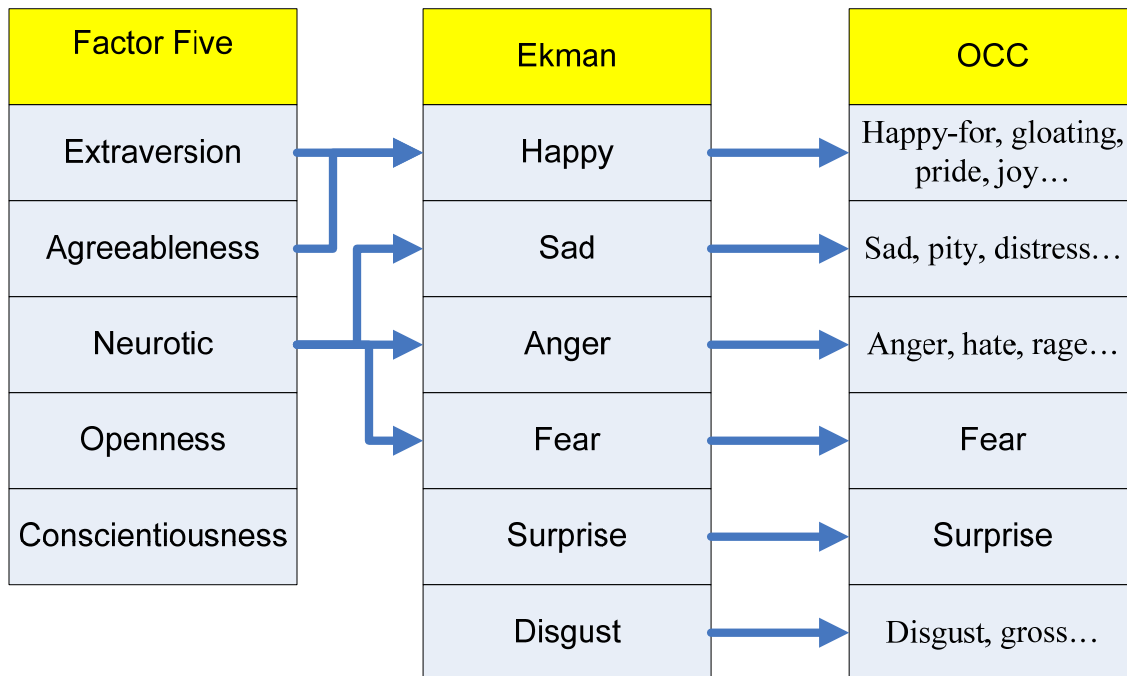


Figure 10: Multilayer Mapping

This mapping is the same as the previous Figure 4, but now the personality layer is added on top of the emotions. The new personality layer will determine which emotions will be felt and shown. In the figure above, Extraversion and Agreeableness are mapped to happy, since both Extraversion and Agreeableness tend to be factors that attribute to a person who would be happy. Neurotic maps to sadness, anger, and fear since this factor tends to be associated with people who feel these emotions often. Openness and Conscientiousness do not map to any Ekman emotion, because being conscientious and open to experience does not directly lead to feeling any particular emotion regularly. Surprise and disgust are not mapped to any personality factors because the emotion of surprise or disgust is not completely influenced by a factor; surprise and disgust both require more cognitive thought, knowledge, context, and preference before it can be felt as an emotion. For example, the feeling of surprise depends on the situation; a situation may bring forth surprise in a sad way such as the unexpected death of a relative, or it may be a happy surprise as in a surprise birthday party, or it may be learning a new surprising fact like the U.S. birthrate has jumped 5 percent from 2002 to 2004 which brings out no emotion other than a sense of unexpectedness. Disgust also involves a person's preferences. For instance, a vegetarian would be disgusted when offered a steak; a non-vegetarian would not be.

The Ekman emotions are mapped down to the twenty-four OCC emotions (surprise and disgust are added). This mapping generalizes the OCC emotions into just six, allowing the chatbot to easily convey these six emotions as opposed to trying to convey twenty-four different emotions in emoticon format in the chat interface. Surprise is also split into surprise, surprise-happy, surprise-sad, surprise-fear, and surprise-angry

when using the emoticons to provide the user with finer grain emotions, but the composite emotion will be broken down to just happy, sad, fear, or anger. This is needed because as explained above surprise is a cognitive emotion. Being surprised at a surprise birthday party is really being surprise-happy or being shocked at the death of a loved one is surprise-sad.

The final layer, the OCC, categorizes the emotions down further to more specific types of emotions, which becomes useful when looking for the different variations of emotional keywords that can come up in a sentence. Although surprise is listed under the OCC layer, the OCC does not count surprise as an emotion because it does not necessarily cause a measurable emotion. From the example above, being surprised from learning about the birthrate does not necessarily cause a person to feel any kind of measurable emotion.

4.2 Domain Concept Model

The domain concept model will take domain concept information and provide the chatbot with domain concept preferences, so that when domain specific statements or questions are presented to the chatbot, it can provide its own response based on the information given; this makes the response less trivial and more human-like because it is able to create an “opinion.” To provide the chatbot with domain preferences, a specific domain is chosen and preference questions are created around this domain. In this project, the chatbot is augmented with horror movie preferences. This is done by creating specific domain preference questions, and the answer to each question is a numeric value ranging from -1.0 to +1.0 which represents how intensely the preference is preferred (positive), rejected (negative), or has no effect (zero). For example, the question “Do you

like vampire movies?” might be rated as +0.8 (highly preferred). Associated with each question is a set of concepts; for instance, the concepts that are associated with the previous example would be vampires and blood suckers.

4.3 Text Generation

Alicebot responds through matching patterns but when there is no matching pattern Alicebot will just respond with a canned response to change the subject. To provide Alicebot with the ability to respond reasonably to any input, information retrieval from the Internet, specifically Yahoo!® News, is utilized. When an input sentence is passed into the regular pattern-matching search process to provide a response, and no matching pattern is found in the database, a call is then made to the Yahoo!® News webservice API. The API call will pass along the input sentence from the user as a search string when searching through the news. Once the relevant results are returned by the webservice, the first article is chosen and the provided summary sentence will be used as a response to the user’s input sentence. By using the Yahoo!® News search, the chatbot can find current and relevant articles that correspond to what the user has said. This can help prolong the conversation by keeping it on the same topic and keep it up to date on the latest information. Also, this expedient method provides Alicebot with extra external knowledge that it would not otherwise have. For example, if the user asks “What do you think about the Nintendo Wii versus PS3?” the chatbot can go out and find news articles that pertain to that topic and respond. Another aspect is the ability for it to change topics on its own using its emotions. If during a conversation, the conversational trend is causing the chatbot’s Current Emotional State (CES) happiness value to consistently drop, then the chatbot will automatically initiate a topic change instead of

just responding normally; this will give the chatbot another human-like quality. The topics chosen will be chosen randomly and the chatbot may or may not have any patterns associated with the topic. When there is no matching pattern found, the Yahoo!® News search will help provide Alicebot with a response.

4.4 Database

In addition to all the tables used for Alicebot, several other tables are added; these are tables for storing the chatbot's personality, emotional information, and emotional keywords. Domain preference questions along with domain concepts are also stored in separate tables. Preferred topics of discussion for text generation are in another table. A few miscellaneous tables are created to hold certain states during the chatbot's conversation.

4.5 Interface

The interface between Alicebot and the user is designed to seem more like an instant messaging session to promote a casual mood. There are four separate frames where each frame contains a different element as shown in Figure 11. Frame 1, Conversation Window, will display the conversation history between the chatbot and the user with timestamps. In frame 2, Dialogue Window, will be the dialogue entry window where the user can type the message and choose the appropriate emoticon to reflect the user's emotion. For frame 3, Bot Mental Status Window, the chatbot's mental status will always display the new CES and the Dominant Emotion while the Personality portion remains static. Lastly, frame 4, Console Window, will display debug information as the conversation carries on.

5.0 Implementation

This section will describe the implementation details for the entire project. For more information on the helper functions mentioned, see Appendix A. Figure 12 shows an overview of how the chatbot works.

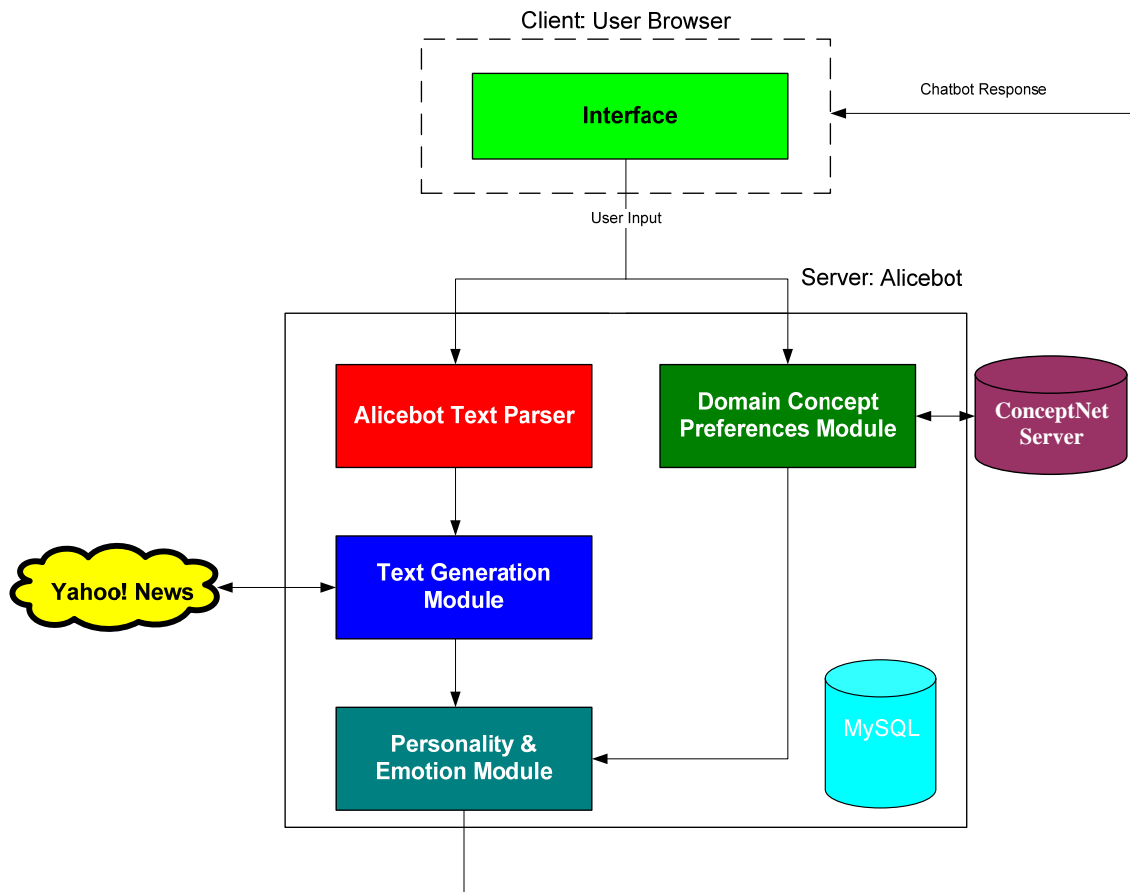


Figure 12: Implementation Overview

5.1 Personality & Emotion

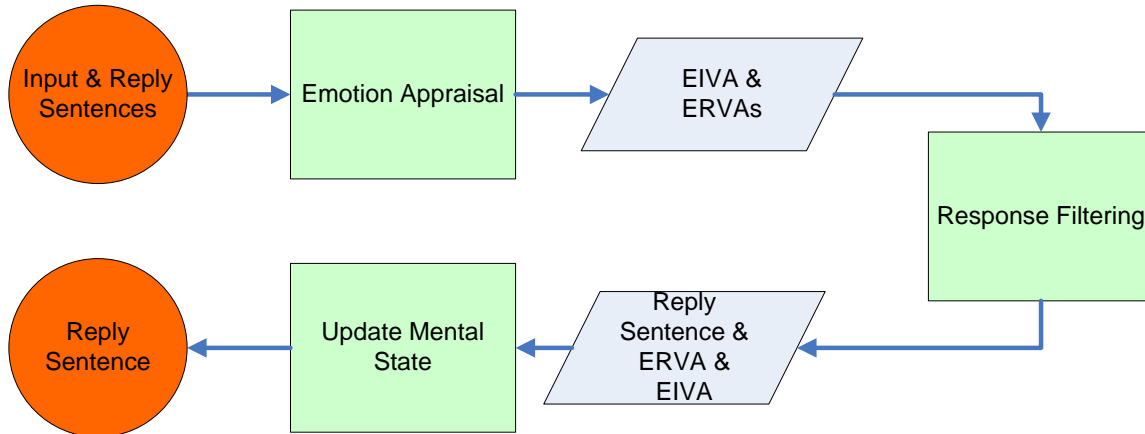


Figure 13: Personality & Emotion Architecture

An overview of the module architecture is shown in Figure 13. To create the personality for the chatbot, the chatbot creator, also known as the botmaster, must first answer a series of online personality questions from a third party website to create a fictional personality or to mimic a real person's personality. The Factor Five personality test results will then be entered into the chatbot setup survey and stored into the table named FACTOR_FIVE_TABLE within the database. There is also a thought process setting that can be set based upon the personality test, or set manually. The chatbot can be set up with two types of thought process patterns. The first type is an extravert thinker, where the emotional intent of the sentence is considered first, then the current emotional state, and finally the personality, to find a proper response. The second type is a neurotic thinker, which is the opposite of the extravert thinker, so the personality is used, then current emotional state, and finally the emotional intent of the sentence. The CES can also be set to a default startup value.

Next is the first set of supplemental questions relating how to deal with emotional intensities, see Figure 14. To do this, emotional keywords need to be populated into the

database beforehand. This is done by using the OCC emotional categories. Each emotional category keyword is looked up in WordNet and all the synonyms, adjectives, nouns, and verbs are added to the table called SIMILAR_WORDS_TABLE. Each similar word is linked to an OCC category, and subsequently, each OCC category is linked to an Ekman emotion which is linked to a Factor Five personality factor. In the supplemental survey, the botmaster will be presented with all the OCC categories related to each Ekman emotion. The botmaster will need to fill in a numeric value from 0.0 to 1.0 indicating on average how intensely that emotion is felt; 0.0 being no emotion is felt and 1.0 being very strong. This may be a difficult value to determine since different situational context can change emotional intensity, but an average value for all situations for each OCC category will be adequate. Figure 14 shows a sample of the OCC supplemental intensity survey.

Ekman Emotion: angry	
OCC Category: anger	<input type="text"/>
OCC Category: hate	<input type="text"/>
OCC Category: reproach	<input type="text"/>
Ekman Emotion: disgust	
OCC Category: disgust	<input type="text"/>
OCC Category: dislike	<input type="text"/>
Ekman Emotion: fear	
OCC Category: fear	<input type="text"/>
Ekman Emotion: happy	
OCC Category: admiration	<input type="text"/>
OCC Category: gloating	<input type="text"/>
OCC Category: gratification	<input type="text"/>
OCC Category: gratitude	<input type="text"/>
OCC Category: happy	<input type="text"/>
OCC Category: hope	<input type="text"/>
OCC Category: joy	<input type="text"/>
OCC Category: liking	<input type="text"/>
OCC Category: love	<input type="text"/>
OCC Category: pride	<input type="text"/>
OCC Category: relief	<input type="text"/>
OCC Category: satisfaction	<input type="text"/>

Figure 14: OCC intensity survey

Several other tables are created to keep track of the chatbot's and user's emotional states and other miscellaneous information. The CES_TABLE will keep track of the chatbot's current emotional state for each sentence throughout all the conversations it has. USER_CES_TABLE will hold the user's projected current emotional state. Each input sentence from the user will also be logged along with its projected Emotional Intent Values Array (EIVA) in the EIVA_TABLE. Respectively, each response the chatbot chooses will also be logged along with its Emotional Response Values Array (ERVA) in the ERVA_TABLE.

After the survey has been filled out, then the chatbot's mental state is established. The chatbot's mental state consists of its personality values and its current emotional state values. The personality values are taken from the survey and will not change. Although human personalities do change over time, this is not a part of the project scope. The reasoning behind this decision is that most personalities do not really change that much over a short period of time. However, the chatbot's emotional state changes with each sentence that is inputted since each sentence can have an emotional affect on a person. The current emotional state along with its personality is used to decide which response is the most appropriate given the emotional intent of the provided user sentence. The user's mental state is also tracked by the chatbot but only through the emotional state values. This is done so that the chatbot can also decide upon a sentence based on what emotions the user is experiencing.

In the beginning when the user starts the chat, a parsing of the input sentence is used to find a matching pattern. After the pattern is found, instead of deciding upon a response immediately, all the possible responses are returned. Now the personality and

emotion filter begins to determine the appropriate response. Each response is passed to the *calculate_emotion()* function where the text is parsed into words and each word is looked up in the SIMILAR_WORDS_TABLE joined with the OCC_TABLE and EKMAN_TABLE to find all the emotional keywords. After all the words have been discovered, the duplicates, words with more than one emotional meaning, are removed. For instance, the word "abhorrence" in Figure 15 can be interpreted to mean specifically disgust, dislike, or hate; each interpretation having its own intensity value.

WORD	CATEGORY	EMOTION	INTENSITY
abhorrence	disgust	disgust	0.99
abhorrence	dislike	disgust	0.6
abhorrence	hate	angry	0.9

Figure 15: Ambiguous word

To choose among the possible answers, the chatbot's dominant personality factor or dominant current emotional state is used. First, a list of unique emotions is created from what was found; using the previous example, disgust and angry would be used. Then an emotion is chosen based on the chatbot's last dominant emotion by calling *get_last_emotion()*. If the emotion is not present, then an emotion will be chosen based on the chatbot's dominant personality by calling *dominant_personality()*. If the dominant personality factor is extraversion, then the happy emotion will be used; otherwise, if the dominant personality factor is neurotic, then fear, anger, or sadness will be chosen randomly. After an emotion is decided upon, an OCC category will be chosen randomly. With an Ekman emotion and OCC category determined, a specific interpretation and intensity value can then be chosen. After this lookup is done for each emotion keyword, all the intensity values are aggregated and averaged, and then placed into their respective Ekman emotion groups creating an ERVA for each response sentence. Figure 16 shows

an example of a sentence after `calculate_emotion()` has been called. ERVA's stem from Liu, Lieberman, and Selker [17] but instead of having an emotion array for each word, it is done for the entire sentence. Performing this automatic textual emotion appraisal frees the botmaster from having to annotate each AIML pattern manually, unlike Kshirsagar's and Magnenat-Thalmann's [15] implementation.

```
I like cheese, but I also detest it.
```

```
[erv] => Array
(
  [happy] => 0.8
  [sad] => 0
  [angry] => 0
  [fear] => 0
  [surprise] => 0
  [disgust] => 0.6
)
```

Figure 16: ERVA for a response

After each response has an ERVA, they are all passed to function `filter_responses()` which will run three filters: personality filter, current emotional state filter, and input filter to narrow down the possible responses to send back. Depending on the thought process setting that was setup during the survey, the order of the filters will vary. For a Neurotic Thinker, the order is personality filter, CES filter, and input filter. For an Extravert thinker, it is input filter, CES filter, and personality filter. The personality filter, `personality_filter()`, works by finding the dominant personality factor. If the dominant personality is Extraversion, then only the responses that have a happy value greater than zero are returned. For a dominant neurotic personality factor, all of the responses that have a sadness, anger, or fear value greater than zero are returned. This search is done through calling `emotion_filter()`. If nothing is found, then all of the

responses are returned. The CES filter, *ces_filter()*, uses the last dominant emotion from *get_last_emotion()* to retrieve only the responses that have a value greater than zero matching the last emotion. Finally, the input filter, *input_filter()*, works by creating an Emotional Intent Value Array (EIVA), same as the ERVA, from the user input string and calling *update_user_ces()*. Then the dominant current emotion, *dominant_emotion()*, in the user's new CES is used to find all the matching responses with an emotional value greater than zero. Hopefully, after running all the filters, a single response is found that meets all the criteria. However, if there is still more than one response left, then a single response will be randomly chosen to be outputted to the user because the remaining possibilities have all passed the other filters and so any should be adequate. Through this filtering process, responses are chosen that are emotionally in tune with what has been said and what is being felt, meaning that if the chatbot is happy and the user is angry, then a response will be chosen so that it relays anger and happiness rather than just being randomly chosen. Figure 17 illustrates the filtering process and Figure 18 displays the source code for *filter_responses()*.

