

text-based emotion analyzer

EMOtion Analyzer

**By**

**We Know Emotions!!!**

**Team**

Table of Contents

[1. Introduction 3](#_Toc450639544)

[1.1 PROBLEM STATEMENT 3](#_Toc450639545)

[1.2 EXISTING SYSTEM 3](#_Toc450639546)

[1.3 PROPOSED SOLUTION 3](#_Toc450639547)

[2. Technology 4](#_Toc450639548)

[2.1 WHAT IS Machine Learning AND WHY? 4](#_Toc450639549)

[3. System Architecture 5](#_Toc450639550)

[4. Software Requirements 8](#_Toc450639551)

[4.1 SOFTWARE USED 8](#_Toc450639552)

[4.2 TECHNOLOGIED USED 8](#_Toc450639553)

[5. Other Design Diagrams 9](#_Toc450639554)

[5.1 LOW LEVEL DESIGN 9](#_Toc450639555)

[5.2 CLASS LEVEL DIAGRAM 11](#_Toc450639556)

[6. User Interface Design 12](#_Toc450639557)

[7. Software Design 14](#_Toc450639558)

[7.1 CODE EXPLANATION 14](#_Toc450639559)

[7.1.1 Controller 14](#_Toc450639560)

[7.1.2 Worker 14](#_Toc450639561)

[7.1.3 Rules 15](#_Toc450639562)

[7.1.4 HomePage.jsp 16](#_Toc450639563)

[6.2 LOGIC 17](#_Toc450639564)

[7.3 DATABASE 18](#_Toc450639565)

[7.3.1 What is MySQL? 18](#_Toc450639566)

[7.3.2 Why to use MySQL? 18](#_Toc450639567)

[7.3.3 Configuration steps 18](#_Toc450639568)

[8. Testing 20](#_Toc450639569)

# 1**. Introduction**

## 1.1 PROBLEM STATEMENT

How do you think your favorite brands or websites learn, what you think about their products or services? Sure they take our comments and feedbacks, but who does the analysis? Is there a person sitting and wasting precious time analyzing millions and millions of text data? Absolutely No, it would be ineffective.

Won’t it be cool if machines could do analyze emotions from text for us? There comes **Machine Learning**. That’s what our project aims at doing to build a better text-based emotion analyzer.

## 1.2 EXISTING SYSTEM

In the present scenario, There are several analyzers already exist in market like “Tone analyzer by IBM Watson”, “NLTK by Stanford”, “TheySay”, etc. Some of them are only sentiment analyzers which will just predict about the sentiments on the scale of positive, negative or neutral. Some predict about the emotions but have less accuracy, and even sometimes give incorrect output.

## 1.3 PROPOSED SOLUTION

The project aims at making a more accurate text based analyzer, by involving the use of Machine Learning. Here in this, our software able to tell the emotions of the input text. Input text can be in the form of paragraphs, words and smileys. Emotions, here, is divided into sad, anger, fear and joy.

