# Chat Box

(A computer program on Android which converses with Humans)

Project Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology.

in

**Computer Science & Engineering**

under the Supervision of

Miss. Ramanpreet Kaur

By

Ankit Srivastava
111320

to


Jaypee University of Information and Technology
Waknaghat, Solan – 173234, Himachal Pradesh

# Certificate

This is to certify that project report entitled "**Chat Box**", submitted by **Ankit Srivastava** in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

**Date:**

**Supervisor's Name**
**Designation**

# Acknowledgment

I would like to express my special thanks of gratitude to my teacher Miss. Ramanpreet Kaur who gave me the golden opportunity to do this wonderful project on the topic Chat Box, which also helped me in doing a lot of Research and I came to know about so many new things I am really thankful to the respected ma'am.

Date:22/12/2014 Ankit Srivastava

# Table of Contents

**Topic** **Page.Number**

# Abbreviation and Symbols

| Abbreviation | FullForm |
|---|---|
| NLP | Natural Language Processor |
| SDK | Software Development toolKit |
| NLG | Natural Language Generator |
| APK | Android Package Kit |
| API | Application Program Interface |

# List of figures and tables

# Abstract

In our busy lives we often find it very hard to find time to communicate with each other. Many times we feel so lonely that we develop many social disorders and often we have to consult a psychiatrist. Sometimes the problem is so severe that people often develop suicidal temperament. This can be totally avoided. If we find time to socialize, with a person or maybe a machine, which not only listens to our problems but also understands them and gives us solutions. This is where technology can come in and play a huge part. It is not only cost effective but also an “always ready” companion. We have tried to develop such a program.
Chat Box is an android application that uses an artificial intelligence algorithm to have conversations with humans. Various similar web applications have been built in the past like “Cleverbot”, ”Eliza” etc.

Unlike other chatterbots, Cleverbot's responses are not programmed. Instead, it "learns" from human input; Humans type into the box below the Cleverbot logo and the system finds all keywords or an exact phrase matching the input. After searching through its saved conversations, it responds to the input by finding how a human responded to that input when it was asked, in part or in full, by Cleverbot.

Right now what we have achieved so far is our sample program processes the responses of the users by using simple parsing and by substitution of them into premade templates and using hardcoded phrases, seemingly continued the conversation.

What we can implement in it is NLP to understand what a user is saying and give solutions to his problems.

Natural language processing (NLP) is a field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and human (natural) languages. As such, NLP is related to the area of human–computer interaction. Many challenges in NLP involve natural language understanding, that is, enabling computers to derive meaning from human or natural language input, and others involve natural language generation.

What has never been implemented till now by all the various programs online is, a database or a “memory” of the program. It isn’t just necessary but it will be a revolutionary step in this field. Because if we implement it, users will feel connected to the program. They will feel that someone really is listening to their problems, not just talking to a machine which can just hold a conversation but can’t solve their problems.

# Introduction

Chat Box is an android application that uses an artificial intelligence algorithm to have conversations with humans. Various similar web applications have been built in the past like “Cleverbot”, ”Eliza” etc.

Much of this report will revolve around these two existing revolutionary programs in this field. Further in this report we will see how these programs work, try to understand their complications and the problems with these programs and try to understand how we can solve them and then make these programs better.

Eliza was first of its kind and it is open source. It was first of its kind. Although it works on a simple algorithm of substitution which isn’t the best algorithm available to us, its working is still important to give us an insight on how these programs work. And the Cleverbot will be the other application which will a subject of our study in this report.

Cleverbot was created by the British AI scientist Rollo Carpenter, who also created Jabberwacky, a similar web application. It is unique in the sense that it learns from humans, remembering words within its AI. In its first decade Cleverbot held several thousand conversations with Carpenter and his associates.

Cleverbot is constantly "learning", growing in data size, and perhaps also in the degree of "intelligence" it appears to display.

Although Cleverbot is much efficient than Eliza, it is of not much importance to us as it is not open source and also because the algorithm used is too vague for us to create a meaningful application which can understand human feelings. However studying its algorithm can help us create a much complex application which can be used to broaden the scope of Chat Box, the program which we are developing.

There will also be some references of “SIRI” an apple’s similar application which isn’t discussed in detail.

We will cover the problems of these programs and how we can improve upon them to develop a program which will be almost near to perfect.

# Literature Review

## *1.1 Components of Chat Box*

The components of the Chat Box include the following components:
# UI
# Background Working
UI: The UI of the chat box has nothing fancy in it. It has been made very simple. It consists of two textviews where the latest conversation can be displayed and a simple edittext where a user can type the input. UI has been created with the help of XML.
Background Working: It consists of these three phases:

Parsing and Substitution
NLP (Natural Language Processor)
Database

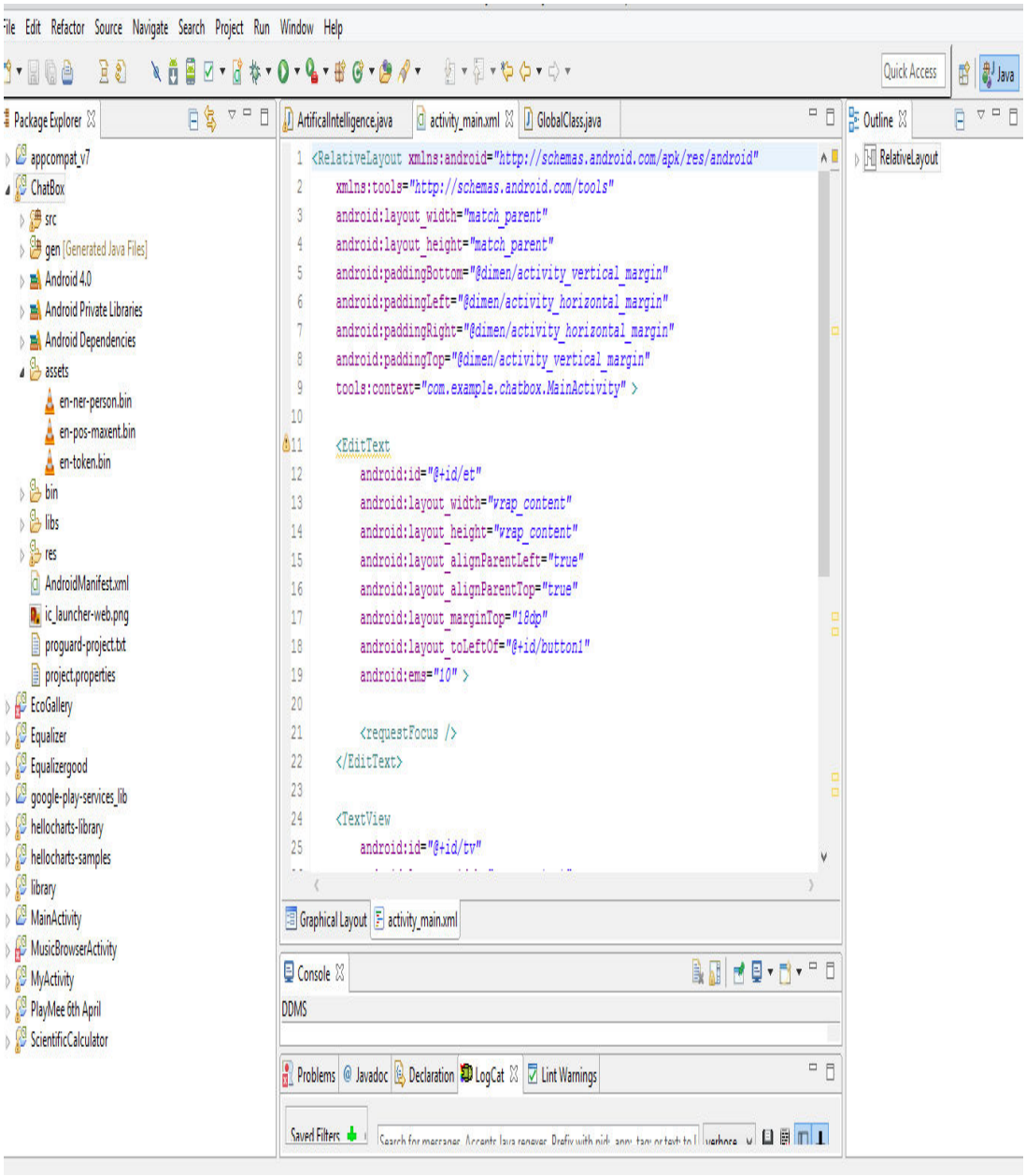
Parsing and Substitution: Whatever user types in, is passed to a class which parses the input and then substitutes words and phrases with various other words and phrases to create a grammatically statement , from a given set of already present words, statements and phrases. The conversation may seem meaningless although they are grammatically correct.

Natural Language Processor: NLP is required so that the parsed data can be "understood" by the application, things such as a user's mood, his feelings, if there is a humor in the statement, the names and places mentioned in the input. Various other nouns, etc. It hasn't been implemented yet and will be implemented in the future.
Database: A Database is required so that the parsed data can be stored by the application, things such as a user's mood, his feelings, and his mood in the past. And if he is feeling too stressed out, a stress call or some other precautionary measures can be taken.

Various other things of importance to the user can also be stored such as names and places. Various other nouns, etc.

A Database can be used so that the user can feel that he is connected to someone and not talking to a machine which forgets every time whatever the user says.

It hasn’t been implemented yet and will be implemented in the future.




Basic XML code

## *1.2 Eliza and the Doctor script*

There have been various programs/applications based upon this concept.
One of the earliest of such a program was Eliza. Eliza is also famous because it is an Open source program.

ELIZA is a computer program and an early example of primitive natural language processing. ELIZA operated by processing users' responses to scripts, the most famous of which was DOCTOR, a simulation of a Rogerian psychotherapist. Using almost **no information about human thought or emotion,** DOCTOR sometimes provided a startlingly human-like interaction. ELIZA was written at MIT by Joseph Weizenbaum between 1964 and 1966.

When the "patient" exceeded the very small knowledge base, DOCTOR might provide a generic response, for example, responding to "My head hurts" with "Why do you say your head hurts?" A possible response to "My mother hates me" would be "Who else in your family hates you?" ELIZA was implemented using simple pattern matching techniques, but was taken seriously by several of its users, even after Weizenbaum explained to them how it worked. It was one of the first chatterbots in existence.

ELIZA ran many scripts, each of which processed the responses of the users by using simple parsing and by substitution of them into premade templates and using hardcoded phrases, seemingly continued the conversation. The most popular script used in the ELIZA program was DOCTOR which was supposed to be a parody of a Rogerian Psychotherapist, i.e. it would ask the questions which are like those favoured by the 60s therapists who indulged in what is known as “Person Centered Therapy” in which the therapist would often return a question asked by the patient to themselves to increase empathy. This choice of a therapy setting was deliberate as questions like “what is your favourite colour” can be returned to the human conversant with replies like “why do you ask?” in such a setting. This kind of setting doesn’t need the system to actually understand the questions and the answer templates are pretty much finite.Thus, Weizenbaum wanted to create something simple enough which will not have requirement to know real world data and thus sidestep issues of natural language processing or any kind of semantics. Nor did

Weizenbaum have any intention to create an NLP system, the ELIZA project was supposed to be something very simple, almost a toy program, or in Weizenbaum's own words, a parody. What DOCTOR does is use a lot of data from the statements the human provides in order to compose responses by using simple templates, these templates having the flavour of stereotypical therapy statements. Thus, DOCTOR does not even attempt to answer the questions in the traditional sense but follows an if-else pattern of using a certain template for a certain trigger. There is no attempt to comprehend the semantics of the statements of the human.

ELIZA was first implemented in the SLIP language (Symmetric List Processor), a language incensed by Weizenbaum himself as an extension to FORTRAN but with better functionality to process doubly linked lists. For its time, ELIZA was revolutionary in many aspects, as interfaces were not really common in the computers of the late 60s due to the absence of personal computing and thus the idea of interactive computation had not arisen yet or entered into popular fancy. Even though the perceived intelligence of it was an illusion (and a very bad illusion) the fact remains that it was the first genuine human machine interface (pretending to be a human – intelligent machine interface) attempting to use natural language. However, what was more interesting about ELIZA was not so much its own implementation or characteristics which were simple enough but that of the reaction of the lay public to it which first illustrated the fascination with "intelligent" systems that became a theme of how people react to computers later. *A sample transcript of the ELIZA program when the conversation is not in the would look like this:*

I am the psychotherapist. Please, describe your problems. Each time you are finished talking, type RET twice.