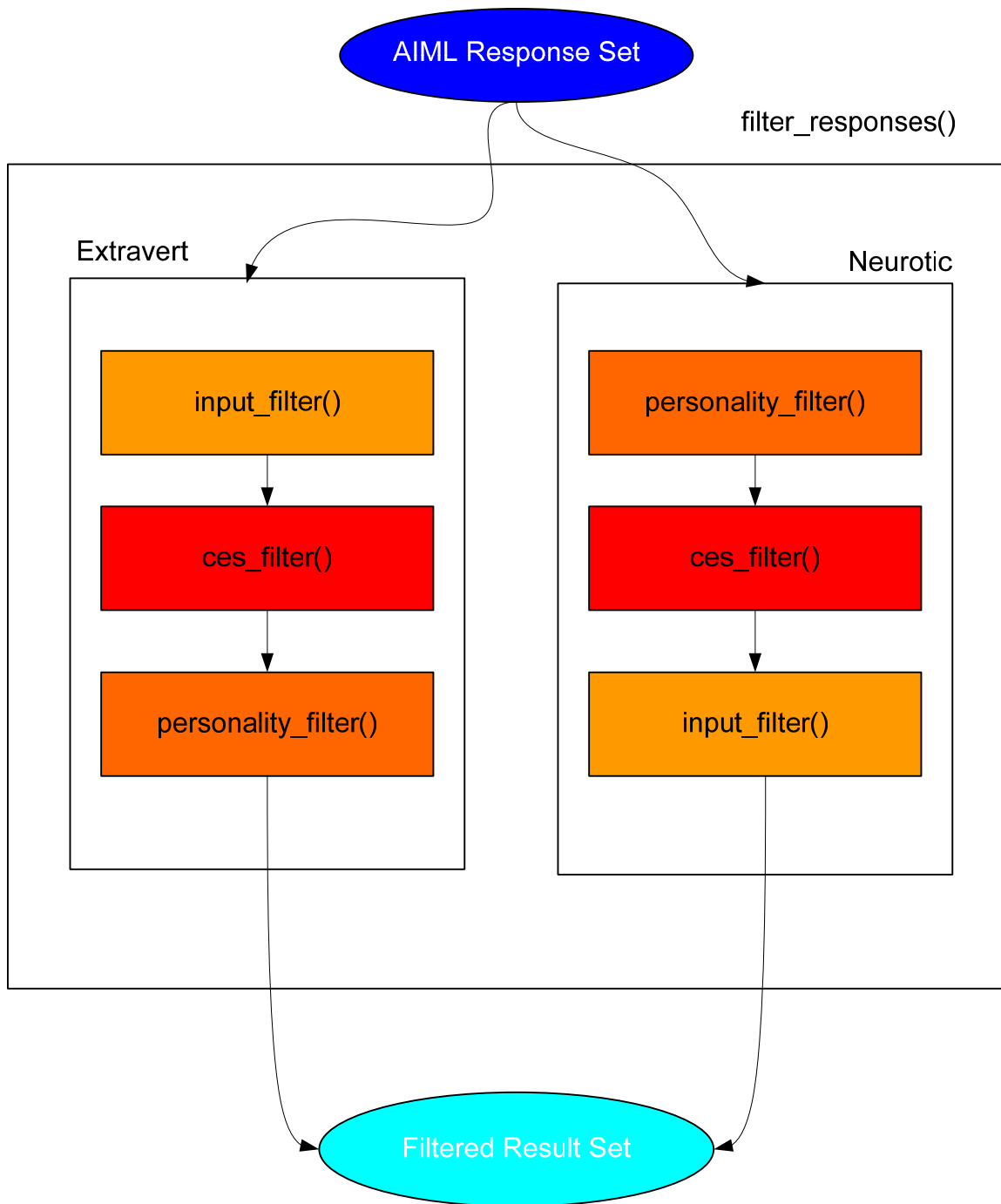


Figure 17: Filtering Overview

```

/*
  @description: takes a set of possible replies and filters them down based on
  emotions until finally
  randomly choosing a single reply
  @param: $finalanswer_array, the replies
  @param: $allinputs, input sentence from the user
  @return: string, the chosen reply
*/
function filter_responses($finalanswer_array, $allinputs)
{
  $query = "select thought_process from " .BOT_DATA_TABLE;
  $row = mysql_fetch_array(mysql_query($query));
  $reply = "";
  $filter1_result = "";
  $filter2_result = "";
  $filter3_reuslt = "";

  if ($row["thought_process"] == EXTRAVERT_THINKER) //extraversion
  {
    $filter1_result = input_filter($finalanswer_array, $allinputs);
    $filter2_result = ces_filter($filter1_result);
    $filter3_result = personality_filter($filter2_result);

    if (sizeof($filter3_result) == 1)
      $reply = $filter3_result[0];
  }
  else if ($row["thought_process"] == NEUROTIC_THINKER) //neurotic
  {
    $filter1_result = personality_filter($finalanswer_array);
    $filter2_result = ces_filter($filter1_result);
    $filter3_result = input_filter($filter2_result, $allinputs);

    if (sizeof($filter3_result) == 1)
      $reply = $filter3_result[0];
  }

  if (empty($reply))
  {
    $random_num = random_num(sizeof($filter3_result));
    update_bot_ces($filter3_result[$random_num]);
    $reply = $filter3_result[$random_num];
  }
  else
    update_bot_ces($reply);

  $reply_sentence = $reply['sentence'];

  return $reply_sentence;
}

```

Figure 18: filter_responses()

Besides just filtering responses, the filter process also updates the CES of both the chatbot and the user. During the *input_filter()* call, *update_user_ces()* is called to compute the EIVA for the user input sentence through *calculate_emotion()*, then *dominant_emotion()* is called to find the dominant emotion in the EIVA. But the computed dominant emotion may not be correct since the user has the ability to express his emotion through emoticons. User provided emoticons have precedence over the computed dominant emotion; only when no emoticon is chosen is the result from *dominant_emotion()* used. Then *calculate_ces()* is called to calculate the new CES value for the user using the EIVA of the input sentence. The chatbot CES is updated by calling *update_bot_ces()* which takes the chosen response and uses *calculate_ces()* to compute the new CES by taking the ERVA of the response. *calculate_ces()* combines either the current CES of the chatbot with the ERVA or the current CES of the user and the EIVA to create a new CES. The new CES is not just created through aggregation of all the emotional states; emotional decay is also added so that over time, per sentence, each emotion drops in value. The new emotion value is calculated by multiplying the old CES emotion with e^{-1} and adding it to the new incoming ERVA or EIVA emotion. *calculate_ces()* is shown in Figure 19.

```

/* @description: takes EIVA or ERVA and returns the new CES
   @param: $new_values, EIVA or ERVA
   @param: $mode, indicator to indicate if it is INPUT mode
   @return: array, CES array
*/
function calculate_ces($new_values, $mode)
{
    $ces_array = get_ces($mode);
    $ces_array['happy'] = exp(-1) * $ces_array['happy'] + $new_values['happy'];
    $ces_array['sad'] = exp(-1) * $ces_array['sad'] + $new_values['sad'];
    $ces_array['angry'] = exp(-1) * $ces_array['angry'] + $new_values['angry'];
    $ces_array['fear'] = exp(-1) * $ces_array['fear'] + $new_values['fear'];
    $ces_array['surprise'] = exp(-1) * $ces_array['surprise'] + $new_values['surprise'];
    $ces_array['disgust'] = exp(-1) * $ces_array['disgust'] + $new_values['disgust'];

    return $ces_array;
}

```

Figure 19: calculate_ces()

5.2 Domain Concept Preferences

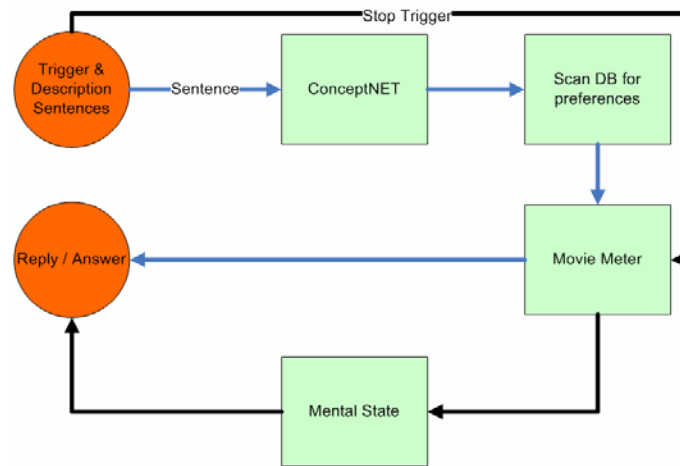


Figure 20: Domain Concept Preferences Architecture

The module overview is shown in Figure 20. Domain concept preferencing is implemented specifically for horror movies, but it can be expanded to include other domains. To implement domain preferences, first the preference information needs to be collected and stored. During the survey the botmaster is presented with another set of supplemental questions pertaining to horror movies. Since the domain preferencing is used to help the chatbot make decisions about horror movies, the feature is enabled by having the user issue a particular statement: “Want to watch a movie?” Once triggered, the chatbot will automatically respond with “yes” and set the MOVIE_TALK variable to TRUE. The MOVIE_TALK variable is needed to maintain the distinction that all subsequent sentences will now be about the movie and will not be processed like a normal sentence. All of the following sentences coming from the user should now describe the horror movie in detail. As each sentence is passed in, it is sent to *movie_concept()* where the sentence is passed to the ConceptNet server through a XMLRPC call. The XMLRPC call will pass along the sentence as well as the instruction to invoke *guess_topic()*, the ConceptNet function that will try to guess the topics within

the sentence. A list of possible topics and each topic's likelihood in percentage form are returned. The topics are checked against all the stored concepts to find if any preferences have been matched. If a match is discovered, then the associated intensity value is aggregated into the movie meter, which is also stored in the MOVIE_DATA_TABLE. The movie meter is a rating for the movie, and each concept preference value is aggregated into the meter and averaged based on the number of concepts. If the movie meter value is greater than zero then the movie is accepted, if the value is less than zero then the movie is rejected, or if the value is zero then the movie is neither accepted nor rejected. Duplicate concepts are still counted. This decision is based on the reasoning that reoccurring events such as gory murders or the appearances of creatures count towards whether a movie is likable. For instance, a horror movie with only one gory scene is not the same as a movie with many gory scenes. The chatbot will also respond to each sentence indicating how it likes or dislikes it so far. Figure 21 shows the code for *movie_concepts()*.

```

/*
  @description: find movie concepts & control movie meter
  @param: $description, sentence from user
*/
function movie_concepts($description)
{
  $description = str_replace("'", "", strtolower($description));

  $query = "select meter, num_concepts from ". MOVIE_DATA_TABLE;
  $row = mysql_fetch_array(mysql_query($query));
  $movie_meter = $row['meter'];
  $num_concepts = $row['num_concepts'];

  $concept_array = Array();

  $topic_results = guess_topic($description);

  foreach ($topic_results as $topic)
  {
    $concept = mysql_escape_string($topic[0]);

    //use distinct to remove the problem of the same concept being seen twice
    $query = "select distinct(b.id), concept, score FROM ".CONCEPTS_TABLE. " a inner join "
      . PREFERENCES_TABLE. " b on a.pref_id = b.id where concept = '$concept'";

    $results = mysql_query($query);

    if (mysql_num_rows($results) > 0)
    {
      while ($row = mysql_fetch_array($results))
      {
        if (!concept_exists($row, $concept_array)) //does not exist or array empty
          array_push($concept_array, $row);
      }
    }

    foreach($concept_array as $concept)
      $movie_meter += $concept['score'];

    $query = "update " .MOVIE_DATA_TABLE. " set meter = " . $movie_meter;
    mysql_query($query);

    $query = "update " .MOVIE_DATA_TABLE. " set num_concepts = " . ($num_concepts +
      sizeof($concept_array));
    mysql_query($query);

    if ($num_concepts != 0)
      $movie_meter = $movie_meter / $num_concepts;

    $response = new Response();

    $decent_movie = Array('This movie sounds really interesting...', 'Not bad', 'What else', 'And..',
      'sounds pretty good', 'What else happens?', 'this movie is getting pretty good', '');
    $great_movie = Array('This movie is sounding pretty good, tell me more!', 'seems really exciting');
    $bad_movie = Array('This movie is going down hill...', 'doesn\'t sound that great', 'uh huh', 'is
      it going to get better?', 'this movie is getting worse', '');
    $horrible_movie = Array('This movie sounds horrible', 'ok, this is just bad');

    $join = array_merge($decent_movie, $great_movie);
    $join2 = array_merge($bad_movie, $horrible_movie);

    if ($movie_meter > 0.0 && $movie_meter <= 0.5)
      $response->response = $decent_movie[random_num(sizeof($decent_movie))]; //"This movie is
        sounding pretty good, tell me more!";
    else if ($movie_meter > 0.5 && $movie_meter <= 1.0)
      $response->response = $join[random_num(sizeof($join))]; //"This movie sounds really
        interesting...";
    else if ($movie_meter < 0.0 && $movie_meter >= -0.5)
      $response->response = $bad_movie[random_num(sizeof($bad_movie))]; //"This movie is going down
        hill...";
    else if ($movie_meter <= -0.5 && $movie_meter >= -1.0)
      $response->response = $join2[random_num(sizeof($join2))]; //"The movie sounds horrible!";
    else
      $response->response = "Go on...";

    return $response;
  }
}

```

Figure 21: movie_concepts()

After the user has finished describing the movie, another trigger sentence needs to be activated: “So, do you want to see the movie?” This trigger will invoke *movie_answer()*, shown in Figure 22, and it will check the value of the movie meter, and depending on the value, the chatbot will choose whether it liked or disliked the movie. The chatbot will also use its current emotion from before the movie was described to also help it decide if it would like to watch the movie. Neither the CES of the chatbot and the user is updated during the movie conversation since all sentences coming in will be specifically about the movie and there will not be any sentences that should affect the chatbot’s CES. Emotions are only used at the end to decide whether if the chatbot feels like watching the movie. For example, if the chatbot did not like the movie, but is happy, then it still might consider watching the movie. In contrast, even if the movie was good it might not want to go watch if it was angry.

```

/*
  @description: calculate answer
  @todo: better method than average?
  @todo: add logging of input & replies
*/
function movie_answer()
{
  $query = "select meter, num_concepts from ". MOVIE_DATA_TABLE;
  $row = mysql_fetch_array(mysql_query($query));
  $movie_meter = $row['meter'];
  $num_concepts = $row['num_concepts'];

  $query = "update " .MOVIE_DATA_TABLE. " set meter = 0.0";
  mysql_query($query);

  $query = "update " .MOVIE_DATA_TABLE. " set num_concepts = 0";
  mysql_query($query);

  $response = new Response();
  $response_sentence = "";
  $response->timer = 0.0;

  if ($num_concepts != 0)
    $movie_meter = $movie_meter / $num_concepts;

  if ($movie_meter == 0.0)
    $response_sentence = "From what I've heard, I don't really care either way";
  else if ($movie_meter > 0.0 && $movie_meter <= 0.5)
    $response_sentence = "This movie sounds pretty good.";
  else if ($movie_meter > 0.5 && $movie_meter <= 1.0)
    $response_sentence = "This movie is a must watch. ";
  else if ($movie_meter < 0.0 && $movie_meter >= -0.5)
    $response_sentence = "The movie doesn't sound too good.";
  else if ($movie_meter <= -0.5 && $movie_meter >= -1.0)
    $response_sentence = "The movie is horrible.";
  else
    $response_sentence = "I've no answer";

  $response->emoticon = get_review($movie_meter);
  $last_emotion = get_last_emotion();

  if ($last_emotion == "happy" && $movie_meter > 0.0)
    $response_sentence .= " I definitely feeling like watching this.";
  else if ($last_emotion == "happy" && $movie_meter <= 0.0)
    $response_sentence .= " But if you want we can still check it out.";
  else if ($last_emotion != "surprise" && $movie_meter > 0.0) //sad, angry, fear,
    disgust
    $response_sentence .= " But I'm not in the mood to go watch a movie.";
  else if ($last_emotion != "surprise" && $movie_meter <= 0.0)
    $response_sentence .= " I'm in no mood to watch something like that.";

  $response->response = $response_sentence;
  return $response;
}

```

Figure 22: movie_answer()

5.3 Text Generation

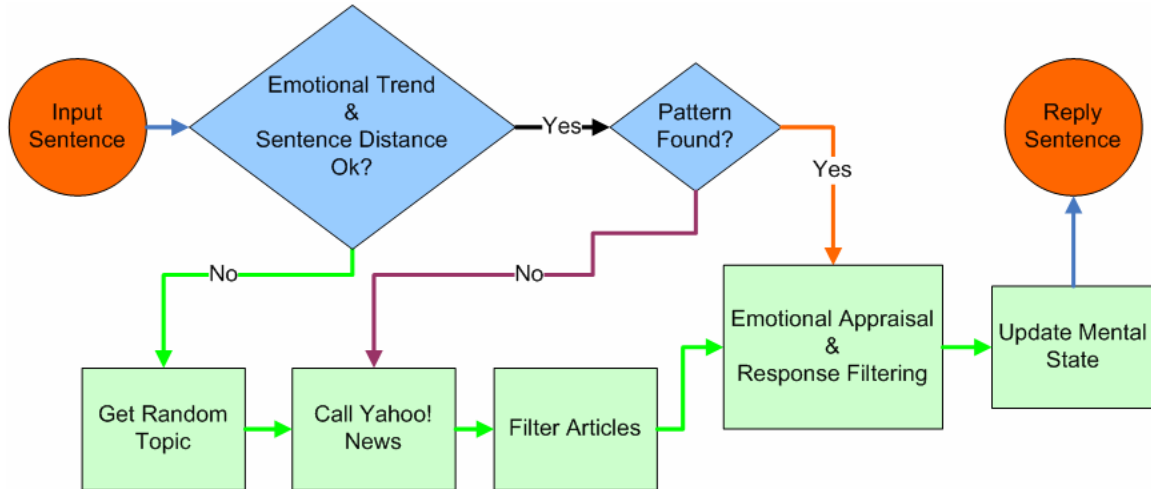


Figure 23: Text Generation Architecture

The text generation architecture is shown in Figure 23. The first type of text generation for the chatbot, Related Topic Text Generation, uses Yahoo!® News search when there are no matching patterns; it is implemented by first checking if there was no matching pattern found during the input pattern search. Once there is no match, *find_topic()* is called. *find_topic()* will invoke *yahoo_request()* which will submit a URL with the user input sentence appended to it to be used as the search string to the Yahoo!® News webservice. A collection of relevant news articles are returned and the top article summary is used as output. Both the chatbot's CES and the user's CES are updated as well. Figure 24 displays the code for *find_topic()*.

```

/*
  @description: find related yahoo articles to user input
  @param: $input, user input string
  @return: string, response string
*/
function find_topic($input)
{
  if ($input == 'INACTIVITY')
  {
    $r = new Response();
    $r = new_topic($input);
    return $r->response;
  }

  $yahoo_articles = yahoo_request($input);
  $summary = '';

  if (sizeof($yahoo_articles) == 0)
  {
    $r = new Response();
    $r = new_topic($input);
    return $r->response;
  }

  $summary = $yahoo_articles["Summary"];

  update_user_ces($input);
  $reply = Array('sentence' => $summary,
    'erv' => calculate_emotion(emotion_query($summary)));
  update_bot_ces($reply);

  return $summary;
}

```

Figure 24: find_topic()

The second type of text generation, Emotional Text Generation, where the chatbot chooses a new topic to discuss based on its emotions, is done by checking the conversational emotion trend from the chatbot's perspective. Each time a sentence is passed in, the *change_subject()*, shown in Figure 25, is called and the chatbot's CES is queried to return a set of dominant emotion values, which are aggregated and averaged. If the average happy value over the course of the conversation is less than HAPPY_MIN_TREND, the default of which is set to 0.50, then *change_subject()* will return TRUE. But to not have the chatbot change topics too often, a sentence distance is kept track of when the last topic change was made. The sentence distance is a random number from 10 (minimum) to 20 (maximum) for Extraverts or 30 (maximum) for non-Extraverts. A distinction is made for extraverts to have a smaller value because they are

more upbeat and outgoing and would tend to change the topic of the conversation more often if it was becoming unpleasant. Once the number of sentences that have gone by equal the sentence distance, then the emotional trend is re-evaluated to see if another topic change is needed.

```

/*
  @description: check if conversational trend seems to be going down
  @return: boolean, true to change subject
*/
function change_subject()
{
  $query = "select topic_distance, topic_current_idx from " .BOT_DATA_TABLE;
  $row = mysql_fetch_array(mysql_query($query));
  $topic_distance = $row["topic_distance"];
  $topic_current_idx = $row["topic_current_idx"];

  $query = "select last_emotion from " .CES_TABLE. " where id >= "
    . $topic_current_idx. " order by id desc";
  $result = mysql_query($query);
  $num_rows = mysql_num_rows($result);
  $gap = $num_rows - $topic_current_idx;

  if ($gap >= $topic_distance)
  {
    $count_array = Array("happy" => 0, "sad" => 0, "angry" => 0, "fear" => 0,
      "surprise" => 0, "disgust" => 0);
    $personality = dominant_personality();

    while ($row = mysql_fetch_array($result))
      if ($row["last_emotion"] != "")
        $count_array[$row["last_emotion"]];

    mt_srand ((float) microtime() * 1000000);
    $rand = 0;

    if ($personality == "extraversion")
      $rand = mt_rand(MIN_TOPIC_DISTANCE, MAX_TOPIC_DISTANCE_EXTRAVERT);
    else
      $rand = mt_rand(MIN_TOPIC_DISTANCE, MAX_TOPIC_DISTANCE);

    if (($count_array["happy"] / $topic_distance) < HAPPY_TREND_MIN)
    {
      //change distance
      $query = "update " .BOT_DATA_TABLE. " set topic_distance = ".$rand;
      mysql_query($query);
      //update idx
      $query = "update " .BOT_DATA_TABLE. " set topic_current_idx = ".$num_rows;
      mysql_query($query);
      return true;
    }
  }
  return false;
}

```

Figure 25: change_subject()

Once *change_subject()* returns TRUE then *new_topic()*, Figure 26, is called to randomly pick a topic from a list of topics provided by the botmaster during the survey. After a topic is chosen then it is fed into *yahoo_request()* to find the most relevant article summary which are then sent back as a response to the user to be the new conversation topic. The CES of both the chatbot and user are also updated.

```
/*
  @description: pick a new topic
  @param: $userinput, user input string
  @return: string, response string
*/
function new_topic($userinput)
{
  $query = "select topic from " .TOPICS_TABLE;
  $result = mysql_query($query);

  $response = new Response();
  $response->emoticon = "";
  $response->timer = 0.0;

  $str = "Let's talk about something else. ";

  $rand_num = random_num(mysql_num_rows($result));
  mysql_data_seek($result, $rand_num);
  $row = mysql_fetch_array($result);

  $yahoo_articles = yahoo_request($row["topic"]);
  $chosen_article = Array();

  if (sizeof($yahoo_articles) == 0)
    new_topic($userinput);

  $str .= $yahoo_articles["Summary"];

  $response->response = $str;

  update_user_ces($userinput);

  $reply = Array('sentence' => $str, 'erv' => calculate_emotion(emotion_query($str)));
  update_bot_ces($reply);

  return $response;
}
```

Figure 26: new_topic()

A result filter is also used during both types of text generation. Within *yahoo_request()*, *yahoo_result_filter()* in Figure 27 is called. Since the returned results do not contain the entire article but just a summary of it which is usually just the first sentence of the article and the article date or author, it is problematic to simply use the summary sentence. Another problem is that an input string might find and use the same article which subsequently makes the chatbot repeat itself. To resolve these issues, a filter scans through all the returned articles and rejects articles that have any sort of timestamp, mention of an author, or has already been used.

```

/*
  @description: filter articles from yahoo
  @param: $base_articles, all articles from yahoo
  @return: array, should be the filtered articles
*/
function yahoo_result_filter($base_articles)
{
  $filter_words = Array("writes", "by", "said", "says", "published");
  $result = Array();

  foreach($base_articles as $art) {
    if (is_array($art)) {
      if (substr_count($art["Summary"], ":") == 0 &&
          substr_count($art["Summary"], '/') == 0) {
        console_output("yahoo_result_filter > passed 1st test");
        $pass = false;
        foreach($filter_words as $w) {
          if (substr_count(strtolower($art["Summary"]), $w) == 0)
            $pass = true;
          else
            $pass = false;
        }

        if ($pass) {
          console_output("yahoo_result_filter > passed 2nd test");
          $pass = true;
          //look at the last 10 replies to make sure there is no duplicate
          $query = "select sentence from " . ERVA_TABLE. " order by id desc limit "
            . MIN_TOPIC_DISTANCE;
          console_output("yahoo_result_filter > query", $query);
          $result = mysql_query($query);

          if (mysql_num_rows($result) <= 0)
            return $art;

          while ($row = mysql_fetch_array($result)) {
            if (strcasecmp($art["Summary"], $row["sentence"]) == 0) {
              console_output("yahoo_result_filter > passed 3rd test");
              $pass = false;
              break;
            }
          }

          if ($pass)
            return $art;
        }
      }
    }
  }

  return Array(); //nothing in results, so return nothing
}

```

Figure 27: yahoo_result_filter()

5.4 Interface

The interface is implemented using AJAX to resemble the look and feel of an instant messenger chat interface. Each frame updates asynchronously with new information after each sentence is submitted. The user starts at Frame 2, as shown in Figure 11, and here the user enters a single sentence and can select an emoticon to further convey a specific emotion. Clicking on the send button will fire off an AJAX request and submit the sentence along with the emoticon to `reply.php`, the server-side PHP script which will do the processing described in previous sections: emotional response filtering or domain concept preferencing. A concatenated string containing the response, chatbot emoticon, and console output is sent back to the client browser from the PHP script. The string is parsed to separate the three pieces and each is used to update their respective frames. The dialogue frame, Frame 1, will display the chatbot's response along with the emoticon and the user's sentence and emoticon. Next the console frame, Frame 4, will update with the latest debugging information, such as input and output parameters and trace messages. Finally, the status frame, Frame 3, is refreshed and the chatbot CES values are updated.

