A PROJECT
ON

Scribe - Natural Language Understanding and Generation Engine

BY

Vandita Shah B-827
Sumer Shende B-834
Vineet Trivedi B-848
Rishabh Vig B-855

Under the guidance of

Internal Guide
Prof. Dnyaneshwar Dhangar

MCT
MANJARA CHARITABLE TRUST
RAJIV GANDHI INSTITUTE OF TECHNOLOGY
Juhu-Versova Link Road Versova, Andheri(W), Mumbai-53

Department of Computer Engineering

University of Mumbai

April – 2016

MANJARA CHARITABLE TRUST
RAJIV GANDHI INSTITUTE OF TECHNOLOGY

Juhu-Versova Link Road Versova, Andheri(W), Mumbai-53

Department of Computer Engineering

CERTIFICATE

This is to certify that

1. Vandita Shah B-827
2. Sumer Shende B-834
3. Vineet Trivedi B-848
4. Rishabh Vig B-855

Have satisfactory completed this project entitled

"Scribe - Natural Language Understanding and Generation Engine"

Towards the partial fulfillment of the

BACHELOR OF

ENGINEERING IN

(COMPUTER ENGINEERING)

as laid by University of Mumbai.

Guide
Prof. Dnyaneshwar Dhangar

H.O.D.
Dr. S. B. Wankhade

Principal
Dr.Udhav Bhosle

Internal Examiner

External Examiner

Project Report Approval for B. E.

This project report entitled “Scribe - Natural Language Understanding and Generation Engine” by Vandita Shah, Sumer Shende, Vineet Trivedi and Rishabh Vig is approved for the degree of ***Bachelor of Computer Engineering***.

External Examiners

1.----------------------------------

2.----------------------------------

Internal Examiner

1.----------------------------------

2.-------------------------------

HOD

Date:
Place:

DECLARATION

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

____

____

Date:

ACKNOWLEDGEMENT

We wish to express our sincere gratitude to Dr. U. V. Bhosle, Principal and Dr. S. B. Wankhade, H.O.D of Computer Department of RGIT for providing us an opportunity to do our project work on “Scribe - Natural Language Understanding and Generation Engine".

This project bears on imprint of many peoples. We sincerely thank our project guide Prof. Dnyaneshwar Dhangar for his/her guidance and encouragement in carrying out this synopsis work.

Finally, we would like to thank our colleagues and friends who helped us in completing the Project work successfully.

ABSTRACT

"SCRIBE" is an AI driven system which automatically generates output in a user defined format based on raw data that is fed to it. It is a tedious and progressively difficult task for a human to process large volumes of data and deliver meaningful output in a scenario where things are changing by the minute. Thus "SCRIBE" reorganizes raw data into furnished output as required by the user whether it be in the form of an essay, article, post or letter. "SCRIBE" collects raw data and stores it in a file system. It applies adequate business rules for data cleaning, standardization and mining.

The polished data is then formatted as per user requirement using narrative science. In essence "SCRIBE" approaches the highest epoch of artificial intelligence to automate creative human tasks like writing.

TABLE OF CONTENTS

TABLE OF FIGURES

CHAPTER 1
INTRODUCTION AND MOTIVATION

1.1 FOREWORD

"SCRIBE" is an AI driven system which automatically generates output in a user defined format based on raw data that is fed to it. It is a tedious and progressively difficult task for a human to process large volumes of data and deliver meaningful output in a scenario where things are changing by the minute. Thus "SCRIBE" reorganizes raw data into furnished output as required by the user whether it be in the form of an essay, article, post or letter. "SCRIBE" collects raw data and stores it in a file system. It applies adequate business rules for data cleaning, standardization and mining.

The goal of the system is to design and build software that will analyze, understand, and generate languages that humans use naturally. For this we use the concept of Natural Language Processing (NLP), which is a field of computer science concerned with interactions between computers and human languages.

1.2 INTRODUCTION TO SCRIBE

The system takes raw data as input and converts it into sentences that may form a small article. But before accepting the raw data the system is taught to understand basic English language syntaxes and the structures of sentences by analyzing various text documents and find links between commonly used words. This is done so that the system knows the difference between various parts of speech. It helps the system break the sentences into elemental components which it saves in the database as links. These links exist between a word and its successive word. Ultimately each word will have a list of words that could follow it, which in turn comes in handy while forming sentences as the system will consider all possible successive words and select the best one based on various factors such as tense, tone etc. Once the system has analyzed the text documents it can efficiently segregate the data in a way that is useful for sentence formation. This entire process is carried out by following the concepts of natural language processing.

After understanding the proper grammar syntaxes and forming the links between words, we move on to the part where the user gives some raw data as input, this could be a set of random words all related to some specific topic. SCRIBE takes this input and finds the link for each word from the input in the database, it then checks the probability of the word and its successive word so as to select the appropriate word for the final sentence. This is done by using tokens and lexical analysis which are a sub topic in natural language generation. The final output would consist of a set of sentences made on the basis of the data entered by the user.

1.3 MOTIVATION

The main motivation behind the project was to eliminate human delay from the process of effective communication. During natural disasters chances of human delay are high and the need to communicate the disaster and its consequences is of paramount importance. Abnormal readings from instruments like seismometers, Saffir-Simpson scale and stream gauges indicating a natural disaster like earthquake, hurricane or flood respectively should be communicated to Scribe. Scribe can then write a piece informing the people of a natural disaster. Scribe can be one of the fastest modes of communicating the occurrence of a natural disaster, the area it has occurred in and its possible impact.

CHAPTER 2
PROBLEM STATEMENT

It is a repetitive and dynamically difficult errand for a human to process substantial volumes of raw data being perceived continuously from different domain, understand and interpret it while creating large amounts of text compositions which are syntactically correct and convey the logic preserving the original meaning.

Scribe is made up of two main modules – Learning modules based on natural language processing; which reads data from thousands of well drafted articles, understands the syntax and stores this information in a knowledge base. This knowledge is later used for NLG to obtain rich output compositions.

The second module of Scribe is the natural language generation module which consists of two main components: the first carries out Analysis and Interpretation, and the second is responsible for Information Delivery. The Analysis and Interpretation stage derives facts and insights from the raw input data and turns them into basic information chunks called messages. The Information Delivery stage works out how to best communicate the information in these messages as a coherent text.

This system revamps crude information into text articles as required by the user. It gathers raw data from sources like:-

1. RSS feeds of multiple websites
2. Social media plug roots
3. Blogging stations
4. News Tickers

For this, Scribe needs to be able to read text from different formats like XML, LateX along with grabbing the correct unformatted text from html documents.

The three key advantages of Scribe-produced text are scalability, tailor ability and consistency. It can produce personalized text at a rate that is simply unattainable using human authors, and the quality of NLG texts does not vary in the way that a human's might. Scribe is ideal for anyone who-

- has data that needs to be interpreted in order to make it actionable
- wants to obtain quick text composition from raw data
- needs to make fast decisions based on data
- has too much data for a human expert to understand and interpret
- requires consistency in their text output
- wants to make their data accessible to people from different domains.

For the implementation of this system, we require a large database that is used to store the entire word POS relations obtained from scanning diverse documents during the NLP phase. This data is stored in an unstructured relational form to make the data fetch more efficient.

In the NLG phase, it is crucial that the system understands the raw data. This can be done by referring its knowledge base to find relevant connections data items and reproduce them to formulate logical sentences. The NLG then should be able to produce grammatically correct sentences using language libraries and finally produce good quality reports.

Lastly, the user should be able to change the text into different formats. User should also be able to give a feedback to the system. This feedback will help the system to calculate its efficiency and check if it is giving satisfactory outputs. Feedback also needs to register in the knowledge base for future use by Scribe.

Proposed System:

We propose an automated system ‘SCRIBE’, based on the concept of robotic journalism as seen at various news organizations, which uses the concepts of artificial intelligence. The system can generate articles and information on a particular event, cricket being the case in point we intend to use here. The system will accept raw data, and accordingly process the provided data and provide the user with an entire full-fledged statements on the same event using the data.

We propose to devise a system that is divided into two main aspects. First being the aspect that deals with reading sample statements to understand the formation of sentences, devise structures, form links between the words, store the links in linked databases that will be further used while forming sentences. The second aspect of the project deals with generation of sentences as the output of the system. In this phase the user will enter the raw data, the system will check the raw data with the databases, use the possible links in the databases and select the most apt link using a supervised learning algorithm, like statistical learning algorithm. Using this algorithm, the sentence can be devised accordingly

Dividing the system into two main functionalities, we intend to establish an organized flow, which involves first learning sentence structure, tokenization, syntax analysis, semantic analysis, devising links and storing them, then learning from the above exhaustive procedure to use the simple raw data to make sentence structures.

We shall limit the domain of this system to the topic of cricket. We shall provide this system with a limited set of statements to devise sentence structures based on their contexts to again limit the scope of possible types of sentences.

When the system receives simple raw data such as “Sachin”, “60 runs” “38 balls”, the system will now look at the word Sachin as a noun, similarly it will find parts of speech for each of the raw words, go back to the database and based on a supervised learning algorithm, it will choose the optimum set of links for the words and devise a sentence accordingly.