I wish to know about you.
Why do you say i might want to know me?
To write my report
What makes you believe to write your report?
I do not understand your question, please clarify
Earlier you said i would like to know me?
yes

Can you elaborate on that?
I wish to know ELIZA
How does it feel to want?
Want what?
Maybe your plans have something to do with this.
Er, what plans?
Is it because of your life that you say er what plans?
It is because of you I say that.
I would appreciate it if you would continue.
I will.

As can be seen in this ‘conversation’ with ELIZA it has a habit of parsing the responses of the user, i.e. me, to frame its next question. After continuing the conversation and after a while the templates it uses to set these responses become predictable. It also has a few set phrases for conversation bits when it can’t find a response in its script, like “Maybe your plans have something to do with this.”

While ELIZA (and its modern, more advanced brethren) may seem convincing for a while to someone who does not know that they are conversing with a chatterbot, at least for a while, these chatterbots will not stand sustained inquiry from a skeptical subject who is actively trying to do a limited version of the Turing test.

## *1.3 Cleverbot*

Cleverbot is a web application that uses an artificial intelligence algorithm to have conversations with humans. It was created by the British AI scientist Rollo Carpenter, who also created Jabberwacky, a similar web application. It is unique in the sense that it learns from humans, remembering words within its AI. In its first decade Cleverbot held several thousand conversations with Carpenter and his associates. Since launching on the web in 1997, the number of conversations held has exceeded 150 million.

Unlike other chatterbots, Cleverbot's responses are not programmed. Instead, it "learns" from human input; Humans type into the box below the Cleverbot logo and the system finds all keywords or an exact phrase matching the input. After searching through its saved conversations, it responds to the input by finding how a human responded to that input when it was asked, in part or in full, by Cleverbot.

Cleverbot participated in a formal Turing Test at the 2011 Technical festival at the Indian Institute of Technology Guwahati on September 3, 2011. Out of the 334 votes cast, Cleverbot was judged to be 59.3% human, compared to the rating of 63.3% human achieved by human participants. A score of 50.05% or higher is often considered to be a passing gradeThe software running for the event had to handle just 1 or 2 simultaneous requests, whereas online Cleverbot is usually talking to around 80,000 people at once.

Cleverbot is constantly "learning", growing in data size, and perhaps also in the degree of "intelligence" it appears to display. Updates to the software have been mostly behind the scenes. In 2014 Cleverbot was upgraded to use GPU serving techniques.

A significant part of the engine behind Cleverbot, and an API for access to serving, has been made available to developers in the form of Cleverscript.

An app that uses the Cleverscript engine to play a game of 20 Questions, has been launched under the name Clevernator. Unlike other such games, the player asks the questions and it is the role of the AI to understand, and answer factually. An app that allows owners to create and talk to their own small Cleverbot-like AI has been launched, called Cleverme! for Apple products.

## *1.4 Problems with Clever Bot*

It doesn’t take anyone too many sentences for it to fail a Turing test, and **figure out it was saying things that had been said to it many times.**
**It can't reference back than a single sentence.**
**It has no core identity**.
**It often answers a "why" question with a "where" answer.**
And the reason is Cleverbot simply stores responses that people give to it in return for things that it says, and to work out what to say to the things said to it, it simply says them to someone else and records their answer. It's then a "simple" matter of finding the nearest equivalent in the database to anything that's just been said to it and then supplying the appropriate reply for that.
**The result is that it can hold a conversation, although a rather empty one.**

For a machine to hold a proper conversation with anyone, it would need to design a system that "understands" what its hearing and saying properly so that it can hold a co-operative conversation. Making that happen is hard. The biggest barrier to it is untangling the tangled mess of complexity that words and ideas are made up of. Once you've got on top of that task, you then need to do something else,

## *1.5 Improving upon Clever Bot*

One goal of a conversation is to construct a model of the other person's motivations, priorities, presumptions and feelings about issues (at least, as far as the other party will reveal this). This process is sometimes called "mind reading" (but not in the ESP sense). If this is done well, you should be able to predict the decisions of the other party, to some extent.

Some parts of this model will be common across all people. This is what cleverbot is doing - creating a mashed-up model of the whole population of cleverbot users (somewhat impaired by the fact that people will intentionally say ridiculous things to Cleverbot , which are then assumed to be normal responses).

However, to do it well, a conversational agent needs to be able to:
Identify individuals, and to build up a model of that person (and not just assume that everyone lives in New Jersey, for example).

It needs to identify gaps in its knowledge about this person, and ask about them. (Knowing what questions are not appropriate is also important - but this may be less of a concern if someone knows they are conversing with a robot.)

Include information from earlier interactions (in the same session, or previous sessions)

It also needs to deduce likely actions from what it knows, and confirm these assumptions by asking questions.

Be able to estimate a confidence in its observations - some people will have a consistent attitude on some topic, but other people will change their attitude depending on context.

Clarifying these seeming inconsistencies is important (without making it sound like an interrogation)

The agent needs to have a similar, consistent, and slowly changing model about itself. Many questions in a normal conversation are about the other party.

There are significant context changes outside our conversations - perhaps that's why (in English) we ask how someone is going, after not seeing them for a while.

There are conversational agents that sometimes cause frustration and/or laughter, such as the predictive text on your mobile phone.

This starts off with some factual information like common words in your language, and their probabilities - but this is a population-wide model, and its initial proposals are often quite different from your intent.

By observing your individual pattern of accepting or rejecting proposed words, modern versions develop a custom model of word usage for their owner (including slang and jargon terms), and should improve their predictions with time.

It is another step to move from single-word interactions up to applying the rules of grammar to predict the next likely word based on the content of the sentence so far.