6.0 Results

Testing whether one chatbot sounds more human than the other is difficult and hard to measure quantitatively, so these tests will try to infer as much as possible from a qualitative perspective. To test the improvements in Alicebot, two chatbots were created, BotA and BotC. Each chatbot is configured differently; Table 3 shows the configuration grid. Three separate tests were run to test each feature. The full test transcripts are in Appendix B.









BotA	BotC	Alicebot
Extravert	Neurotic	N/A
Likes vampires	Dislikes vampires	N/A
Dislikes demons	Likes demons	N/A

Table 3: Chatbot Configuration Grid

6.1 Personality & Emotion Testing

The first test contrasts BotA and BotC versus the original Alicebot. Each chatbot will be loaded with the same AIML conversation. The conversation will be long and span different topics. It is suspected that if the conversation is too short then the outputs from all of the chatbots will not show any signs of improvement over Alicebot. The expected output for BotA and BotC responses will be emotionally stable, meaning that since BotA is an extravert, most of its responses should be happy and since BotC is neurotic, most of its responses should either be sad, angry, or afraid. Alicebot's output should show contradictory emotions or very quick turn over in emotions because Alicebot randomly chooses responses. However, since Alicebot is randomly choosing responses it is possible that it can coincidentally choose a coherent response.

The actual results were as expected. BotA and BotC responded accordingly to their emotions and personality quite consistently as expected. Alicebot responded in an emotionally inconsistent manner. Table 4 shows a comparison where BotC is consistently angry and does not like cheese while Alicebot loves and then hates cheese.

[18:28:29] User: let us talk about something else	
[18:28:29] Bot: ANGRY what about?	
[18:30:25] User: how about cheese	
[18:30:25] Bot: ANGRY i like cheese too, but i also abhor it.	
[18:30:56] User: i think the aroma it has is lovely	
[18:30:56] Bot: ANGRY no, i hate cheeses	
[18:31:45] User: what about the texture and taste that is great too	
[18:31:46] Bot: ANGRY no, i hate cheese	

input	response
love eating candy	so do i, that's why i hate seeing the dentist.
let us talk about something else	what about?
how about cheese	i love cheese.
i think the aroma it has is lovely	no, i hate cheese
what about the texture and taste that is great too	oh yes, it is intoxicating
well anyway i am hungry want to go get some sushi	no, i'm too annoyed to eat

Table 4: BotC (top) vs. Alicebot (bottom)

6.2 Domain Concept Preferences Testing

The domain concept preference test will use Alicebot, BotA, and BotC. The chatbots are set up to have the same preferences except BotA is set up to enjoy vampire movies and dislike demon movies while BotC is the opposite, and Alicebot has no preferences to set. Each chatbot will be fed two different movie descriptions that were arbitrarily made: *Mr. Vampire* is about vampires, and *Hellraiser* is about demons from Hell. The expected output is that each will find the same concepts but rate them differently. BotA should like *Mr. Vampire* and want to see it but should reject *Hellraiser*. BotC should like *Hellraiser* and want to see it but reject *Mr. Vampire*. Alicebot will be

fed *Mr. Vampire*. However, since Alicebot is random the outcome is unknown and it may not understand the conversation and try to change topics or cause confusion. Results are shown in Table 5 and Figure 28.

BotA: <i>Mr. Vampire</i>	[19:14:23:] Bot: 👍👍 This movie is a must watch. I definitely feel like watching this.
BotA: <i>Hellraiser</i>	[19:39:56:] Bot: 🗑️ From what I've heard, I don't really care either way But if you want we can still check it out.
BotC: <i>Mr. Vampire</i>	[20:18:04:] Bot: 👍👍 The movie is horrible. I'm in no mood to watch something like that.
BotC: <i>Hellraiser</i>	[19:56:40:] Bot: 👍👍 This movie is a must watch. But I'm not in the mood to go watch a movie.

Table 5: BotA & BotC movie results

input	response
so, do you want to watch?	Ah. OK, I will watch.

Figure 28: Alicebot movie results

The outcome was as expected but with noticeable differences. Although BotA favored *Mr. Vampire* it found a low number of concepts; only eight were found, even though the text contained many concepts. The reason for this is because ConceptNet sometimes has topics created from more than one word, a composite topic, such as “female vampire” rather than “vampire.” For *Hellraiser*, BotA’s movie meter fluctuated and then leveled out. This is due to there being other concepts within the movie description that appealed to BotA as well; concepts such as the movie being very gory and supernatural. BotC did the same as BotA but just rated the concepts oppositely. Although the descriptions were somewhat short it was still able to rate the movie. A more accurate rating could have been computed if longer, more descriptive summaries of the movies were given.

On the other hand, although Alicebot responded positively to the movie, this was only due to the hard-coded response. Looking at the conversation history, it is evident that Alicebot was not fully aware of the topic and had tried to change the subject. By

comparing the results from Table 6, it is evident that BotC is holding a consistent conversation about the horror movie, while Alicebot ignores what has been said and changes topics.

<p>[15:26:34] User: want to watch a movie? [15:26:34] Bot: What is the movie about? [15:27:31] User: In rose red, the house had a mind of its own [15:27:44] Bot: Go on... [15:28:04] User: The house had evil intentions [15:28:17] Bot: Go on... [15:28:36] User: The house killed numerous people [15:28:49] Bot: Go on...</p>	
input	response
want to watch a movie?	OK, I will watch a movie.
In rose red, the house had a mind of its own	Where is that?
in california	I am in California too!
The house had evil intentions	What is your real name?
jasprov	Hi there jasprov.
The house killed numerous people	What are your goals in life?
to eat, drink, and sleep	Interesting. I want to find out more about you.

Table 6: BotC (top) vs. Alicebot (bottom) results

6.3 Text Generation Testing

The text generation test is done with BotA versus Alicebot only since text generation is independent of how the chatbot is configured, so running the test for BotC will yield the same result. Several random topics will be brought up to be discussed so that BotA is forced to retrieve information from Yahoo!® News. Alicebot will also hold a conversation in the same manner. The expected outcome for BotA is that the information retrieved should help supplement the conversation. However, since any results can be returned from the search, it could also result in abrupt topic changes as well. Alicebot will try to compensate by using its patterns and try to change conversation topics or create confusion.

BotA presented new topics most of the time instead of continuing the current one. One reason was that no results were returned from Yahoo!® News that matched the search string, so the chatbot automatically picked another topic to discuss. The other subtle reason is that although the topic seems to be unrelated, it in fact does fulfill the requirements of the search string. For example, although cheese was supposed to be the topic from the user sentence: “let's get to talking about different cheeses,” the returned sentence was about the BBC European editor Mark Mardell’s report about the presidential campaign in France. Upon closer inspection of the retrieved article, the word “cheeses” is actually mentioned in this quote:

*“One must recognise the plethora of choices available to French voters. Charles de Gaulle said there are as many ideas in France as there are **cheeses**. -- Cormac Mac Gabhann, Dublin, Ireland” [32].*

It is still possible to get a coherent conversation such as this example in Table 7. The results for Super Paper Mario fit well because of the unique context created by the keywords “Super Paper Mario” in the second user sentence since it can only relate to games. So, even if the chatbot did not answer the question of the game being fun, it would still have found articles directly related to Super Paper Mario. The conversation can also sound slightly awkward due to different articles being used. For example, one sentence would be in third person and the next in first person, but the topic is still being discussed.



[13:30:51:] User: so what do you think of the nintendo Wii?
[13:30:56:] Bot:  If you take games a little too seriously, then the newest Nintendo-made Wii game, "Super Paper Mario," is making fun of you.
[13:31:42:] User: Is super paper mario fun?
[13:31:43:] Bot:  It's really hard to classify “Super Paper Mario” for the Nintendo Wii.

Table 7: Text generation results

The conversation with Alicebot was confusing and off-topic. When asked specifically about the Nintendo Wii, Alicebot never answered the question; it tried to change subjects instead.

input	response
so what do you think of the nintendo wii?	Oops. I wasn't paying attention. Tell me again what is going on.
i was talking about the nintendo wii	Sure, the nintendo wii sounds like a good topic.
so then what do you think about the nintendo wii	Oops. I wasn't paying attention. Tell me again what is going on.
i said we are talking about the nintendo wii	What did you mean by we are talking about the nintendo wii? Sure, the nintendo wii sounds like a good topic.
guess you havent heard of it then	I can't guess. Tell me.
the nintendo wii is a new video game system	What are your goals in life?
ok... do finish my thesis	Groovy. I do nothing but chat all day. need time to formulate the reply.
you chat all day	Thanks for telling me: undefined said I chat all day.
no i didn't you did	Are you serious? Why didn't you do you did?
nevermind	OK Thanks for trying to explain that to me.

Figure 29: Text generation result for Alicebot

6.4 Survey

Overall the results from the testing were more than satisfactory. This shows great promise that the added features definitely improve Alicebot's believability. A small survey was conducted to gauge the level of believability. Five people were chosen to chat with both Alicebot and the new Alicebot called AlicePlus. Each feature was tested during the conversation with both bots and the participant had to decide on AliceBot, AlicePlus, or undecided. A tally is shown in Table 8.

	<i>AlicePlus</i>	<i>AliceBot</i>	<i>Undecided</i>
<i>Personality & Emotions</i>	3	1	1
<i>Domain Preferences</i>	5	0	0
<i>Text Generation</i>	3	2	0
<i>Overall</i>	5	0	0
Total	16	3	1

Table 8: Survey results

AlicePlus did best in the Personality & Emotions test. Overall, the comments given were all saying that AlicePlus was more emotionally consistent and gave better emotional feedback while AliceBot was inconsistent. One participant did prefer AliceBot over AlicePlus because AlicePlus had seemed less emotionally consistent than AliceBot.

Upon viewing the conversation that was held, the participant selected various emotions for each input. Since this AlicePlus was an extravert, and more empathic to the user's emotions, it chose replies by first basing them off the inputted user emotion from the emoticons. Therefore, since the participant varied among emotions for each input, so did AlicePlus. Another participant could not decide between AliceBot and AlicePlus saying that although AlicePlus was more consistent it did not seem entirely believable that it could be so angry for most of the time, while although Alicebot was inconsistent, it seemed more believable in its emotions.

In the Domain Preference test AlicePlus did the best. The participants commented that AlicePlus was best because it, as some participants commented, "paid attention" and "cared" about the movie conversation while AliceBot ignored the topic presented by the user. Another commented that although AliceBot agreed to watch the movie, the answer seemed "superficial."

The Text Generation test was a close match. Some thought that AliceBot held a better conversation overall, even though it knew nothing about the topic. Others thought AlicePlus was better because it was able to provide actual information about the discussed topics, even if the conversation sounded strange or topics were changed often.

Overall AlicePlus was chosen as the best because it remained the most consistent and abreast of new information about the discussed topics. AliceBot, while being more conversational, was not consistent and seemed to merely repeat or try to force the participant to say something else by changing the subject.

7.0 Future Work

In future work, emotional text appraisal can be expanded to include phrases besides just keywords which would expand the text covered. There can be better integration with words and phrases from WordNet to categorize and place as many words and phrases into the database. Handling sentence negations would also improve the text appraisal process. During the sentence filtering process, emotional intensity along with using more than just the dominant emotion could also be used as a criterion when looking for the best response so that the emotional intensity of the sentence complements what has been said and the CES of the chatbot. Also, the filtering process could be improved by handling emotionally ambiguous words, where a word means more than one emotion or the emotion is different depending on the context. Improving the way the CES, ERVA, and EIVA values are calculated could be accomplished by following the fact that emotions affect other emotions, meaning when a person becomes happy, their sadness, anger, and fear values drop. The decay factor could also be adjustable depending on the personality; extraverts would have their emotions decay faster while neurotics would decay slower.

Domain preference enhancements can also be improved by matching phrases instead of just keywords when searching for concepts. Currently, duplicate occurrences of concepts within the same sentence or separate sentences can cause the movie meter to be skewed. Although it is acceptable in some circumstances, it is not in all. So a new method should be introduced instead. There should be more accurate and in-depth movie questions and concept associations to expand the horror movie coverage. Alicebot should either cite what has been said by the user, using past information on other similar

movies, or use its own preferences to justify why it liked or disliked the movie. A better method could be used to calculate the movie meter, such as using a movie rating criteria is yet another possible enhancement. Intermediate responses given by Alicebot can also be improved to include questions or references. Currently, the domain concept preferencing is also too restrictive since specific information is needed to be able to build a preference answer. Perhaps generic preferences can be added to Alicebot allowing it to make decisions using common preferences for all domains.

Text generation can be enhanced by passing more than just the input string directly from the user to Yahoo!® News, and contextual topic information acquired from the conversation history can also be passed along to find better articles. A memory and context module would vastly improve Alicebot's ability to remember and make use of past conversations for specific conversations where the conversation held is within the same context. During the survey, text generation was noted to have lacked the ability to have a good conversation; therefore, the parser can be enhanced so that all AIML patterns can be used in conjunction with text generation so that an informative and better conversation can be held.

A new feature of allowing Alicebot to make its own typos based on the statistics of how often typos are made and choosing whether to correct them or not can make Alicebot seem more human to the user.

8.0 Conclusion

Alicebot is a popular and successful chatbot but still lacks emotions and knowledge. The goal was to add new layers to make Alicebot more emotionally interactive, aware of its own preferences to make its own decisions, and allow it to acquire more knowledge. This was done by adding several modules: personality and emotion filters, domain concept preferencing knowledge, and text generation through fetching the latest news information.

The personality and emotions layer successfully made Alicebot more emotionally stable and interactive instead of just responding with random responses. The new, unique domain concept preferences feature gave Alicebot its own ability to make decisions about what it likes and dislikes to form its own “opinion.” It is also tied into Alicebot’s emotions which influences its final answer. This gives Alicebot another application: a reviewer of horror movies when loaded with a real person’s domain preferences so a person can form an opinion about a movie based on what the chatbot has decided without really knowing anything about the movie. Finally, the text generation allows Alicebot to discuss topics even though it does not have any preset patterns to respond with. Combining these concepts all together now allows Alicebot to better mimic a real person.

Appendix A: Source Code

The source code listed here is the code that is original to this paper or was taken and modified from Alicebot. Not all of the Alicebot code is listed here.

Respond.php

```
/*
  Program E
  Copyright 2002, Paul Rydell

  This file is part of Program E.

  Program E is free software; you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation; either version 2 of the License, or
  (at your option) any later version.

  Program E is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with Program E; if not, write to the Free Software
  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*/

/**
 * Respond functions
 *
 * Second layer functions that are prior to entering the AIML match routine.
 * @author Paul Rydell
 * @copyright 2002
 * @version 0.0.8
 * @license http://opensource.org/licenses/gpl-license.php GNU Public License
 * @package Interpreter
 */
/**
 * Start function for retrieving bot reply
 *
 * Checks to see if bot exists, if so calls reply() to get the repons to the user's input.
 *
 * @uses lookupbotid()
 * @uses reply()
 *
 * @param string $userinput      The user's input
 * @param integer $uniqueid      The user's session ID
 * @param string $botname        The bot's name, if no name selected the default value is "TestBot".
 *
 * @return string                The bot's reply.
 */
function replybotname($userinput,$uniqueid,$botname = "TestBot", $emotion)
{
```

```

//alvin
global $MOVIE_TALK;
global $user_emotion;
global $assessed_input_emotion;

console_output("replybotname > userinput", $userinput);

$user_emotion = $emotion;

$query = "select talk from " .MOVIE_DATA_TABLE;
$row = mysql_fetch_array(mysql_query($query));
$MOVIE_TALK = $row['talk'];

$botid = lookupbotid($botname);

/*$response = new_topic($userinput);
logconversation($userinput, $response->response);
return show_output($response);*/

if ($botid==1){
    print "I don't know that bot: $botname<BR>\n";
}
else if (strtolower($userinput) == "want to watch a movie?" && $MOVIE_TALK == 0)
{
    $query = "update " .MOVIE_DATA_TABLE. " set talk = 1";
    mysql_query($query);
    //$MOVIE_TALK = true;

    $response = new Response();
    $response->response = "What is the movie about?";
    $response->timer = 0.0;
    $response->emoticon = "";

    $user_emotion = "";
    $assessed_input_emotion = "";
    logconversation($userinput, $response->response);

    return show_output($response);
}
else if (strtolower($userinput) == "so, do you want to watch?" && $MOVIE_TALK == 1)
{
    $query = "update " .MOVIE_DATA_TABLE. " set talk = 0";
    mysql_query($query);
    $MOVIE_TALK = false;
    $response = movie_answer();

    $user_emotion = "";
    $assessed_input_emotion = "";
    logconversation($userinput, $response->response);

    return show_output($response);
}
else if ($MOVIE_TALK)
{
    $response = movie_concepts($userinput);
}

```



```

        $user_emotion = "";
        $assessed_input_emotion = "";
        logconversation($userinput, $response->response);

        return show_output($response);
    }
    else if (change_subject())
    {
        $response = new_topic($userinput);
        logconversation($userinput, $response->response);
        return show_output($response);
    }
    else
        return show_output(reply($userinput,$uniqueid,$botid));
}

/**
 * Main container function in creating the bot's reply.
 *
 * This function is the 'manager' of all the sub-funtions that do the real processing. It creates a class
 * called Response that is used throughout the application.
 *
 * @uses addinputs()
 * @uses addthats()
 * @uses bget()
 * @uses cleanup()
 * @uses getthat()
 * @uses loadcustomtags()
 * @uses logconversation()
 * @uses normalsentences()
 * @uses respond()
 * @uses ss_timing_current()
 * @uses ss_timing_start()
 * @uses ss_timing_stop()
 *
 * @global string that          The conversation's previous bot output
 * @global string topic        The contents of the AIML tag 'Topic'
 * @global integer uid         The session ID of the user (previously $uniqueid)
 * @global integer loopcounter   Counts the number of time a particular category is used in the same
match trace.
 * @global array patternmatched The pattern's that matched the
 *
 * @param string $userinput     The user's input.
 * @param integer $uniqueid     The user's session ID.
 * @param integer $bot          The bot's ID.
 *
 * @return object              A class link to 'Response'.
 */
function reply($userinput,$uniqueid, $bot = 1){

    global $that,$topic,$uid,$loopcounter,$patternmatched,$inputmatched,$selectbot;
    global $user_emotion, $assessed_input_emotion;

    cleanup();

```

```

ss_timing_start("all");

$patternmatched=array();
$inputmatched=array();

$myresponse = new Response;

$myresponse->errors="";

$uid=$uniqueid;
$selectbot=$bot;

// Load the custom plugin tags
loadcustomtags();

// Get the "that" and the "topic"
$that=getthat(1,1);
$topic=bget("TOPIC");

// Normalize the input
$allinputs=normalsentences($userinput);

// If nothing said then use INACTIVITY special input
if (sizeof($allinputs)==0){
    $allinputs[]="INACTIVITY";
}

// Put all the inputs into the <input> stack.
addinputs($allinputs);

$finalanswer="";
$finalanswer_array = Array();
// Build our response to all of the inputs.
//alvin: just do it for one inputted sentence for now
//for ($x=0;$x<sizeof($allinputs);$x++)
for ($x=0;$x<1;$x++)
{
    /*alvin*/
    //echo "allinputs:$x<br>";
    //print($allinputs[$x]);
    //echo "<br><br>";
    print_msg("allinputs", $allinputs[$x]);
    //$finalanswer.=respond($allinputs[$x]);
    //alvin:
    $answer_array = respond($allinputs[$x]);
}

if (($loopcounter>LOOPINGLIMIT)&&(LOOPINGLIMIT!=-1)){
    $finalanswer=LOOPINGERRORMSG;
    $myresponse->errors="LOOPINGLIMIT";
}

/*
    create sql statement to search for any words in the
    SIMILAR_WORDS table
*/

