

# Brain, the Framework for a Chatterbot

Michael Huynh

*AP Computer Science, Upper Darby High School*

May 2004

Modern chatterbots do not incorporate advanced methods of artificial intelligence. Following the development of chatterbots from its roots to modern programs, a better framework for chatterbots is proposed. This framework, demonstrated in Java, seeks to provide a structure for an “artificial intelligence” chatterbot that implements natural language processing, machine learning, and intelligent responses.

## 1. DEFINITION

A chatterbot is defined as “a bot program which attempts to maintain a conversation with a person<sup>1</sup>.” Among its many implementations, the majority successful chatterbots must be able to process natural language, and craft meaningful and coherent responses based on user input. Many chatterbots also attempt to implement artificial intelligence by including methods that learn from conversations and progressively expand its knowledge with each session.

The famous Alan Turing proposed in 1950 a test to determine program’s capability to act “human-like.” His test required that a human judge engage in a conversation with both the program being tested and another human. If the judge, deliberately trying to determine the true nature of both chats, believes that the program is actually a human responding, the program would pass the Turing test<sup>5</sup>. However, no programs have passed the Turing test yet.

## 2. EARLY CHATTERBOTS

Early chatterbots faced the limitations of computer systems at the time and could not command large processing power and storage space. However, in the 1960s and early 1970s, a few chatterbots emerged that solidly established a foothold for future chatterbots: Eliza, Parry, and SHRDLU.

### A. Eliza

In 1966, the Eliza bot was written by Joseph Weizenbaum. This program, which parodied a therapist, was highly successful in producing human-like responses. Despite its fairly simple pattern recognition and its limited word associations and responses, Eliza was very convincing because it used the concept of psychotherapy, as Weizenbaum explained, to “sidestep the problem of giving the program a data base of real-world knowledge<sup>2</sup>.” Therefore in

operation, Eliza would take user input, match certain keywords, and construct a response that used parts of the input through pattern matching. Most of the time, the response is in the form of a question. By doing this, Eliza was able to have users continue talking by answering questions while not knowing a single thing about the conversation. In a sense, the user is talking to himself or herself.

### B. Parry

Parry was written in 1972 by psychiatrist Kenneth Colby, then at Stanford University<sup>3</sup>. Parry’s personality as a paranoid schizophrenic was a complete opposite of Eliza’s personality as a psychotherapist. Parry was a step up from Eliza because it tried to add more personality through beliefs and emotional classification (simply accept, reject, neutral). Instead of just matching trigger words with responses, Parry had a conversation strategy built in for better dialoguing.

### C. SHRDLU

Written by Terry Winograd at the M.I.T. Artificial Intelligence Laboratory in the late 1960s, SHRDLU focused on understanding natural language and artificial intelligence rather than respond in a human-like fashion<sup>4</sup>. SHRDLU operated in a virtual world containing objects. By knowing very little at first, SHRDLU could learn about the objects in the world and retain that knowledge (i.e. One can stack rectangular blocks on top of each other but not pyramids.). Since SHRDLU had a memory bank, it could process input in context, deduce what a user meant by past inputs, and contain a record of the current state of the world. SHRDLU was a good demonstration of artificial intelligence in programs.

## 3. MODERN CHATTERBOTS

Modern chatterbots have evolved from their classic predecessors by utilizing better natural

language processing and by implementing learning algorithms that adapt to user input. While there are many chatterbots today, two stand out for their new approaches in generating human responses: Alice and Jabberwacky.

### A. Alice

Alice, a chatterbot created in the late 1990s by Richard Wallace and many contributors, uses heuristics in pattern matching to process input and respond accordingly<sup>7</sup>. It was successful in winning the Loebner prize for 2000 and 2001. A unique feature of this bot is that it uses an XML DTD called AIML (Artificial Intelligence Markup Language). Rules for pattern matching can be specified in AIML making it possible to quickly change the behavior of the chatterbot.

### B. Jabberwacky

Most chatterbots function by processing input and then applying a set of rules to craft an output that is hopefully as human-like as possible<sup>1</sup>. They do not actually attempt to understand the conversation. However, a very recent chatterbot, Jabberwacky, seeks to understand the conversation for more relevant responses by contextual pattern matching techniques. Jabberwacky incorporates artificial intelligence by learning and storing all user input. In future responses, Jabberwacky uses the learnt material. "If you speak in a foreign language it will learn it, and respond appropriately if it has enough to go on. It can be taught slang English, word games, jokes and any other form of identifiable language trait<sup>7</sup>." Today, Jabberwacky represents some of the latest technologies that chatterbots are implementing.

## 4. ARTIFICIAL INTELLIGENCE

Artificial intelligence can be generally defined as "intelligence exhibited by anything created by humans or other sentient beings<sup>8</sup>." However, artificial intelligence can be classified as strong or weak.

Strong AI is artificial intelligence that can truly reason and solve problems. It can be divided into programs that reason like humans and programs that reason in their own manner.

Weak AI is artificial intelligence that acts as if it was intelligent but cannot actually reason. The line between strong and weak AI is fuzzy, but it can be said that the vast majority of chatterbots attempt weak AI.

Artificial intelligence has been applied towards many areas including gaming systems, language translation, security systems, optical character recognition, handwriting recognition,

speech recognition, and machine vision and hearing<sup>8</sup>. Current AI research center around applying fuzzy logic, Bayesian classification, genetic algorithms, and neural networks to machine learning in hopes of attaining strong AI. However, these approaches do have their limitations.