It is yet another step to analyse the topic of an individual conversation, and to propose words most relevant to that conversation.
It is also important to identify if there is a defined context for a conversation, eg

"Have a conversation with this person, and work together to determine information X (perhaps a short-term goal for AI), or accomplish task X (perhaps a longer-term task for AI)." This provides an initial context for the conversation.
"Have a conversation with this person that you don't know, and may never see again". This doesn't really have a goal, and is much more open-ended. Part of a good conversation is identifying mutual goals, and exploring them.

I think Apple's "Siri" has been a fairly successful example of a conversational agent, with the goal of becoming a personal assistant.

It includes a great deal of factual knowledge about the phone owner, including a directory of commonly called numbers, a diary of events, etc.
It also includes considerable factual information about the world, through its link with Wolfram Alpha

The text <-> speech conversion seems to work fairly well
Parsing the text into actions seems to work OK
It often proposes a hypothetical interpretation for confirmation.

A lot of that does sound pretty complex, but I wouldn't think it would be as hard to provide some kind of short term memory within a single session about what was said a few sentences ago. It would prevent repeating questions that had already been answered, or making certain contradictory statements (Cleverbot doesn't just forget what I said earlier, he forgets what he said as well.) It might also duplicate the recursive nature of discussions, where one responds not just to the content of the very last statement but makes a deduction (or says something relevant) based on the information in several statements. That never happens with Cleverbot, as far as I can tell.

## *1.6 Other Applications*

An application like Chat Box can be essential for many other purposes,

One such use is Emotional Therapy. More advanced versions of Chat Box can be used when feeling lonely and can provide us company or a “companion”.

Other future uses include, an advanced operating system, a personal assistant etc etc. There are many possibilities.
Siri: An app developed by apple acts like a search tool as well as an assistant. It works on voice commands instead of text input, but the working remains the same. Siri can read your latest text messages. Like when someone texts you while you're driving, and you know you're not supposed to look at your iPhone because it's a deadly risk, not just for you. Siri can read your email too.

A Chat Box like app can be used purely entertainment purposes. The possibilities are many. Only the future can tell us what it holds for us and Artificial Intelligence.

# Chat Box

---

## *2.1 The general problem*

### # Making the program understand English language:

The main problem with the chat box was how to make a computer understand the human language, and finally came to a conclusion it is too difficult to implement.

But it doesn't mean it cannot be done . It can be done to an extent. One such model is distinguishing the type of sentence

If a computer can distinguish type of a sentence (the user input) then this problem can be solved to some extent.

Here are the four parts:

**1. a command (asking the bot to perform something)**
**2. a question (asking the bot to search for an answer (not perform))**
**3. a statement (telling bot to learn something from the input) and**
**4. a simple response (like say hello, how are you doing)**

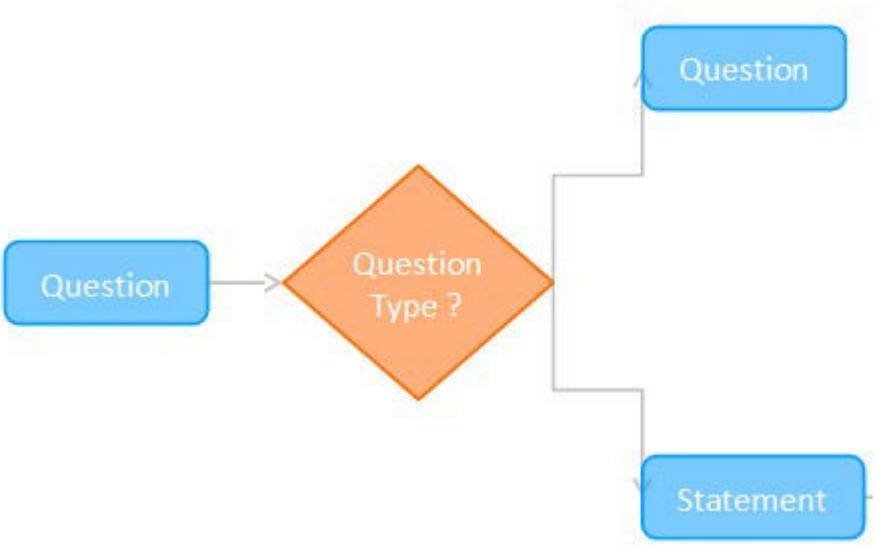
To minimize the scope we will be using only the 2 parts.

Which are:

**A Statement.**

**A Question.**

The diagrammatically representation of the division of the statement or the question entered by the user.




So this was the main idea for me into making the working logic of the chat bot,
A method of pattern recognition for database search has been used in our chat box.

Then the  next problem i faced is ,If a user replies “srk is an actor”, how the program will  understand that srk is a human not an object, ie when another user ask “what is srk,” it would be bad to reply “srk is actor”, or worst “who is bus” to “he is an vehicle”.
The solution was to do this by finding a keywords that represents humans (like “actor”) from the input.

These are the possibilities arising from a single statement.

“My friend raj has an iPod.”

Friend  Who is my friend?
Raj  Who is raj? / What does raj have?
IPod  Who has an iPod?

But this will be going out of the scope. We are trying to limit the use cases and the scope since this field is very vast and that's why we are trying to solve the problem of depression arising from relationships.
Hence we haven't covered the above problem.

## *2.2 Understanding our scope of work*

The purpose of our application the “Chat Box” is to provide therapy to the user. Now to provide emotional therapy ,we need to know who our user is ,their basic bio ,such as his name ,age ,history of the user's mood,their gender.

