Technothon2015-16

DESIGN SPECIFICATION

Emotion Analyzer

Date Created	30th March, 2016
Author	Mr. Ganesh Mane, Mr. Darshan Shinde, Ms. Kiran Bisht, Ms. Deepa Upparwar
Last Updated by	Mr. Darshan Shinde
Date updated	26th April, 2016
Project Name	Emotion Analyzer

Contents

1. Introduction

Scope:

This document describes the design specification of “Emotion Analyzer” project.

There are several text-based analyzers already present in the market, but they predict only sentiments of the input data, as positive, negative or neutral. Many other analyzers are also present, but the accuracy is low. Those analyzers just do word matching like if a “good” word is there in input, it will directly say it’s a case of joy, even though “not” word is there before “good” i.e. consider a scenario like “It is not good”, they will just match “good” word and predict joy where it should be sadness. To overcome all those issues, we have come with the product “Emotion analyzer”. We have add many features on the top of word matching.

Emotion Analyzer is able to tell the emotions of the input text. It takes words, sentences or paragraphs as input and predicts the emotions. Emotions, here are classified as sadness, anger, fear and joy.

The automatic detection of emotions in texts is important for applications such as: opinion mining and market analysis, effective computing, natural language interfaces, and e-learning environments, including educational games.

References:

1. nlp.stanford.edu/software/tagger.html

2. https://wordnet.princeton.edu/

3. rednoise.org/rita

4. https://tone-analyzer-demo.mybluemix.net/

5. http://text-processing.com/demo/sentiment/

6. George A. Miller (1995). WordNet: A Lexical Database for English.Communications of the ACM Vol. 38, No. 11: 39-41.

7. Christiane Fellbaum (1998, ed.) WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.

8. justgage.com/

2. DB Infrastructure

Database name: dict

Table name: anger1

Field	Type	Null	Key	Default
Word	Varchar(30)	No	PRI	Null
Score	Double	No		Null
Emotion	Varchar(30)	No		Null

Word: It will contain words.

Score: It will contain impact score decimal value in range 0~1.

Emotion: It will contain emotion associated with that word out of joy, anger, fear, sad.

Word is the primary key to avoid duplicates and all fields are set as not null.

3. System Architecture

As in figure, “Emotion Analyzer” systems will collect the data from user which will be preprocessed and will be passed to the system as input; system will analyze the input and classify the data as per emotions in several categories. Output from the system is interpreted by the browser, which will be displayed to user.

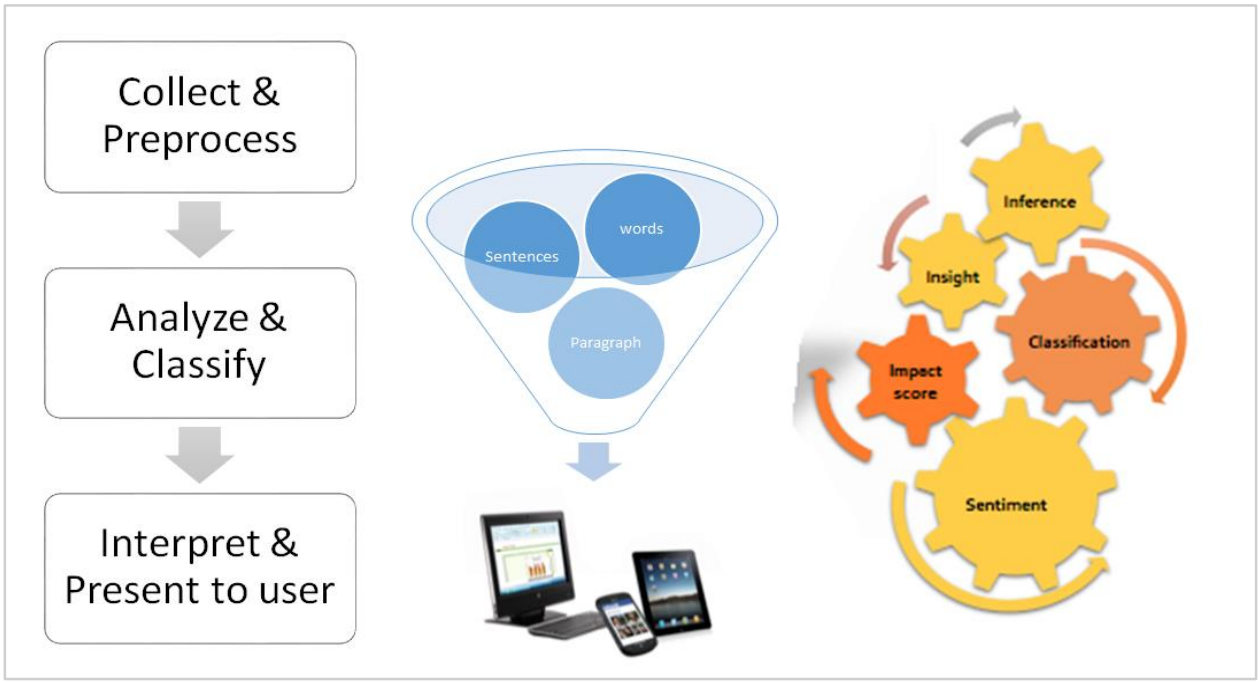

Figure: Architecture - Emotion Analyzer

The data taken from the user is treated as raw input to the system. We process that raw data by applying POS tagger from coreNLP of Stanford NLP library to get the part of speech (POS) tagging to each word of input data. That preprocessed data is treated as an actual input to our system (Emotion Analyzer). We have analyzed near about 15000~20000 sentences during the development time and built a database of words. Database contains words along with the emotions associated with it and impact value of that word over the sentence.

If the word is not found in database, control will pass to WordNet, which has a dictionary of all words. We will search for synonyms or antonyms for that word. Result word list will be passed to the database and then will search for any one of those in database. If any word is found, system will result with the emotion of that word and subsequently add that word automatically to the database. In this manner, database will improve automatically; this is how machine learns by itself.

Now preprocessed data which is the actual input to the system goes to database, search for the words given in input and wrap into a package for each word. Each package contains word along with emotions associated with that word and impact score. Output from the database will be the set of packages for words in input.