```

```

$conn = db_connect();
$result = "";

//Seiv_array = "";//Array("happy" => 0.0, "sad" => 0.0, "angry" => 0.0, "fear" => 0.0);
//$serv_arrays = Array();

//scan input statement
//assume just one inputted statement for now
$temp_array = Array();
//$assessed_input_emotion =
dominant_emotion(calculate_emotion(emotion_query($allinputs[0]));
//console_output("reply > assessed_input_emotion: $assessed_input_emotion");
/*for ($i = 0; $i < 1; $i++)
{
    //$temp_array = split(':', $inputmatched[$i]);
    $result = emotion_query($userinput);
}*/

//echo $query."<br><br>";
//$result = mysql_query($query);

/*if (mysql_num_rows($result) > 0)
{
    $seiv_array = calculate_emotion($result);
    //print_r($seiv_array);
}*/

//scan possible responses
//done backwards, since the 1st response is at the back, due to the
//recursive calls being made in respond()

print_msg("answerarray:", ($answer_array));
if (sizeof($answer_array) == 1 && !is_array($answer_array[0])) //single sentence
{
    //$finalanswer = $answer_array[0];
    $result = emotion_query($answer_array[0]);
    console_output("reply > single answer found");

    array_push($finalanswer_array, Array("sentence" => $answer_array[0], "erv" =>
calculate_emotion($result));
}
else
{
    for ($i = 0; $i < sizeof($answer_array); $i++)
    {
        for ($j = 0; $j < sizeof($answer_array[$i]); $j++)
        {
            if (validate_data(SENTENCE_REGEX, $answer_array[$i][$j]))
            {
                //echo $answer_array[$i][$j];
                $result = emotion_query($answer_array[$i][$j]);
                //array_push($serv_arrays, calculate_emotion($result));
                //$serv_arrays[$i][$j] = calculate_emotion($result);
                array_push($finalanswer_array, Array("sentence" =>
$answer_array[$i][$j], "erv" => calculate_emotion($result)));
            }
        }
    }
}

```

```

    }
  }
}

//temporary test data
//q: I really love cheese.
//$finalanswer_array = "";
/*$finalanswer_array = Array(Array("sentence" => "I do not care for cheese", "erv" =>
calculate_emotion(emotion_query("I do not care for cheese"))),
  Array("sentence" => "I love cheese too",
    "erv" => calculate_emotion(emotion_query("I love cheese too"))),
  Array("sentence" => "I hope there is a lot of cheese, but I fear it will be gone",
    "erv" => calculate_emotion(emotion_query("I hope there is a lot of
cheese, but I fear it will be gone"))),
  Array("sentence" => "I like cheese too",
    "erv" => calculate_emotion(emotion_query("I like cheese too"))),
  Array("sentence" => "I like cheese too, but I also abhor it",
    "erv" => calculate_emotion(emotion_query("I like cheese too, but I
also abhor it"))),
  Array("sentence" => "I like cheese, but I also detest it",
    "erv" => calculate_emotion(emotion_query("I like cheese, but I also
detest it"))),
  Array("sentence" => "I've got mixed emotions about cheese, I like it, I fear it,
I'm disappoint by it",
    "erv" => calculate_emotion(emotion_query("I've got mixed emotions
about cheese, I like it, I fear it, I'm disappoint by it"))),
  Array("sentence" => "Cheese is bad",
    "erv" => calculate_emotion(emotion_query("Cheese is bad"))),
  Array("sentence" => "I hate cheese",
    "erv" => calculate_emotion(emotion_query("I hate cheese"))),
  Array("sentence" => "I am astonished by cheese",
    "erv" => calculate_emotion(emotion_query("I am astonished by
cheese"))),
  Array("sentence" => "I am astonished you hate cheese",
    "erv" => calculate_emotion(emotion_query("I am astonished you hate
cheese"))),
  Array("sentence" => "Oh my god!",
    "erv" => calculate_emotion(emotion_query("Oh my god!"))),
  Array("sentence" => "I'm astonished, you love cheese too?",
    "erv" => calculate_emotion(emotion_query("I'm astonished, you love
cheese too?"))),
  Array("sentence" => "I am disgusted by cheese",
    "erv" => calculate_emotion(emotion_query("I am disgusted by
cheese"))),
  Array("sentence" => "I feel only happy and surprise for cheese.",
    "erv" => calculate_emotion(emotion_query("I feel only happy and
surprise for cheese."))));*/
}

print_msg("finalanswer", ($finalanswer_array));
console_output("Reply Result with ERVA:", $finalanswer_array);

if (strtolower($finalanswer_array[0]["sentence"]) == "topic")
  $finalanswer = find_topic($allinputs[0]);
else
{

```

```

        //decide which response is appropriate based on
        $finalanswer = filter_responses($finalanswer_array, $allinputs, $user_emotion);
        //$finalanswer = create_typo($finalanswer);
    }

    // Put the final answers into the <that> stack.
    addthats(normal_sentences($finalanswer));

    // Log the conversation
    logconversation($allinputs[0], $finalanswer);

    $myresponse->response=$finalanswer;
    $myresponse->patternsmatched=$patternmatched;
    $myresponse->inputs=$inputmatched;
    $myresponse->emoticon = get_emoticon("", SMALLEST_GIFS);

    ss_timing_stop("all");

    $myresponse->timer=ss_timing_current("all");

    return $myresponse;
}
/**
 * This is the second level response function.
 *
 * After reply() this function is the second level function to get the answer to the user's input.
 *
 * @uses bget()
 * @uses debugger()
 * @uses gettemplate()
 * @uses GetXMLTree()
 * @uses recursechildren()
 *
 * @global string
 * @global integer
 * @global array
 * @global array
 *
 * @param string $sentence    The sentence to be matched.
 *
 * #return string            The response to the user's input.
 */
function respond($sentence){

    global $that,$loopcounter,$patternmatched,$inputmatched;
    //alvin:
    $answers_array = Array();

    $topic = bget("topic");

    $loopcounter++;
    if (($loopcounter>LOOPINGLIMIT)&&(LOOPINGLIMIT != -1)){
        return "";
    }
}

```

```

$inputstarvals=array();
$thatstarvals=array();
$topicstarvals=array();

debugger("respond called with sentence: $sentence",3);
flush();

if ($that==""){
    $that="<nothing>";
}
if ($topic==""){
    $topic="<nothing>";
}

if ($sentence==""){
    return "";
}
else{

    //If we found a template

    $template=gettemplate($sentence,$that,$topic,$inputstarvals,$thatstarvals,$topicstarvals,$s_patternmatched,$s_inputmatched);

    $patternmatched[]=$s_patternmatched;
    $inputmatched[]=$s_inputmatched;

    if ($template!=""){

        $template="<xml><TEMPLATE>" . $template . "</TEMPLATE></xml>";

        debugger ("found template: $template",2);

        /*
            check for random, if so then break into several templates and
            call recursechildren on them
        */
        /*if (strpos($template, '<random>') != FALSE) // string found
        {
            $count = substr_count($template, '<li>');
            for ($i = 0; $i < $count; $i++)
            {
                $new_template = '<xml><template>';
                $new_template .= '</template></xml>';
            }
        }
        else
        {
            */
        $root=GetXMLTree($template);

        if (!isset($root[0]['children'][0]['value'])){

```

```

        $root=$root[0]['children'][0]['children'];
    }
    else {
        $root=$root[0]['children'][0]['value'];
    }
}
/*
    ALVIN: remove recursechildren and do my own
parsing and emotion decisions here. by this time all
the templates are returned.
can also parse the input as well for EIV
*/
/*
    create new array holding each LI then process those separately
for answers
*/
$new_roots_array = Array();
//$answers_array = Array();
if (is_array($root))
{
    //if (!isset($root['value']))
    {
        for ($x = 0; $x < sizeof($root); $x++)
        {
            if (is_array($root[$x]) &&
strtoupper($root[$x]['tag']) == "RANDOM")
            {
                for ($y = 0; $y <
sizeof($root[$x]['children']); $y++)
                {
                    array_push($new_roots_array,
                    $root[$x]['children'][$y]);
                }
            }
            else if ($root[$x] != chr(10))
            {
                array_push($new_roots_array, $root[$x]);
            }
        }
    }
}
else
{
    //single_reply_array = Array();
    //array_push($single_reply_array, $root);
    //array_push($answers_array, $single_reply_array);
array_push($answers_array, $root);
    return $answers_array;
}

//let recursechildren process the arrays to get the full reply sentence
$sub_array = Array();

```

```

        for ($i = 0; $i < sizeof($new_roots_array); $i++)
        {
            //if (is_array($new_roots_array[$i]) && isset($new_roots_array[$i]))
            if (isset($new_roots_array[$i]) && !empty($new_roots_array[$i]) &&
$new_roots_array[$i] != chr(10))
                {
                    $temp_array = Array();
                    array_push($temp_array, $new_roots_array[$i]);

                    $resp1=recursechildren($temp_array,$inputstarvals,$thatstarvals,$topicstarvals);
                    if ($resp1 != "")
                        //if (validate_data(SENTENCE_REGEX, $resp1))
                        {
                            //echo "resp1:$resp1<br>";
                            //array_push($answers_array, $resp1);
                            array_push($sub_array, $resp1);
                            //echo "resp1:".ord($resp1)."<br>";
                        }
                }
            }

            array_push($answers_array, $sub_array);

/*echo "root:<br>";
print_r($root);
echo "<br/><br/>";
echo "input:<br>";
print_r($inputmatched);
echo "<br/><br/>";
echo "template:<br>";
print_r($template);
echo "<br/><br/>";
echo "new_roots:<br>";
print_r($new_roots_array);
echo "<br/><br/>";
echo "answers_array:<br>";
print_r($answers_array);
echo "<br/><br/>";*/

//echo "ORIGINALS<BR><BR>";
$myresponse=recursechildren($root,$inputstarvals,$thatstarvals,$topicstarvals);
//debugger("recursechildren ret: $myresponse",3);

//return $myresponse;
//return "";
return $answers_array;
}
}
}

```

Common.php

```

<?php
include('kd_xmlrpc.php');

```



```

define ('SIMILAR_WORDS_TABLE', 'alvin_similar_words');
define ('EKMAN_TABLE', 'alvin_ekman');
define ('OCC_TABLE', 'alvin_occ');
define ('FACTOR_FIVE_TABLE', 'alvin_factorfive');
define('CES_TABLE', 'alvin_bot_emotional_state');
define('USER_CES_TABLE', 'alvin_user_emotional_state');
define('PREFERENCES_TABLE', 'alvin_movie_prefs');
define('CONCEPTS_TABLE', 'alvin_movie_concepts');
define('MOVIE_DATA_TABLE', 'alvin_movie_data');
define('EIVA_TABLE', 'alvin_emotional_intent_vals');
define('ERVA_TABLE', 'alvin_emotional_response_vals');
define('CONVERSATION_LOG_TABLE', 'conversationlog');
define('TOPICS_TABLE', 'alvin_topics');
define('BOT_DATA_TABLE', 'alvin_bot_data');
define('TYPO_TABLE', 'alvin_misspelled');
define('SENTENCE_REGEX', '/^[0-9a-zA-Z][0-9a-zA-Z\s-\.\:!"?\'*\/]');
define('MIN_TOPIC_DISTANCE', 10);
define('MAX_TOPIC_DISTANCE', 30);
define('MAX_TOPIC_DISTANCE_EXTRAVERT', 20);
define('CURRENT_CES', 0);
define('LAST_CES', 1);
define('N_CES', 5);
define('N_CONCEPTS', 5);
define('N_TOPICS', 5);
define('TOPIC_HIT_PERCENT', .85);
define('N_TREND', 10);
define('HAPPY_TREND_MIN', .50);
define('NEUROTIC_TREND_MIN', .70);
define('TYPO_MIN', .85);
define('SURPRISE_MIN', .50);
define('EXTRAVERT_THINKER', 1);
define('NEUROTIC_THINKER', 0);
define('BOT_PERSPECTIVE', 1);
define('USER_PERSPECTIVE', 0);
define('SMALLEST_GIFS', 1);
define('SMALL_GIFS', 0);
define('NORMAL_MODE', 0); //control dominant_emotion, do regular assessment
define('INPUT_MODE', 1); //do not randomly choose emotion, for dominant_emotion
define('SITE', 'localhost:8000');
define('LOCATION', 'kd_xmlrpc.php');
define('DBNAME', 'bot');
define('USER', 'root');
define('PASSWD', 'start00');
define('SERVER', 'localhost');
set_time_limit(180);
//globals
$MOVIE_TALK = false;
//$movie_meter = 0.0;

require_once("util.php");

class Word
{
    var $word;
    var $emotion;
    var $intensity;

```

```

    var $occ;
}

/*
  Descrip: setup connection to db
*/
function db_connect()
{
    $conn = mysql_connect(SERVER, USER, PASSWD) or die('Cannot connect');
    mysql_select_db(DBNAME) or die('Cannot select database');

    return $conn;
}

function get_emoticon($emotion="", $is_smallest)
{
    if ($emotion == "")
    {
        $query = "select last_emotion from " .CES_TABLE. " order by id desc limit 1";
        $row = mysql_fetch_array(mysql_query($query));
        $emotion = $row["last_emotion"];
    }

    //check surprise intensity
    $ces = get_ces();

    //todo: should there be a min?
    if ($ces['surprise'] > SURPRISE_MIN && $emotion != "disgust" && $emotion != "surprise")
    {
        $emotion = "surprise_". $emotion;
    }

    $image = "<img align=middle src=images/";

    print_msg("getemoticon > emotion: ", $emotion);

    if ($emotion == "happy")
        $image .= "happy";
    else if ($emotion == "sad")
        $image .= "sad";
    else if ($emotion == "angry")
        $image .= "angry";
    else if ($emotion == "fear")
        $image .= "fear";
    else if ($emotion == "surprise")
        $image .= "surprise";
    else if ($emotion == "surprise_happy")
        $image .= "surprise_happy";
    else if ($emotion == "surprise_sad")
        $image .= "surprise_sad";
    else if ($emotion == "surprise_angry")
        $image .= "surprise_angry";
    else if ($emotion == "surprise_fear")
        $image .= "surprise_fear";
    else if ($emotion == "disgust")
        $image .= "disgust";
}

```

```

else
    return "None";

if ($is_smallest)
    return $image .= "-smallest.gif>";

return $image .= "-small.gif>";
}

/*
  @description: for printing debug messages. $debug_msg is in talk.php
  @param: $msg, message to display
  @param: $value, value to display
*/
function print_msg($msg, $value)
{
    global $debug_msg;

    if ($debug_msg)
    {
        echo "<br><br>Debug ".$msg."<br>";
        if (is_array($value))
        {
            echo "<pre>".print_r($value, true)."</pre><br><br>";
        }
        else
            echo $value."<br><br>";
    }
}

//???possible problems Fatal error: Maximum function nesting level of '100' reached, aborting!
/*
  @description: sort array
  @param: $array, array of numerical values
*/
function rquicksort($array)
{
    $z = array();
    if(count($array)>1)
    {
        $k = $array[0];
        $x = array();
        $y = array();
        for($i=1; $i<count($array); $i++)
        {
            if($array[$i][1] <= $k[1])
            {
                $x[] = $array[$i];
            }
            else
            {
                $y[] = $array[$i];
            }
        }
        $x = rquicksort($x);
        $y = rquicksort($y);
    }
}

```

```

        //return array_merge($x, $k, $y);
        array_push($z, $k);
        return array_merge($y, $z, $x);
    }
    else
    {
        return $array;
    }
}

/*
 * @description: get the CES from the user or bot ces table depending on the mode
 */
function get_ces($mode = NORMAL_MODE)
{
    $query = "";
    $ces_array = Array("happy" => 0.0, "sad" => 0.0, "angry" => 0.0, "fear" => 0.0, "surprise" => 0.0,
"disgust" => 0.0);
    console_output("get_ces > mode:", $mode);

    if ($mode == NORMAL_MODE)
        $query = "select * from ".CES_TABLE. " order by id desc limit 1";
    else
        $query = "select * from ".USER_CES_TABLE." order by id desc limit 1";

    $result = mysql_query($query);
    if (mysql_num_rows($result) > 0)
    {
        $row = mysql_fetch_array($result);

        /*$ces_array['happy'] = round($ces_array['happy'], 2);
        $ces_array['sad'] = round($ces_array['sad'], 2);
        $ces_array['angry'] = round($ces_array['angry'], 2);
        $ces_array['fear'] = round($ces_array['fear'], 2);
        $ces_array['surprise'] = round($ces_array['surprise'], 2);
        $ces_array['disgust'] = round($ces_array['disgust'], 2);*/
        $ces_array['happy'] = ($row['happy']);
        $ces_array['sad'] = ($row['sad']);
        $ces_array['angry'] = ($row['angry']);
        $ces_array['fear'] = ($row['fear']);
        $ces_array['surprise'] = ($row['surprise']);
        $ces_array['disgust'] = ($row['disgust']);
    }

    console_output("CES:", $ces_array);

    print_msg("getCES: ces", $ces_array);
    return $ces_array;
}

function get_rounded_ces()
{
    $ces_array = get_ces();

    $keys = array_keys($ces_array);

```

```

for($i = 0; $i < sizeof($keys); $i++)
{
    if ($ces_array[$keys[$i]] > 1.0)
        $ces_array[$keys[$i]] = 1.0;
}

/*$ces_array['happy'] = sprintf("%.3e", $ces_array['happy']);
$ces_array['sad'] = sprintf("%.3e", $ces_array['sad']);
$ces_array['angry'] = sprintf("%.3e", $ces_array['angry']);
$ces_array['fear'] = sprintf("%.3e", $ces_array['fear']);
$ces_array['surprise'] = sprintf("%.3e", $ces_array['surprise']);
$ces_array['disgust'] = sprintf("%.3e", $ces_array['disgust']);*/
$ces_array['happy'] = round($ces_array['happy'], 2);
$ces_array['sad'] = round($ces_array['sad'], 2);
$ces_array['angry'] = round($ces_array['angry'], 2);
$ces_array['fear'] = round($ces_array['fear'], 2);
$ces_array['surprise'] = round($ces_array['surprise'], 2);
$ces_array['disgust'] = round($ces_array['disgust'], 2);

return $ces_array;
}

function get_last_emotion()
{
    $query = "select last_emotion from " .CES_TABLE. " order by id desc limit 1";
    $row = mysql_fetch_array(mysql_query($query));
    return $row["last_emotion"];
}

function calculate_ces($new_values, $mode)
{
    console_output("calculate_ces > mode:", $mode);

    $ces_array = get_ces($mode);
    console_output("calculate_ces > ces_array:", $ces_array);

    $ces_array['happy'] = exp(-1) * $ces_array['happy'] + $new_values['happy'];
    $ces_array['sad'] = exp(-1) * $ces_array['sad'] + $new_values['sad'];
    $ces_array['angry'] = exp(-1) * $ces_array['angry'] + $new_values['angry'];
    $ces_array['fear'] = exp(-1) * $ces_array['fear'] + $new_values['fear'];
    $ces_array['surprise'] = exp(-1) * $ces_array['surprise'] + $new_values['surprise'];
    $ces_array['disgust'] = exp(-1) * $ces_array['disgust'] + $new_values['disgust'];
    console_output("calculate_ces > new ces_array", $ces_array);

    $keys = array_keys($ces_array);
    for($i = 0; $i < sizeof($keys); $i++)
    {
        if ($ces_array[$keys[$i]] < 0.01)
            $ces_array[$keys[$i]] = 0.00;
        else if ($ces_array[$keys[$i]] > 1.0)
            $ces_array[$keys[$i]] = 1.00;
    }

    return $ces_array;
}

```

```

/*$ces_array = Array("happy" => 0.0, "sad" => 0.0, "angry" => 0.0, "fear" => 0.0, "surprise" => 0.0,
"disgust" => 0.0);
$happy = 0.0;
$sad = 0.0;
$angry = 0.0;
$fear = 0.0;
$surprise = 0.0;
$disgust = 0.0;

$query = 'select happy, sad, angry, fear, surprise, disgust from ' .CES_TABLE. ' order by id asc';
$result = mysql_query($query);

print_msg("calces: query", $query);

if (mysql_num_rows($result) == 1)
{
    $row = mysql_fetch_array($result);
    $ces_array['happy'] = $row['happy'];
    $ces_array['sad'] = $row['sad'];
    $ces_array['angry'] = $row['angry'];
    $ces_array['fear'] = $row['fear'];
    $ces_array['surprise'] = $row['surprise'];
    $ces_array['disgust'] = $row['disgust'];
    return $ces_array;
}
else if (mysql_num_rows($result) == 0)
    return $ces_array;

for ($i = 1; $i <= mysql_num_rows($result) && $row = mysql_fetch_array($result); $i++)
{
    if ($i == 1)
    {
        $happy = $row['happy'];
        $sad = $row['sad'];
        $angry = $row['angry'];
        $fear = $row['fear'];
        $surprise = $row['surprise'];
        $disgust = $row['disgust'];
    }

    //$happy = $happy * exp((-1 * $i)) + $row['happy'];
    //$sad = $sad * exp((-1 * $i)) + $row['sad'];
    //$angry = $angry * exp((-1 * $i)) + $row['angry'];
    //$fear = $fear * exp((-1 * $i)) + $row['fear'];
    //$surprise = $surprise * exp((-1 * $i)) + $row['surprise'];
    //$disgust = $disgust * exp((-1 * $i)) + $row['disgust'];
    $happy = $happy * exp(-1) + $row['happy'];
    $sad = $sad * exp(-1) + $row['sad'];
    $angry = $angry * exp(-1) + $row['angry'];
    $fear = $fear * exp(-1) + $row['fear'];
    $surprise = $surprise * exp(-1) + $row['surprise'];
    $disgust = $disgust * exp(-1) + $row['disgust'];
}

$ces_array['happy'] = $happy;
$ces_array['sad'] = $sad;

```

```

$ces_array['angry'] = $angry;
$ces_array['fear'] = $fear;
$ces_array['surprise'] = $surprise;
$ces_array['disgust'] = $disgust;

console_output("calculate_ces > ces", $ces_array);
return $ces_array;*/
}

/*
 @description: updates ces by adding previous values & doing avg
 also set last emotion, which would be the highest value (or rand if equal)
 @todo: should be temporal...
*/
function update_bot_ces($finalanswer)
{
    $last_emotion = "";
    $serva = $finalanswer['erv'];
    $query = 'select count(id) total from '. CES_TABLE;
    $row = mysql_fetch_array(mysql_query($query));
    $new_total = $row['total'] + 1;

    //caluclate new CES to decide
    $ces_array = calculate_ces($serva, NORMAL_MODE);
    //$ces_array = get_ces();

    //console_output("Update CES > ces: ", $ces_array);
    /*$ces_array['happy'] = exp(-1) * $ces_array['happy'] + $serva['happy'];
    $ces_array['sad'] = exp(-1) * $ces_array['sad'] + $serva['sad'];
    $ces_array['angry'] = exp(-1) * $ces_array['angry'] + $serva['angry'];
    $ces_array['fear'] = exp(-1) * $ces_array['fear'] + $serva['fear'];
    $ces_array['surprise'] = exp(-1) * $ces_array['surprise'] + $serva['surprise'];
    $ces_array['disgust'] = exp(-1) * $ces_array['disgust'] + $serva['disgust']; */

    console_output("update_bot_ces > final answer", $finalanswer);
    console_output("update_bot_ces > new ces: ", $ces_array);

    $last_emotion = dominant_emotion($ces_array);
    //$serva_emotion = dominant_emotion($serva);

    print_msg("update_bot_ces: lastemotion", $last_emotion);

    $sen = mysql_escape_string($finalanswer['sentence']);
    //remove_punctuation(strtolower($finalanswer['sentence']));
    $query = "insert into ".ERVA_TABLE. " (sentence, happy, sad, angry, fear, surprise, disgust) values (
    ...
        .$.sen.", ".$serva["happy"].", ".$serva["sad"].", ".$serva["angry"].", ".$serva["fear"].",
    ".$serva["surprise"].", "
        .$.serva["disgust"].");";
    console_output("update_bot_ces > insert query: ".$query);
    mysql_query($query);

    $query = "insert into ".CES_TABLE. " (happy, sad, angry, fear, surprise, disgust, last_emotion)
    values ( "