# **2. Technology**

## 2.1 **WHAT IS** Machine Learning **AND WHY**?

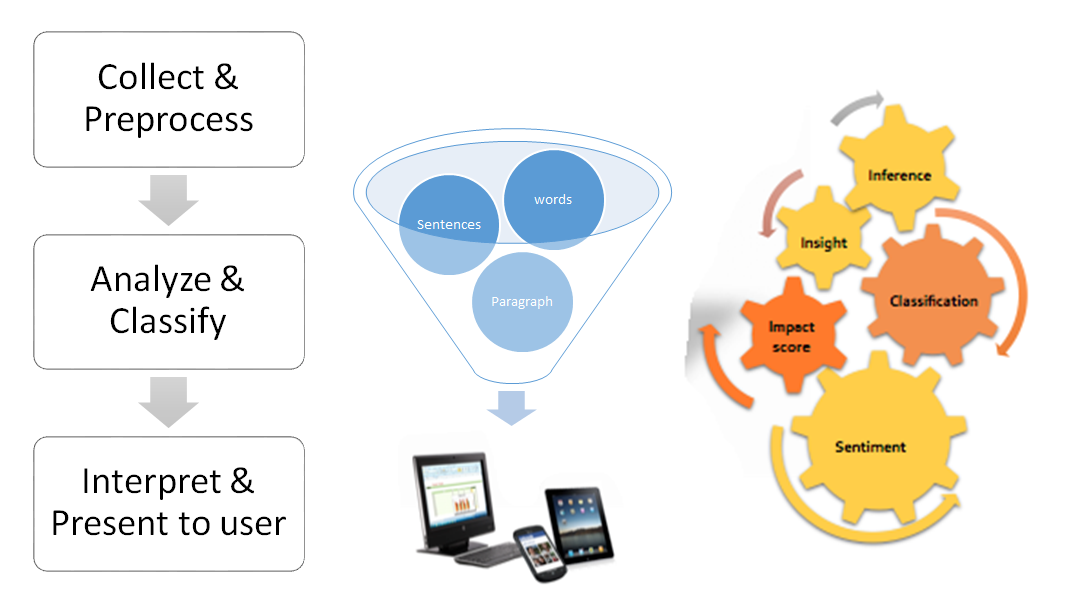
Machine learning is a subfield of [computer science](https://en.wikipedia.org/wiki/Computer_science) that evolved from the study of [pattern recognition](https://en.wikipedia.org/wiki/Pattern_recognition) and [computational learning theory](https://en.wikipedia.org/wiki/Computational_learning_theory) in [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence).

Machine learning studies computer algorithms for learning to do stuff. We might, for instance, be interested in learning to complete a task, or to make accurate predictions, or to behave intelligently. The learning that is being done is always based on some sort of observations or data, such as examples (the most common case in this course), direct experience, or instruction. So in general, machine learning is about learning to do better in the future based on what was experienced in the past.

In **text-based analysis**, it is quite impossible to have every word to be stored in database. In our project, we have built a rich dictionary of words along with emotions and score. But sometimes it may be possible that it may encounter a new word. This problem is solved by using Machine Learning. We developed a new algorithm that solves this issue.

The algorithm adds the new word into the database along with score.

# **3. System Architecture**

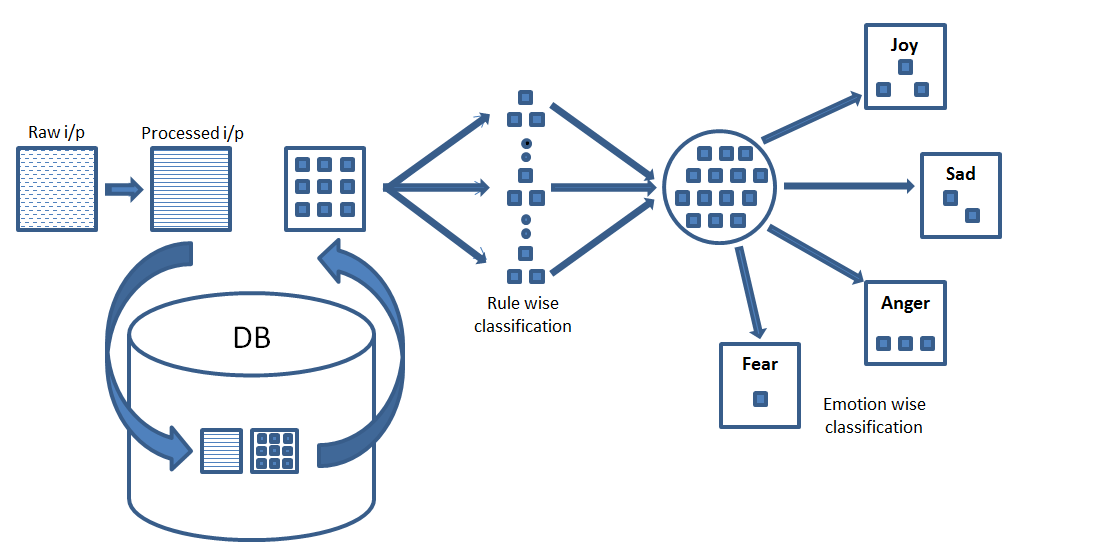
As in figure, “Emotion Analyzer” systems will collect the data from user which will be preprocessed and will be passed to the system as input; system will analyze the input and classify the data as per emotions in several categories. Output from the system is interpreted by the browser, which will be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

