

# Saya – Mini Project

## Text Based Emotion Estimation

Ben-Gurion University of the Negev

Department of Computer Science

September 2008

Advisors: Prof. Shlomi Dolev &

Mr. Michael Orlov

Submitted by: Itamar Chazanovitz

Meital Greenwalc

Table of Contents

Introduction..... 3

Motivation..... 4

Solution..... 5-10

    Parsing Phase..... 5

    Emotion Extraction Phase..... 6

    Applying Negation..... 9

    Sentence Tagging..... 9

    Summery..... 10

Results and Conclusions..... 11

    Execution Examples..... 11

    Conclusions and Future Extensions..... 11

User Guide..... 12

Bibliography..... 13

## Introduction

Saya was invented and developed by Hiroshi Kobayashi, an associate professor at the Tokyo University of Science.

Saya's human-like face and ability to express facial expressions similar to human beings paved her way to become the ultimate receptionist .

The Department of Computer Science of "Ben-Gurion University of The Negev" saw this potential in Saya and not long after appointed her to be the receptionist of the new "Alon Hi-Tech Building" lobby. Saya was brought to serve as a learning and research tool on which students can apply their new learned knowledge and experience with some Hi-Tech technologies .

This paper presents an approach to emotion estimation that assesses the affective content from textual messages. Our main goals are to detect emotion from text messages and to produce emotional reasoning based on the textual interaction .

In this paper, the emotion estimation module is applied to text messages produced by a chat system in Saya or text messages coming from the voice-recognition system, and by that, allowing Saya to act out the assessed emotions of messages through facial gestures.

## Motivation

Human beings are not only able to audio process the spoken conversation, but also to analyze the conversation according to the facial expressions of their conversational partner. The ability to react with accordance to the emotional value of an input or output sentence can turn a conversation with a robot to be much more interesting.

The prospect of such an ability implemented in a robot drew us to form the text based emotion estimation module. Therefore, we wanted Saya to be able to act more human-like and to give her responses liveliness through her ability to perform facial gestures such as Anger, Surprise, Happy, Sad, Disgust, and Fear.



The main issue was how to interpret an input string, coming from either the speech recognition module or the output of the chat system in Saya, to one of the specific emotional gestures Saya is capable of.

In order to do so we had to analyze the input text in a way that will allow multiple emotions in the input, and also to be able to deal with negation affecting any word that has relevant emotional value.

We needed to resolve the sense of the given input sentence and in case we found that the sentence had any emotional value, to act on it.

## Solution

We approach the solution with an algorithm that consists of 4 main phases:

- Parsing Phase
- Emotion Extraction Phase
- Negation Detection
- Sentence Tagging

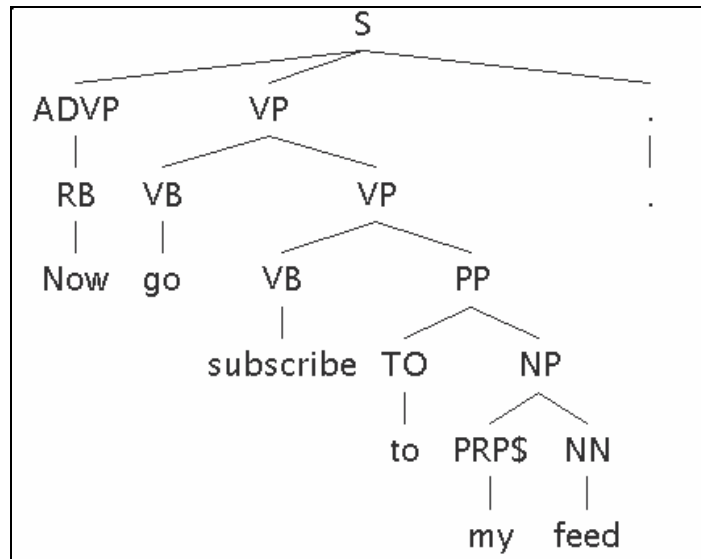
### Parsing Phase –

The first stage after receiving an input sentence is to create a parse tree using the Stanford Parser. Our choice of parsing language was the English-PCFG package (Probabilistic Context Free Grammar); this parsing language was constructed using the Penn Treebank which is a large set of parsed trees formed out of sentences from the "Wall Street Business Journal". Probabilistic parsers use knowledge of language gained from hand-parsed sentences to try to produce the most likely analysis of new sentences. These statistical parsers still make some mistakes, but commonly work rather well.