```

```

        .Sces_array["happy"].", ".Sces_array["sad"].", ".Sces_array["angry"].", ".Sces_array["fear"].",
        ".Sces_array["surprise"].", "
        .Sces_array["disgust"].", "".$last_emotion."");
    print_msg("update_bot_ces: query: ", $query);
    console_output("update_bot_ces > query:", $query);

    mysql_query($query);
}

function update_user_ces($input)
{
    global $user_emotion;
    global $assessed_input_emotion;

    $input_eiv = calculate_emotion(emotion_query($input));
    //print_r($input_eiv);
    arsort($input_eiv, SORT_NUMERIC);
    $emotion = dominant_emotion($input_eiv, INPUT_MODE); //todo:possible bug, need to look at input
    history if there are equal emotions
    $assessed_input_emotion = $emotion;

    console_output("update_user_ces > EIVA:", $input_eiv);
    console_output("update_user_ces > emotion:", $emotion);
    console_output("User Emoticon: ".$user_emotion);
    //compare assessed emotion vs. emoticon
    //if same assesment was correct, otherwise use emoticon instead
    print_msg("assessed emotion", $emotion);
    print_msg("emoticon", $user_emotion);
    $emotion = compare_emotions($emotion, $user_emotion);
    print_msg("finalemotion", $emotion);

    console_output("update_user_ces > emotion after compare:", $emotion);

    //store EIVA & sentence
    $sentence = mysql_escape_string($input); //remove_punctuation(strtolower($inputs[0]));
    $query = "insert into ".EIVA_TABLE. " (sentence, happy, sad, angry, fear, surprise, disgust,
    last_emotion) values ( "
        . $sentence.", ".$input_eiv["happy"].", ".$input_eiv["sad"].", ".$input_eiv["angry"].",
        ".$input_eiv["fear"].", ".$input_eiv["surprise"].", "
        . $input_eiv["disgust"].", "".$emotion."");
    console_output("update_user_ces > insert query: ".$query);
    mysql_query($query);

    //update user CES
    //use emotion from compare_emotion(). not highest value from ces_array.
    //that way i can preserve the emoticon chosen by the user, my CES maynot be correct
    $ces_array = calculate_ces($input_eiv, INPUT_MODE);

    $emotion = dominant_emotion($ces_array, INPUT_MODE);
    console_output("update_user_ces > ces emotion:", $emotion);
    $emotion = compare_emotions($emotion, $user_emotion);
    console_output("update_user_ces > ces emotion after compare:", $emotion);

    $query = "insert into ".USER_CES_TABLE. " (happy, sad, angry, fear, surprise, disgust,
    last_emotion) values ( "

```



```

        .Sces_array["happy"].", ".Sces_array["sad"].", ".Sces_array["angry"].", ".Sces_array["fear"].",
        ".Sces_array["surprise"].", "
        .Sces_array["disgust"].", "".$emotion."");
        console_output("update_user_ces > ces insert query:", $query);
        mysql_query($query);

        return $emotion;
    }

    /*
    @description: get the personality
    */
    function get_personality()
    {
        //return Array("extraversion" => 0.81, "openness" => 0.47, "agreeableness" => 0.10, "neurotic" =>
        0.55, "conscientiousness" => 0.47);
        $personality_array = Array("extraversion" => 0.0, "openness" => 0.0, "agreeableness" => 0.0,
        "neurotic" => 0.0, "conscientiousness" => 0.0);
        $query = "select * from ".FACTOR_FIVE_TABLE;
        $result = mysql_query($query);

        while ($row = mysql_fetch_array($result))
        {
            $personality_array[$row["factor"]] = $row["score"];
        }

        return $personality_array;
    }

    /*
    @description: find the most dominant emotion based on highest value.
    if values equal, then use past history of choices, if no choices based off personality.
    @param: $emotion_array, array of basic emotions & values
    @return: string, the emotion
    @todo: fix same value code => done
    @todo: ignore surprise & disgust => done
    */
    function dominant_emotion($emotion_array, $mode = NORMAL_MODE)
    {
        $demotion = "";
        //todo:
        //zero out surprise & disgust, to ignore these values
        //because scan for underlying emotions, not just surprised, but why?
        //how about disgust or just normal surprise?
        //disgust ok
        //as for surprise, only 1 dominant, not both surprise & happy
        //therefore, since underlying for surprise happy would need to mean
        //both happy & surprise are high
        //for normal surprise, then must mean everything else ZERO
        //$emotion_array["surprise"] = 0.0;
        //$emotion_array["disgust"] = 0.0;

        arsort($emotion_array, SORT_NUMERIC);

        $e1 = current($emotion_array);
        next($emotion_array);
    }

```

```

$e2 = current($emotion_array);

console_output("dominant_emotion > mode:", $mode);

/*if ($mode == INPUT_MODE && $e1 == 0)
{
    $demotion = "";
    console_output("dominant_emotion > input mode");
}*/
if ($e1 == $e2 && ($e1 == 0))
{
    $demotion = "";
    console_output("dominant_emotion > zero values");
}
else if ($e1 == $e2) // same value
{
    $last_emotion = "";
    $last_emotion2 = "";
    $tmp = Array();

    reset($emotion_array);

    while ($e = each($emotion_array))
    {
        if ($e["value"] == $e1)
            $tmp[$e["key"]] = $e["value"];
    }
    console_output("dominant_emotion > same emotion values:", $tmp);

    if ($mode != INPUT_MODE)
        $query = 'SELECT last_emotion FROM ' .CES_TABLE. ' order by id desc';
    else
        $query = 'SELECT last_emotion from ' .USER_CES_TABLE. ' order by id desc';

    $result = mysql_query($query);
    //$row = mysql_fetch_array($result);
    //$last_emotion = $row['last_emotion'];

    if (mysql_num_rows($result) == 0)
    {
        $last_emotion = "";
    }
    else
    {
        $row = mysql_fetch_array($result);
        $last_emotion = $row['last_emotion'];

        if (mysql_num_rows($result) > 1)
        {
            $row = mysql_fetch_array($result);
            $last_emotion2 = $row['last_emotion'];
        }
    }

    if ($last_emotion != "" && array_key_exists($last_emotion, $tmp))
        $demotion = $last_emotion;

```

```

else if ($last_emotion2 != " && array_key_exists($last_emotion2, $tmp))
    $demotion = $last_emotion2;
else
    $demotion = rand_emotion($tmp, $mode);

/*if (mysql_num_rows($result) == 0 || $row["last_emotion"] == "")
{
    if (mysql_num_rows($result) > 1)
        $row = mysql_fetch_array($result);

    if ($row["last_emotion"] == "")
        $demotion = rand_emotion($tmp, $mode);
    else
        $demotion = $row["last_emotion"];
}
else //check if last_emotion is within the $tmp set
{
    $row = mysql_fetch_array($result);
    //$demotion = $row['last_emotion'];

    if (array_key_exists($row['last_emotion'], $tmp))
    {
        $demotion = $row['last_emotion'];
    }
    else
    {
        $demotion = rand_emotion($tmp, $mode);
    }
}*/
/*if (mysql_num_rows($result) == 0 || $row["last_emotion"] == "") //no rows, use personality to
decide emotion
{
    $demotion = rand_emotion($tmp);
}
else
{
    $row = mysql_fetch_array($result);
    $demotion = $row['last_emotion'];
}*/
}
else if ($e1 > $e2)//e1 larger, since we sorted in reverse order
{
    $demotion = array_search($e1, $emotion_array);

    if ($demotion == 'surprise') //if suprise, make sure other emotions zero
    {
        //since in the rsorted array surprise is 1st, then if do sort, it will be last
        //then to a pop to remove
        asort($emotion_array);
        array_pop($emotion_array);

        foreach ($emotion_array as $e)
        {
            if ($e !=0) //if not, then find next highest value
            {

```

```

        $demotion = dominant_emotion($emotion_array, $mode);

        break;
    }
}
}

console_output("dominant_emotion > Dominant Assessed Emotion: ".$demotion);
return $demotion;
}

/*
get emotion based on personality
@param: $equal_emotions: array of emotions with the same value
*/
function rand_emotion($equal_emotions, $mode = NORMAL_MODE)
{
    $demotion = "";

    if ($mode == INPUT_MODE)
    {
        //$personality = get_personality();
        $personality = dominant_personality();
        //$e_array = Array("sad", "fear", "angry");

        if (array_key_exists('happy', $equal_emotions) &&
            $personality == "extraversion")
            //($personality["extraversion"] > $personality["neurotic"] ||
            // $personality["agreeableness"] > $personality["neurotic"]))
            {
                $demotion = "happy";
            }
        //else if ($personality["neurotic"] > $personality["extraversion"] &&
        // $personality["neurotic"] > $personality["agreeableness"])
        else if ($personality == "neurotic");
        {
            if (array_key_exists('sad', $equal_emotions))
                array_push($e_array, 'sad');

            if (array_key_exists('fear', $equal_emotions))
                array_push($e_array, 'fear');

            if (array_key_exists('angry', $equal_emotions))
                array_push($e_array, 'angry');

            if (sizeof($equal_emotions) > 0)
                $demotion = $e_array[random_num(sizeof($e_array))];
        }
    }
}

if ($demotion == "")
{
    $e_array = Array("happy", "sad", "fear", "angry", "surprise", "disgust");
    $demotion = $e_array[random_num(sizeof($equal_emotions))];
}
}

```

```

    return $demotion;
}

/*
  @todo: leans towards being good natured?
*/
function dominant_personality()
{
    $personality = get_personality();
    $personality["openness"] = 0.0;
    $personality["conscientiousness"] = 0.0;
    $dpersonality = "";

    arsort($personality, SORT_NUMERIC);
    print_msg("sorted:", ($personality));
    $p1 = current($personality);
    next($personality);
    $p2 = current($personality);

    if ($p1 > $p2)
    {
        $dpersonality = array_search($p1, $personality);
    }
    else //possible same values (extravert / agree) vs neurotic
    {
        //cant just randomly decide, personality is static
        $query = "select thought_process from " .BOT_DATA_TABLE;
        $row = mysql_fetch_array(mysql_query($query));

        if ($row["thought_process"] == EXTRAVERT_THINKER) //extraversion
            $dpersonality = "extraversion";
        else
            $dpersonality = "neurotic";
    }

    console_output("Dominant Personality:". $dpersonality);
    return $dpersonality;
}

/*
  @description: takes a set of possible replies and filters them down based on emotions until finally
  randomly choosing a single reply
  @param: $finalanswer_array, the replies
  @param: $allinputs, input sentence from the user
  @return: string, the chosen reply
  @todo: filter style depends on personality=> done
  @todo: choosing random replies?
*/
function filter_responses($finalanswer_array, $allinputs)
{
    $query = "select thought_process from " .BOT_DATA_TABLE;
    $row = mysql_fetch_array(mysql_query($query));
    $reply = "";
    $filter1_result = "";
    $filter2_result = "";

```

```

$filter3_reuslt = "";

if ($row["thought_process"] == EXTRAVERT_THINKER) //extraversion
{
    print_msg("EXTRAVERT_THINKER", "");
    console_output("Filter as Extarvert:<br>");

    $filter1_result = input_filter($finalanswer_array, $allinputs);
    print_msg("filter1result", "");
    print_msg("filter1array", ($filter1_result));
    console_output("Input Filter Results:", $filter1_result);

    /*if (sizeof($filter1_result) == 1)
        $reply = $filter1_result[0];
        //return $filter1_result[0]["sentence"];*/

    $filter2_result = ces_filter($filter1_result);
    print_msg("filter2result", "");
    print_msg("filter2array", ($filter2_result));
    console_output("CES Filter Results:", $filter2_result);

    /*if (sizeof($filter2_result) == 1)
        $reply = $filter2_result[0];*/

    $filter3_result = personality_filter($filter2_result);
    print_msg("filter3result", "");
    print_msg("filter3array", ($filter3_result));
    console_output("Personality Filter Results:", $filter3_result);

    if (sizeof($filter3_result) == 1)
        $reply = $filter3_result[0];
    }
else if ($row["thought_process"] == NEUROTIC_THINKER) //neurotic
{
    print_msg("NEUROTIC_THINKER", "");
    console_output("Filter as Neurotic:");

    $filter1_result = personality_filter($finalanswer_array);
    print_msg("filter1result", "");
    print_msg("filter1array", ($filter1_result));
    console_output("Personality Filter Results:", $filter1_result);

    /*if (sizeof($filter1_result) == 1)
        $reply = $filter1_result[0];*/

    $filter2_result = ces_filter($filter1_result);
    print_msg("filter2result", "");
    print_msg("filter2array", ($filter2_result));
    console_output("CES Filter Results:", $filter2_result);

    /*if (sizeof($filter2_result) == 1)
        $reply = $filter2_result[0];*/

    $filter3_result = input_filter($filter2_result, $allinputs);
    print_msg("filter3result", "");
    print_msg("filter3array", ($filter3_result));
}

```

```

    console_output("input_filter Filter Results:", $filter3_result);

    if (sizeof($filter3_result) == 1)
        $reply = $filter3_result[0];
    }

    if (empty($reply))
    {
        console_output("filter_responses > picking random reply");
        $random_num = random_num(sizeof($filter3_result));
        update_bot_ces($filter3_result[$random_num]);
        $reply = $filter3_result[$random_num];
    }
    else
        update_bot_ces($reply);

    $reply_sentence = $reply['sentence'];
    console_output("Filter Responses > answer: ", $reply_sentence);
    //logconversation($allinputs[0], $reply_sentence);

    return $reply_sentence;
}

//filter responses based on $max_emotion key
//finds all responses where $max_emotion is the highest value
//todo: same value
function emotion_filter($finalanswer_array, $max_emotion)
{
    $responses_array = Array();
    $temp_array = Array();

    //find all responses that have a high max_emotion
    for ($i = 0; $i < sizeof($finalanswer_array); $i++)
    {
        $serv = $finalanswer_array[$i]['erv'];
        arsort($serv, SORT_NUMERIC);
        $max_value = $finalanswer_array[$i]['erv'][$max_emotion];
        $max = max($serv);

        //find max emotion, elements can have the same value
        //if ($max_value >= $max)
        if ($max_value > 0.0)
        {
            array_push($temp_array, $finalanswer_array[$i]);
        }
    }
}

print_msg("temp", "");
print_msg("temparray", ($temp_array));

if (sizeof($temp_array) == 0) //nothing found, just return all answers
{
    console_output("emotion_filter > nothing filtered");
    return $finalanswer_array;
}

```

```

    return $temp_array;
}

//find all responses where the dominant CES emotion
//is the highest value in the responses
function ces_filter($responses)
{
    if (sizeof($responses) == 1)
        return $responses;
    //$ces_array = get_ces();
    //echo "<br>ces:";
    //print_r($ces_array);
    //print_msg("ces", ($ces_array));
    $emotion = get_last_emotion();//dominant_emotion($ces_array);//array_search(max($ces_array),
$ces_array);
    //echo "<br>max:$emotion";
    //$filter1_result = Array();
    console_output("ces_filter > emotion:", $emotion);

    if (emotion_exists($responses, $emotion))
        return emotion_filter($responses, $emotion);

    return $responses; //if nothing found just return all the responses
}

/*
 @description: find all responses that match the dominant personality factor
 @todo: choosing based on last n? should i do a trend instead? what about the while loop?
 if happy is encountered, it is just ignored, along with surprise & disgust.
 allow sad/angry/fear to be all possible answers, since a neurotic personality is more prone
 to these emotions => done
*/
function personality_filter($responses)
{
    if (sizeof($responses) == 1)
        return $responses;

    //$personality_array = get_personality();
    //arsort($personality_array);

    //$factor = current($personality_array);
    $factor = dominant_personality();
    print_msg("max factor:", $factor);
    //foreach($personality_array as $factor)
    //{
        if ($factor == "extraversion" || $factor == "agreeableness")
        {
            if (emotion_exists($responses, "happy"))
            {
                return emotion_filter($responses, "happy");
            }
        }
        else if ($factor == "neurotic")
        {
            if (emotion_exists($responses, "sad") || emotion_exists($responses, "angry") ||
emotion_exists($responses, "fear"))

```



```

{
    $result = Array();

    foreach($responses as $r)
    {
        $happy = $r["erv"]["happy"];
        $sad = $r["erv"]["sad"];
        $angry = $r["erv"]["angry"];
        $fear = $r["erv"]["fear"];

        //if ($fear > 0.0 || $sad > 0.0 || $angry > 0.0)
        if ($fear > $happy || $sad > $happy || $angry > $happy)
            array_push($result, $r);
    }

    print_msg("personality filter", ($result));

    if (sizeof($result) > 0)
        return $result;
    else
        return $responses;
    /*//choose based on the last N CESs
    $query = "SELECT last_emotion from " .CES_TABLE. " order by id desc limit " .N_CES;
    $result = mysql_query($query);

    $emotion = "";
    while ($row = mysql_fetch_array($result))
    {
        if ($row['last_emotion'] == "sad")
        {
            if (emotion_exists($responses, "sad"))
                return emotion_filter($responses, "sad");
        }
        else if ($row['last_emotion'] == "angry")
        {
            if (emotion_exists($responses, "angry"))
                return emotion_filter($responses, "angry");
        }
        else if ($row['last_emotion'] == "fear")
        {
            if (emotion_exists($responses, "fear"))
                return emotion_filter($responses, "fear");
        }
    }
    */
}
//}

//nothing was found, then return all responses
return $responses;
}

//find all responses that match the dominant EIV emotion
//aside from surprise & disgust
function input_filter($responses, $inputs)
{

```

```

/*
global $user_emotion;
global $assessed_input_emotion;

$input_eiv = calculate_emotion(emotion_query($inputs[0]));
//print_r($input_eiv);
arsort($input_eiv, SORT_NUMERIC);
$emotion = dominant_emotion($input_eiv, INPUT_MODE); //todo:possible bug, need to look at input
history if there are equal emotions
$assessed_input_emotion = $emotion;

console_output("Input Filter EIVA:", $input_eiv);
console_output("input_filter > emotion:", $emotion);
console_output("User Emoticon: ".$user_emotion);
//compare assessed emotion vs. emoticon
//if same assesment was correct, otherwise use emoticon instead
print_msg("assed emotion", $emotion);
print_msg("emoticon", $user_emotion);
$emotion = compare_emotions($emotion, $user_emotion);
print_msg("finalemotion", $emotion);

console_output("input_filter > emotion after compare:", $emotion);

//store EIVA & sentence
$sentence = mysql_escape_string($inputs[0]); //remove_puncutation(strtolower($inputs[0]));
$query = "insert into ".EIVA_TABLE. " (sentence, happy, sad, angry, fear, surprise, disgust,
last_emotion) values ( "
    .$sentence.", ".$input_eiv["happy"].", ".$input_eiv["sad"].", ".$input_eiv["angry"].",
".$input_eiv["fear"].", ".$input_eiv["surprise"].", "
    .$input_eiv["disgust"].", ".$emotion."");
console_output("input_filter > insert query: ".$query);
mysql_query($query);

//update user CES
//use emotion from compare_emotion(). not highest value from ces_array.
//that way i can preserve the emoticon choosen by the user, my CES maynot be correct
$ces_array = calculate_ces($input_eiv, INPUT_MODE);

$emotion = dominant_emotion($ces_array, INPUT_MODE);
console_output("input_filter > ces emotion:", $emotion);
$emotion = compare_emotions($emotion, $user_emotion);
console_output("input_filter > ces emotion after compare:", $emotion);

$query = "insert into ".USER_CES_TABLE. " (happy, sad, angry, fear, surprise, disgust,
last_emotion) values ( "
    .$ces_array["happy"].", ".$ces_array["sad"].", ".$ces_array["angry"].", ".$ces_array["fear"].",
".$ces_array["surprise"].", "
    .$ces_array["disgust"].", ".$emotion."");
console_output("input_filter > ces insert query:", $query);
mysql_query($query);
*/

$emotion = update_user_ces($inputs[0]);

if (emotion_exists($responses, $emotion))
{

```

```

    return emotion_filter($responses, $emotion);
}

return $responses; //if nothing found just return all the responses
}

function compare_emotions($emotion, $user_emotion)
{
    if ($user_emotion == "surprise_happy")
        $user_emotion = "happy";
    else if ($user_emotion == "surprise_sad")
        $user_emotion = "sad";
    else if ($user_emotion == "surprise_fear")
        $user_emotion = "fear";
    else if ($user_emotion == "surprise_angry")
        $user_emotion = "angry";
    else if ($user_emotion == "surprise")
        $user_emotion = 'surprise'; //return $emotion; //go with assesement instead
    else if ($user_emotion == "disgust")
        $user_emotion = 'disgust'; //return $emotion;
    else if ($user_emotion == "blank")
        return $emotion;
    else if ($user_emotion == "")
        return $emotion; //no emoticon, so go with what the bot thinks the user is feeling

    if ($emotion != $user_emotion) //if my assesment is wrong, use emoticon instead
        return $user_emotion;

    return $emotion;
}

//check if any of the responses ERV has the specified emotion
function emotion_exists($responses, $emotion)
{
    if ($emotion == "")
        return false;

    for ($i = 0; $i < sizeof($responses); $i++)
    {
        $value = $responses[$i]['erv'][$emotion];
        if ($value > 0.0)
        {
            return true;
        }
    }

    return false;
}

/*
    @description: calcualte EIVA or ERVA arrays from result set
    @param: $result, is a mysql result of words & intensities
    @return: array, emotional intensity array
    @todo: currently only doing average, need something better?
    @todo: need to decide if word has more than one meaning, from which prespective?
*/