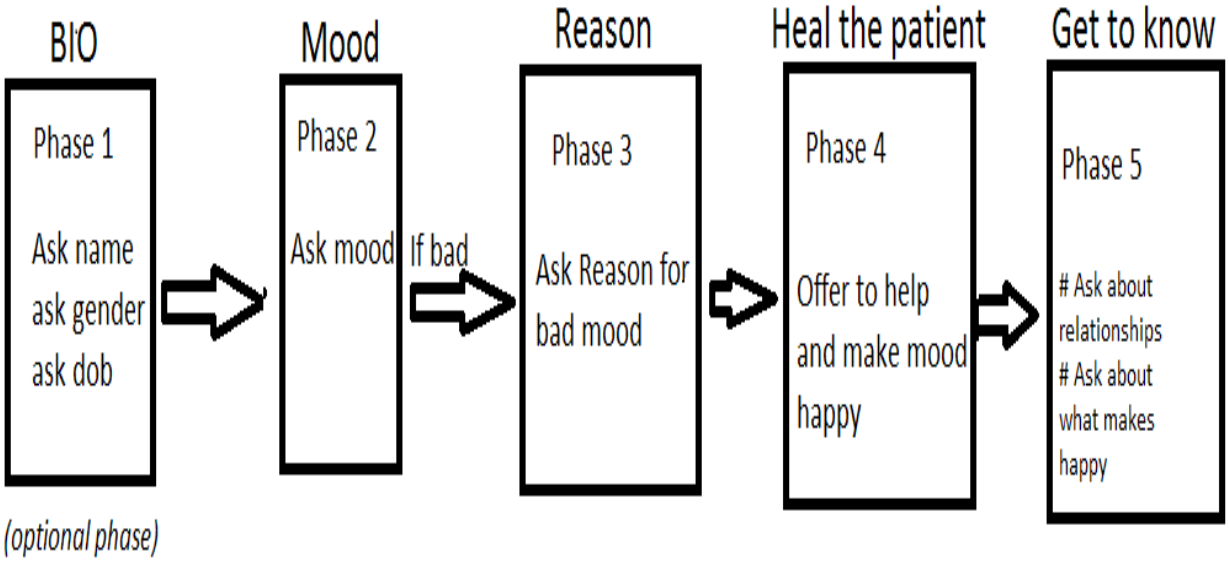
Now to limit the scope of this application and to limit the models made to understand the problem we have assumed that the user will be facing problems only related to relationships.

By making this assumption ,we need only one model.
Which is relationships of the users. Which can be defined as
Name-Relation-Relationship .
Where Name represents the name of the person ,relation is the basic relation with the person,such as if he is a brother,a friend or a sister etc.

Anything outside of this scope will not be entertained by the application and the user will be asked to keep the conversation within the scope of this application.

## *2.3 The Phases Explained*

The whole flow of the app has been divided into 5 phases. So that the mood of the user can be changed if he is feeling sad. The ultimate goal of this app is to provide emotional therapy to the user. Which is ensured by dividing the conversation into 5 parts or phases.
The five phases are as below:




**Phase 1**:

This is an optional phase and the conversation only goes through this phase when the app has been started for the first time or there isn't much information present about the user. In this phase the program asks the user about his personal bio. Things such as name,gender and age are asked and then are stored in the database for later use.
The user is asked premeditated questions stored in the database.
Questions such as :

```
"Hello I am Chat Box. Please tell me your name to get started"

"Hey I am your therapist. Please tell me your name"

"Seems like it's your first time here. Please tell me your name to get started"
```

Then the response of the user is searched for the names ,age and other such details using NLP which is then stored inside the database.
The phase is shifts to the second phase once this information is provided and stored.

## Phase 2:

There are two ways in which this phase can be initiated.
If the user opens the app not the first time then this phase is initiated ,ignoring the first phase.
This phase is also initiated after the first phase is completed.

The user is asked about how he is feeling. If the user is feeling fine,then the 5th phase is initiated,
If he isn't then phase 3 is initiated. Basically we want to know if the user is feeling bad so that therapy can be provided.
The user is asked premeditated questions stored in the database.
Questions such as :

**"Hello X , How are you feeling today ?"**

**"Hey X , How are you doing today ?"**

**"How are you X ?"**

Here "X" is replaced by the user name stored in the **Phase 1.**
The POS tagger of the NLP is used to tag each word into different parts of speech to understand how the user is feeling.
The mood of the user is then stored in the memory And a different phase is initiated accordingly.

## Phase 3:

The user is asked the reason for bad mood.
Again premeditated questions stored in the database are asked.
Questions such as:

**"Why are you feeling Y , X ?"**

**"What is the reason for your Y mood , X ?"**

**"Why is it that you are feeling Y , X "**

Here "X" is replaced by the user name stored in the **Phase 1.**
And "Y" Is replaced by the mood of the user stored in the **Phase 2.**
Here the program tries to understand the reason why the user is feeling bad.
Assuming that the user is feeling sad due to his relationships.
Hence the user will only talk about the people and his relationships with them.
This assumption has been made so that the scope of this program is constrained or else there will be a lot of use cases and a lot models should be made to understand the problem of the user.
By making this assumption ,we need only one model.
Which is:
Name-Relation-Relationship
Only these three parts of a particular entry should be known to provide advice on it.

# Phase 4:

In this  phase,the problem of the user is known and accordingly various measures are provided.
The statement is generated from the Simple NLG and the database which stores the various responses according to the problem and the age of the user.
Various parameters are used to generate the response.
They are as follows:

**Type of the problem user is facing(currently supporting only relationships)**

**Age of the user :** The statement generated takes care of the age of the user so that the problem is appropriate according to the age group

**The mood of the user throughout the time period etc**
If the user has been upset for a long time and continuous period of time,then a different solution can be provided.