**Figure: Architecture - Emotion Analyzer**

As we have taken data that is to be analyzed by the user is treated as raw input to the system. We processed that raw data by applying POS tagger from coreNLP of Stanford NLP library to get part of speech (POS) tagging to each word of input data. That preprocessed data is treated as actual input to our system (Emotion Analyzer). As we have analyzed near about 15000~20000 sentences during the development time and build 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 pass to the database and will search for any one from 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 actual input to the system goes to database, search for the words given in input into our database 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, we have used decision tree algorithm for classification. We have used classification algorithm twice throughout the code. First time, we are doing classification as per rules that we have designed which follows 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 as per rules. After 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 have done classification second time, 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. Software Requirements**

## 4.1 SOFTWARE USED

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

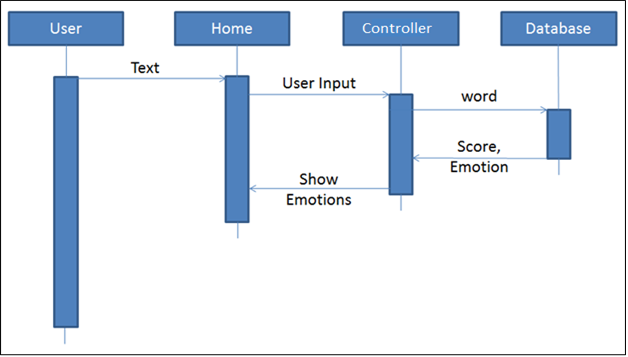
## 4.2 TECHNOLOGIED USED

* HTML5
* CSS3
* JavaScript
* JSP/Servlet

# **5. Other Design Diagrams**

## 5.1 LOW LEVEL DESIGN

As user enters text data on webpage, from where input is passed from webpage to controller.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 HomePage.jsp webpage which will call gauge.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 Controller.java. Then, Controller.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 HomePage.jsp webpage which will display result to user.

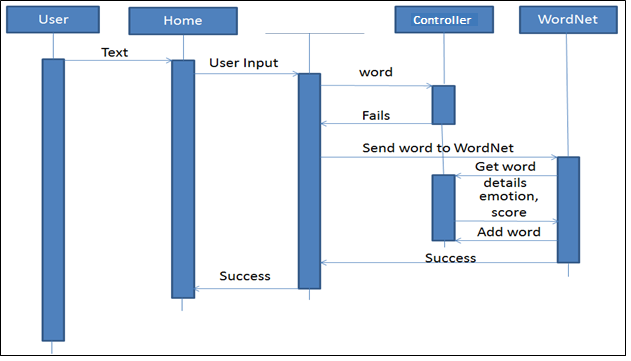


Figure: Sequence diagram (when word not found in database)

## 5.2 CLASS LEVEL DIAGRAM



Figure: Emotion Analyzer – Class level diagram

# 6. User Interface Design

6.