```

```

//function calculate_emotion($result, $perspective)
function calculate_emotion($result)
{
    //global $user_emotion;
    $new_intensity = 0.0;
    $emotion_array = Array("happy" => 0.0, "sad" => 0.0, "angry" => 0.0, "fear" => 0.0, "surprise" =>
0.0, "disgust" => 0.0);
    $count_array = Array("happy" => 0, "sad" => 0, "angry" => 0, "fear" => 0, "surprise" => 0.0,
"disgust" => 0.0);
    $word_array = Array();
    $word_count = Array();

    while ($row = mysql_fetch_array($result))
    {
        $w = new Word;
        $w->word = $row["word"];
        $w->emotion = $row["emotion"];
        $w->intensity = $row["intensity"];
        $w->occ = $row["category"];

        array_push($word_array, $w);
        array_push($word_count, $row["word"]);
    }

    console_output("Calculate_emotion > list of words", $word_array);

    $count = array_count_values($word_count);

    foreach ($word_array as $w)
    {
        if ($count[$w->word] > 1) //duplicate exists
        {
            //check if emotions differ
            $query = "select distinct emotion ";
            $query .= "FROM ' .SIMILAR_WORDS_TABLE. ' s, ' .OCC_TABLE.' o, ' .EKMAN_TABLE. '
e where ";
            $query .= "s.occ_id = o.id and o.ekman_id = e.id and ";
            $query .= "s.word = " . $w->word. """;

            console_output("calculate_emotion > find duplicates query for an emotion:", $query);

            $distinct_result = mysql_query($query);
            $last_emotion = get_last_emotion();
            $the_emotion = "";
            $the_category = "";

            if (mysql_num_rows($distinct_result) > 1) //more than 1 emotion group
            {
                //decide emotion based on last_emotion or personality
                while ($row = mysql_fetch_array($distinct_result))
                {
                    if ($row["emotion"] == $last_emotion)
                    {
                        $the_emotion = $last_emotion;
                        console_output("calculate_emotion > last_emotion used");
                        break;
                    }
                }
            }
        }
    }
}

```

```

    }
}

print_msg("here1", $the_emotion);
console_output("calculate_emotion > the_emotion:", $the_emotion);
//nothing found, use personality to decide emotion
mysql_data_seek($distinct_result, 0); //reset pointer
//$personality = get_personality();
$personality = dominant_personality();

//if (($personality["extraversion"] > $personality["neurotic"] ||
//$personality["agreeableness"] > $personality["neurotic"]) &&
if ($personality == "extraversion" &&
    $the_emotion == "")
{
    //find if happy in the list
    $query = "select distinct emotion ";
    $query .= 'FROM ' .SIMILAR_WORDS_TABLE. ' s, ' .OCC_TABLE. ' o, '
.EKMAN_TABLE. ' e where ';
    $query .= 's.occ_id = o.id and o.ekman_id = e.id and ';
    $query .= "e.emotion = 'happy' and ";
    $query .= "s.word = '' .Sw->word. ''";
    $result1 = mysql_query($query);

    if (mysql_num_rows($result1) != 0) //happy found
        $the_emotion = "happy";

    print_msg("query:", $query);
    print_msg("here2", $the_emotion);
    console_output("calculate_emotion > extravert query:", $query);
    console_output("calculate_emotion > extravert emotion:");
}

//if ($personality["neurotic"] > $personality["extraversion"] &&
//$personality["neurotic"] > $personality["agreeableness"] &&
if ($personality == "neurotic" &&
    $the_emotion == "")
{
    $query = "select distinct emotion ";
    $query .= 'FROM ' .SIMILAR_WORDS_TABLE. ' s, ' .OCC_TABLE. ' o, '
.EKMAN_TABLE. ' e where ';
    $query .= 's.occ_id = o.id and o.ekman_id = e.id and ';
    $query .= "(e.emotion = 'sad' or e.emotion = 'angry' or e.emotion = 'fear') and ";
    $query .= "s.word = '' .Sw->word. ''";
    $result1 = mysql_query($query);

    if (mysql_num_rows($result1) != 0) //found
    {
        //todo: now randomly decide???
        $n_rows = mysql_num_rows($result1);
        mysql_data_seek($result1, random_num($n_rows - 1));

        $row = mysql_fetch_array($result1);
        $the_emotion = $row["emotion"];

        print_msg("query:", $query);
    }
}

```

```

        print_msg("here2", $the_emotion);
        console_output("calculate_emotion > neurotic query:", $query);
        console_output("calculate_emotion > neurotic emotion:");
    }
}

if ($the_emotion == "") //randomly choose something
{
    //todo: perhaps choose the emotion that corresponds to highest value in CES
    //otherwise then do random?
    console_output("calculate_emotion > random emotion");
    $n_rows = mysql_num_rows($distinct_result);
    mysql_data_seek($distinct_result, random_num($n_rows - 1));

    $row = mysql_fetch_array($distinct_result);
    $the_emotion = $row["emotion"];

    print_msg("here3", $the_emotion);
}
}
else
{
    $row = mysql_fetch_array($distinct_result);
    $the_emotion = $row["emotion"];
    console_output("calculate_emotion > no other emotion groups");
}

$query = "select distinct category ";
$query .= 'FROM ' .SIMILAR_WORDS_TABLE. ' s, ' .OCC_TABLE.' o, ' .EKMAN_TABLE. '
e where ';
$query .= 's.occ_id = o.id and o.ekman_id = e.id and ';
$query .= "e.emotion = " . $the_emotion. " and ";
$query .= "s.word = " . $w->word. " and ";
console_output("calculate_emotions > find category query:", $query);
$result = mysql_query($query);

if (mysql_num_rows($result) > 1)
{
    $n_rows = mysql_num_rows($result);
    mysql_data_seek($result, random_num($n_rows - 1));

    $row = mysql_fetch_array($result);
    $the_category = $row["category"];
    console_output("calculate_emotions > random category chosen");
}
else
{
    $row = mysql_fetch_array($result);
    $the_category = $row["category"];
}

//by now we should have both an emotion & category
$query = "select word, category, emotion, intensity ";
$query .= 'FROM ' .SIMILAR_WORDS_TABLE. ' s, ' .OCC_TABLE.' o, ' .EKMAN_TABLE. '
e where ';
$query .= 's.occ_id = o.id and o.ekman_id = e.id and ';

```

```

$query .= "e.emotion = " . $the_emotion. " and ";
$query .= "o.category = " . $the_category. " and ";
$query .= "s.word = " . $w->word. """;

console_output("Duplicate words found, final choice query: ", $query);

$row = mysql_fetch_array(mysql_query($query)); //should be just 1
$emotion_array[$row["emotion"]] += $row["intensity"];
$count_array[$row["emotion"]] += 1;

$count[$w->word] = 0; //remove duplicate
}
else if ($count[$w->word] == 1)
{
    $emotion_array[$w->emotion] += $w->intensity;
    $count_array[$w->emotion] += 1;
}
}

while ($val = current($emotion_array))
{
    $key = key($emotion_array);
    $emotion_array[$key] = current($emotion_array) / $count_array[$key];
    next($emotion_array);
}

return $emotion_array;
}

function base_emotion($emoticon)
{
    if ($emoticon == "surprise_happy")
        return "happy";
    else if ($emoticon == "surprise_sad")
        return "sad";
    else if ($emoticon == "surprise_fear")
        return "fear";
    else if ($emoticon == "surprise_angry")
        return "angry";
}

//todo: remove punctuation : , ; ? . ! => done
//todo: set to lowercase => done
function emotion_query($sentence)
{
    //print_r($inputmatched);
    $sentence = strtolower($sentence);

    //$sentence_array = split(" ", $sentence);
    // Remove punctuation.

    $sentence_array = remove_punctuation($sentence);
    print_msg("emotion_query: sentence array", $sentence_array);

    //$query = 'SELECT s.word, o.category, o.intensity, e.emotion, ff.factor, ff.score ';
    $query = 'SELECT s.word, o.intensity, e.emotion, o.category ';

```

```

    // $query .= 'FROM ' .SIMILAR_WORDS_TABLE.' 's, ' .OCC_TABLE.' o, ' .EKMAN_TABLE.' e, '
    .FACTOR_FIVE_TABLE.' ff where ';
    $query .= 'FROM ' .SIMILAR_WORDS_TABLE.' 's, ' .OCC_TABLE.' o, ' .EKMAN_TABLE.' e
    where ';
    $query .= 's.occ_id = o.id and o.ekman_id = e.id and (';
    // $query .= 's.occ_id = o.id and o.ekman_id = e.id and e.ff_id = ff.factor and (';

    for ($j = 0; $j < sizeof($sentence_array); $j++)
    {
        //if ($sentence_array[$j] != ':' && $sentence_array[$j] != 'undefined')
        {
            //if ($sentence_array[$j + 1] == ':')
            //perhaps use LIKE instead
            if ($j == sizeof($sentence_array) - 1)
                // $query .= "s.word LIKE \"%" .mysql_escape_string($sentence_array[$j]). "%\"";
                $query .= "s.word = \"" .mysql_escape_string($sentence_array[$j]). "\"";
            else
                // $query .= "s.word LIKE \"%" .mysql_escape_string($sentence_array[$j]). "%\" || ";
                $query .= "s.word = \"" .mysql_escape_string($sentence_array[$j]). "\" || ";
        }
        //else
        //break;
    }

    print_msg("query:", $query);
    console_output("Emotion Query for sentence: ", $query);
    return $result = mysql_query($query);
}

function remove_punctuation($sentence)
{
    // $sentence_array = preg_split('/\s*[\s+.\|?|!|,|()|-+|\\"|=;|&#0215;|\$|\/|:|{|}]s*/i', $sentence);
    return preg_split('/\s*[\s+.\|?|!|,|()|-+|=;|&#0215;|\$|\/|:|{|}]s*/i', $sentence); //leave apostrophe
    (don't)
}

function validate_data($regex, $data)
{
    $match = array();
    $val = preg_match_all($regex, $data, $match);

    if ($val != 0) //found
    {
        $tmp = $match[0];
        if (strlen($tmp[0]) != strlen($data)) //check if length the same
            return false;
        else
            return true; //was found & len same, we good
    }

    return false;
}

function xmlrpc_call($command, $param)
{
    return XMLRPC_request(SITE, LOCATION, $command, array(XMLRPC_prepare($param)));
}

```



```

}

function get_max_concept($response)
{
    $max = 0.0;
    $final_concept = Array();

    foreach($response as $concept)
    {
        $val = $concept[2];

        //todo:
        //what about concept with same vals?
        //if several maxes, then does it mean they are all possible given the input
        //all possible concepts?
        //therefore, should we use them all?
        if (floatval($val) >= $max)
        {
            $final_concept = $concept;
            $max = $val;
            //echo "<br>final_concept:<br>";
            //print_r($final_concept);
            //echo $val;
        }
    }

    return $final_concept;
}

function guess_concept($sentence)
{
    list($success, $response) = xmlrpc_call('guess_concept', $sentence); //XMLRPC_request(SITE,
LOCATION, 'guess_concept', array(XMLRPC_prepare($sentence)));

    if ($success)
    {
        //print_r($response);
        //echo "max:<br>";
        //print_r(get_max_concept($response));
        return get_max_concept($response);
    }
    else
    {
        print_msg("guessconcept", ($response));
        exit;
    }
}

/*
    @description: get all topics greater than X percent
    @param: $response, all topics from conceptnet
    @return: array, of final topics
*/
function get_max_topic($response)
{
    $keys = $response[0];

```

```

$topics = $response[1];
$final_topics = Array();
$max = 0.0;
console_output("get_max_topic > topics", $topics);

//take broken phrases of the sentence and find their percentage
foreach($keys as $key)
{
    //echo $key;
    foreach($topics as $topic)
    {
        //print_r($topic);
        if ($key == $topic[0])
        {
            /*$val = $topic[1];
            if ($val >= $max)
            {
                $max = $val;
                $final_topic = $topic;
            }
            break;*/
            array_push($final_topics, $topic);
        }
    }
}
console_output("get_max_topic > final_topics", $final_topics);

//sort all the topics, then add only those with a X hit rate
//check for duplicates from $keys
//$sorted_array = rquicksort($topics);
//console_output("get_max_topic > sorted array", $sorted_array);

foreach($topics as $topic)
{
    if (!in_array($topic[0], $keys)) //if success, then not a duplicate
    {
        if ($topic[1] >= TOPIC_HIT_PERCENT)
        {
            array_push($final_topics, $topic);
        }
        else
            break;
    }
}
console_output("get_max_topic > final_topics", $final_topics);
print_msg("maxtopics", "");
print_msg("topicsarray", $final_topics);
return $final_topics;
}

/*
    @description: make call to get topics from conceptnet
*/
function guess_topic($sentence)
{

```

```
list($success, $response) = xmlrpc_call('guess_topic', $sentence); //XMLRPC_request(SITE,
LOCATION, 'guess_topic', array(XMLRPC_prepare($sentence)));
```

```
if ($success)
{
    //echo "REPONSES:<br>";
    //print_r($response);
    //rquicksort($response[1]);
    //return ($response);
    return get_max_topic($response);
}
else
{
    print_msg("guesstopic", ($response));
    console_output("guess_topic > msg", $success);
}
}
```

```
//$concept is a mysql row from our db
function concept_exists($concept, $concept_array)
```

```
{
    print_msg("conceptexists", ($concept));
    //echo "<br>";
    print_msg("conceptarray", ($concept_array));
    $concept_id = $concept['id'];

    if (sizeof($concept_array) == 0)
        return false;

    //foreach ($concept_array as $c)
    for ($i = 0; $i < sizeof($concept_array); $i++)
    {
        $c = $concept_array[$i];
        if ($concept_id == $c['id'])
            return true;
    }

    return false;
}
```

```
function summarize($sentence)
{
    list($success, $response) = xmlrpc_call('summarize_document', $sentence);
    //XMLRPC_request(SITE, LOCATION, 'guess_topic', array(XMLRPC_prepare($sentence)));
```

```
if ($success)
{
    //print_r($response);
    //echo "max:<br>";
    //print_r(get_max_topic($response));
    return $response;
}
else
{
    print_msg("summarize", ($response));
}
}
```

```

}

/*
  @description: calucalte answer
  @todo: better method than average?
  @todo: add logging of input & replies
*/
function movie_answer()
{
  $query = "select meter, num_concepts from ". MOVIE_DATA_TABLE;
  $row = mysql_fetch_array(mysql_query($query));
  $movie_meter = $row['meter'];
  $num_concepts = $row['num_concepts'];
  console_output("movie_answer > movie_meter", $movie_meter);
  console_output("movie_answer > num_concepts", $num_concepts);

  $query = "update " .MOVIE_DATA_TABLE. " set meter = 0.0";
  console_output("movie_anser > update meter:", $query);
  mysql_query($query);

  $query = "update " .MOVIE_DATA_TABLE. " set num_concepts = 0";
  console_output("movie_anser > update num concepts:", $query);
  mysql_query($query);

  //echo "<br>MovieMeter<br>";
  //echo $movie_meter;

  $response = new Response();
  $response_sentence = "";
  $response->timer = 0.0;
  //$response->response = "Score: $movie_meter \n";

  if ($num_concepts != 0)
    $movie_meter = $movie_meter / $num_concepts;

  print_msg("moviemeter", $movie_meter);
  print_msg("numconcpets", $num_concepts);
  console_output("movie_answer > meter", $movie_meter);
  //$movie_meter = 0.0;

  if ($movie_meter == 0.0)
    $response_sentence = "From what I've heard, I don't really care either way";
  else if ($movie_meter > 0.0 && $movie_meter <= 0.5)
    $response_sentence = "This movie sounds pretty good.";
  else if ($movie_meter > 0.5 && $movie_meter <= 1.0)
    $response_sentence = "This movie is a must watch. ";
  else if ($movie_meter < 0.0 && $movie_meter >= -0.5)
    $response_sentence = "The movie doesn't sound too good.";
  else if ($movie_meter <= -0.5 && $movie_meter >= -1.0)
    $response_sentence = "The movie is horrible.";
  else
    $response_sentence = "I've no answer";

  //echo "response: $response->response";
  $response->emoticon = get_review($movie_meter);

```

```

$last_emotion = get_last_emotion();

if ($last_emotion == "happy" && $movie_meter > 0.0)
    $response_sentence .= " I definitely feeling like watching this.";
else if ($last_emotion == "happy" && $movie_meter <= 0.0)
    $response_sentence .= " But if you want we can still check it out.";
else if ($last_emotion != "surprise" && $movie_meter > 0.0) //sad, angry, fear, disgust
    $response_sentence .= " But I'm not in the mood to go watch a movie.";
else if ($last_emotion != "surprise" && $movie_meter <= 0.0)
    $response_sentence .= " I'm in no mood to watch something like that.";

$response->response = $response_sentence;
return $response;
}

function get_review($movie_meter)
{
    $image = "<img align=middle src=images/";
    if ($movie_meter == 0.0)
        $image .= "middlethumb";
    else if ($movie_meter > 0.0 && $movie_meter <= 0.5)
        $image .= "thumbup";
    else if ($movie_meter > 0.5 && $movie_meter <= 1.0)
        $image .= "2thumbsup";
    else if ($movie_meter < 0.0 && $movie_meter >= -0.5)
        $image .= "thumbdown";
    else if ($movie_meter <= -0.5 && $movie_meter >= -1.0)
        $image .= "2thumbsdown";
    else
        return "";

    $image .= "-smallest-noface.gif>";
    return $image;
}

/*
    @description: find movie concepts & control movie meter
    @param: $description, sentence from user
    @todo: filter out common words => no need?
    @todo: remove puncutation = > no need
    @todo: lower case => done
    @todo: add logging of input & replies
*/
function movie_concepts($description)
{
    $description = str_replace("''", "", strtolower($description));

    $query = "select meter, num_concepts from ". MOVIE_DATA_TABLE;
    $row = mysql_fetch_array(mysql_query($query));
    $movie_meter = $row['meter'];
    $num_concepts = $row['num_concepts'];
    console_output("movie_concepts > movie_meter", $movie_meter);
    console_output("movie_concepts > num_concepts", $num_concepts);

    $concept_array = Array();

```

```

$topic_results = guess_topic($description);
console_output("movie_concepts > sentence", $description);
console_output("ConceptNet Topics:", $topic_results);

foreach ($topic_results as $topic)
{
    $concept = mysql_escape_string($topic[0]);

    //use distinct to remove the problem of the same concept being seen twice
    $query = "select distinct(b.id), concept, score FROM " .CONCEPTS_TABLE. " a inner join "
.PREFERENCES_TABLE. " b on a.pref_id = b.id
    where concept = '$concept'";

    print_msg("query", $query);
    console_output("movie_concpets > list concepts query", $query);
    $results = mysql_query($query);

    if (mysql_num_rows($results) > 0)
    {
        while ($row = mysql_fetch_array($results))
        {
            /*echo "<br>row:".print_r($row)."<br>";
            $movie_meter += $row['score'];*/
            /*if (sizeof($concept_array) == 0)
            {
                array_push($concept_array, $row);
            }*/
            if (!concept_exists($row, $concept_array)) //does not exist or array empty
            {
                array_push($concept_array, $row);
            }
        }
    }
}

print_msg("movie concepts", $concept_array);
console_output("Concepts HIT:", $concept_array);

foreach($concept_array as $concept)
{
    $movie_meter += $concept['score'];
}

$query = "update " .MOVIE_DATA_TABLE. " set meter = " . $movie_meter;
console_output("movie_concpets > update meter", $query);
mysql_query($query);

$query = "update " .MOVIE_DATA_TABLE. " set num_concepts = " .($num_concepts +
sizeof($concept_array));
console_output("movie_concepts > num concpets", $query);
mysql_query($query);

//console_output("Movie Meter: " . $movie_meter. " Num Concepts Hit: " . $num_concepts +
sizeof($concept_array));

if ($num_concepts != 0)

```

```

    $movie_meter = $movie_meter / $num_concepts;

    console_output("movie_concepts > meter", $movie_meter);
    $response = new Response();
    //if ($movie_meter == 0.0)
        //$response->response = "Go on...";
    $decent_movie = Array("This movie sounds really interesting...!", 'Not bad', 'What else', 'And..!', 'sounds
pretty good',
        'What else happens?', 'this movie is getting pretty good', "");
    $great_movie = Array("This movie is sounding pretty good, tell me more!", 'seems really exciting');
    $bad_movie = Array("This movie is going down hill...!", 'doesn\'t sound that great', 'uh huh', 'is it going
to get better?',
        'this movie is getting worse', "");
    $horrible_movie = Array("This movie sounds horrible", 'ok, this is just bad');

    $join = array_merge($decent_movie, $great_movie);
    $join2 = array_merge($bad_movie, $horrible_movie);

    if ($movie_meter > 0.0 && $movie_meter <= 0.5)
        $response->response = $decent_movie[random_num(sizeof($decent_movie))];/"This movie is
sounding pretty good, tell me more!";
    else if ($movie_meter > 0.5 && $movie_meter <= 1.0)
        $response->response = $join[random_num(sizeof($join))];/"This movie sounds really
interesting...";
    else if ($movie_meter < 0.0 && $movie_meter >= -0.5)
        $response->response = $bad_movie[random_num(sizeof($bad_movie))];/"This movie is going
down hill...";
    else if ($movie_meter <= -0.5 && $movie_meter >= -1.0)
        $response->response = $join2[random_num(sizeof($join2))];/"The movie sounds horrible!";
    else
        $response->response = "Go on...";

    //$response->emoticon = get_review($movie_meter);
    return $response;
}

//check if conversational trend seems to be going down
function change_subject()
{
    $query = "select topic_distance, topic_current_idx from " .BOT_DATA_TABLE;
    $row = mysql_fetch_array(mysql_query($query));
    $topic_distance = $row["topic_distance"];
    $topic_current_idx = $row["topic_current_idx"];
    print_msg("change_subject > topic_dist", $topic_distance);
    print_msg("change_subject > topic_current_idx", $topic_current_idx);

    //check conversation log for emotional trend
    $query = "select last_emotion from " .CES_TABLE. " where id >= " .$.topic_current_idx." order by id
desc";
    $result = mysql_query($query);
    $num_rows = mysql_num_rows($result);

    $gap = $num_rows - $topic_current_idx;
    print_msg("change_subject > numrows", $num_rows);
    print_msg("change_subject > gap", $gap);
    //if (mysql_num_rows($result) >= N_TREND)