**Things which make them happy**
The program can talk about the things which make the user happy.

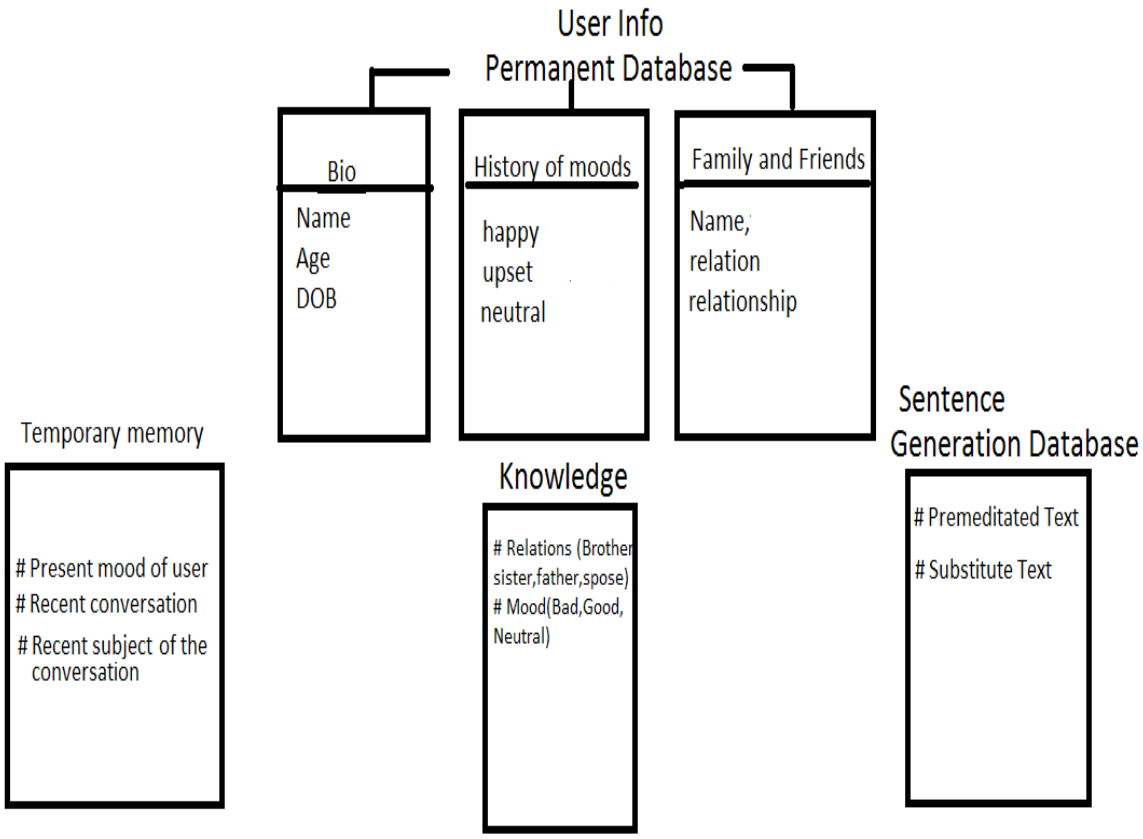
# Phase 5:

This is the last phase of the conversation .
In this phase either the solution has been provided or either the solution was not needed at the first place.
Now the user can either choose between the following to options

**Talk more about himself ,and the things which make them happy**
So that the program knows more about the user and can generate better solutions for the user,next time when they are unhappy.

**Exit the program.**
To exit the program and discontinue the conversation

## *2.4 The Storage Explained*



# Storage System provided by android :

Android Software Development Toolkit provides us with 2 permanent storage options. They are as follows.

**1. Shared preferences**

**2. Database management system.**

Shared preferences

Interface for accessing and modifying preference data returned by getSharedPreferences(String, int). For any particular set of preferences, there is a single instance of this class that all clients share. Modifications to the preferences must go through an SharedPreferences.Editorobject to ensure the preference values remain in a consistent state and control when they are committed to storage. Objects that are returned from the various get methods must be treated as immutable by the application.

It is generally preferred when organization of the data is not required and the data is small. It is very easy to implement.

And requires just 2 lines of code to store and to fetch.

Database management system

It is implemented using the inbuilt functions provided by the android sdk. And with the help of basic knowledge of SQLITE.

We have currently used a mixture of both to achieve the desired results.

Some files are even stored in the .bin format

# Implementation:

We have divided the storage into 4 categories .They are as follows:

**User Information**

**Knowledge of the program**

**Database for Sentence Generation**

**Temporary Memory**

User Information

This section stores three things about the user.

It stores basic bio of the user ,that is,things like the age ,the name ,the gender etc.

It stores relationships and friends of the users. This information is used to generate a response in the Phase 4 of the conversation and is generally acquired in the phase 5.

It also stores the past mood of the user.

Knowledge of the program

This consists of the information required for the NLP to work properly which includes files such as the bin files which stores the English names and files which contains various parts of speech .

It stores information related to the various kinds of moods and their classification into positive or negative mood and the various kinds of relationships so that when the program is in learning phase,it can match them from this database and learn.

Database for Sentence Generation

It stores the information about which word is to be replaced by what etc etc.

And the premeditated texts which appear when there is almost no information available about the user.

For e.g one of the three sentences is generated when the user opens the app for the first time

```
"Hello I am Chat Box. Please tell me your name to
get started"
"Hey I am your therapist. Please tell me your name"
"Seems like it's your first time here. Please tell
me your name to get started"
```

These are stored inside the sentence generation database

Temporary Memory

This memory stores the present mood of the user

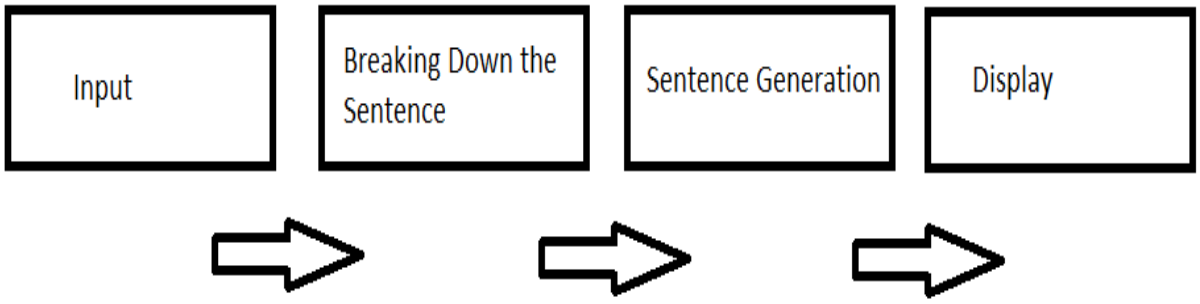
And it also stores the information related to the current conversation.

This hasn't been implemented in the program yet.

## *2.5 The Working (Technical) Explained*

# Basic Working:

The application behaves according to the phase of the conversation. The basic working has been summarized in the figure below.



The basic working of the app can be divided into 4 simple parts.