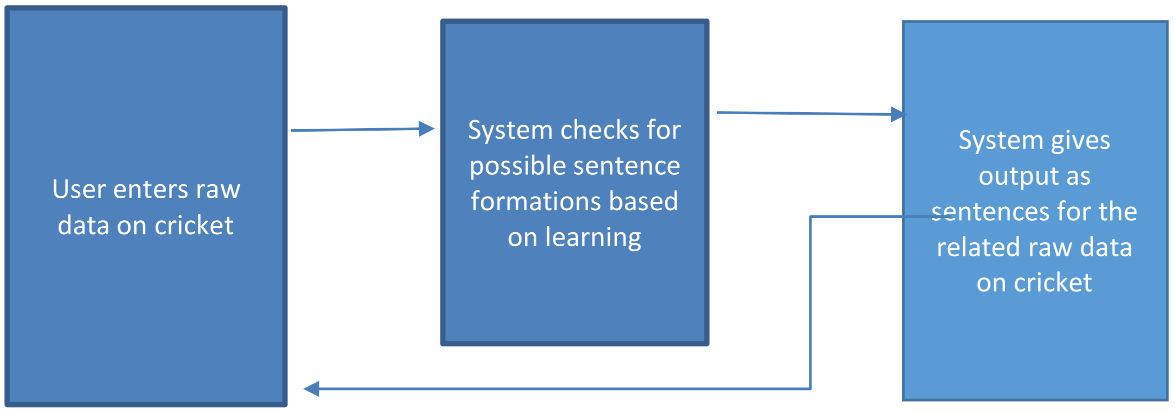

User enters raw data on cricket
System checks for possible sentence formations based on learning
System gives output as sentences for the related raw data on cricket

Fig 2.1 block diagram for basic proposed system

CHAPTER 3

LITERATURE SURVEY

The research work in the natural language processing has been increasingly addressed in the recent years. The natural language processing is the computerized approach to analysing text and being a very active area of research and development. The literature distinguishes the main application of natural language processing and the methods to describe it. For our project, we have reviewed multiple papers, but among these our prominent ideas come from the following papers:-

1) A Robust Human-Robot Communication System Using Natural Language for HARMS (The 12th International Conference on Mobile Systems and Pervasive Computing, MobiSPC 2015)
2) SimpleNLG: A realisation engine for practical applications. (Proceedings of the 12th European Workshop on Natural Language Generation)
3) Collaborative Writing Support Tools on the Cloud (IEEE Journal on Learning Technologies, VOL. 4, NO. 1, March 2011)
4) Discourse strategies for generating natural-language text(Artificial Intelligence (Elsevier, Volume 27, Issue 1))
5) Natural Language Generation as Incremental Planning Under Uncertainty: Adaptive Information Presentation for Statistical Dialogue Systems (IEEE/ACM Transactions on Audio Speech And Language Processing, VOL. 22, NO. 5, May 2014)
6) Big Data and the SP Theory of Intelligence (Digital Object Identifier 10.1109/ACCESS.2014.2315297)

7) Using Wikipedia and Conceptual Graph Structures to Generate Questions for Academic Writing Support (IEEE Transactions On Learning Technologies, VOL. 5, NO. 3, July-September 2012)

- Natural language processing for knowledge generation:

  This is based on the text obtained from various documents and social media, which is the first input into the system. [1] [5] It uses this high level modules for text analysis. It uses the sentence segmentation which deals with punctuation marks with a simple decision tree. There are several main techniques used in analysing natural language processing. Some of them can be briefly described as follows.

- Pattern matching:-

  Idea here is to interpret input word utterances as a whole father than building up their interpretation by combining the structure and meaning of words or other lower level constituents. That means the interpretations are obtained by matching patterns of words against the input utterance. For a deep level of analysis in pattern matching a large number of patterns are required even for a restricted domain. This problem can be ameliorated by hierarchical pattern matching in which the input is gradually canonicalized through pattern matching against sub-phrases. Another way to reduce the number of patterns is by matching with semantic primitives instead of words. [1] [5]

- Syntactically driven Parsing:-

  Syntax means ways that words can fit together to form higher level units such as phrases, clauses and sentences. Therefore syntactically driven parsing means interpretation of larger groups of words are built up out of the interpretation of their syntactic constituent words or phrases. In a way this is the opposite of pattern matching as here the interpretation of the input is done as a whole. Syntactic analyses are obtained by application of a grammar that determines what sentences are legal in the language that is being parsed. [1] [5]

- Semantic Grammars:-

  Natural language analysis based on semantic grammar is bit similar to syntactically driven parsing except that in semantic grammar the categories used are defined semantically and syntactically. [5] Here the semantic grammar is also complex, formed of more than one phrases and inter-intra sentence parsing.

- Case frame instantiation:-

  Case frame instantiation is one of the major parsing techniques under NLP. It has some very useful computational properties such as its recursive nature and its ability to combine bottom-up recognition of key constituents with top-down instantiation of less structured constituents. [5]

Natural language Generation for text composition:

Automatic generation makes use of natural language processing techniques based on grammars syntax checking. It uses the context free grammars for representing syntax of that language through the spotlighting addition of automatic summarization including indexing, which extracts the gist of the phrase placing transcriptions in order to deal with Information retrieval and alternative generation syntax. [8] [10]

The simplest approach is to build a separate module for each task, and connect these modules via a one-way pipeline. In such an architecture, the content determination module first decides on all the messages to be included in the text; the discourse planning module then organises these messages into a discourse structure tree; and so on. From a pragmatic perspective the most common architecture in present-day applied NLG is a three-stage pipeline with the following stages:

- Text Planning:-

This stage combines the content determination and discourse planning tasks described above. This rejects the fact that in many real applications, it can be difficult to separate these two activities. [8] [10]

- Sentence Planning:-

Sentence planning combines sentence aggregation, lexicalization, and referring expression generation. This combination is not universally accepted in the NLG held. Nevertheless, most applied NLG systems have chosen to combine these three tasks into one stage, and that is the approach we will take in Scribe. [8] [10]

- Linguistic Realisation:-

The task of linguistic realisation involves syntactic, morphological, and orthographic processing. [8] [10]

CHAPTER 4
REQUIREMENT ANALYSIS

After doing an extensive research on the requirements of a computer running an NLP process we came up with the following hardware requirements:

1. SAN storage network interface.
2. Solid State Hard drive.
3. Computer for user interface.

Apart from the hardware we required:

1. An NLP Library
2. An NLU Library
3. A schema free database to store semi structured data
4. A repository of cricket articles to serve as training data
5. Sources to extract information for the input article
6. Java JRE 6+
7. Eclipse
8. Windows/Linux Operating Environment

CHAPTER 5

PROJECT DESIGN

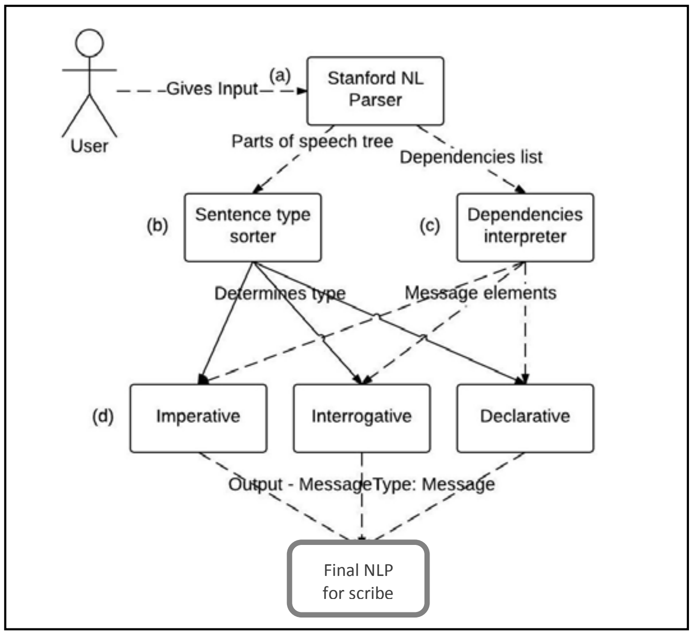


Diagram 5.1 Overall Design

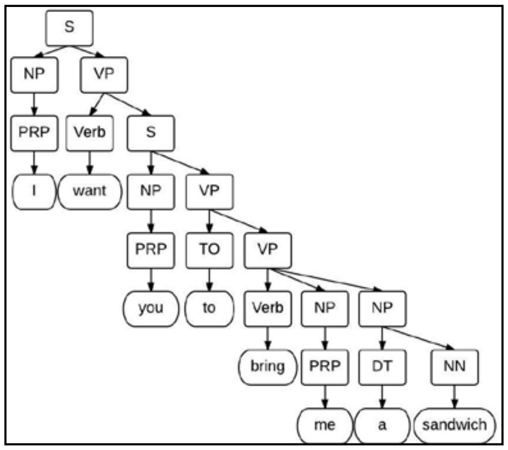


Diagram 5.2 Breakdown of sentences:

Input for token analysis

```
{ "Scribe", "ANALYSIS", "PROJECT", "'s",
   "Fourth-year", "group", "said", "they", "reached", "a", "tentative",
"deadline",
   "extending", "its", "deadline", "with", "professor", "XYZ", "to",
   "provide", "structural", "parts", "for", "SCRIBE", "'s", “Java",
   "Modules", "." };
```

Output of these words:

```
{ "NNP", "NNP", "NNP", "POS", "NNP", "NN",
   "VBD", "PRP", "VBD", "DT", "JJ", "NN", "VBG", "PRP$", "NN",
"IN",
   "NNP", "NNP", "TO", "VB", "JJ", "NNS", "IN", "NNP", "POS",
"CD", "NNS",
   "." };
```

5.1 Architecture of system

“A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.”