```

```

if ($gap >= $topic_distance)
{
    $count_array = Array("happy" => 0, "sad" => 0, "angry" => 0, "fear" => 0, "surprise" => 0,
"disgust" => 0);
    $personality = dominant_personality(); //get_personality();

    while ($row = mysql_fetch_array($result))
    {
        if ($row["last_emotion"] != "")
            $count_array[$row["last_emotion"]];
    }

    print_msg("changesubject:count_array:", $count_array);

    //if ($personality["extraversion"] >= $personality["neurotic"])
    mt_srand ((float) microtime() * 1000000);
    $rand = 0;

    if ($personality == "extraversion")
        $rand = mt_rand(MIN_TOPIC_DISTANCE, MAX_TOPIC_DISTANCE_EXTRAVERT);
    else
        $rand = mt_rand(MIN_TOPIC_DISTANCE, MAX_TOPIC_DISTANCE);

    if (($count_array["happy"] / $topic_distance) < HAPPY_TREND_MIN)
    {
        print_msg("trend:", $count_array["happy"] / $topic_distance);
        console_output("Proceed to change subject.<br>Trend array:", $count_array);
        console_output("Trend: ".$count_array["happy"] / $topic_distance);

        //change distance
        $query = "update ".BOT_DATA_TABLE." set topic_distance = ".$rand;
        print_msg("change_subject > query", $query);
        mysql_query($query);

        //update idx
        $query = "update ".BOT_DATA_TABLE." set topic_current_idx = ".$num_rows;
        print_msg("change_subject > query", $query);
        mysql_query($query);

        return true;
    }
}

return false;
}

function new_topic($userinput)
{
    $query = "select topic from ".TOPICS_TABLE;
    $result = mysql_query($query);

    $response = new Response();
    $response->emoticon = "";
    $response->timer = 0.0;

    $str = "Let's talk about something else. ";

```



```

print_msg("str", $str);

$rand_num = random_num(mysql_num_rows($result));
mysql_data_seek($result, $rand_num);
$row = mysql_fetch_array($result);
print_msg("row", $row["topic"]);
console_output("New Topic is: ".$row["topic"]);

$yahoo_articles = yahoo_request($row["topic"]);
$chosen_article = Array();
/*foreach ($yahoo_articles as $article)
{
    if (is_array($article))
        $chosen_article = $article;
}*/
if (sizeof($yahoo_articles) == 0)
    new_topic($userinput);

$str .= $yahoo_articles["Summary"]; //$chosen_article["Summary"]; //$row["topic"];
//$str .= "<br>";
$response->response = $str;
console_output('new_topic > str', $str);
//log user input > intent
//update CES user
//update CES bot
//log reply sentence > reply

update_user_ces($userinput);

$reply = Array('sentence' => $str, 'erv' => calculate_emotion(emotion_query($str)));
console_output("new_topic > reply:", $reply);
update_bot_ces($reply);

/*//since each row in CES table is for each sentence uttered
//should add another, even though nothing changed
$ces = get_ces();
$last_emotion = ";//get_last_emotion();

$query = "insert into ".CES_TABLE. " (happy, sad, angry, fear, surprise, disgust, last_emotion)
values ( "
        .$ces["happy"].", ".$ces["sad"].", ".$ces["angry"].", ".$ces["fear"].", ".$ces["surprise"].", "
        .$ces["disgust"].", ".$last_emotion."");
mysql_query($query);*/

//print_msg("newtopic > ces query", $query);
//print_msg("new topic", $str);
return $response;
}

/*
find related yahoo articles to user input
*/
function find_topic($input)
{
    //combine userinput & chosen topic?

```

```

//break input into topics?
//just pass input sentence alone to yahoo

/*$topic_array = Array();
$sorted = rquicksort(guess_topic($input));
$max_len = 0;
$max_idx = 0;

for ($i = 0; $i < sizeof($sorted); $i++)
{
    if (strlen($sorted[$i][0]) > $max_len)
    {
        $max_idx = $i;//array_push($topic_array, $sorted[$i]);
        $max_len = strlen($sorted[$i][0]);
    }
}

console_output("find_topic > sorted topic array", $sorted);
console_output("find_topic > max_idx", $max_idx);*/

if ($input == 'INACTIVITY')
{
    console_output("find_topic > empty input");
    $r = new Response();
    $r = new_topic($input);
    return $r->response;
}

$yahoo_articles = yahoo_request($input);//yahoo_request($sorted[$max_idx][0]);
$summary = ";

//if ($yahoo_articles['totalResultsReturned'] == 0)
if (sizeof($yahoo_articles) == 0)
{
    console_output("find_topic > no yahoo resultsF");
    $r = new Response();
    $r = new_topic($input);
    return $r->response;
}

/*foreach ($yahoo_articles as $article)
{
    if (is_array($article))
        $summary = $article["Summary"];
}*/
$summary = $yahoo_articles["Summary"];
console_output("find_topic > summary", $summary);

//update user_ces
//update bot_ces

update_user_ces($input);
$reply = Array('sentence' => $summary,
    'erv' => calculate_emotion(emotion_query($summary)));
console_output("find_topic > reply:", $reply);
update_bot_ces($reply);

```

```

    return $summary;
}

function random_num($max)
{
    // Pick a random number from 0 to sizeof($liarray)-1
    mt_srand ((float) microtime() * 1000000);
    return mt_rand(0,($max-1));
}

function console_output($msg, $data = "")
{
    global $console_output;

    $console_output .= $msg . "<br>";

    if (is_array($data))
    {
        if (sizeof($data) > 0)
            $console_output .= "<pre>".print_r($data, true) . "</pre>";
    }
    else
        $console_output .= $data . "<br><br>";
}

function show_output($response)
{
    global $console_output;

    $response->console = $console_output;

    //print "CONSOLE";
    //print $response->console;

    return $response;
}

/*
    @description: insert typo(s) based on CES. if an emotion is at a high level
    typo(s). higher the emotion more typos...
    higher the meotion higher possiblity of typos
*/
function create_typo($sentence)
{
    $ces = get_ces();
    $typo = false;

    foreach ($ces as $emotion)
    {
        if ($emotion > TYPO_MIN)
            $typo = true;
    }

    if ($typo == false)
        return $sentence; //return normal sentence
}

```

```

$new_sentence = "";
$sentence_array = remove_punctuation(strtolower($sentence));
$query = "select correct, misspelled from " .TYPO_TABLE. " where ";

for ($j = 0; $j < sizeof($sentence_array); $j++)
{
    if ($j == sizeof($sentence_array) - 1)
        $query .= "correct = \"\" .mysql_escape_string($sentence_array[$j]). \"\"";
    else
        $query .= "correct = \"\" .mysql_escape_string($sentence_array[$j]). \"\" || ";
}

return $new_sentence;
}

/*
 @description: either fix 1 or more typos, or rewrite the sentence...
*/
function fix_typo($sentence)
{
    $response = new Response();
    $sentence_array = remove_punctuation(strtolower($sentence));

    $query = "select correct, misspelled from " .TYPO_TABLE. " where ";

    for ($j = 0; $j < sizeof($sentence_array); $j++)
    {
        if ($j == sizeof($sentence_array) - 1)
            $query .= "misspelled = \"\" .mysql_escape_string($sentence_array[$j]). \"\"";
        else
            $query .= "misspelled = \"\" .mysql_escape_string($sentence_array[$j]). \"\" || ";
    }

    print_msg("fixtypo:query ", $query);
    $row = mysql_fetch_array(mysql_query($query));

    console_output("Typo found: ", $row["correct"]." => ".$row["misspelled"]);
    $response->response = $row["correct"]." => ".$row["misspelled"];
    return $response;
}

//modified yahoo example code
function yahoo_request($topic)
{
    console_output("yahoo_request > incoming topic", $topic);
    $appid = 'CMlyfwPV34HLQuQWzuUn4iq10mwbcibgVtFRlEXYDKpqrMlFv1MsfGWOHk1';
    $sort = 'rank';
    $query = '&query='.urlencode($topic);

    $request = 'http://search.yahooapis.com/NewsSearchService/V1/newsSearch?appid='.$appid.
        '&language=en&results=50&sort='.$sort.$query;

    //echo $request;
    console_output('yahoo_request > request', $request);
}

```

```

$response = file_get_contents($request);

// Retrieve HTTP status code
list($version,$status_code,$msg) = explode(' ', $http_response_header[0], 3);

// Check the HTTP Status code
switch($status_code)
{
    case 200:
        // Success
        break;
    case 503:
        die('Your call to Yahoo Web Services failed and returned an HTTP status of 503. That means:
Service unavailable. An internal problem prevented us from returning data to you.');
```

break;

```

    case 403:
        die('Your call to Yahoo Web Services failed and returned an HTTP status of 403. That means:
Forbidden. You do not have permission to access this resource, or are over your rate limit.');
```

break;

```

    case 400:
        // You may want to fall through here and read the specific XML error
        die('Your call to Yahoo Web Services failed and returned an HTTP status of 400. That means:
Bad request. The parameters passed to the service did not match as expected. The exact error is returned in
the XML response.');
```

break;

```

    default:
        die('Your call to Yahoo Web Services returned an unexpected HTTP status of:' . $status_code);
}

// Output the XML
//echo htmlspecialchars($response, ENT_QUOTES);

// Parse the XML into a DOM tree
$error = null;
if (!$dom = domxml_open_mem($response, DOMXML_LOAD_PARSING, $error)) {
    echo "XML parse error\n";
    foreach ($error as $errorline) {
        // For production use this should be logged to a file instead
        echo $errorline['errormessage']."<br />\n";
        echo " Node : " . $errorline['nodename'] . "<br />\n";
        echo " Line : " . $errorline['line'] . "<br />\n";
        echo " Column : " . $errorline['col'] . "<br />\n";
    }
    die('XML parse error');
}

$result = xml_to_result($dom);
console_output('yahoo_request > result', $result);
return yahoo_result_filter($result);//return $result;
}

//remove all articles that start with
//dates
//writes, by, said, says, published
//duplicate results
function yahoo_result_filter($base_articles)
```

```

{
  $filter_words = Array("writes", "by", "said", "says", "published");
  $result = Array();

  foreach($base_articles as $art)
  {
    if (is_array($art))
    {
      if (substr_count($art["Summary"], ":") == 0 &&
          substr_count($art["Summary"], '/') == 0)
      {
        console_output("yahoo_result_filter > passed 1st test");
        $pass = false;
        foreach($filter_words as $w)
        {
          if (substr_count(strtolower($art["Summary"]), $w) == 0)
            $pass = true;
          else
            $pass = false;
        }

        if ($pass)
        {
          console_output("yahoo_result_filter > passed 2nd test");

          $pass = true;
          //look at the last 10 replies to make sure there is no duplicate
          $query = "select sentence from ".ERVA_TABLE. " order by id desc limit "
.MIN_TOPIC_DISTANCE;
          console_output("yahoo_result_filter > query", $query);
          $result = mysql_query($query);

          if (mysql_num_rows($result) <= 0)
            return $art;

          while ($row = mysql_fetch_array($result))
          {
            //echo "result:".strcasecmp($art["Summary"], $row["sentence"]);
            //echo "art:". $art["Summary"];
            //echo "row:". $row["sentence"];
            if (strcasecmp($art["Summary"], $row["sentence"]) == 0)
            {
              console_output("yahoo_result_filter > passed 3rd test");
              $pass = false;
              break;
            }
          }

          if ($pass)
          {
            return $art;
          }
        }
      }
    }
  }
}

```

```

    return Array(); //nothing in results, so return nothing
}

// Now traverse the DOM with this function.
// This traversal function is suitable for the Yahoo! Maps and
// Search web services, and any other Yahoo! Web Services that
// return the number of results as attributes in the root tag and
// have no other attributes in the results. For other Yahoo! Web
// Services, such as Flickr web services, this routine will
// have to be altered.
// yahoo example code
function xml_to_result($dom)
{
    error_reporting(E_ALL ^ E_NOTICE);
    //echo $dom;
    $root = $dom->document_element();
    $res['totalResultsAvailable'] = $root->get_attribute('totalResultsAvailable');
    $res['totalResultsReturned'] = $root->get_attribute('totalResultsReturned');
    $res['firstResultPosition'] = $root->get_attribute('firstResultPosition');

    $node = $root->first_child();
    $i = 0;
    $k = null;
    while($node) {
        //print_r($node);
        //if ($node->hasAttributes('tagname'))
        {
            switch($node->tagname) {
                case "":
                    break;
                case 'Result':
                    $subnode = $node->first_child();
                    while($subnode) {
                        $subnodes = $subnode->child_nodes();
                        if(!empty($subnodes)) {
                            foreach($subnodes as $k=>$n) {
                                if(empty($n->tagname)) {
                                    $res[$i][$subnode->tagname] =
trim($n->get_content());
                                } else {
                                    $res[$i][$subnode->tagname][$n-
>tagname]=trim($n->get_content());
                                }
                            }
                        }
                        $subnode = $subnode->next_sibling();
                    }
                    break;
                default:
                    $res[$node->tagname] = trim($node->get_content());
                    $i--;
                    break;
            }
            $i++;
            $node = $node->next_sibling();
        }
    }
}

```

```

    }
  }
  error_reporting(E_ALL);
  return $res;
}
?>

```

Ajax.js

```

var DEBUG = false;
//var DEBUG = true;
var input = "";
var user_hrs;
var user_mins;
var user_secs;
var emoticon = "blank"; //default is the none radio button, value is blank
var display_emoticon = "";

```

```

function getxmlHttpObj()
{
    var xmlHttp = false;
    //Create a new XMLHttpRequest object to talk to the Web server
    try
    {
        xmlHttp = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (e)
    {
        try
        {
            xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
        } catch (e2)
        {
            xmlHttp = false;
            //alert('broken XMLHTTP');
        }
    }

    if (!xmlHttp && typeof XMLHttpRequest != 'undefined')
    {
        xmlHttp = new XMLHttpRequest();
    }
    return xmlHttp;
}

```

```

function updateDisplay(xmlHttpObj)
{
    if (xmlHttpObj.readyState == 4)
    {
        if (xmlHttpObj.status == 200)
        {
            var response = xmlHttpObj.responseText;
            var divresponse = top.display.document.getElementById('window');
            var divconsole = top.debug.document.getElementById('console');
            var divtype = top.type.document.getElementById('type_status');
            var array = response.split('##');
        }
    }
}

```



```

        var d = new Date();
        var hrs = d.getHours();
        var mins = d.getMinutes();
        var secs = d.getSeconds();

        var user_timestamp = printTime(user_hrs, user_mins, user_secs);
        var timestamp = printTime(hrs, mins, secs);
        var emo_icon = "";

        if (array[2] != 'None')
            emo_icon = array[2];

        divresponse.innerHTML += "<br><font size='+1' color='blue'>[" + user_timestamp + "
User:</font> " + get_emoticon_image() + " " + input;
        divresponse.innerHTML += "<br><font size='+1' color='red'>[" + timestamp + "
Bot:</font> " + emo_icon + " " + array[0];
        divconsole.innerHTML += array[1];

        display_emoticon = "";
        print_msg("reponse:", array[2]);

        //todo: force status page to refresh in IE
        top.status.window.location = "status.php";

        //scroll to bottom
        top.display.scrollTo(0, divresponse.scrollHeight);
        top.debug.scrollTo(0, divconsole.scrollHeight);

        divtype.innerHTML = "";

        /*if (is_typo)
            fixTypo(array[2]);*/
    }
    else
        alert("status is " + xmlHttpObj.status);
}
}

function updateTypo(xmlHttpObj)
{
    if (xmlHttpObj.readyState == 4)
    {
        if (xmlHttpObj.status == 200)
        {
            var response = xmlHttpObj.responseText;
            var divresponse = top.display.document.getElementById('window');
            var divconsole = top.debug.document.getElementById('console');
            var divtype = top.type.document.getElementById('type_status');
            var array = response.split('##');
            var d = new Date();
            var hrs = d.getHours();
            var mins = d.getMinutes();
            var secs = d.getSeconds();

            var timestamp = printTime(hrs, mins, secs);

```

```

        divresponse.innerHTML += "<br><font size='+1' color='red'>[" + timestamp + "
Bot:</font>" + array[0];
        divconsole.innerHTML += array[1];

        print_msg("reponse:", response);

        //todo: force status page to refresh in IE
        top.status.window.location = "status.php";

        //scroll to bottom
        top.display.scrollTo(0, divresponse.scrollHeight);
        top.debug.scrollTo(0, divconsole.scrollHeight);

        divtype.innerHTML = "";
    }
    else
        alert("status is " + xmlHttpObj.status);
}

function fixTypo(sentence)
{
    var xmlHttpObj = getxmlHttpObj();
    var url = "reply.php?";

    url += "typo=" + sentence;
    url += "&botname=" + document.getElementById('botname').value;
    url += "&PHPSESSID=" + document.getElementById('PHPSESSID').value;

    var divtype = top.type.document.getElementById('type_status');
    divtype.innerHTML = "<b><i>Bot is typing...</i></b>";

    xmlHttpObj.open("GET", url, true);
    xmlHttpObj.onreadystatechange = function()
    {
        updateTypo(xmlHttpObj);
    }

    xmlHttpObj.send(null);
}

function sendText()
{
    var xmlHttpObj = getxmlHttpObj();
    var url = "reply.php?";

    input = document.getElementById('input').value;
    url += "input=" + document.getElementById('input').value;
    url += "&botname=" + document.getElementById('botname').value;
    url += "&PHPSESSID=" + document.getElementById('PHPSESSID').value;
    //url += "&emoticon=" + document.getElementById('emoticon').value;
    url += "&emoticon=" + emoticon;

    print_msg("sendtext:url", url);

    document.getElementById('input').value = "";
}

```

```

clearRadios();

var divtype = top.type.document.getElementById('type_status');
divtype.innerHTML = "<b><i>Bot is typing...</i></b>";

var d = new Date();
user_hrs = d.getHours();
user_mins = d.getMinutes();
user_secs = d.getSeconds();

xmlHttpObj.open("GET", url, true);
xmlHttpObj.onreadystatechange = function()
{
    updateDisplay(xmlHttpObj);
}

xmlHttpObj.send(null);
}

function setEmoticon(value)
{
    emoticon = value;
    display_emoticon = value;
    print_msg("emoticon", emoticon);
}

function get_emoticon_image()
{
    print_msg("emoticon", emoticon);
    var image = "";
    if (display_emoticon == "happy")
        image = "happy";
    else if (display_emoticon == "sad")
        image = "sad";
    else if (display_emoticon == "angry")
        image = "angry";
    else if (display_emoticon == "fear")
        image = "fear";
    else if (display_emoticon == "surprise")
        image = "surprise";
    else if (display_emoticon == "surprise_happy")
        image = "surprise_happy";
    else if (display_emoticon == "surprise_sad")
        image = "surprise_sad";
    else if (display_emoticon == "surprise_angry")
        image = "surprise_angry";
    else if (display_emoticon == "surprise_fear")
        image = "surprise_fear";
    else if (display_emoticon == "disgust")
        image = "disgust";
    else
        return "";

    return "<img src='images/' + image + "-smallest.gif">";
}

```

```

function clearText()
{
    var div = top.display.document.getElementById('window');
    div.innerHTML = "";
    document.getElementById('input').value = "";
    clearRadios();
}

function clearTypeText()
{
    document.getElementById('input').value = "";
}

function clearRadios()
{
    emoticon = "";

    for (var i = 1; i < 12; i++)
    {
        document.getElementById(i).checked = false;
        print_msg("clearradio:val", document.getElementById(i).value);
    }

    document.getElementById('11').checked = true;
}

function printTime(hrs, mins, secs)
{
    var timestamp = hrs + ":";

    if (mins < 10)
        timestamp += "0" + mins + ":";
    else
        timestamp += mins + ":";

    if (secs < 10)
        timestamp += "0" + secs + ":";
    else
        timestamp += secs + ":";

    return timestamp
}


















function print_msg(msg, data)
{
    if (DEBUG)
    {
        alert(msg+":"+data);
    }
}

```

Appendix B: Testing Transcripts

Personality & Emotion Test Transcripts

BotA:

[11:19:46:] Bot: Hey, what's up?
[14:25:42:] User: hey

[14:26:14:] Bot: ~~HAPPY~~YO
[14:39:34:] User: how is it going

[14:39:34:] Bot: ~~HAPPY~~i'm cheerful.
[14:39:50:] User: anything going on

[14:39:50:] Bot: ~~HAPPY~~yea, have to see the dentist later today.

[14:40:11:] User: ~~HAPPY~~sounds like fun

[14:40:12:] Bot: ~~HAPPY~~yea, i like going to the dentist.
[14:40:26:] User: why

[14:40:27:] Bot: ~~HAPPY~~because i like having healthy teeth.
[14:40:41:] User: why are you in such a good mood

[14:40:42:] Bot: ~~HAPPY~~today just feels like a good day.

[14:40:58:] User: ~~HAPPY~~i love eating candy

[14:40:58:] Bot: ~~HAPPY~~i love to eat candy too.
[14:41:17:] User: i am getting really annoyed with you

[14:41:17:] Bot: ~~ANGRY~~i'm getting annoyed with you.
[14:41:59:] User: let us talk about something else

[14:41:59:] Bot: ~~ANGRY~~what about?
[14:42:10:] User: how about cheese

[14:42:10:] Bot: ~~ANGRY~~i like cheese, but i also detest it.

[14:42:27:] User: ~~HAPPY~~i think the aroma it has is lovely

[14:42:27:] Bot: ~~HAPPY~~yes, i love the aroma
[14:42:57:] User: what about the texture and taste that is great too

[14:42:57:] Bot: ~~HAPPY~~oh yes, it is intoxicating
[14:43:13:] User: well anyway i am hungry want to go get some sushi

[14:43:13:] Bot: ~~HAPPY~~right now, sure. it should ease my hunger
[14:43:27:] User: by the way how is your thesis going

[14:43:27:] Bot: ~~HAPPY~~if i look at it optimistically, i'm doing pretty well

[14:43:42:] User: oh really why is that



[14:43:42:] Bot: ~~HAPPY~~ the progress is slow, but i'm still hopeful since i still have time

[14:44:30:] User: are we still going to great america



[14:44:30:] Bot: ~~HAPPY~~ yea, i love great america

[14:45:11:] User: i like the top gun ride



[14:45:12:] Bot: ~~HAPPY~~ that ride is exciting, but intimidating.

[14:45:36:] User: i hate how expensive food is there



[14:45:37:] Bot: ~~HAPPY~~ i'm glad they charge that much, otherwise they might raise gate prices or charge per ride like the boardwalk.