The parser works out the grammatical structure of sentences, for instance: which groups of words go together as "phrases" and which word is the subject or the object of a verb. The parser can read various forms of plain text input and can output various analyses formats, including part-of-speech tagged text, phrase structure trees and a grammatical relations (typed dependency) format.

We use the part of speech (POS) tagged text and the grammatical relations from which we derive the typed dependencies. Once we have the dependency data, we analyze it in order to find if there is a negation dependency in the sentence and prepare the needed information to be used later on.

An example of an output parse tree:



Here we can see that the original sentence was "Now go subscribe to my feed", and some of the PCFG rules that were used in the parsing process are shown:  $S \rightarrow ADVP$ ,  $ADVP \rightarrow RB$ ,  $S \rightarrow VP$ ,  $VP \rightarrow VB VP$  etc.

The Penn Treebank is based upon a special Tag-Set, therefore the parse trees are tagged with this particular tag-set. The tag-set is very rich and must be narrowed down to the four main part of speech types which are: noun, verb, adjective and adverb. The reason that the algorithm must roll back each tag into its basic POS type is because that in the next phase the WordNet database is divided into these basic POS types, therefore must be accessed with only these part-of-speech types.

**Emotion Extraction Phase –**

At this phase we assign every word with an object that will hold the following information: array of emotions (happiness, sadness, anger, fear, surprise and disgust), negation information, the dominant emotion of the word and the word itself.

Once we've established the POS type for each word in the sentence, we proceed by extracting the possible senses hidden behind each word using an open source Java implementation of WordNet v2.1 called JWordnet, and the XML files of the WordNet-Affect v1.1.

**WordNet** is a large lexical database of English, developed under the direction of George A. Miller.

In this database, nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms called "synsets", each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations, resulting in the formation of a network of meaningfully related words and concepts.

For each synset there exists a "synset offset" in the form:

<POS-Type>#<8 digit number> where the <POS-Type> is for example: "n" stands for "Noun" and "vb" stands for "Verb."

**WordNet-Affect** – in order to understand what WordNet-affect is, we need first to understand the concept of semantic domains. Semantic domains provide a natural way to establish semantic relations among word senses, which can be profitably used for Computational Linguistics. Semantic domains are areas of human discussion, such as politics, economy, sports, which exhibit their own terminology and lexical coherence.

WordNet-Affect is a hierarchy of affective domain labels which were developed by selecting suitable synsets from WordNet which represent affective concepts and dividing them into subsets of affective data .

It is important to state that WordNet-Affect v1.1 was built using WordNet v1.6 and the synset offsets are relevant to this WordNet version only.

Here is an example of the WordNet-Affect hierarchy in its XML form:

```
<categ name="positive-emotion" isa="emotion"/>

<categ name="negative-emotion" isa="emotion"/>

<categ name="neutral-emotion" isa="emotion"/>

<categ name="ambiguous-emotion" isa="emotion"/>

<categ name="joy" isa="positive-emotion"/>
```

Here we can see that in the hierarchy the sub-category "joy" is located under "positive-emotion" which is located along side with "negative-emotion" etc, and all together are located under the sub-category "emotion". Each category has its own synset offset related to it and the mapping between the synset-offsets and the category are also located in a XML file. Those two files together form the WordNet-Affect database.

We use WordNet-Affect to construct a mapping between synset offsets from WordNet, and one of the possible emotion types. In order to do that, we needed to choose base words that will represent each of the emotion types .

For example: the base word for the emotion "Happy" is "positive-emotion". The word "content" has number of senses in WordNet, one of those senses has the offset (in WordNet v1.6) n#05595732; and in WordNet-Affect this synset offset is related to the affective category "happiness" which is located in the hierarchy under "joy" → "positive-emotion". As a result, the emotion paired with the given synset offset in the mapping is "Happy ."

We can now start describing the actual process of the emotion extraction.

Using WordNet we first extract the synsets of every word in the sentence, we then extract the synsets offsets and with the use of a mapping between WordNet v2.1 and WordNet v1.6, we assume the correct offsets which are now relevant to the mapping that was built using WordNet-Affect which, as stated before, was created using WordNet v1.6 .

At the end of this stage we now know which of the synsets has an emotional value as described above, allowing us to update the emotion array of the object holding the word being analyzed, and eventually assign a word with its most probable emotional sense out of the possible emotional senses available.