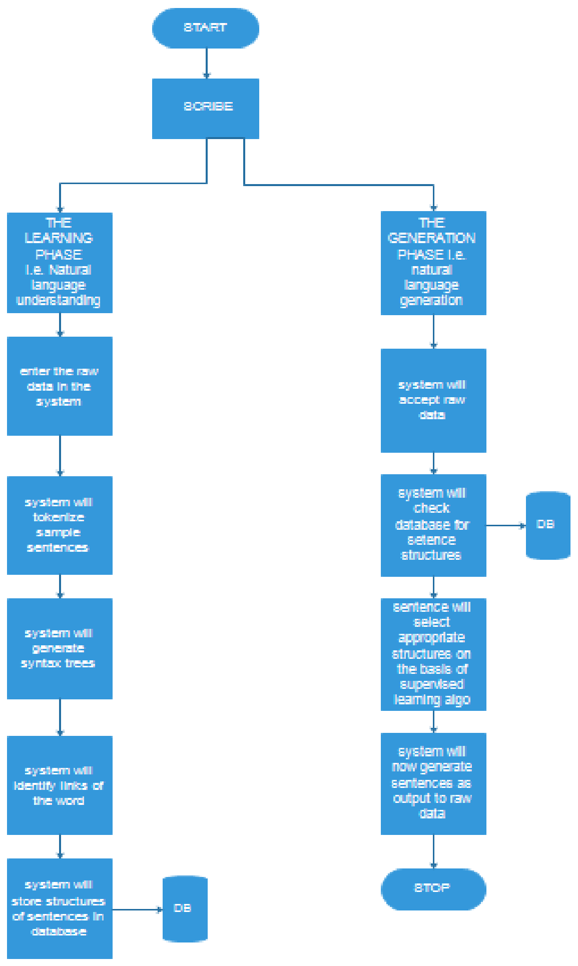

START
SCRIBE
THE LEARNING PHASE i.e. Natural language understanding
enter the raw data in the system
system will tokenize sample sentences
system will generate syntax trees
system will identify links of the word
system will store structures of sentences in database
DB
THE GENERATION PHASE i.e. natural language generation
system will accept raw data
system will check database for setence structures
DB
sentence will select appropriate structures on the basis of supervised learning algo
system will now generate sentences as output to raw data
STOP

Fig 5.3: Architecture of system

The architecture contains the internal and external entities as well as the central database to store the data. Based on the diagram, we know that a user will enter a set of information into the system; the information will be entered into the system, which will be be compared with the database for available sentences or sentence structures. Based on the available structures the NLG module will generate sentences. In the NLP phase, the the system will be feed with documents, these will be tokenized as per sentence structures and stored in the database, further used in the NLG phase

5.2 Modular design of system

“Modular design, or "modularity in design", is a design approach that subdivides a system into smaller parts called modules or skids that can be independently created and then used in different systems. A modular system can be characterized by functional partitioning into discrete scalable, reusable modules, rigorous use of well-defined modular interfaces, and making use of industry standards for interfaces.”

Besides reduction in cost (due to less customization, and shorter learning time), and flexibility in design, modularity offers other benefits such as augmentation (adding new solution by merely plugging in a new module), and exclusion.

Modular design is an attempt to combine the advantages of standardization (high volume normally equals low manufacturing costs) with those of customization. A downside to modularity (and this depends on the extent of modularity) is that low quality modular systems are not optimized for performance. This is usually due to the cost of putting up interfaces between modules.

5.2.1 Module 1: Parser

Takes the sentence as an input, generates POS tags, dependencies etc. the entire file of sentence structure undergoes data cleaning and is exported to MongoDB.

5.2.2 Module 2: MongoDB formatter

Here, the structure of the sentences storesd in mongoDB are organized into a required format.

5.2.3 Module 3: NLG Realizer

It is an important module of our system that deals with the input of the user. Accepting an input from the user, the module with process the different parts of the input sentences of the user, compare with the database for previous sentence structures and finally give us a fully formed sentence.

CHAPTER 6

IMPLEMENTATION DETAILS

6.1 System Implementation:

"The systems implementation is a process in terms of construction and delivery phases of the life cycle. Systems implementation is the construction of the new system and the delivery of that system into production. Every project is based on some hardware and software. Without this the project can't be completed."

Implementation is the realization of system specifications of an application through programming and deployment. System implementation consists of several tasks such as adopting a methodology, identifying the software, hardware and network requirements, planning and scheduling but the major task is realizing the technical specifications into a working product according to given requirements.

The project is divided into the following phases:

1. Learning phase
2. Knowledge Implementation phase

Phase 1:

In this phase we teach the system about the English language, the structure of sentences and characteristics of each word. The aim is that the system should develop a correct grammatical sense. We use open source libraries like open nlg and Stanford NLG for this purpose. These libraries help establish links between words and categorize the words into nouns, pronouns, verbs, adverbs etc.

Once the system is equipped to understand the language we feed data to it. Numerous texts related to cricket are fed into the system. The libraries then breakdown these articles into sentences and provide the structure of each sentence. Thus the system learns how sentences are formed, which sentence structure is used most often and under what circumstances. This is the growth part where the system reads the documents and garners experience from them.

Phase 2:

Next phase is the Knowledge Implementation phase. Here the system uses all the knowledge it has acquired through the learning and growth phase to create the article.

An article needs to provide the following primary information winning team, match details, runs scored by each team, man of the match or any other outstanding performance by a player. So the system will try to convey this information. It will use a sentence to convey each point. The structure of this sentence is decided by experience. If a team has scored 300 runs in 50 overs and 7 wickets then the most frequently used structure type matching this scenario is used. The most frequently used structure type is obtained from articles fed to the system during the learning and growing phase.

The system creates a sentence to convey each of the above points and when we put all the sentences together we have the article in its entirety.

6.2 Implemented flow of the system

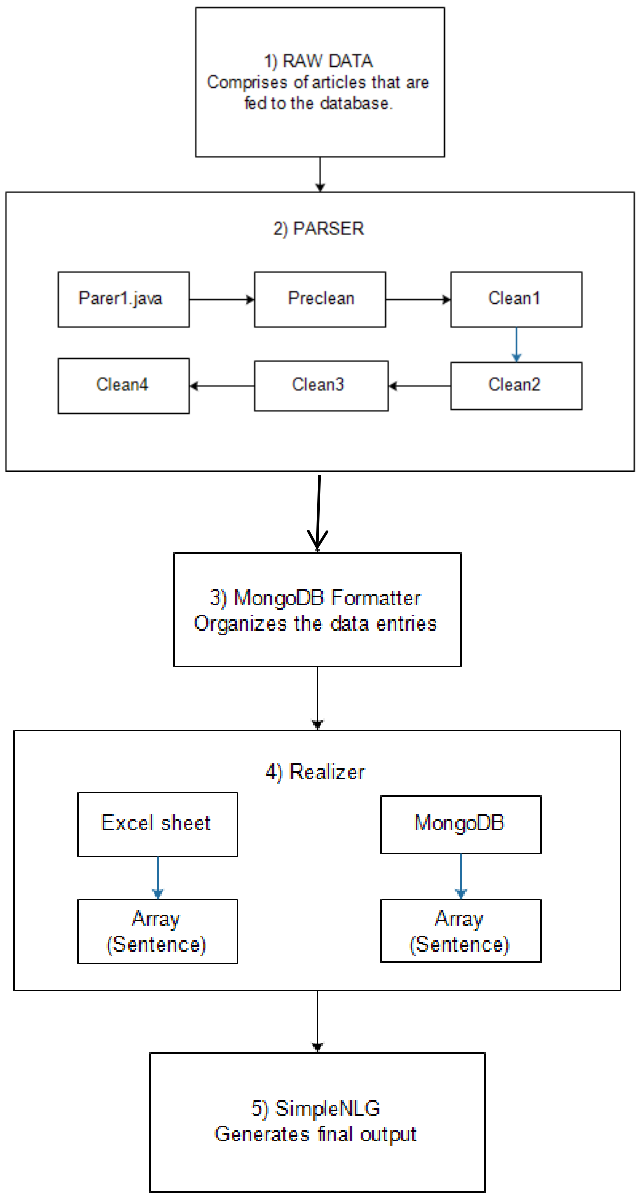
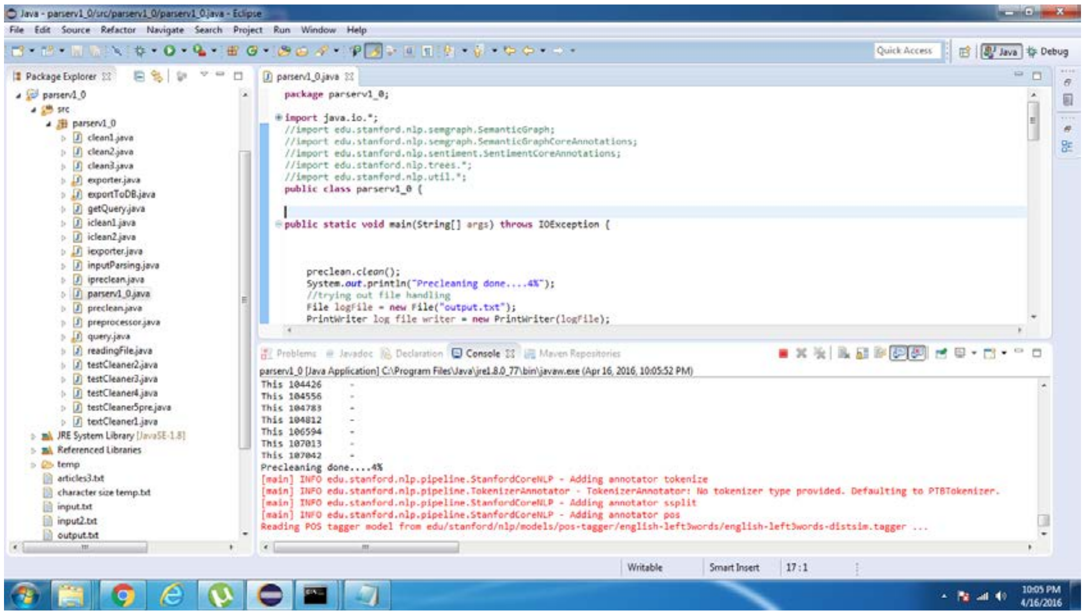