The following frameworks are used to design the UI:

* CSS3
* HTML5
* JavaScript

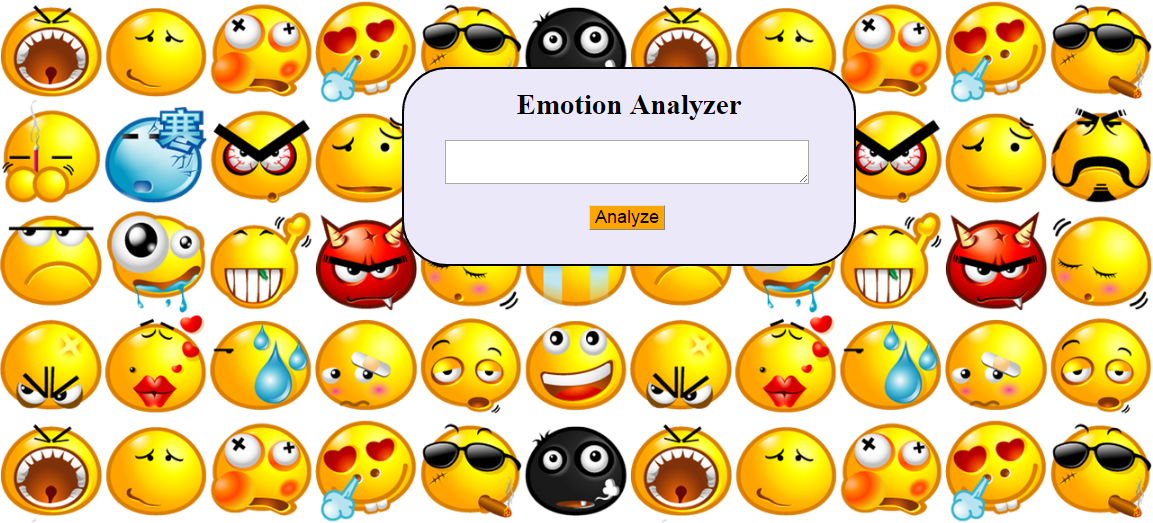


Figure: Blank UI



Figure: UI with paragraph as input & output

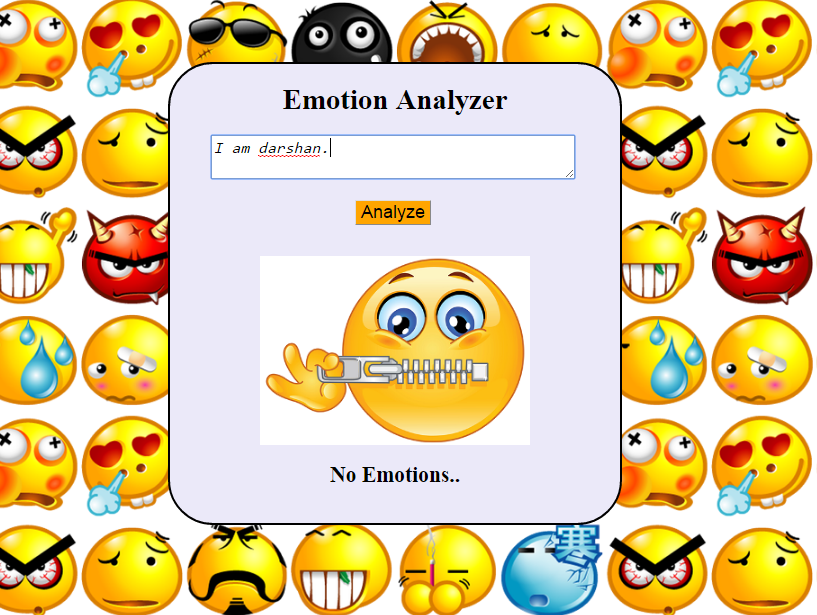


Figure: UI with sentence having no emotions

# 7. Software Design

## 7.1 CODE EXPLANATION

The code contains 3 packages: controller, worker and rules.

### 7.1.1 Controller

Package contains the controller.java which works as bridge to view portion to actual working server. It receives request from the user i.e. HomePage.jsp and as per request and conditions, it forwards to respective worker. It works as like as receptionist in any firm. As any user reaches to him/her, he/she just listens and navigates user in appropriate direction.