They are

**Input**

**Breaking down the sentence**

**Sentence Generation**

**Display the result**

Input

Input is taken from the user using the EditText class of the android sdk.

It is like a box where the text can be inputted.

By pressing the button after typing the information goes to the next part where the sentence is broken down into various parts of speech.

Breaking down the sentence

After the input is received it is broken down into various parts ,called as parts of speech by the help of POS tagger provided by the OPENNLP api

Sentence Generation

Simple NLG is used to generate grammatically correct sentences. The database is also used to generate the sentence.

Display the result

The result is then shown inside the Text View.

# Sentence Generation explained:

Sentence generation is incomplete without the breakdown of the sentence

Sentence breakdown is achieved by POS tagging which means parts of speech tagging. In this process each word is tagged into various parts of speech such as :

noun

adjective

verb ,etc

POS tagging is achieved with the help of OPENNLP where nlp stands for natural language processor.

Sentences breakdown is different for various phases and hence sentence generation is  different according to the various phases of the conversation.

**Sentence Generation inside Phase 1**:

For the first phase sentence is searched only for name ,which is achieved by pattern matching performed over the file containing the various predefined names.

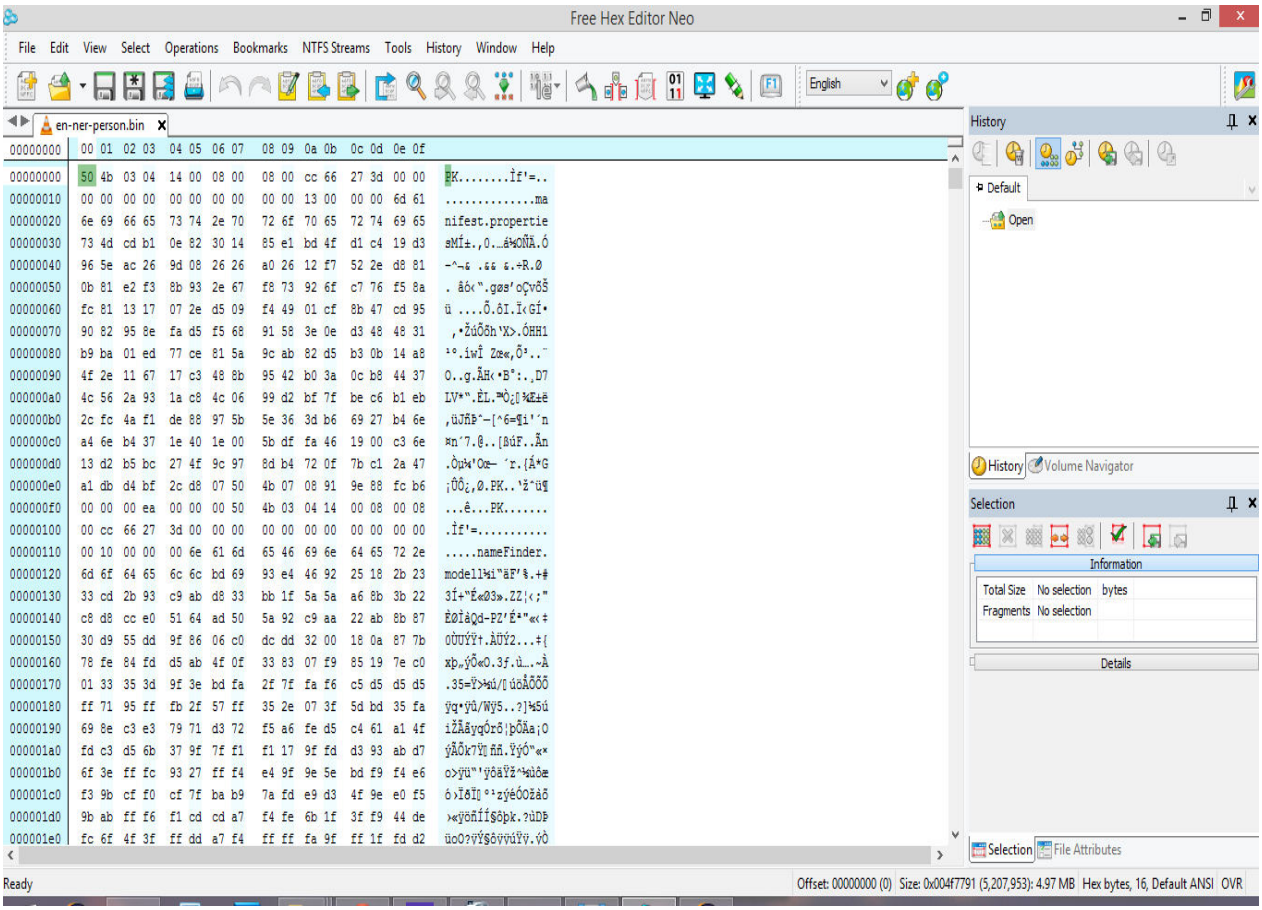
This file is in .bin format and can be edited using a hex editor.

This file can be edited to add more names so that it it is able to identify various other names even other than English names.

Here is an example of this file looks.

This phase doesn't involve the machine learning part.

It only stores the user name and bio of the user.




**Hex Editor used to edit bin files**

**Sentence Generation inside Phase 2**:

Inside the phase 2 the program only breaks down the sentence using POS Tagging with the help of OPENNLP to find the mood.

Basically sad,happy , angry are the basic moods. They are also adjectives .

There are two ways of identifying moods,either using parts of speech or classifying the various known moods inside the database and using pattern matching.

We have currently used the pattern matching technique to keep the problem simple.

This phase doesn't involve the machine learning part.

It stores only the type of the mood of the user.

**Sentence Generation inside Phase 3:**

In this phase ,the program breaks down the sentence Using POS Tagging and also tries to find names inside it.

We assumed that the problem is related only to humans on which we created a 3 entry model. The three entries were

Name

Relationship

Relations

So the programs tries to find such information from the text inputted by the user.

Suppose the user enters a sentence like.”My friend is very bad”

Now the program will try to learn from the Name-Relation-Relationship model and will find that Name  entry is missing.

So the program ask for things like “What is the name of your friend?” and will expect a user gives away the name of that friend and hence will try to find the name inside the next statement generated by the user.