Fig 6.1 Flowchart for implementation

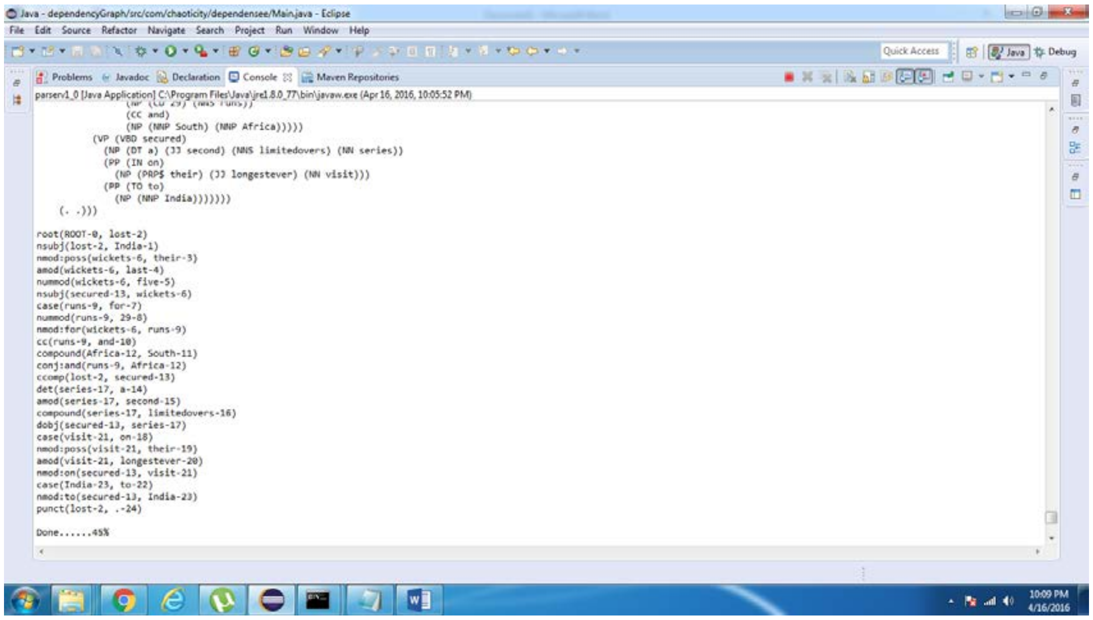
6.3 Implemented system screenshot.



```
package parserv1_0;

import java.io.*;
//import edu.stanford.nlp.semgraph.SemanticGraph;
//import edu.stanford.nlp.semgraph.SemanticGraphCoreAnnotations;
//import edu.stanford.nlp.sentiment.SentimentCoreAnnotations;
//import edu.stanford.nlp.trees.*;
//import edu.stanford.nlp.util.*;
public class parserv1_0 {

public static void main(String[] args) throws IOException {



    preclean.clean();
    System.out.println("Precleaning done....4%");
    //trying out file handling
    File logFile = new File("output.txt");
    PrintWriter log file writer = new PrintWriter(logFile);
```

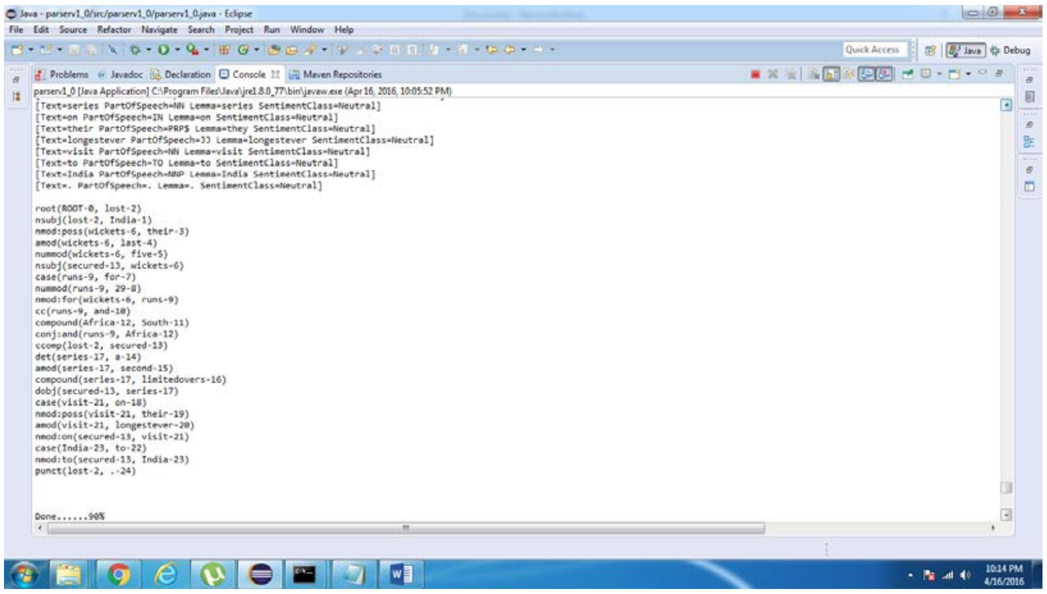
```
parserv1_0 [Java Application] C:\Program Files\Java\jre1.8.0_77\bin\javaw.exe (Apr 16, 2016, 10:05:52 PM)
This 104426     -
This 104556     -
This 104783     -
This 104812     -
This 106594     -
This 107013     -
This 107042     -
Precleaning done....4%
[main] INFO edu.stanford.nlp.pipeline.StanfordCoreNLP - Adding annotator tokenize
[main] INFO edu.stanford.nlp.pipeline.TokenizerAnnotator - TokenizerAnnotator: No tokenizer type provided. Defaulting to PTBTokenizer.
[main] INFO edu.stanford.nlp.pipeline.StanfordCoreNLP - Adding annotator ssplit
[main] INFO edu.stanford.nlp.pipeline.StanfordCoreNLP - Adding annotator pos
Reading POS tagger model from edu/stanford/nlp/models/pos-tagger/english-left3words/english-left3words-distsim.tagger ...
```

Fig 6.2 Preprocessing output


Java - dependencyGraph/src/com/chaoticity/dependensee/Main.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console Maven Repositories
10:09 PM 4/16/2016