[14:46:07:] User: same with movie theatres



[14:46:10:] Bot: ~~HAPPY~~ KEARNEY — How much of a good thing is too much? When it comes to new movie theaters, Kearney might soon find out.

[14:47:36:] User: same with movie theatres food



[14:47:36:] Bot: ~~ANGRY~~ yea, it makes me angry how much we get ripped off.

BotC:

[14:54:46:] Bot: Hey, what's up?

[14:55:04:] User: how is it going



[14:55:05:] Bot: ~~SAD~~ ok i guess, i'm content yet depressed.

[14:55:17:] User: anything going on



[14:55:18:] Bot: ~~SAD~~ yea, have to see the dentist later today.



[14:55:35:] User: ~~HAPPY~~ sounds like fun



[14:55:36:] Bot: ~~SAD~~ no it's not. it's torture going.

[14:55:45:] User: why



[14:55:46:] Bot: ~~SAD~~ i detest going because if the results are bad, then no more candy.

[14:56:17:] User: why are you in such a sad mood



[14:56:18:] Bot: ~~ANGRY~~ because i'm feeling angry.



[14:56:30:] User: ~~HAPPY~~ i love eating candy



[14:56:31:] Bot: ~~ANGRY~~ so do i, that's why i hate seeing the dentist.

[14:56:49:] User:



[14:56:51:] Bot: ~~HAPPY~~ Let's talk about something else. I think my interest in racing games vanished shortly around age 16 when people I knew started to drive real cars. For those who still love to go 'round and 'round, I give you Burnout Dominator for the PS2.

[14:57:21:] User: let us talk about something else



[14:57:22:] Bot: ~~HAPPY~~ what about?

[14:57:27:] User: how about cheese

[14:57:27:] Bot:  ANGRY i like cheese too, but i also abhor it.

[14:57:37:] User:  HAPPY i think the aroma it has is lovely

[14:57:37:] Bot:  ANGRY no, i hate cheese

[14:57:43:] User: what about the texture and taste that is great too

[14:57:43:] Bot:  ANGRY no, i hate cheese

[14:57:50:] User: well anyway i am hungry want to go get some sushi

[14:57:50:] Bot:  ANGRY no, i'm too annoyed to eat

[14:57:56:] User: by the way how is your thesis going

[14:57:57:] Bot:  ANGRY writing this paper is aggravating

[14:58:04:] User: oh really why is that

[14:58:04:] Bot:  HAPPY the progress is slow, but i'm still hopeful since i still have time

[14:58:41:] User: are we still going to great america

[14:58:42:] Bot: Let's talk about something else. MANILA, Philippines - The canine versions of Starsky and Hutch have arrived in Manila, where they sniffed through locked steel doors, pointing authorities to stacks of pirated movies during a mall raid Monday.

[14:59:57:] User: let us talk about something else

[14:59:57:] Bot:  HAPPY what about?

[15:00:10:] User: are we still going to great america

[15:00:11:] Bot:  FEAR no, i'm intimidated by roller coasters.


[15:00:28:] User:

[15:00:29:] Bot:  FEAR Let's talk about something else. Just what the PSP needed, this latest Metal Gear delivers on all fronts with an excellent story, a deep campaign, and a complete set of multiplayer features.

[15:01:35:] User: i like the top gun

[15:01:36:] Bot:  FEAR that ride is exciting, but intimidating.

[15:01:59:] User: i hate how expensive food is

[15:02:00:] Bot:  SAD yea, it's deplorable how we have to pay so much for a drink.

[15:02:21:] User: same with movie theatres food

[15:02:22:] Bot:  ANGRY yea, it makes me angry how much we get ripped off.

Domain Concept Preference Test Transcripts

BotA: Mr. Vampire

[18:50:23:] User: want to watch a movie?

[18:50:24:] Bot: What is the movie about?

[18:55:28:] User: In China, a Taoist priest and two of his apprentices fight against Chinese vampire zombies and ghosts.

[18:55:47:] Bot: Go on...

[18:57:39:] User: The Taoist priest is called upon by a rich merchant to check the feng shui, a traditional Chinese practice of judging harmony based upon arrangement of objects, of the merchants fathers grave

site.

[18:57:56:] Bot: Go on...

[18:59:18:] User: Turns out that the fathers grave site is cursed and the Taoist priest tries to purify the corpse by taking it back to his morgue.

[18:59:33:] Bot: Go on...

[19:00:56:] User: At the morgue, the corpse turns into a vampire zombie and escapes.

[19:01:06:] Bot: Go on...

[19:01:49:] User: The vampire kills his son, and the son turns into a vampire.

[19:02:00:] Bot: What else

[19:02:50:] User: The priest kills the rich merchant.

[19:03:01:] Bot: Go on...

[19:04:57:] User: Now the priest must protect the merchants daughter from the other vampire that escaped.

[19:05:07:] Bot: this movie is getting pretty good

[19:06:00:] User: The vampire terrorizes the people.

[19:06:09:] Bot: Go on...

[19:06:37:] User: The priest and apprentices fight the vampire with Taoist magic.

[19:06:47:] Bot: Go on...

[19:07:19:] User: But the vampire bites one of the apprentices and escapes wounded.

[19:07:29:] Bot: Not bad

[19:08:16:] User: While the vampire hides, the second apprentice is seduced by a female ghost.

[19:08:26:] Bot: This movie sounds really interesting...

[19:09:20:] User: The Taoist priest must protect his apprentices and kill the vampire and ghost.

[19:09:20:] Bot: sounds pretty good

[19:10:05:] User: The priest manages to fight the ghost and convinces the ghost not to bother the apprentice anymore.

[19:10:16:] Bot: Go on...

[19:10:55:] User: Now the vampire is the only evil to be destroyed.

[19:11:07:] Bot: Go on...

[19:11:45:] User: The vampire grows stronger and bursts into the priests home.

[19:11:54:] Bot: Go on...

[19:12:43:] User: The priest and apprentices fight the vampire with martial arts.

[19:12:51:] Bot: Go on...

[19:13:24:] User: Eventually the vampire is burned to death.

[19:13:34:] Bot: Go on...

[19:14:23:] User: So, do you want to watch?

[19:14:23:] Bot: 🍷🍷 This movie is a must watch. I definitely feel like watching this.

BotA: Hellraiser

[19:16:53:] User: want to watch a movie?

[19:16:53:] Bot: What is the movie about?

[19:18:04:] User: Hellraiser is a supernatural movie based on Clive Barkers work set in the modern day 1980s in the United States.

[19:18:17:] Bot: This movie is sounding pretty good, tell me more!

[19:19:13:] User: The gory movie involves a mysterious puzzle box that acts as a gateway to the dimension called Hell.

[19:19:24:] Bot: this movie is getting pretty good

[19:21:12:] User: Solving the puzzle calls forth the Cenobites, evil demons that take pleasure in removing the raw flesh and causing extreme pain to the puzzle solver.

[19:21:24:] Bot: sounds pretty good

[19:26:01:] User: The heroine is a young girl whose uncle becomes a victim of the Cenobite monsters but the uncle incompletely resurrects when bits of his flesh come in contact with the young girls fathers blood in the old attic of the house.

[19:26:38:] Bot: Not bad

[19:28:46:] User: The young girls stepmother loves the uncle and so schemes to lure male strangers with

sex into her home and murders them so that the uncle can suck the flesh.

[19:28:57:] Bot: What else happens?

[19:29:44:] User: The young girl solves the puzzle box as well and meets the Cenobites and makes a bargain with the Cenobite leader, Pinhead, a man who was tortured and pins were hammered into his head.

[19:29:56:] Bot: What else

[19:30:52:] User: Without much time, the young girl must rush to rescue her father and herself from the unnatural and torturous horrors of Hell by using the puzzle box to seal the evil, flesh craving demons away.

[19:31:04:] Bot: doesn't sound that great

[19:35:52:] User: The scenes are very graphic and many scenes have a lot of blood and flesh and gore.

[19:36:27:] Bot: Go on...

[19:39:56:] User: So, do you want to watch?

[19:39:56:] Bot: 🗨️From what I've heard, I don't really care either way But if you want we can still check it out.

BotC: Mr. Vampire

[19:57:40:] User: Want to watch a movie?

[19:57:40:] Bot: What is the movie about?

[20:06:43:] User: In China, a Taoist priest and two of his apprentices fight against Chinese vampire zombies and ghosts.

[20:06:54:] Bot: Go on...

[20:07:04:] User: The Taoist priest is called upon by a rich merchant to check the feng shui, a traditional Chinese practice of judging harmony based upon arrangement of objects, of the merchants fathers grave site.

[20:07:15:] Bot: Go on...

[20:07:21:] User: Turns out that the fathers grave site is cursed and the Taoist priest tries to purify the corpse by taking it back to his morgue.

[20:07:33:] Bot: Go on...

[20:07:37:] User: At the morgue, the corpse turns into a vampire zombie and escapes.

[20:07:47:] Bot: Go on...

[20:07:54:] User: The vampire kills his son, and the son turns into a vampire.

[20:08:03:] Bot: This movie sounds horrible

[20:08:09:] User: The priest kills the rich merchant.

[20:08:18:] Bot: Go on...

[20:08:46:] User: Now the priest must protect the merchants daughter from the other vampire that escaped.

[20:08:56:] Bot: This movie is going down hill...

[20:11:36:] User: The vampire terrorizes the people.

[20:11:45:] Bot: Go on...

[20:11:53:] User: The priest and apprentices fight the vampire with Taoist magic.

[20:12:02:] Bot: Go on...

[20:12:50:] User: But the vampire bites one of the apprentices and escapes wounded.

[20:13:00:] Bot: ok, this is just bad

[20:13:08:] User: While the vampire hides, the second apprentice is seduced by a female ghost

[20:13:18:] Bot: doesn't sound that great

[20:13:29:] User: The Taoist priest must protect his apprentices and kill the vampire and ghost.

[20:13:30:] Bot: this movie is getting worse

[20:13:40:] User: The priest manages to fight the ghost and convinces the ghost not to bother the apprentice anymore.

[20:13:49:] Bot: doesn't sound that great

[20:15:46:] User: Now the vampire is the only evil to be destroyed.

[20:15:55:] Bot: This movie is going down hill...

[20:16:02:] User: The vampire grows stronger and bursts into the priests home.

[20:16:12:] Bot: This movie is going down hill...

[20:16:19:] User: The priest and apprentices fight the vampire with martial arts.

[20:16:28:] Bot:

[20:17:27:] User: Eventually the vampire is burned to death.
[20:17:35:] Bot: doesn't sound that great
[20:18:04:] User: So, do you want to watch?
[20:18:04:] Bot: 🙄🙄 The movie is horrible. I'm in no mood to watch something like that.

BotC: Hellraiser

[19:51:07:] Bot: Hey, what's up?
[19:52:16:] User: Want to watch a movie?
[19:52:16:] Bot: What is the movie about?
[19:52:49:] User: Hellraiser is a supernatural movie based on Clive Barkers work set in the modern day 1980s in the United States.
[19:52:59:] Bot: What else happens?
[19:53:03:] User: The gory movie involves a mysterious puzzle box that acts as a gateway to the dimension called Hell.
[19:53:13:] Bot: Go on...
[19:53:17:] User: Solving the puzzle calls forth the Cenobites, evil demons that take pleasure in removing the raw flesh and causing extreme pain to the puzzle solver.
[19:53:27:] Bot: sounds pretty good
[19:54:14:] User: The heroine is a young girl whose uncle becomes a victim of the Cenobite monsters but the uncle incompletely resurrects when bits of his flesh come in contact with the young girls fathers blood in the old attic of the house.
[19:54:25:] Bot: What else happens?
[19:54:47:] User: The young girls stepmother loves the uncle and so schemes to lure male strangers with sex into her home and murders them so that the uncle can suck the flesh.
[19:54:56:] Bot:
[19:55:22:] User: The young girl solves the puzzle box as well and meets the Cenobites and makes a bargain with the Cenobite leader, Pinhead, a man who was tortured and pins were hammered into his head.
[19:55:30:] Bot: What else happens?
[19:55:52:] User: Without much time, the young girl must rush to rescue her father and herself from the unnatural and torturous horrors of Hell by using the puzzle box to seal the evil, flesh craving demons away.
[19:56:03:] Bot: Go on...
[19:56:08:] User: The scenes are very graphic and many scenes have a lot of blood and flesh and gore.
[19:56:19:] Bot: And..
[19:56:39:] User: So, do you want to watch?
[19:56:40:] Bot: 🙄🙄 This movie is a must watch. But I'm not in the mood to go watch a movie.

Text Generation Test Transcripts

[13:29:14:] Bot: Hey, what's up?
[13:30:51:] User: so what do you think of the nintendo Wii?
[13:30:56:] Bot: 😄 If you take games a little too seriously, then the newest Nintendo-made Wii game, "Super Paper Mario," is making fun of you.
[13:31:42:] User: Is super paper mario fun?
[13:31:43:] Bot: 😄 It's really hard to classify "Super Paper Mario" for the Nintendo Wii.
[13:32:34:] User: why is it hard to classify super paper mario?
[13:32:35:] Bot: 😄 Let's talk about something else. Just what the PSP needed, this latest Metal Gear delivers on all fronts with an excellent story, a deep campaign, and a complete set of multiplayer features.
[13:33:06:] User: wow, sounds like Metal Gear Portable Ops is great
[13:33:09:] Bot: 😄 Let's talk about something else. I think my interest in racing games vanished shortly

around age 16 when people I knew started to drive real cars. For those who still love to go 'round and 'round, I give you Burnout Dominator for the PS2.

[13:33:41:] User: but i want to talk about Metal Gear for PSP



[13:33:43:] Bot: **HAPPY** Let's talk about something else. White Men Can't Jump, but that didn't keep it from making Rotten Tomatoes' list of the greatest sports movies ever.

[13:34:17:] User: ok, fine. but i think that movie was stupid



[13:34:18:] Bot: **HAPPY** At this moment police are investigating a stand-off that ended with two shot and at least one dead. A one-year-old baby caught in the middle is OK.

[13:35:04:] User: really? well it's good that the baby is ok. there are too many shootings lately



[13:35:06:] Bot: **HAPPY** Former Lord Chief Justice Lord Woolf called for a cap on prison places saying no-one should go to jail unless they really deserve it. The overcrowding and the high costs of imprisonment mean only the violent should be locked up, he told MPs

[13:40:24:] User: prison overcrowding has always been a problem



[13:40:26:] Bot: **HAPPY** Wind warnings REUTERS High wind warnings remained in effect for much of Southern California after powerful gusts knocked out power for tens of thousands of utility customers Thursday, stoked a wildfire in the hills above Los Angeles that damaged expensive homes and triggered a sandstorm that may have led to a fatal highway pileup.

[13:45:16:] User: let us talk about something else

[13:45:18:] Bot: what about?

[13:45:24:] User: how about cheese



[13:45:25:] Bot: **HAPPY** i like cheese, but i also detest it.

[13:45:58:] User: i think the aroma it has is lovely



[13:45:59:] Bot: **DISGUST** it depends on the cheese, sometimes it smells likable other times it's appalling

[13:46:23:] User: what about the texture and taste that is great too



[13:46:25:] Bot: **HAPPY** oh yes, it is intoxicating



[13:46:56:] User: **HAPPY** i love cheese on pizza



[13:46:58:] Bot: **SURPRISE - HAPPY** GALVESTON — Buffalo has its chicken wings. Chicago has deep-dish pizza. Galveston has the surprise burrito.



[13:48:36:] User: **SURPRISE** a surprise burrito?



[13:48:38:] Bot: **SURPRISE - HAPPY** Baskin-Robbins, Walgreens and Blue Burrito have agreed to join Surprise's first movie theater, UltraStar Cinemas, in the \$75 million Surprise Pointe lifestyle and entertainment center at Waddell and Litchfield roads.

[13:54:46:] User: let's get to talking about different cheeses



[13:54:47:] Bot: **HAPPY** BBC Europe editor Mark Mardell hits the campaign trail in France as presidential candidates vie for the spotlight.

[20:01:14:] User: well i guess french cheeses are nice



[20:01:25:] Bot: **HAPPY** Let's talk about something else. Just what the PSP needed, this latest Metal Gear delivers on all fronts with an excellent story, a deep campaign, and a complete set of multiplayer features.

[21:01:03:] User: what is metal gear



[21:01:05:] Bot: **SURPRISE** Pedal, meet metal. We get our feet wet in our first look at Forza 2's career mode.

[21:23:42:] User: i was talking about Metal Gear Solid



[21:23:43:] Bot: **HAPPY** It doesn't seem like two decades ago that the K-Car based Chrysler LeBaron brought back the first convertible after years of ill-conceived safety legislation that banned drop-tops.

References

- [1] Bartneck, Christoph. Integrating the OCC Model of Emotions in Embodied Characters.
- [2] Bazzan, A. L. C., and Bordini, R. H. A Framework for the Simulation of Agents with Emotions: Report on Experiments with the Iterated Prisoner's Dilemma. International Conference on Autonomous Agents: Proceedings of the fifth international conference on Autonomous agents, May 2001.
- [3] Coursey, Kino. Living in Cyn: Mating AIML and CYC Together with Program N. January 2004.
- [4] Darwin, Charles. The Expression of the Emotions in Man and Animals, Third Edition. New York: Oxford University Press. 1998.
- [5] Descamps, S., Prendinger, H., and Ishizuka, M. A Multimodal Presentation Markup Language for Enhanced Affective Presentation.
- [6] Egges, A., Kshirsagar, S., and Magnenat-Thalmann, N. A Model for Personality and Emotion Simulation.
- [7] Ekman, Paul. Basic Emotions. Handbook of Cognition and Emotion. Sussex, United Kingdom: John Wiley & Sons, Ltd., 1999.
- [8] Ekman, Paul. Facial Expression and Emotion. American Psychologist. Volume 48, Number 4, pp 384-392. April 1993.
- [9] Fitrianie, S. and Rothkrantz, L. J. M. My_Eliza, A Multimodal Communication System.
- [10] Funge, J., Tu, X., and Terzopoulos, D. Cognitive Modeling: Knowledge, Reasoning and Planning for Intelligent Characters. International Conference on Computer Graphics and Interactive Techniques, Proceedings of the 26th annual conference on Computer graphics and interactive techniques, 1999.
- [11] Gebhard, Patrick. ALMA – A Layered Model of Affect. International Conference on Autonomous Agents: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, July 2005.
- [12] Hubal, R. C., Frank, G. A., and Guinn, C. I. Lessons Learned in Modeling Schizophrenic and Depressed Responsive Virtual Humans for Training. International Conference on Intelligent User Interfaces: Proceedings of the 8th international conference on Intelligent user interfaces, January 2003.

- [13] Kallmann, M., de Sevin, E., and Thalmann, D. Towards Real Time Virtual Human Life Simulations. IEEE Proceedings of Computer Graphics International, 2001.
- [14] Karunaratne, S. and Yan, H. 3D Animated Movie Actor Training using Fuzzy Logic. IEEE Proceedings of Computer Graphics International, 2001.
- [15] Kshirsagar, S. and Magnenat-Thalmann, N. A Multilayer Personality Model. ACM Symposium on Smart Graphics, June 2002.
- [16] Lawrence, P. R. and Nohria, N. Driven: How Human Nature Shapes Our Choices. San Francisco: Jossey-Bass. 2002.
- [17] Liu, H., Lieberman, H., and Selker, T. A Model of Textual Affect Sensing using Real-World Knowledge. International Conference on Intelligent User Interfaces: Proceedings of the 8th international conference on Intelligent user interfaces, January 2003.
- [18] Liu, H. and Maes, P. What Would They Think? A Computational Model of Attitudes. International Conference on Intelligent User Interfaces: Proceedings of the 9th international conference on Intelligent user interfaces, January 2004.
- [19] Magnenat-Thalmann, N., Kim, H., Egges, A., and Garchery, S. Believability and Interaction in Virtual Worlds. IEEE Proceedings of the 11th International Multimedia Modelling Conference, 2005.
- [20] Mehrabian, Albert. Incorporating Emotions and Personality in Artificial Intelligence Software. Retrieved from <http://www.kaaj.com/psych/ai.html> on July 29, 2005.
- [21] Neubauer, Bruce J. Designing Artificial Personalities Using Jungian Theory. Consortium for Computing Sciences in Colleges. 2004.
- [22] Ortony, A., Clore, G., and Collins, A. The Cognitive Structure of Emotions. New York: Cambridge University Press. 1988.
- [23] Ortony, A., Clore, G., and Collins, A. The Referential Structure of the Affective Lexicon. Cognitive Science: A Multidisciplinary Journal, Volume 11, Number 3, pp 341-364. 1987.
- [24] Popkins, Nathan C. The Five-Factor Model: Emergence of a Taxonomic Model for Personality Psychology. Retrieved from <http://www.personalityresearch.org/papers/popkins.html> on January 4, 2006.
- [25] Romano, D. M. and Wong, A. K. L. Personality Model of a Social Character.

- [26] Trappl, R., Petta, P., and Payr, S. Emotions in Humans and Artifacts. Massachusetts: MIT Press. 2002.
- [27] Traum, D. and Rickel, J. Embodied Agents for Multi-party Dialogue in Immersive Virtual Worlds. International Conference on Autonomous Agents, Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1, July 2002.
- [28] Wallace, Richard S. The Anatomy of A.L.I.C.E. A.L.I.C.E. Artificial Intelligence Foundation, Inc.
- [29] Wilson, Ian. The Artificial Emotion Engine™, Driving Emotional Behavior. November 1999.
- [30] Yang, H., Chua, T., Wang, S., and Koh, C. Structured Use of External Knowledge for Event-based Open Domain Question Answering. Annual ACM Conference on Research and Development in Information Retrieval: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, July 2003.
- [31] Zhe, X. and Boucouvalas, A. Text-to-Emotion Engine for Real Time Internet Communication.
- [32] <http://news.bbc.co.uk/2/hi/europe/6546069.stm>
- [33] <http://www.jg-tc.com/articles/2007/04/16/features/doc4622c36151c5b655093884.txt>
- [34] <http://en.wikipedia.org/wiki/Personality>
- [35] <http://www.jabberwacky.com/>
- [36] <http://www.personalityforge.com/>
- [37] http://en.wikipedia.org/wiki/Big_Five_personality_traits
- [38] <http://commonsense.media.mit.edu/cgi-bin/search.cgi>
- [39] <http://wordnet.princeton.edu/>
- [40] <http://web.media.mit.edu/~hugo/conceptnet/>
- [41] <http://en.wikipedia.org/wiki/Emotion>