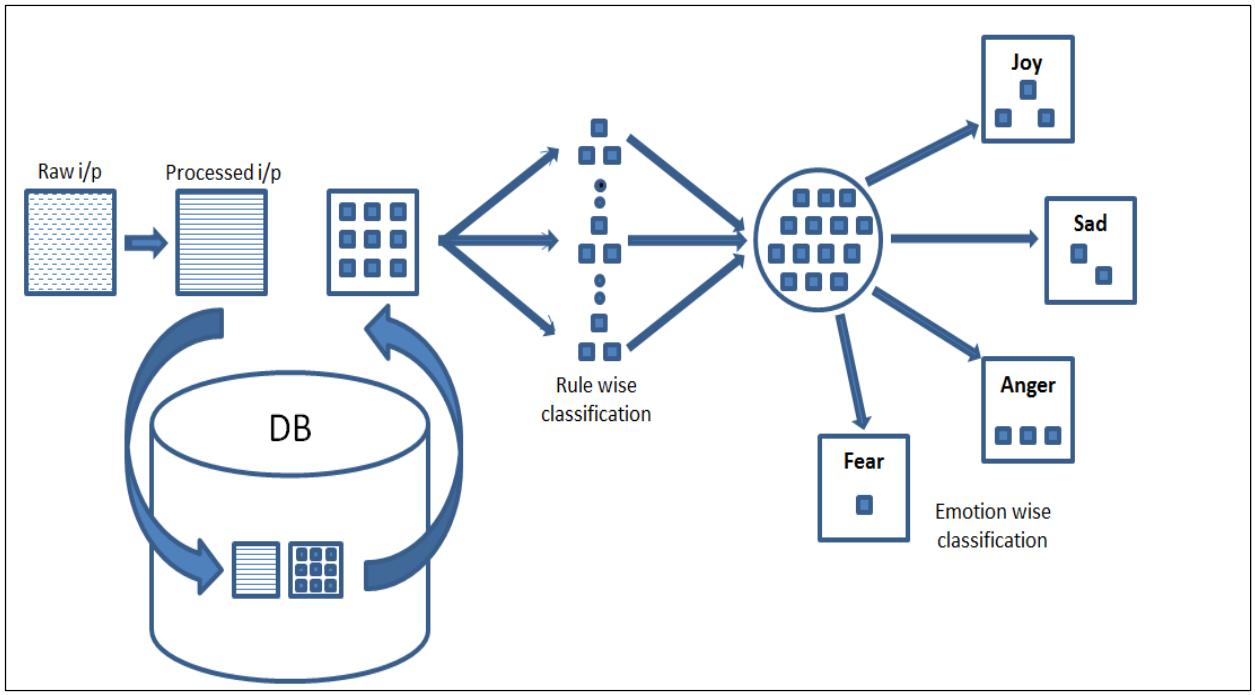

Figure: Detailed Architecture - Emotion Analyzer

For developing system (Emotion Analyzer), we have used rule based supervised approach and decision tree algorithm for classification. We have used classification algorithm twice throughout the code. First time, we do the classification as per rules that we have designed which follow the basics of English language. There are prerequisite conditions defined for each rule. As per the conditions, packages are classified into several clusters where different calculations are proposed as per the rules. By calculations, score associated with each package and emotions get changed in some clusters. After the value change, all packages are collected to a single location where we get our final values for each package. From all that values, we calculate emotion value for complete input.

Later, we do the classification again, this time as per emotions. As our system analyses 4 types of emotions i.e. anger, fear, joy, sad. Classifications are done in those 4 clusters. Here we get our actual output i.e. value for each emotions which is further passed to bar.js where we are drawing our graphical result on GUI.

4. Tools & Technologies

Tools used:

- Notepad ++
- Eclipse IDE
- MySql Command Line Client
- Apache tomcat server

Technologies used:

- HTML5
- CSS3
- JavaScript
- JSP/Servlet

5. Low level design

As user enters text data on web page, input is passed from webpage to tester.java where paragraphs are classified into sentences and words. System searches those words into database. Database will return emotion associated with that word and impact value of that word to tester.java. That result is further passed to home.jsp webpage which will call bar.js with the emotion value string as input to draw result in graphical format on webpage.

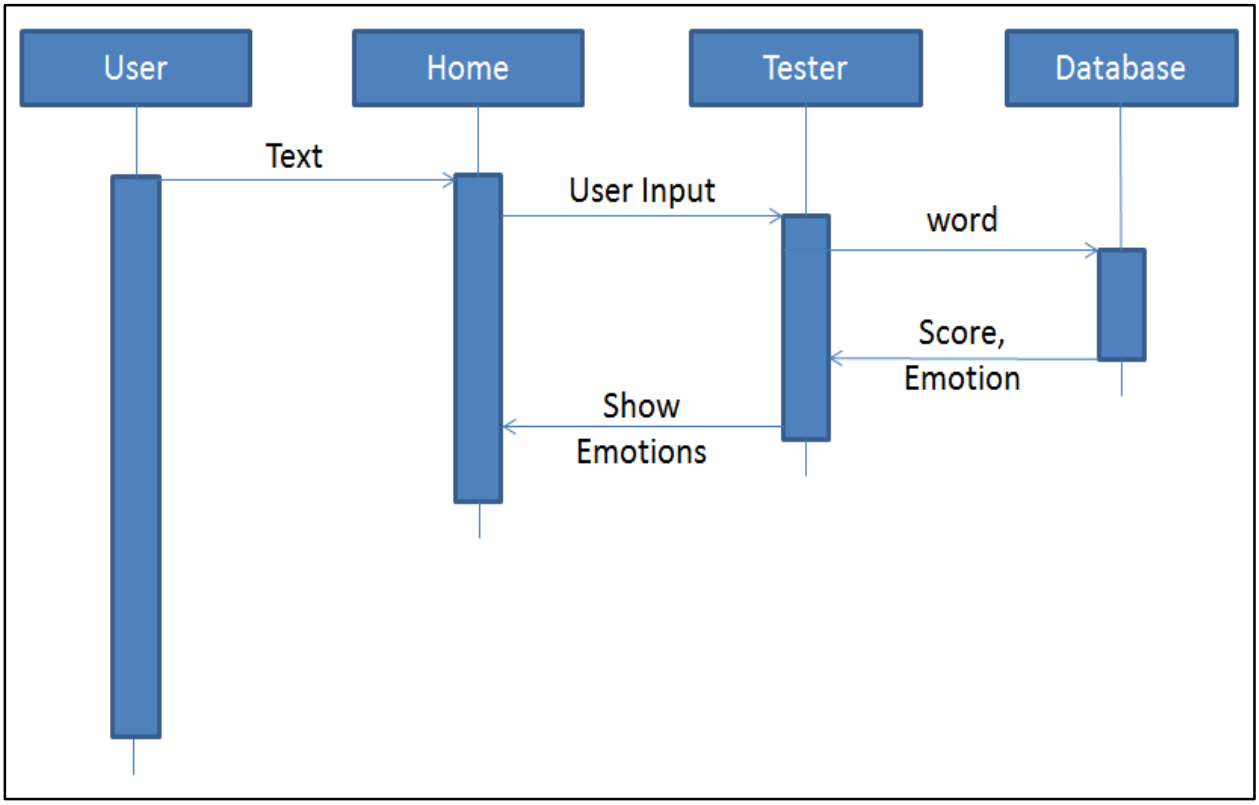

Figure: Sequence diagram (Ideal case)

If the word is not present in database, database will return with failure call to tester.java. Then, tester.java will forward the request to WordNet.java for that word. WordNet.java will return synonyms and antonyms for that word to database. Database will search for those words and return emotion for that word to WordNet.java. Wordnet.java will add that word into the database and returns emotion and impact score for that word to the tester.java. At last, those results will forward to home.jsp webpage which will display result to user.