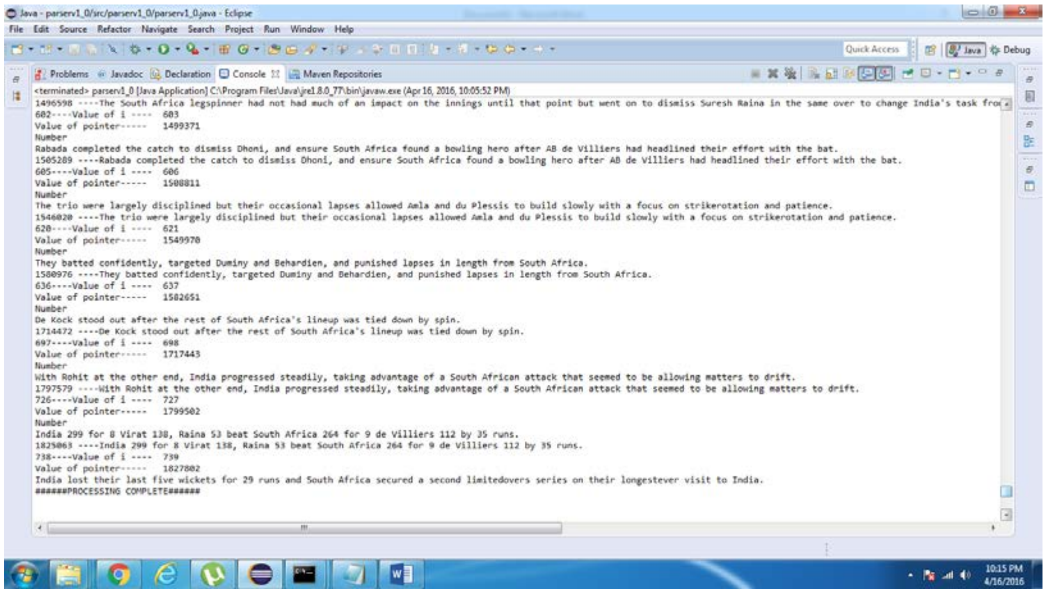
```
parserv1_0 [Java Application] C:\Program Files\Java\jre1.8.0_77\bin\javaw.exe (Apr 16, 2016, 10:05:52 PM)
                (NP (CD 29) (NNS runs))
                (CC and)
                (NP (NNP South) (NNP Africa)))))
          (VP (VBD secured)
            (NP (DT a) (JJ second) (NNS limitedovers) (NN series))
            (PP (IN on)
              (NP (PRP$ their) (JJ longestever) (NN visit)))
            (PP (TO to)
              (NP (NNP India)))))))
    (. .)))

root(ROOT-0, lost-2)
nsubj(lost-2, India-1)
nmod:poss(wickets-6, their-3)
amod(wickets-6, last-4)
nummod(wickets-6, five-5)
nsubj(secured-13, wickets-6)
case(runs-9, for-7)
nummod(runs-9, 29-8)
nmod:for(wickets-6, runs-9)
cc(runs-9, and-10)
compound(Africa-12, South-11)
conj:and(runs-9, Africa-12)
ccomp(lost-2, secured-13)
det(series-17, a-14)
amod(series-17, second-15)
compound(series-17, limitedovers-16)
dobj(secured-13, series-17)
case(visit-21, on-18)
nmod:poss(visit-21, their-19)
amod(visit-21, longestever-20)
nmod:on(secured-13, visit-21)
case(India-23, to-22)
nmod:to(secured-13, India-23)
punct(lost-2, .-24)

Done......45%
```

Fig 6.3 Cleaning phase 1 output



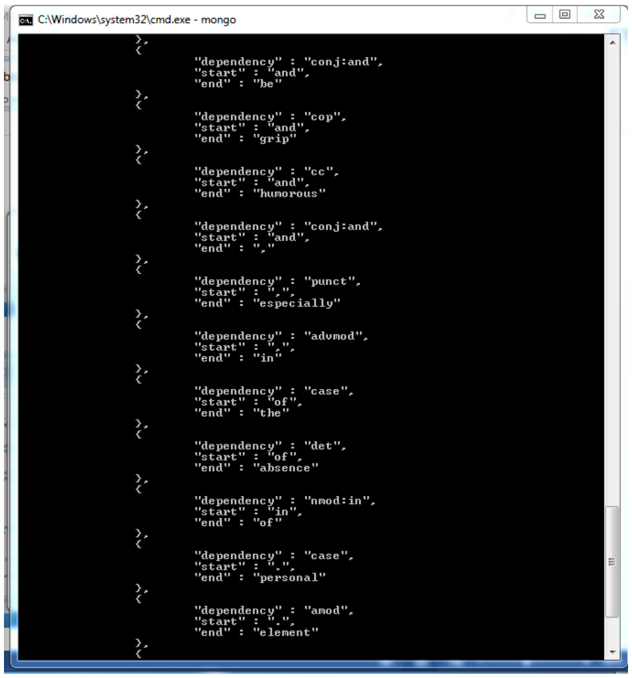
```
parserv1_0 [Java Application] C:\Program Files\Java\jre1.8.0_77\bin\javaw.exe (Apr 16, 2016, 10:05:52 PM)
[Text=series PartOfSpeech=NN Lemma=series SentimentClass=Neutral]
[Text=on PartOfSpeech=IN Lemma=on SentimentClass=Neutral]
[Text=their PartOfSpeech=PRP$ Lemma=they SentimentClass=Neutral]
[Text=longestever PartOfSpeech=JJ Lemma=longestever SentimentClass=Neutral]
[Text=visit PartOfSpeech=NN Lemma=visit SentimentClass=Neutral]
[Text=to PartOfSpeech=TO Lemma=to SentimentClass=Neutral]
[Text=India PartOfSpeech=NNP Lemma=India SentimentClass=Neutral]
[Text=. PartOfSpeech=. Lemma=. SentimentClass=Neutral]

root(ROOT-0, lost-2)
nsubj(lost-2, India-1)
nmod:poss(wickets-6, their-3)
amod(wickets-6, last-4)
nummod(wickets-6, five-5)
nsubj(secured-13, wickets-6)
case(runs-9, for-7)
nummod(runs-9, 29-8)
nmod:for(wickets-6, runs-9)
cc(runs-9, and-10)
compound(Africa-12, South-11)
conj:and(runs-9, Africa-12)
ccomp(lost-2, secured-13)
det(series-17, a-14)
amod(series-17, second-15)
compound(series-17, limitedovers-16)
dobj(secured-13, series-17)
case(visit-21, on-18)
nmod:poss(visit-21, their-19)
amod(visit-21, longestever-20)
nmod:on(secured-13, visit-21)
case(India-23, to-22)
nmod:to(secured-13, India-23)
punct(lost-2, .-24)


Done......90%
```

Fig 6.4 Cleaning phase 2 output



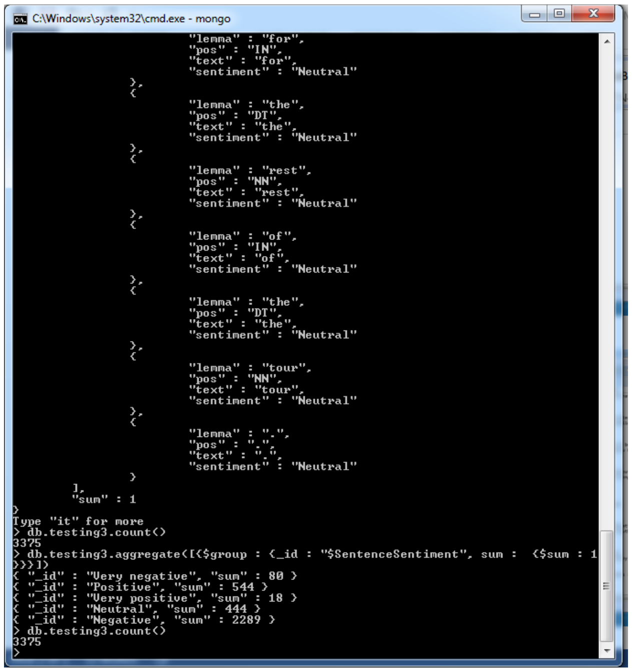
```
<terminated> parserv1_0 [Java Application] C:\Program Files\Java\jre1.8.0_77\bin\javaw.exe (Apr 16, 2016, 10:05:52 PM)
1496598 ----The South Africa legspinner had not had much of an impact on the innings until that point but went on to dismiss Suresh Raina in the same over to change India's task fro
602----Value of i ----  603
Value of pointer-----   1499371
Number
Rabada completed the catch to dismiss Dhoni, and ensure South Africa found a bowling hero after AB de Villiers had headlined their effort with the bat.
1505289 ----Rabada completed the catch to dismiss Dhoni, and ensure South Africa found a bowling hero after AB de Villiers had headlined their effort with the bat.
605----Value of i ----  606
Value of pointer-----   1508811
Number
The trio were largely disciplined but their occasional lapses allowed Amla and du Plessis to build slowly with a focus on strikerotation and patience.
1546020 ----The trio were largely disciplined but their occasional lapses allowed Amla and du Plessis to build slowly with a focus on strikerotation and patience.
620----Value of i ----  621
Value of pointer-----   1549970
Number
They batted confidently, targeted Duminy and Behardien, and punished lapses in length from South Africa.
1580976 ----They batted confidently, targeted Duminy and Behardien, and punished lapses in length from South Africa.
636----Value of i ----  637
Value of pointer-----   1582651
Number
De Kock stood out after the rest of South Africa's lineup was tied down by spin.
1714472 ----De Kock stood out after the rest of South Africa's lineup was tied down by spin.
697----Value of i ----  698
Value of pointer-----   1717443
Number
With Rohit at the other end, India progressed steadily, taking advantage of a South African attack that seemed to be allowing matters to drift.
1797579 ----With Rohit at the other end, India progressed steadily, taking advantage of a South African attack that seemed to be allowing matters to drift.
726----Value of i ----  727
Value of pointer-----   1799502
Number
India 299 for 8 Virat 138, Raina 53 beat South Africa 264 for 9 de Villiers 112 by 35 runs.
1825063 ----India 299 for 8 Virat 138, Raina 53 beat South Africa 264 for 9 de Villiers 112 by 35 runs.
738----Value of i ----  739
Value of pointer-----   1827802
India lost their last five wickets for 29 runs and South Africa secured a second limitedovers series on their longestever visit to India.
######PROCESSING COMPLETE######
```