This phase involves the machine learning part and whatever is learned about the user is stored inside the respective database and fetched to generate the statements.

---

**Sentence Generation inside Phase 4:**

Inside this phase of the program,the information learned from the phase 3 and information about the user is used to generate the statements.

For example

Suppose in the phase 3 the user types

“My friend is very bad”

A response can be

“Perhaps you shouldn't spend time with such friends(name) who are bad to you.

But if, instead of friend ,wife was used.

The response can change to

“If the things aren't going well,you should go to a marriage counselor ”

The personal information such as age of the user is also take into consideration while generating the response.

**Sentence Generation inside Phase 5:**

In this phase the problem is solved and the program asks for the things which the user likes. These things can be hobbies,games,sports(activities) which can be done or the company of other people. These things are searched for ,in this phase and the information is learned so that it can be used in the phase 4 while generating a response.

---

# Tools and Techniques

Tools used:

# A computer System running on Windows
# Eclipse was used as an IDE.
# Android SDK and ADT bundle
# OpenNLP (Natural Language Processor API)
# An Android Device

Techniques, languages and algorithms used:

# Java
# XML
# SQLITE
# Parsing and substitution, the same as used in Eliza algorithm.
# PosTagging NLP(Natural Language Processor)
# Database management System

# Conclusion and Future work

What we see from the above scenario and implementation is that bigger the database and more the models and use cases,the better is the response generated for the user.
But the problems are many.
For adapting more uses,the scope changes from the study of AI to language study.
We also have to remember that we are working on a mobile device.
The NLP processes are very resource extensive and perhaps using them on a server and dividing this application into client and server side application can solve the speed issue as when we do that ,the speed of the program wont be limited to the hardware of the mobile phones.

We live in an era of intelligent technology. Our watches tell us not only the time, but they also remind us to exercise. Our phones recommend the best places to dine, and our computers predict our preferences, helping us to do our daily work more efficiently.

Still, all of these digital assistants demonstrate only a tiny sliver of artificial intelligence (AI).

Apple's Siri is a rudimentary form of artificial intelligence that millions of people use every day.

Most of the consumer-level artificial-intelligence applications we're interacting with are programmed to use accumulated data stored in their databases to improve a reaction to inputs, which leads to a better response within predetermined parameters.

ELIZA, while itself a relatively simple system, is important from the point of view of understanding human intelligence and hypothetical machine intelligence as it provokes us to inquire what intelligence actually is and what can be considered genuine intelligence or "original" intentionality in a machine. It also makes us consider the amount of what is perceived as intelligent behavior of an agent lies.

Although the conversations are grammatically correct, they don't convey a meaningful conversation as the program doesn't seem to be able to understand whatever is being said by the user and hence the user never feels connected while

conversing with Eliza or Cleverbot. There seems to be no "memory" and not even a sense of identity. Such a machine is of no use to use.

However if we develop a program which can hold all the records and the "memories" of the user, his feelings the names and all the things which matter to him/her, we can really come close to build something which can help out a user

when he is stressed and act like a counsellor. This can be achieved with the following two components which we plan on implementing in the future.

Natural Language Processor: NLP is required so that the parsed data can be "understood" by the application, things such as a user's mood, his feelings, if there is a humor in the statement, the names and places mentioned in the input. Various other nouns, etc.
Database: A Database is required so that the parsed data can be stored by the application, things such as a user's mood, his feelings, and his mood in the past. And if he is feeling too stressed out, a stress call or some other precautionary measures can be taken.
Various other things of importance to the user can also be stored such as names and places. Various other nouns, etc.
A Database can be used so that the user can feel that he is connected to someone and not talking to a machine which forgets every time whatever the user says.

# Future Applications

There is a movie based on Artificial Intelligence. The movie, Her, in which a man falls in love with his computer, was awarded Best Original Screenplay at last night's Academy Awards. The concept may seem laughable, but advances in artificial intelligence are bringing us closer to our machines than ever before.

In the movie Theodore Twombly (played by Joaquin Phoenix) falls in love with his operating system Samantha, voiced by Scarlett Johansson. The genesis of "Her" was inspired by Cleverbot, a web application using a similar artificial intelligence algorithm to SmarterChild.

Her is set in the near future; around 2100 or so. To some extent, our relationship with technology already matches that depicted in the film; for example, when you look around a subway carriage it's not at all unusual to see the majority of passengers utterly engrossed in their phones. We already have that level of disconnect. But in terms of having a romantic relationship with our technology, we're still quite some way off.

By the 1970s, programmes could be created that could answer series of factual questions, but they were extremely limited. Now the algorithms are much more sophisticated, and it's much easier to feel you're having an actual conversation with a programme such as the iPhone's Siri as opposed to it firing answers back at you.

# Bibliography

## Books:

[1] Weizenbaum, Joseph. Computer Power and Human Reason (New York: Freeman, 1976)

[2] Weizenbaum, Joseph. "ELIZA - A Computer Program for the Study of Natural Language Communication between Man and Machine," Communications of the Association for Computing Machinery 9 (1966): 36-45.

[3] Suchman, Lucy A. Plans and Situated Actions: The problem of human-machine communication (Cambridge
University Press, 1987)

## Online Links:

http://www.planet-source-code.com/vb/scripts/ShowCode.asp?txtCodeId=5369&lngWId=3

http://en.wikipedia.org/wiki/Artificial_intelligence

http://en.wikipedia.org/wiki/Cleverbot

http://en.wikipedia.org/wiki/ELIZA

http://developers.slashdot.org/story/11/09/11/2051222/has-cleverbot-passed-the-turing-test

http://www.telegraph.co.uk/technology/technology-topics/10672747/Her-Could-you-ever-fall-in-love-with-a-computer.html

http://www.thenakedscientists.com/forum/index.php?topic=50684.0;prev_next=prev#new

http://www.pandoras-brain.com/

http://www.discovery.com

http://www.codeproject.com/Articles/12454/Developing-AI-chatbots

https://www.chatbots.org/ai_zone/viewthread/696/

## **Other online resources** :

DaxArts Virtual Interact  documentation of the sdk