### 7.1.2 Worker

Controller passes the control to worker to perform actual work. Worker actually performs tasks and returns result to controller. There are several workers that are developed in system as Database connection generator (DBConnector.java, DBConnectorImpl.java), POSTagger (POSTag.java), WordNet dictionary (WordNet.java), Logger (CreateLog.java) & Output Generator (OutputGenerator.java, OutputGeneratorImpl.java). Each worker does its specific task which is assigned to him.

* Database connection generator:

There is DBConnector.java interface which is further implemented by DBConnectorImpl.java. This worker generates connection with database and returns java.sql.Connection object. It contains details about database URL, port number, username and password.

* POSTagger:

POSTagger possesses the main functionality of the system. POSTag.java is the java code to implement POSTagging of Stanford NLP library. It tags each word of sentence as per the parts of speech which are further used to identify emotions.

* WordNet Dictionary:

WordNet is the dictionary which contains most of the words of the English language which is used in case when the word not present in database, to search the synonym of that word which is present in database. Java code to implement this entire algorithm is capsulate in WordNet.java. It searches for the synonym of the word which is present in database and adds that new word to database by picking emotion and score value from its relative synonym present in database.

* Logger:

Logger keeps the log of each input given by user to system, output generated by system and error logs to a text file on server machine. Code for this functionality is encapsulated in CreateLog.java.

* Output Generator:

Output generator is the worker who performs in final stage of actual working of each request. It generates response message for client side system as per the output generated by all other workers. For that, OutputGenerator.java is interface which is further implemented by OutputGeneratorImpl.java to fulfill this functionality.

### 7.1.3 Rules

Rules are developed to help workers to perform their tasks. Rules which follow Basic English language. Rules are defined with pre-requisite conditions which are to be satisfied by the request. When condition matches, worker will take steps which are coded in respective rules. There are several types of rules; to check negative condition (NegationRule.java), to check quantification condition (QuantifierRule.java), to clean the word (WordClean.java), to check but conditions (ButRule.java).

* WordClean.java:

WordClean rule is basically designed for to clean the word via some former symbols may be present in or along with word like as comma (,), full stop (.), apostrophe ('), hyphen (-), semi-colon (;). These symbols may create problem while finding words in dictionary or database, in such cases it is important to remove such symbols. For that purpose, this rule is defined.

* NegationRule.java:

Negation rule is the one of the most important rule out of all. Because few negation words in the sentence can change the mood completely by 360 degree. To trace such conditions and perform respective actions in such conditions, negation rule is defined.

* QuantifierRule.java:

Quantifier rule is also equally important as negation rule because quantifier words like very, much, more, short, little, most, etc. perform scaling operation on emotions. To get more accuracy, systems need to check such conditions and perform actions. For that purpose this quantifier rule is defined.

* ButRule.java:

Many times, words like but, none the less, in fact, etc. change the meaning of previous or current sentence with respect to other. These words impact not only on the current sentence but also on others. To identify such conditions and sentences on which they impact and perform actions, this but rule is defined.

### 7.1.4 HomePage.jsp

HomePage.jsp is the part of view side for user interface part. User gives input to the system from this HomePage.jsp. This page generates request and sends it to controller on server side. Controller sends back response to HomePage.jsp which is read and graphically displayed to the user.

|  |  |  |
| --- | --- | --- |
| **Libraries Used** | **Used in** | **Purpose** |
| Stanford Core NLP | public static String stemming(String word) :  defined in POSTag.java under worker package. | Stanford core NLP lib is used for stemming the word. Stemming means getting root word of any word for example bank is root word for banking. |
| WordNet | WordNet.java under worker package | WordNet is open source lexical database for English. In WordNet words are organized in structure form and it also defines relation between words such as synonym, antonyms, ISA relation. |
| RiTa Wordnet(rita.jar) | Imported in WordNet.java under worker package | WordNet is only lexical database but to access the WordNet we have to take help of one of the lib i.e. rita.jar. It provides functions to access WordNet db. |
| justgauge.js | HomeBar.jsp under WEB-INF package | It is java script lib. It is used to show the result in more interactive manner i.e. in the form of gauge. |

Table: Emotion Analyzer –Libraries Used

## 6.2 LOGIC

Figure: Emotion Analyzer – Logic Diagram

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.