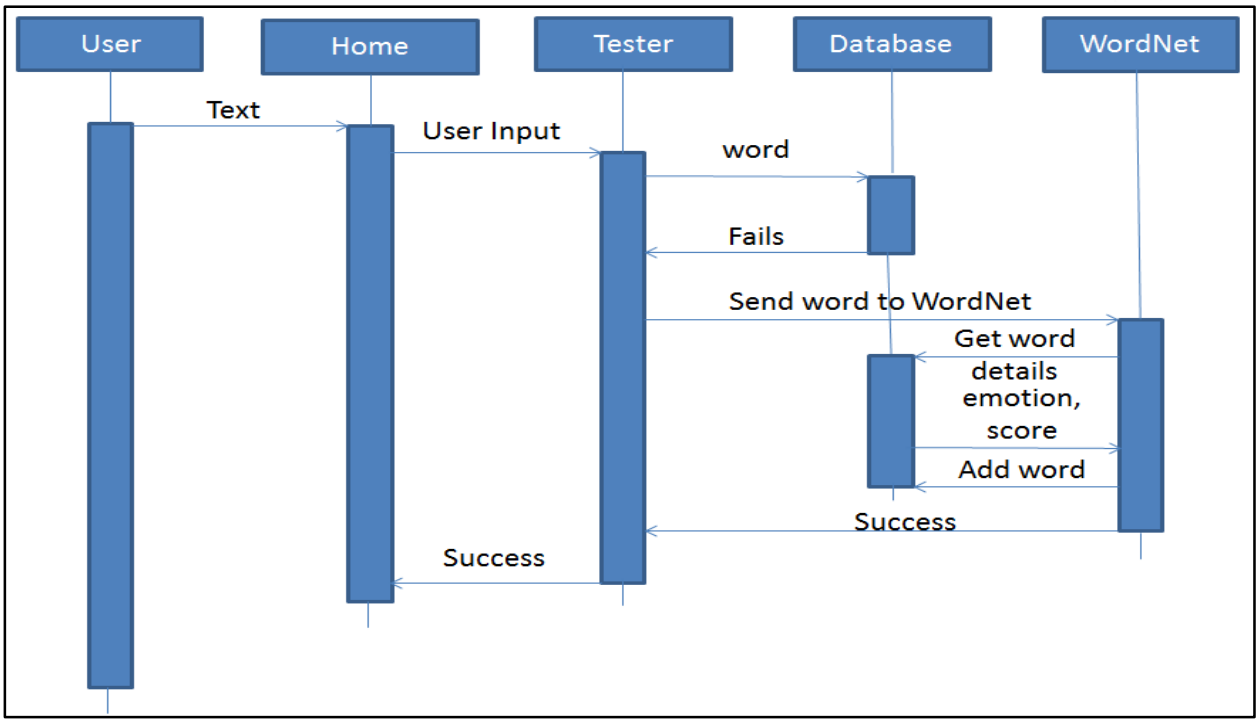

Figure: Sequence diagram (when word not found in database)

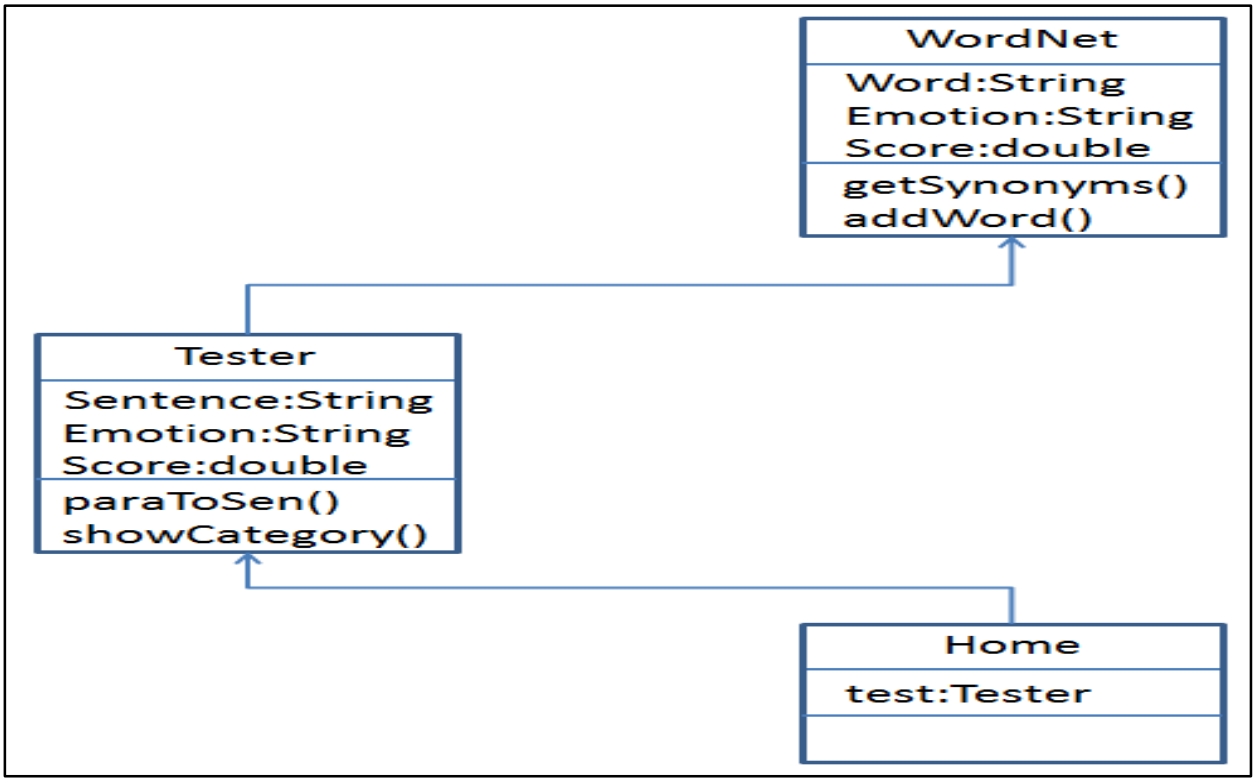

Figure: Class diagram

6. Implementation

The detailed implementation of “Emotion Analyzer” system is as follows:

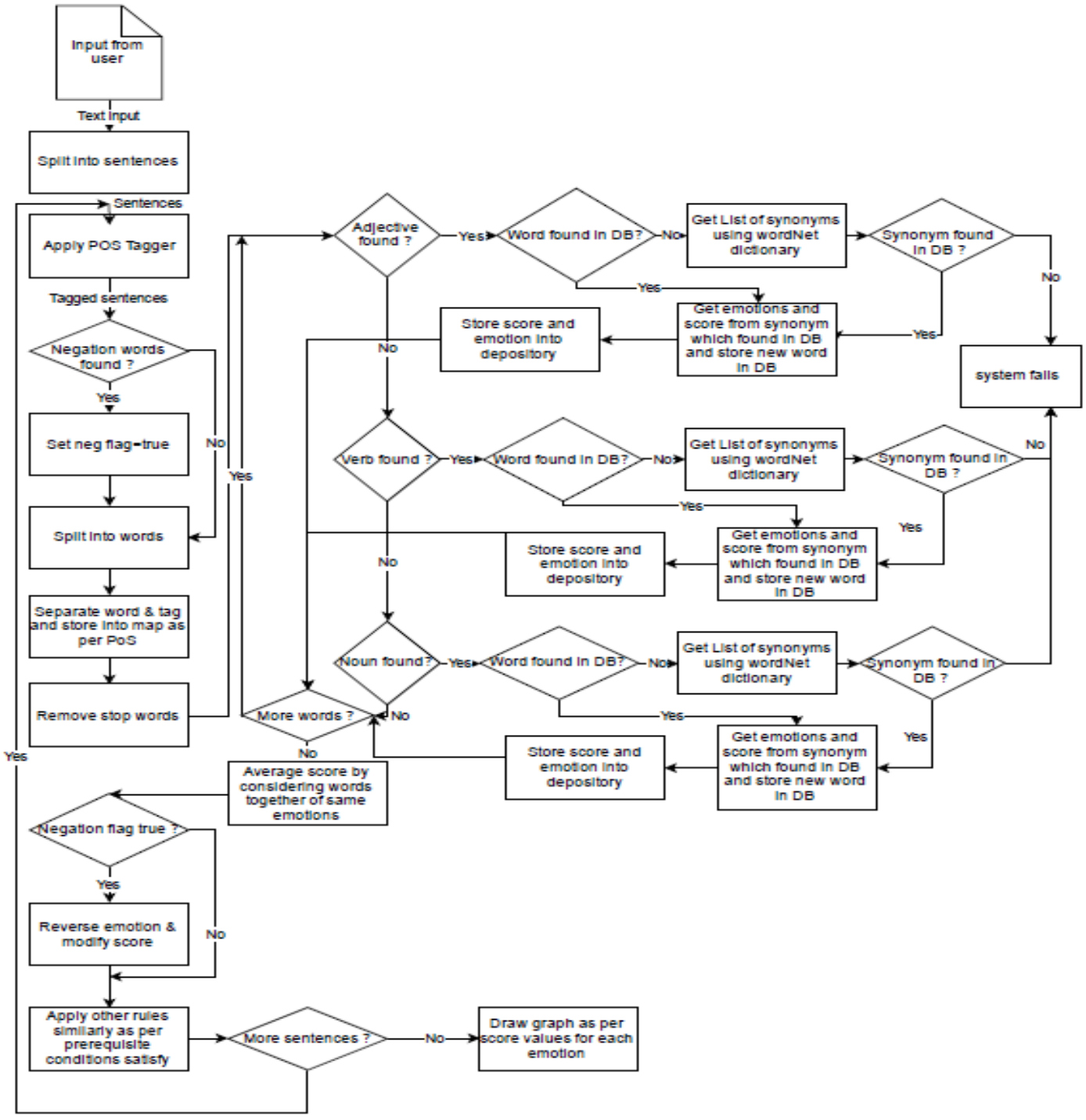

Figure: Emotion Analyzer implementation

As described in the diagram, textual input that system got from user is split into sentences. Each sentence is analyzed separately one-by-one initially and at last output for whole input is calculated and displayed to user.

While analyzing each sentence, PoS tagger is applied to tag the sentence as per the parts of speech. As a result we get tagged sentences which will be split into words. Now system will check for negative words like not, neither, never, none, no one, nothing, etc. and set *neg* flag to true if found. As a next step, words are separated from tag associated with it and all words are stored into Map data structure where part of speech is treated as key and set of words as value.

Before applying PoS tagger, system will search for stop words and remove all stop words from the sentence which will not be analyzed because such words don't have any emotions. This approach will reduce the response time, result into improved performance.

After doing all these preprocessing tasks, actual work starts from here. Several rules are designed in algorithm to analyze the sentence and calculate result. One of those is described in diagram. System will search for adjective first, if found, system will search for those words in database and get the score value and emotion for that word and save it into temporary repository. If adjective is not present, it will search for verb and noun accordingly and do same task. Now all data is classified according to emotions and average value is calculated where adjective has got a higher weightage than verb and noun.

Along with that, several other rules are applied. Such as, system will search for controversy words which basically reverse the meaning of the sentence. If such words are found, like but, although, etc. it will reduce the weightage of the previous sentence over the next sentence accordingly. Consider a situation where user enters a sentence "I was happy but now I am sad." As there are two words after removing stop words, happy and sad. We will find two types of emotions as joy and sadness. There is a word 'but' which is changing the context of the sentence from happiness to sadness. System will catch such controversy words and reduce impact of previous part of the sentence which will result in reduction of joy factor and now sadness factor is clearly visible over joy factor.

If input from user contains more than one sentence, that will be treated as a paragraph. In general, first sentence of paragraph introduces the paragraph while last sentence concludes the paragraph which means there is higher impact of last sentence over the paragraph's first and middle sentences. The same rule is designed in algorithm. When there is paragraph as input, system will give higher weightage to last sentence while computing final combined output for complete paragraph.

If *neg* flag is set to true, which is set during preprocessing, means sentence is of type negation, in such cases, system will reverse the emotion of the sentence with modified score value. As the sentence, "I am not happy", means sadness is there but not as higher as if it will be in "I am happy". So in our algorithm, we are reducing the score while reversing the emotion associated with it.