Fig 6.5 Exporting to database



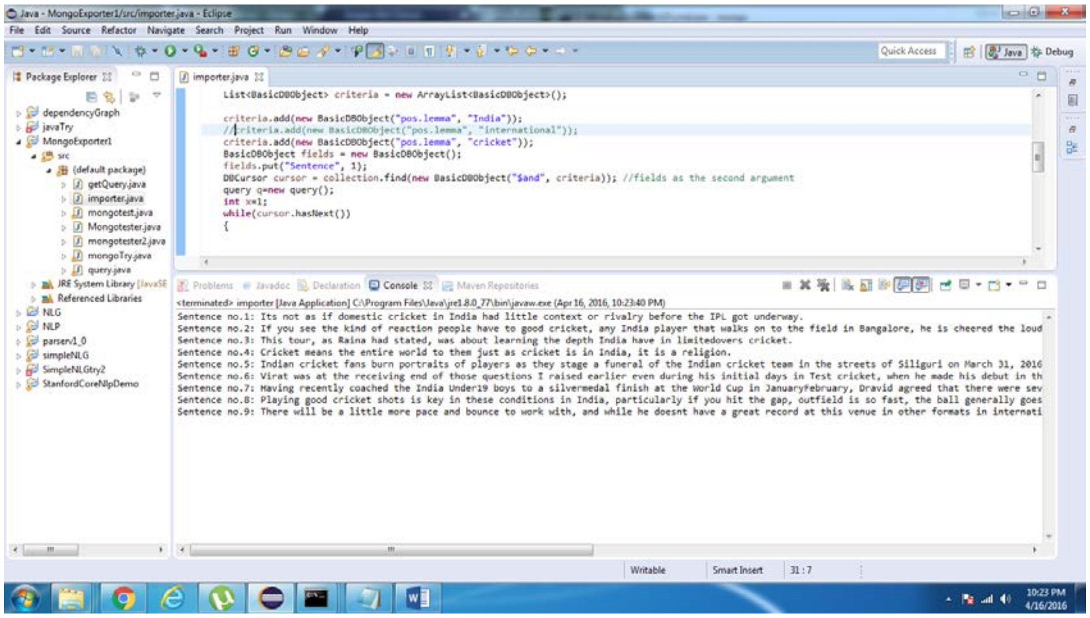
```
C:\Windows\system32\cmd.exe - mongo
                },
                {
                        "dependency" : "conj:and",
                        "start" : "and",
                        "end" : "be"
                },
                {
                        "dependency" : "cop",
                        "start" : "and",
                        "end" : "grip"
                },
                {
                        "dependency" : "cc",
                        "start" : "and",
                        "end" : "humorous"
                },
                {
                        "dependency" : "conj:and",
                        "start" : "and",
                        "end" : ","
                },
                {
                        "dependency" : "punct",
                        "start" : ",",
                        "end" : "especially"
                },
                {
                        "dependency" : "advmod",
                        "start" : ",",
                        "end" : "in"
                },
                {
                        "dependency" : "case",
                        "start" : "of",
                        "end" : "the"
                },
                {
                        "dependency" : "det",
                        "start" : "of",
                        "end" : "absence"
                },
                {
                        "dependency" : "nmod:in",
                        "start" : "in",
                        "end" : "of"
                },
                {
                        "dependency" : "case",
                        "start" : ".",
                        "end" : "personal"
                },
                {
                        "dependency" : "amod",
                        "start" : ".",
                        "end" : "element"
                },
                {
```

Fig 6.6 Checking the imported data



```
C:\Windows\system32\cmd.exe - mongo
                        "lemma" : "for",
                        "pos" : "IN",
                        "text" : "for",
                        "sentiment" : "Neutral"
                },
                {
                        "lemma" : "the",
                        "pos" : "DT",
                        "text" : "the",
                        "sentiment" : "Neutral"
                },
                {
                        "lemma" : "rest",
                        "pos" : "NN",
                        "text" : "rest",
                        "sentiment" : "Neutral"
                },
                {
                        "lemma" : "of",
                        "pos" : "IN",
                        "text" : "of",
                        "sentiment" : "Neutral"
                },
                {
                        "lemma" : "the",
                        "pos" : "DT",
                        "text" : "the",
                        "sentiment" : "Neutral"
                },
                {
                        "lemma" : "tour",
                        "pos" : "NN",
                        "text" : "tour",
                        "sentiment" : "Neutral"
                },
                {
                        "lemma" : ".",
                        "pos" : ".",
                        "text" : ".",
                        "sentiment" : "Neutral"
                }
        ],
        "sum" : 1
}
Type "it" for more
> db.testing3.count()
3375
> db.testing3.aggregate([{$group : {_id : "$SentenceSentiment", sum :  {$sum : 1
}}}])
{ "_id" : "Very negative", "sum" : 80 }
{ "_id" : "Positive", "sum" : 544 }
{ "_id" : "Very positive", "sum" : 18 }
{ "_id" : "Neutral", "sum" : 444 }
{ "_id" : "Negative", "sum" : 2289 }
> db.testing3.count()
3375
>
```

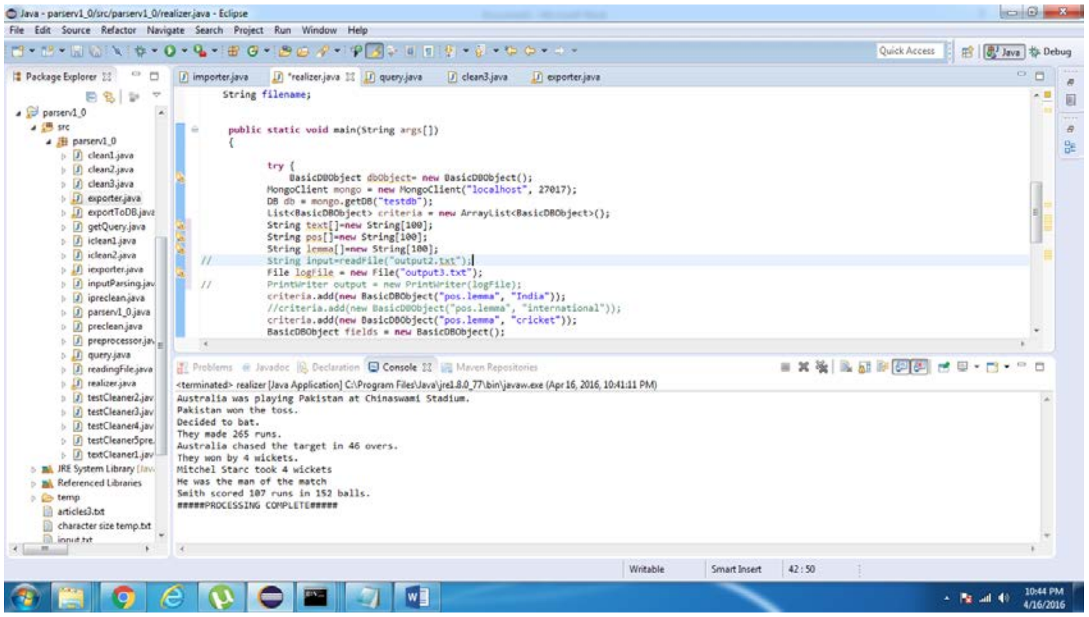
Fig 6.7 Checking for diversity of data



```
List<BasicDBObject> criteria = new ArrayList<BasicDBObject>();

criteria.add(new BasicDBObject("pos.lemma", "India"));
//criteria.add(new BasicDBObject("pos.lemma", "international"));
criteria.add(new BasicDBObject("pos.lemma", "cricket"));
BasicDBObject fields = new BasicDBObject();
fields.put("Sentence", 1);
DBCursor cursor = collection.find(new BasicDBObject("$and", criteria)); //fields as the second argument
query q=new query();
int x=1;
while(cursor.hasNext())
{
```

```
<terminated> importer [Java Application] C:\Program Files\Java\jre1.8.0_77\bin\javaw.exe (Apr 16, 2016, 10:23:40 PM)
Sentence no.1: Its not as if domestic cricket in India had little context or rivalry before the IPL got underway.
Sentence no.2: If you see the kind of reaction people have to good cricket, any India player that walks on to the field in Bangalore, he is cheered the loud
Sentence no.3: This tour, as Raina had stated, was about learning the depth India have in limitedovers cricket.
Sentence no.4: Cricket means the entire world to them just as cricket is in India, it is a religion.
Sentence no.5: Indian cricket fans burn portraits of players as they stage a funeral of the Indian cricket team in the streets of Siliguri on March 31, 2016
Sentence no.6: Virat was at the receiving end of those questions I raised earlier even during his initial days in Test cricket, when he made his debut in th
Sentence no.7: Having recently coached the India Under19 boys to a silvermedal finish at the World Cup in JanuaryFebruary, Dravid agreed that there were sev
Sentence no.8: Playing good cricket shots is key in these conditions in India, particularly if you hit the gap, outfield is so fast, the ball generally goes
Sentence no.9: There will be a little more pace and bounce to work with, and while he doesnt have a great record at this venue in other formats in internati
```