Using this process we analyze each word in the input sentence. Once completing this phase, we turn to use the negation information we have extracted during the parsing phase.

### **Applying Negation –**

The intuitive way to deal with negation is to emphasize the counter emotion of the emotion found as most dominant in the word. Considering the fact that the only two emotions that we can consider as opposite to one another are "Happy" and "Sad", the negation will turn a word marked with emotional value "Happy", to be marked with emotional value "Sad" and vice versa. As for any other word holding a dominant emotional value which differs from the ones stated above, the negation will simply disable the emotional value of this word and by that turning the emotional value of it to be ambiguous.

### **Sentence Tagging –**

At this phase we construct the output. In order for Saya to be able to express the emotions found in the sentence, we tag the output sentence with XML tags such as <angry> ... </angry>.

The method we use to deal with multi-emotional sentence is: When we reach a word with an emotional value, we open an appropriate tag and close this tag either when we reach a word with a different emotional value, or at the end of the sentence. In case we reached a word with a different emotional value, we open a new emotion tag and in case that the emotional value is similar to the previous one, we continue on to the rest of the sentence.

**Summary –**

The process described above gives us a word-by-word evaluation of the input text and the assignment of actual emotional value to words in the input that holds a semantic value that is of interest to us. We've also shown that different dependencies can be analyzed and integrated into the input classification, and by that enrich the overall analysis capabilities of the module.

The tagging stage allows the communication with the module to be smooth. In case a sentence has no specific emotional sense it will remain untagged, which will insure the conversations continues flow.

## Results and Conclusions

### Execution Examples –

I was <fear> apprehensive before the big test </fear>

>surprise> wow it is amazing to finally meet you </surprise>

I was <sad> melancholy after my dog's death </sad>

I became <angry> infuriated when my favorite pen disappeared </angry>

The <disgust> foul odor of the dead animal revolted me </disgust>

### **Negation –**

I was <happy> content to sit in the sun </happy>

I was not <sad> content to sit in the sun </sad>

### **Multi – Sense –**

I was <happy> happy to meet you and </happy> <sad> sorry to hear about the bad news </sad>

### Conclusions and Future Extensions –

The algorithm we chose to implement is based on an approach of a word-by-word evaluation of the input sentence. We found the results that the module has shown us to be satisfying.

In future projects it is recommended to come up with an additional algorithm that performs the evaluation based on each sentence as a whole using analysis of the structure of the parse tree generated from the sentence. By crossing the evaluation results of the two approaches we assume that a higher level of performance might be achieved.

It is also possible to enrich the current data with an algorithm that upon failure of the module to distinguish an emotion in a sentence, will add the synset relevant to the WordNet-Affect database.

## User Guide

The interface for the **Emotion Estimation Module** is located in the `EmotionEstimation.java` file.

The method **init()** initialize the EEM by loading the Stanford Parser and the construction of the different mappings described. The initialization process is time consuming.

The file **eem.properties** holds all the file-paths needed for the module to function correctly and other relevant information. In this file the section needed alternation before mounting on a different system is marked.

The main method signature is - **String estimateEmotion(String input)** which activates the module on a given input.

The actual method that performs the word-by-word evaluation of the input (sense wise) is `buildSenses(...)`

## Bibliography

- Carlo Strapparava and Alessandro Valitutti.  
"WordNet-Affect: an affective extension of WordNet".  
In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, Lisbon, May 2004, pp. 1083-1086.
- Emotion Estimation and Reasoning Based on Affective Textual Interaction by Chunling Ma, Helmut Prendinger, and Mitsuru Ishizuka
  
- [The Stanford Parser](http://nlp.stanford.edu/software/lex-parser.shtml) - <http://nlp.stanford.edu/software/lex-parser.shtml>
  
- [Penn Treebank](http://www.cis.upenn.edu/~treebank/home.html) - <http://www.cis.upenn.edu/~treebank/home.html>
  
- [Penn Tree Tag-Set](http://www.comp.leeds.ac.uk/amalgam/tagsets/upenn.html) -  
<http://www.comp.leeds.ac.uk/amalgam/tagsets/upenn.html>
  
- [WordNet – a lexical database for the English language](http://wordnet.princeton.edu/) -  
<http://wordnet.princeton.edu/>
  
- [JWordnet](http://jwn.sourceforge.net/) – Pure Java API and browser interface to Princeton's WordNet database - <http://jwn.sourceforge.net/>.