First, the sentences will be split into words. Then some rules are applied:-

1. Now system will check for quantifier words like very, little etc. and if present set quantifier flag true.

If a sentence contains quantifier words like more, most, very, higher, less etc., system catches such words and modifies the score as per the position of words present on first/second part.

2. If negating words like not, neither, never, none, no one, nothing, etc. are found and set neg flag to true.

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.

3. If any special symbol is present, then remove those symbols from sentences.

If a sentence contains question mark that means, input is question. Question basically doesn’t possess any emotions 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’.

Then words are checked that if present in database. If present, emotion along with score are taken for that word and final calculation is done.

One major problem in this approach is:

We are searching for words into database. English language has 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 the word is not found in database, then system will find 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 found at least one synonym in database from where we got 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 found that word directly into database. So the database keeps improving automatically and that 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 found any synonym in databases. 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.

## 7.3 DATABASE

### 7.3.1 What is MySQL?

MySQL is an open source relational database management system ([RDBMS](http://searchsqlserver.techtarget.com/definition/relational-database-management-system)) based on Structured Query Language ([SQL](http://searchsqlserver.techtarget.com/definition/SQL)).

MySQL runs on virtually all platforms, including [Linux](http://searchenterpriselinux.techtarget.com/definition/Linux), [UNIX](http://searchenterpriselinux.techtarget.com/definition/Unix), and [Windows](http://searchwindowsserver.techtarget.com/definition/Windows).

### 7.3.2 Why to use MySQL?

The MySQL database server provides the ultimate in scalability, sporting the capacity to handle deeply embedded applications with a footprint of only 1MB to running massive data warehouses holding terabytes of information.

MySQL can meet the most demanding performance expectations of any system.

### 7.3.3 Configuration steps

**Application Server:**

1. Configure Apache Tomcat 7 webserver.
2. Deploy project war file to web server.

**MySQL Database:**

1. Configure MySQL Database server. It should run over port number 3306.
2. Set username as **root** and password as **root**.
3. Import sql database backup to MySQL using MySQL query browser.

Later on you can run and use the system by URL as:

[*http://[webserver\_ip\_address]:[webserver\_port\_number]/EmotionAnalyzer/homepage*](http://[webserver_ip_address]:[webserver_port_number]/EmotionAnalyzer/homepage)

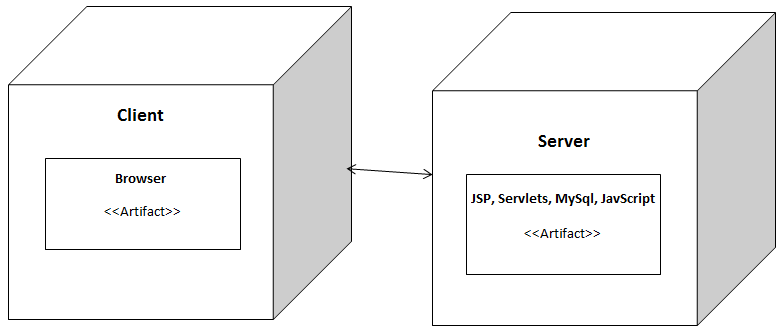


Figure: Deployment diagram

Deployment Package will contain following materials:

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

# 8. Testing

We have designed test data of database very intelligently, though there are 1-2 % chances that we would not found any synonym in databases. 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.

For testing purpose you can refer text file, named as **sentences.txt**. The text file is attached along with the code.