Fig 6.8 Querying lemmas in the database



```
    String filename;


    public static void main(String args[])
    {

            try {
                BasicDBObject dbObject= new BasicDBObject();
            MongoClient mongo = new MongoClient("localhost", 27017);
            DB db = mongo.getDB("testdb");
            List<BasicDBObject> criteria = new ArrayList<BasicDBObject>();
            String text[]=new String[100];
            String pos[]=new String[100];
            String lemma[]=new String[100];
//          String input=readFile("output2.txt");
            File logFile = new File("output3.txt");
//          PrintWriter output = new PrintWriter(logFile);
            criteria.add(new BasicDBObject("pos.lemma", "India"));
            //criteria.add(new BasicDBObject("pos.lemma", "international"));
            criteria.add(new BasicDBObject("pos.lemma", "cricket"));
            BasicDBObject fields = new BasicDBObject();
```

```
<terminated> realizer [Java Application] C:\Program Files\Java\jre1.8.0_77\bin\javaw.exe (Apr 16, 2016, 10:41:11 PM)
Australia was playing Pakistan at Chinaswami Stadium.
Pakistan won the toss.
Decided to bat.
They made 265 runs.
Australia chased the target in 46 overs.
They won by 4 wickets.
Mitchel Starc took 4 wickets
He was the man of the match
Smith scored 107 runs in 152 balls.
#####PROCESSING COMPLETE#####
```

Fig 6.9 Production of sentences

CHAPTER 7
TECHNOLOGIES USED

7.1 Hardware Requirements

- SAN storage network interface.
  This storage is used to store data from the automatic identification, capture, and summary from web pages, knowledge bases, emails, chats, social media, transcripts, conversations, and other unstructured formats.
- Solid State Hard drive.
  Databases can benefit from solid state storage. This can reduce the access speeds and querying delay by almost half. This system will be very memory extensive and will require a shorter access cycle for optimal performance.
- Computer for user interface.
  This will allow users to connect to the NLG portal and run the software.

7.2 Software Requirements

- Stanford CoreNLP
  The Stanford CoreNLP library is a machine learning based toolkit for the processing of natural language text. It supports the most common NLP tasks, such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and coreference resolution.
- SimpleNLG
  SimpleNLG is a library and not an application. SimpleNLG library is used to create a program that will generate grammatically correct sentences in English. The library consists of classes which allow the user to define the subject, the object, the verb and additional compliments to the sentence.

- MongoDB / NOSQL

  MongoDB (from humongous) is a cross-platform document-oriented database. Classified as a NoSQL database, MongoDB eschews the traditional table-based relational database structure, having unstructured database architecture.
- Java JRE 6+

  Java JRE will allow us to run applets that provides user interface and running environment as the entire application is developed in Java. Java 2 Runtime Environment is essential to establish a connection between popular browsers and the Java platform. Java allows applications to be downloaded over a network and run within a guarded sandbox. Security restrictions are easily imposed on the sandbox.
- Eclipse

  Using Eclipse Integrated Development Environment for providing a development workspace and easy export and sharing of development files between team members along with building and publishing the final application. Eclipse uses plug-ins to provide all the functionality within and on top of the runtime system.
- Windows/Linux Operating Environment

  Provides and environment to sun the java application along with providing an interface to build and access the SAN network. This system should have access to the Internet, which is essential for connecting to the database.

CHAPTER 8
TEST CASES

8.1 Test Cases of system

"A test case specifies the pretest state of the IUT and its environment, the test inputs or conditions, and the expected result. The expected result specifies what the IUT should produce from the test inputs. This specification includes messages generated by the IUT, exceptions, returned values, and resultant state of the IUT and its environment. Test cases may also specify initial and resulting conditions for other objects that constitute the IUT and its environment."

Unit Test Cases (UTC)

These are very specific to a particular unit. The basic functionality of the unit is to be understood based on the requirements and the design documents. Generally, Design document will provide a lot of information about the functionality of a unit. The Design document has to be referred before UTC is written, because it provides the actual functionality of how the system must behave, for given inputs.

Test conditions for invalid symbols that our system will not accept

Test	**Description**	**Test Inputs**	**Expected Results**	**Actual results**	**Pass/Fail**
1	invalid symbols in the documents fed to the system.	Sachin was #livingitup	Inputs should not be accepted. Sentence will be ignored.	The system will ignore the sentence altogether. It makes an exception and does not display any error	Pass
2	Checking with a valid statement and no wrong symbols.	Sachin brought the cup back	Input entry will be accepted.	Accepted with no errors.	Pass

Integration Test Cases

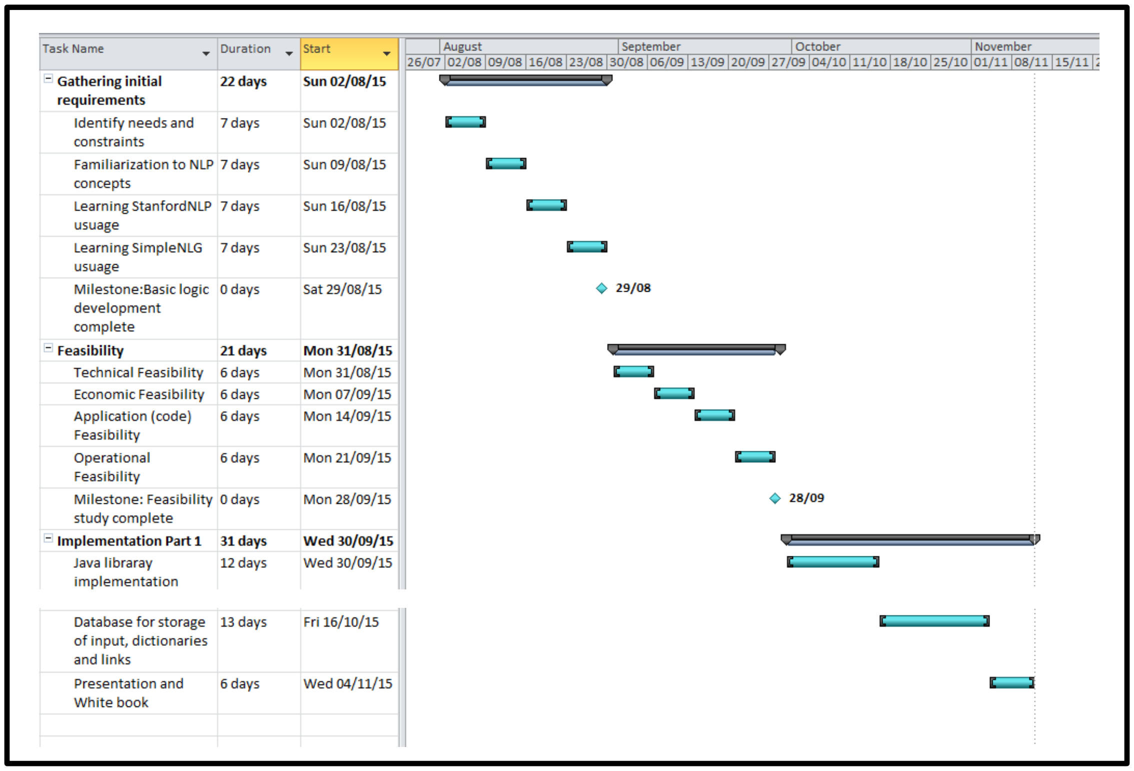
The main aim of integration test cases is that it tests the multiple modules together. By executing these test cases the user can find out the errors in the interfaces between the Modules.

In the following example we will test whether the statement in a document is stored and dependencies and tags are developed.