If a sentence contains quantifier words like more, most, very, higher, less etc., system catches such words and modifies the score as per the words on upper/lower side.

If a sentence contains question mark that means, input is question. Question basically doesn't possess any emotion, so in such cases system will return '*no emotions*' message. If after removing stop words, there is nothing remaining to analyze in such case, system gives result as '*no emotions*'. E.g., "My name is Parvati." There are all stops words which don't have any emotions. In such case system will return 'no emotions'.

As such few more rules are designed in algorithm and implemented and final output is calculated.

One major problem in this approach is:

We are searching for words into database. English language have trillions of words which are not possible to cover. The main problem is, if a word is not found into database, then system may fail. To handle this issue, we come with a new idea. We are using WordNet dictionary. If word is not found into database, then system will get all synonyms for that word by using WordNet dictionary. And then search for all synonyms in database. The database is prepared smartly, as we will find out at least one synonym in database from where we get the emotion and score for that word. And that new word is automatically added to database. First time system will take this long route, but by adding that word to database, system will save time from next time as system will find that word directly from the database. As the database keeps improving automatically. This is the machine learning part.

As we have designed test data of database very intelligently, though there are 1-2 % chances that we would not find any synonym in the database. In such case, system will return with output as 'no emotions'. In such cases, system will fail.

Even though, we tried to cover all defects which are present in existing system, there are few limitations of the "Emotion Analyzer" system. It is tested in very restricted and supervised environment. We have tested it on a huge data set, we are getting 73-74% accuracy. Outside the boundary, it is not tested. In such cases, it may fail. Currently the boundary of that restricted area is very small. We are striving to spread boundaries of the system. But it will need more research work which will take time.

7. User Interface Design

The following frameworks are used to design the UI:

- CSS3
- HTML5
- Javascript

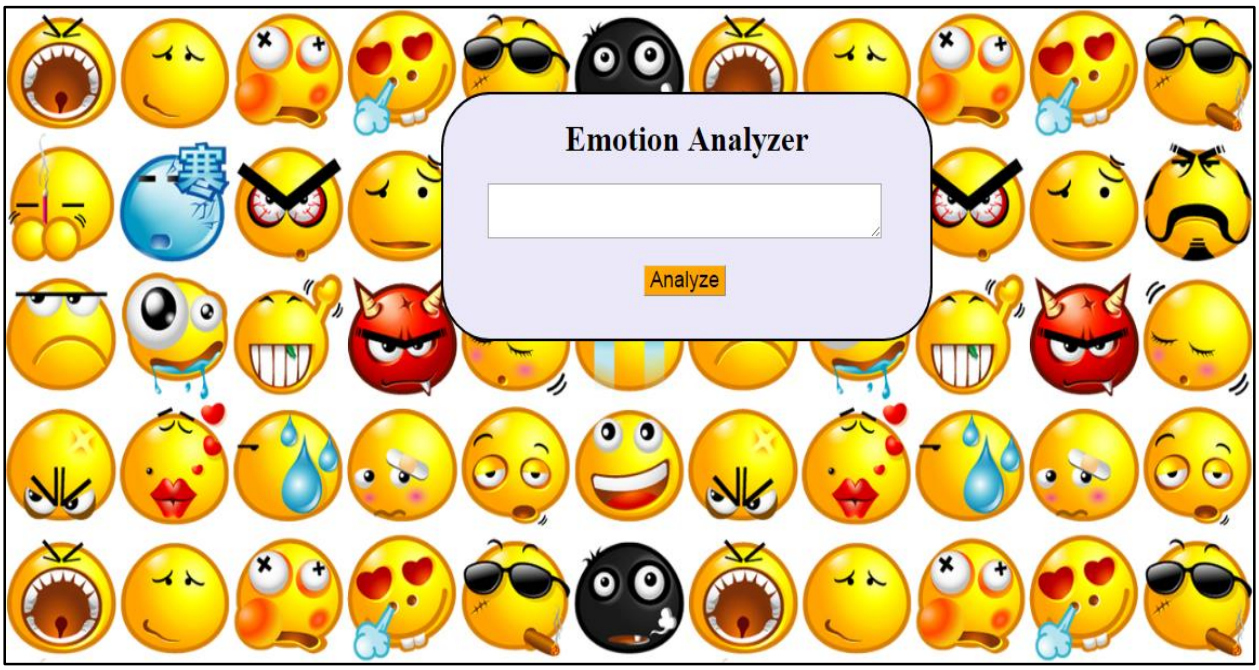

Figure: Blank UI

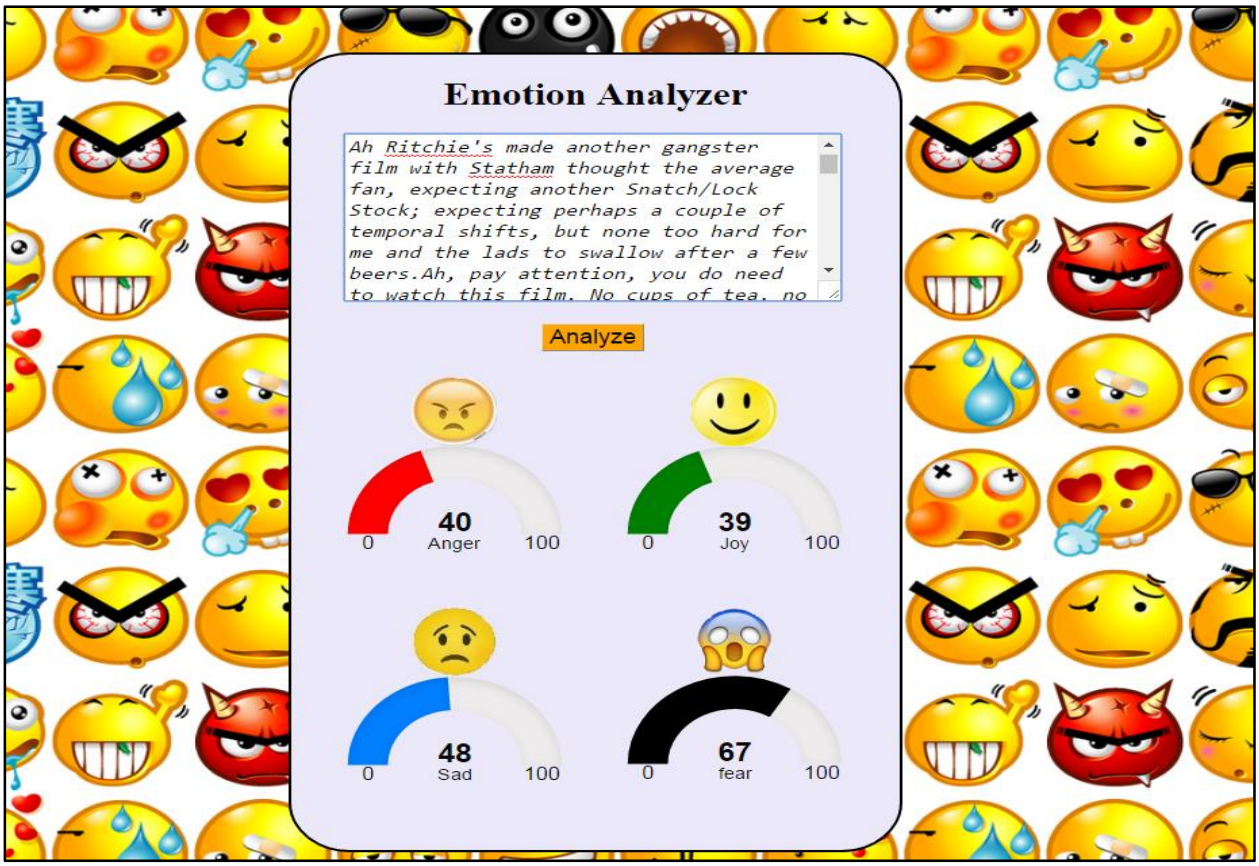

Figure: UI with paragraph as input & output

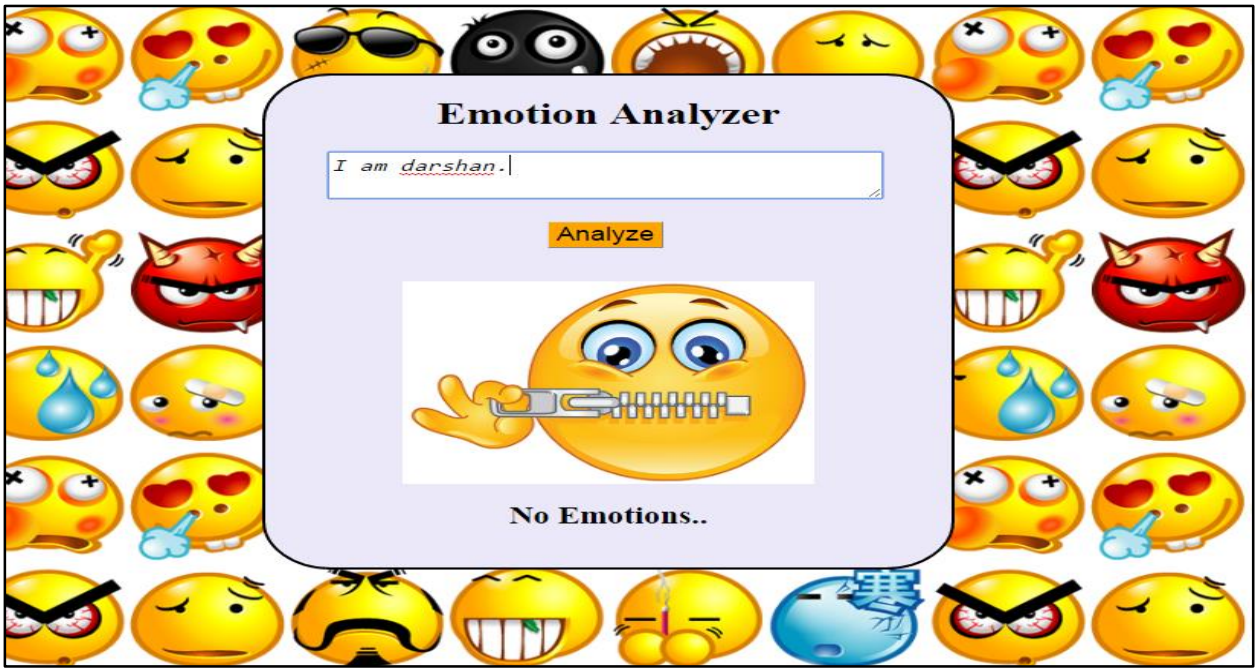

Figure: UI with sentence having no emotions

8. Deployment Model

We are using Apache Tomcat 7 webserver. We need to deploy our project '.war' file to web server. Along with the web server we need to set up a database server too. We are using MySQL database server. MySQL should run over port number 3306 with username as 'root' and password as 'root'. We need to set up library of words into MySQL. For that we are providing sql queries along with our project. User needs to import sql database backup to MySQL using MySQL query browser; the database will be automatically setup to the client side. Later on the user can run and use the system by URL as:

http://[webserver_ip_address]:[webserver_port_number]/EmotionAnalyzer/home1

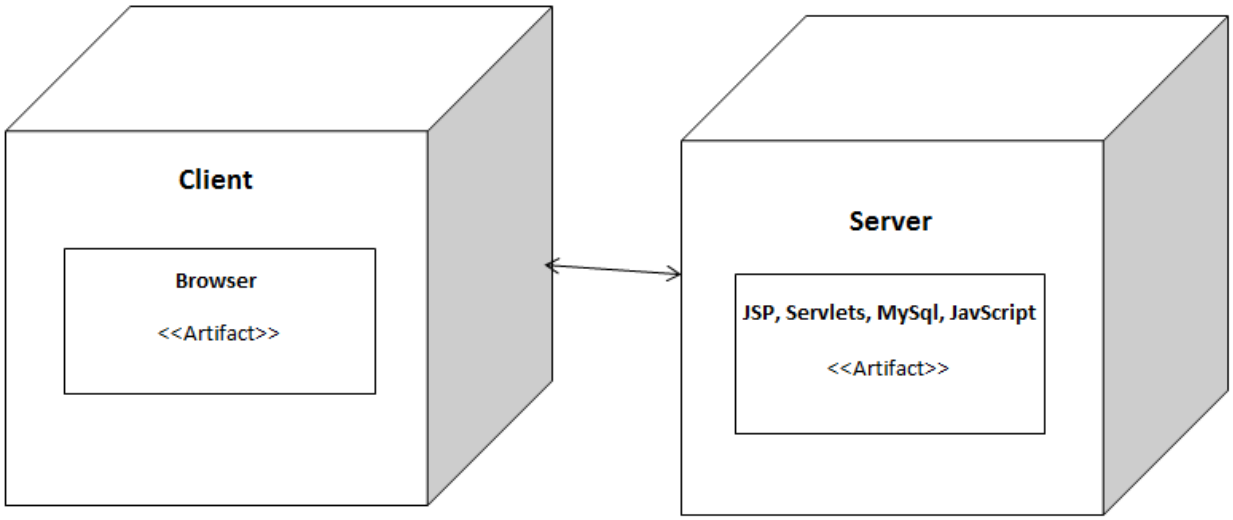

Figure: Deployment diagram

Deployment Package will contain following materials:

- Emotion Analyzer war file
- .sql database backup file
- User guide (setup help)

9. Browser compatibility

The web pages supported in most major browsers.