Test	**Description**	**Test Inputs**	**Expected Results**	**Actual results**	**Pass/Fail**
1	Checking whether our entry is tokenized with POS tags using NLP	Sachin played 100 on 80 balls.	Inputs should be tokenized and tagged, dependencies. This must be stored in MongoDB	Statement accepted by NLP	Pass
2	Checking Mongodb for statement	Sachin brought the cup back	Input entry will be accepted.	On querying we will find the statement in the database.	Pass
3	Complete inter-module exception handling	Kohli : 106/3 no ball	Input is tokenized into a vague sentence generating an “irrelevant sentence” exception	Generated POS tags along with exception pointer	Pass

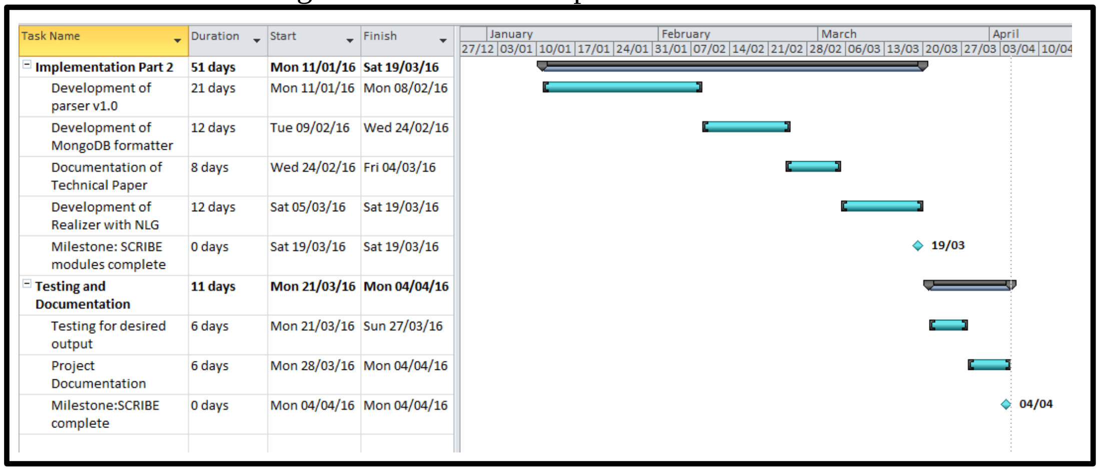
CHAPTER 9

PROJECT TIME LINE



Task Name	Duration	Start
Gathering initial requirements	**22 days**	**Sun 02/08/15**
Identify needs and constraints	7 days	Sun 02/08/15
Familiarization to NLP concepts	7 days	Sun 09/08/15
Learning StanfordNLP usuage	7 days	Sun 16/08/15
Learning SimpleNLG usuage	7 days	Sun 23/08/15
Milestone:Basic logic development complete	0 days	Sat 29/08/15
Feasibility	**21 days**	**Mon 31/08/15**
Technical Feasibility	6 days	Mon 31/08/15
Economic Feasibility	6 days	Mon 07/09/15
Application (code) Feasibility	6 days	Mon 14/09/15
Operational Feasibility	6 days	Mon 21/09/15
Milestone: Feasibility study complete	0 days	Mon 28/09/15
Implementation Part 1	**31 days**	**Wed 30/09/15**
Java libraray implementation	12 days	Wed 30/09/15
Database for storage of input, dictionaries and links	13 days	Fri 16/10/15
Presentation and White book	6 days	Wed 04/11/15

Fig 9.1 Timeline for Implementation Part 1



Task Name	Duration	Start	Finish
Implementation Part 2	**51 days**	**Mon 11/01/16**	**Sat 19/03/16**
Development of parser v1.0	21 days	Mon 11/01/16	Mon 08/02/16
Development of MongoDB formatter	12 days	Tue 09/02/16	Wed 24/02/16
Documentation of Technical Paper	8 days	Wed 24/02/16	Fri 04/03/16
Development of Realizer with NLG	12 days	Sat 05/03/16	Sat 19/03/16
Milestone: SCRIBE modules complete	0 days	Sat 19/03/16	Sat 19/03/16
Testing and Documentation	**11 days**	**Mon 21/03/16**	**Mon 04/04/16**
Testing for desired output	6 days	Mon 21/03/16	Sun 27/03/16
Project Documentation	6 days	Mon 28/03/16	Mon 04/04/16
Milestone:SCRIBE complete	0 days	Mon 04/04/16	Mon 04/04/16

Fig 9.2 Timeline for Implementation Part 2

CHAPTER 10
TASK DISTRIBUTION

Responsibilities:

1. Java libraries for sentence structure and parts of speech:

   Sumer Shende
   Rishabh Vig

2. Database for storage of input, dictionary and links created by the java programs:

   Vineet Trivedi
   Vandita Shah

3. Documentation (White Book) and Presentation
   Sumer Shende
   Vandita Shah
   Vineet Trivedi
   Rishabh Vig

Time Frame:
August-November

Responsibilities:

1. Java program for formation of word links and sentences:

   Sumer Shende

2. Technical Paper

   Vandita Shah
   Vineet Trivedi

3. Final program that accepts input and integrates all sectors to obtain final output
   Sumer Shende
   Vandita Shah
   Vineet Trivedi
   Rishabh Vig

4. Documentation (White Book) and Presentation
   Sumer Shende
   Vandita Shah
   Vineet Trivedi
   Rishabh Vig

Time Frame: December-March

CHAPTER 11
CONCLUSION AND FUTURE WORK

Scribe can expand its data sources. It can be made capable enough to draw raw data from sources like RSS feeds of multiple website, social media plug roots, blogging stations, news tickers etc. By expanding its sources of data, the richness and quantity of data can be increased. For this, Scribe needs to be able to read text from different formats like XML, LateX along with grabbing the correct unformatted text from html documents.

Scribe's use is not bounded to only cricket. Scribe can easily learn how to write articles on other sports like football, hockey, baseball etc. For this Scribe needs to be fed training data in the form of articles from the respective sports and it will learn the structure and details of each sport from the training data itself. With minimal changes made to the Scribe it will be ready to use for any other sport.

However the most important use of Scribe is the coverage of natural disasters. As stated earlier one of the key problems that this system was designed to tackle and eliminate was human delay in communication. During natural disasters chances of human delay are high and the need to communicate the disaster and its consequences is of paramount importance, Saffir-Simpson scale and stream gauges indicating a natural disaster like earthquake, hurricane or flood respectively should be communicated to Scribe. Scribe can then write a piece informing the people of a natural disaster. Scribe can be one of the fastest modes of communicating the occurrence of a natural disaster, the area it has occurred in and its possible impact. Incorporating the above suggestions in scribe will make the proposed system a multifaceted all-purpose journalist capable of writing an article on any topic.

CHAPTER 12
REFERENCES

[1] Abhimanyu Chopra, Abhinav Prashar, and Chandresh Sain, 2013, “Natural Language Processing”, International Journal of Technology Enhancements and Emerging Engineering Research.

[2] E. Cambria and B. White, 2014, “Jumping NLP Curves: A Review of Natural Language Processing Research”, IEEE Computational Intelligence Magazine.

[3] J. M. Huerta and D. Lubensky, 2003, “Graph-based representation and techniques for NLU application development”, IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03).

[4] N. Ito and M. Hagiwara, 2011, “Natural language generation using automatically constructed lexical resources” The 2011 International Joint Conference on Neural Networks (IJCNN).

[5] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard and David McClosky, 2014, “The Stanford CoreNLP Natural Language Processing Toolkit”, Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations.

[6] Zachary Parker, Scott Poe and Susan V. Vrbsky, 2013, “Comparing NoSQL MongoDB to an SQL DB”, Proceedings of the 51st ACM Southeast Conference.

[7] Veronika Abramova and Jorge Bernardino, 2013, “NoSQL databases: MongoDB vs cassandra”, Proceedings of the International C* Conference on Computer Science and Software Engineering.

[8] Albert Gatt and Ehud Reiter, 2009, "SimpleNLG: A realisation engine for practical application", Proceedings of the 12th European Workshop on Natural Language Generation.

[9] A Robust Human-Robot Communication System Using Natural Language for HARMS (The 12th International Conference on Mobile Systems and Pervasive Computing, MobiSPC 2015)

[10] SimpleNLG: A realisation engine for practical applications. (Proceedings of the 12th European Workshop on Natural Language Generation)

[11] Collaborative Writing Support Tools on the Cloud (IEEE Journal on Learning Technologies, VOL. 4, NO. 1, March 2011)

[12] Discourse strategies for generating natural-language text(Artificial Intelligence (Elsevier, Volume 27, Issue 1))

[13] Natural Language Generation as Incremental Planning Under Uncertainty: Adaptive Information Presentation for Statistical Dialogue Systems (IEEE/ACM Transactions on Audio Speech And Language Processing, VOL. 22, NO. 5, May 2014)

[14] Big Data and the SP Theory of Intelligence (Digital Object Identifier 10.1109/ACCESS.2014.2315297)

[15] Using Wikipedia and Conceptual Graph Structures to Generate Questions for Academic Writing Support (IEEE Transactions On Learning Technologies, VOL. 5, NO. 3, July-September 2012)

CHAPTER 13
APPENDIX

USER MANUAL:

Right now, since this project of ours is at a nascent stage, it is designed to function with minimum requirements of the user.

The two aspects of the projects as discussed throughout this document has been natural language understanding and natural language generation. For the most of the NLU aspect, that is the aspect that concerns itself with updating the database with files of sentences, which are text documents, POS tagging, finding dependencies etc. all of these are a back ground process that does not concern the user. What concerns our user however will be how to directly use the system in the most simple manner.

For the very same reason, we will restrict the scope of this manual to the aspect of user entries.

Here the user will simply need to provide the system will an excel sheet consisting of the information on a match. This will consist of the various scores and statistics of a given match. The user will provide the system with this basic file. Based on this file, the system will accept the users input and generate an article based on the important information mentioned in the excel sheet.

The system will automatically read the file and will not need any special inputs or entries to do the same.