The browsers supported for the web pages are as follows:

- Firefox 45.0 and later
- Google Chrome 48.0 and later
- Opera 36.0 and later
- Internet Explorer 10 and later

The web pages are supported in desktop as well as mobile browsers.

10. Emotion Analyzer and other analyzers

There are several analyzers already exist in market like “Tone analyzer by IBM Watson”, “NLTK by Stanford”, “Theysay”, etc. Those analyzers just do the word matching. Some of them are only sentiment analyzers which just predict about the sentiments on the scale of positive, negative or neutral. Some predict about the emotions but give incorrect output.

Let’s see a case of NLTK, it is a semantic analyzer. They just check for sentiments. Accuracy is also very less. In many cases it will result as neutral even though positivity or negativity exists. When it doesn’t know result in such case it will return with result as neutral. Polarity value is also inaccurate.

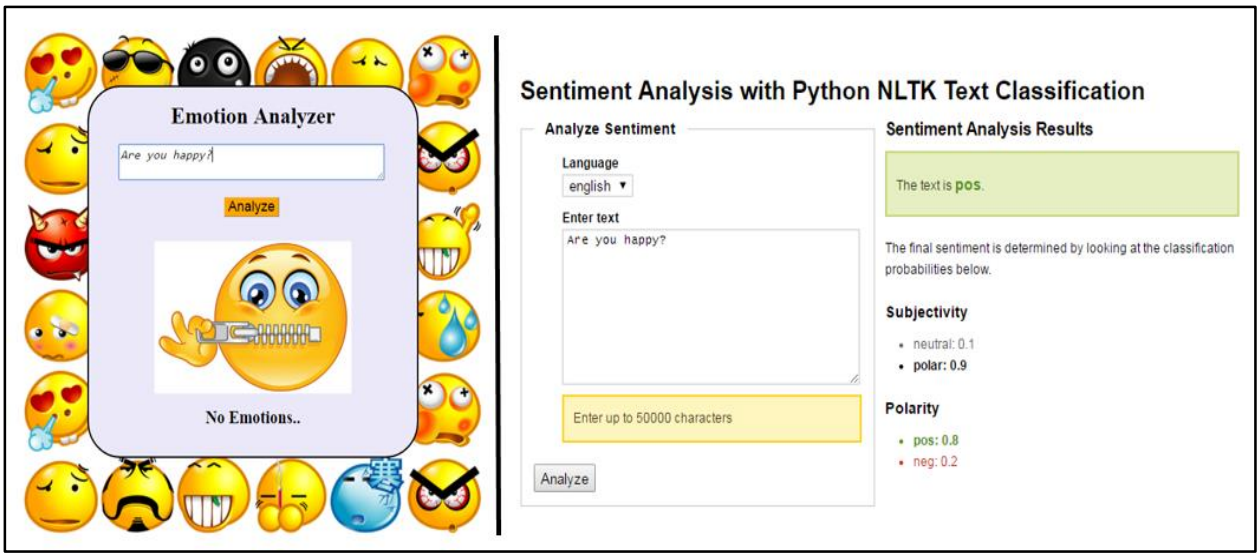

Figure: Emotion Analyzer | NLTK

Theysay, is one of the sentiment analyzer with better accuracy. But the major issue is, it just matches the word and predicts output. Fails for complex sentences. Sometimes fails for basic sentences also.

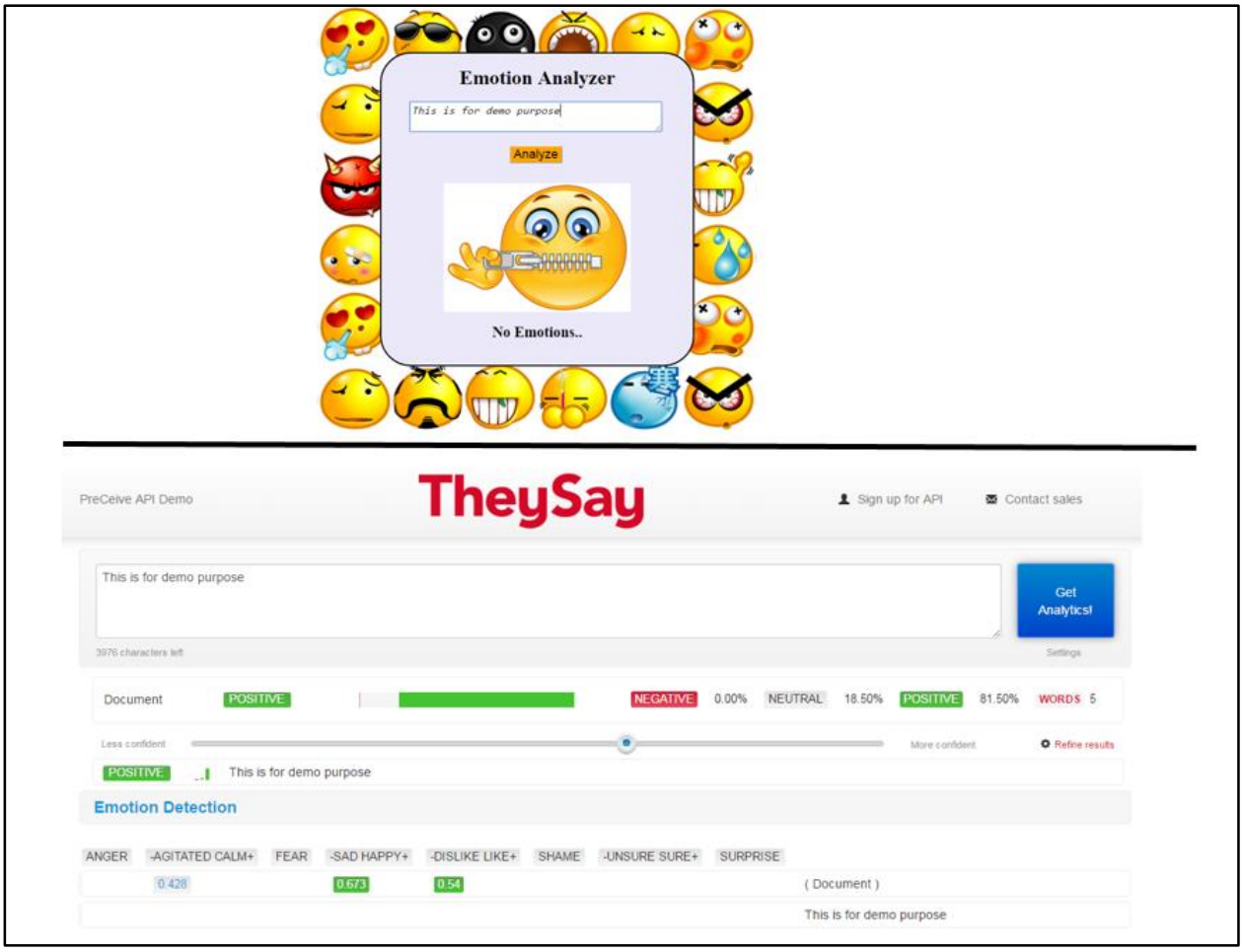

Figure: Emotion Analyzer | TheySay

IBM Watson's Tone analyzer is the one of the best analyzer, which already exists in market. But there are several issues with tone analyzer such as if result is not known it results any random output. As the sentence doesn't have any emotions, in such cases it will give any random output. It fails for controversy sentences like 'None of the projects were good', it will result as Joy which is absolutely wrong. It fails for some basic sentences also like as 'I will fire you', basically sentence suggests anger, where tone analyzer is predicting fear. Inaccurate result for emoji.

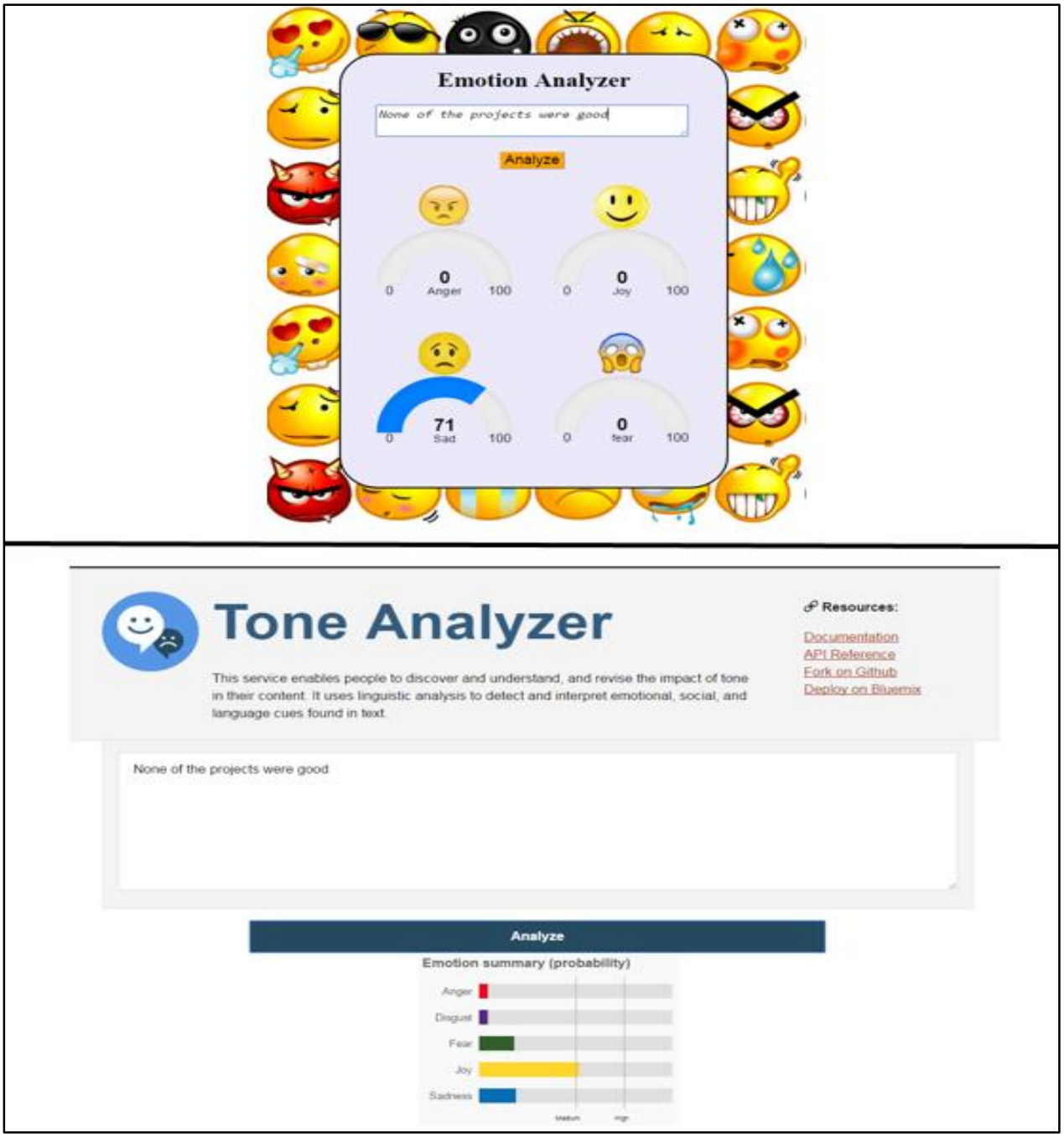

Figure : Emotion Analyzer | IBM Watson's Tone Analyzer

On the other hand, “Emotion Analyzer” is pretty good than all of those. It predicts emotions along with sentiments. Currently supports 4 types of emotions as anger, fear, sad, joy. More accurate result than other. Auto learning as machine learning approach is implemented, accuracy is continuously improving. Supports emoji. Gives correct result in case of complex sentence, controversy sentences, negating sentences, emotionless sentences, etc. Not for comparison with others, but we have tested with

same huge data set IBM's tone analyzer and Emotion Analyzer, we are getting better result than tone analyzer. Sometimes IBM may fail and our system will give correct answer, sometimes it may be reverse, but overall result Emotion Analyzer is more accurate.

11. Pros and Cons

There are several pros and cons of the Emotion Analyzer. Those are as follows:

Pros:

- Predicts Emotions (fear, anger, sad, joy)
- Supports text based input (word, set of words, sentence, paragraphs, story)
- More accurate result
- Correct result in case of controversy sentences
- Works fine with huge input
- Auto learning (support machine learning)
- Reduces response time
- Better UI
- Simplified output in eye catchy style

Cons:

- Tested in restricted and supervised environment, so boundary area is very small
- If any synonym of word is not found in database then system fails
- Static emotion scores
- Fails in cases where same words have multiple meaning e.g., like (interest and comparison)