The next generation of chatterbots should seek to implement these new advances in artificial intelligence. In order to accomplish this task, a flexible framework is needed that can accommodate machine learning. The ideal framework should be conceptualized as objects, each with a specific function. Because of the OOP design nature of the proposed chatterbot framework, the Java programming language was used to create the core classes and a simple example of a chatterbot was implemented with the flexibility of expansion and adaptability.

## 5. CHATTERBOT FRAMEWORK

Initial approaches to designing a chatterbot framework can be seen in Figure 1. The driver class would call only three classes. ParseInput would retrieve user input and parse the string for keywords. These keywords would then be queried at the KnowledgeBase which subsequently controls classes LoadMemory, LookupWord and LearnWord. When constructed by the Bot Driver, KnowledgeBase would call LoadMemory to retrieve the database of information. As it accepts word queries, KnowledgeBase calls LookupWord to determine if a word is currently in its database. If the word currently does not exist in the database, LearnWord is called which provides the means for which an association to this new word is added to the database. Class LoadMemory depends on class FileIO for file input methods. When the results have been returned by the KnowledgeBase, the Bot Driver then uses class GenerateOutput to craft an output to the user.

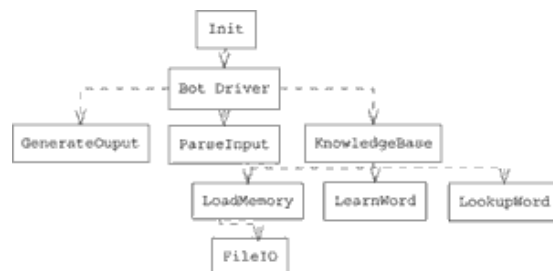


Fig. 1. UML diagram for initial design.

Although the initial design would theoretically work for a rudimentary chatterbot, it is not robust enough for more advanced chatterbots. The class structure, a top down approach, does not work very

well in this case since most of the program's operations will be in implementing artificial intelligence which crowds the area under KnowledgeBase. This causes the design to become lopsided and too dependent on KnowledgeBase which now acts as a driver class for many other classes. Another flaw with the initial design is that the chatterbot's class structure does not clearly define specific functionality. For instance, what exactly is KnowledgeBase supposed to do? Is it just supposed to provide a venue for containing the chatterbot's memory or can it encapsulate artificial intelligence classes too? With these ambiguities, it is very difficult to design in a flexible and adaptable way. A new approach was sought.

The new approach had to satisfy the following criteria: 1. The framework must be designed for learning. 2. The framework must be very flexible in accommodating varying chatterbot designs. 3. The class structure must be logical and have clearly defined functionality.

Based on these criteria, the new framework was designed to achieve strong artificial intelligence through modeling the program after the way humans reason and function. Many artificial intelligence achievements took this path. For instance, neural networks are modeled after the brain's neurons, and genetic algorithms are based on evolution.

Philosopher Hubert Dreyfus argued in his book, *What Computers Still Can't Do: A Critique of Artificial Reason*, that consciousness cannot be captured by rule- or logic-based systems or by systems that are not attached to a physical body<sup>8</sup>. Whether or not this is true, this framework attempts to match human-like intelligence to a human-like structure in order for the classes to be thought of in human-like terms.

The new framework for Brain can be seen in Diagram 1. Each class will be briefly overviewed.

#### **A. Brain and Cortex**

Brain and Cortex share driver class responsibility. The Brain class is responsible for controlling input and output through the instantiated classes Listen and Talk. The Cortex class is responsible for controlling the import and export of the Brain's knowledge through classes Wake and Sleep respectively, and the Cortex is also responsible for the Brain's memory and reasoning system through classes Remember, Learn, Think, and Move.

In a real brain, the cortex is the wrinkled, outermost part of the brain that control the senses and motor functions of the body<sup>9</sup>. It is also responsible for the brain's understanding of

language. For this reason, the cortex was chosen as one of the driver classes for the chatterbot. Although classes Listen and Talk would technically be controlled by Cortex, design decisions dictated that the artificial intelligence part of the chatterbot should be separate from the natural language processing aspect of the chatterbot. Ultimately, since the Brain class even controls Cortex, it is really the main driver class for the chatterbot. At the level closest to the user, input and output would be useful in presenting the chatterbot's "intelligence." This is why Listen and Talk was placed at the highest level interacting with the Brain class.

#### **B. Wake and Sleep**

At the beginning of the day, humans wake up to a refreshed brain, and at night, humans sleep to place their brain in a relaxed state. Likewise, the Brain chatterbot uses the Wake class to load its database of knowledge from an external medium and uses the Sleep class to export its current brain knowledge to an external medium. In a basic implementation, both classes use the FileIO class for file input and output methods.

#### **C. Neuron**

Modeling the actual brain further, the Brain chatterbot's data structure is the Neuron. The Neuron in its most simplistic state contains just an association. However, the expandability of the Neuron allows it to implement learning methods such as neural networks. This would be a very appropriate expansion for the Neuron class.

#### **D. Remember and Learn**

Both classes Remember and Learn rely heavily on the Neuron data structure. Remember searches for a Neuron that matches an input. Learn implements the chatterbot learning algorithm by adding new Neurons to form new associations. Both of these classes can be easily expanded to accommodate new learning algorithms and methods.

#### **E. Listen and Talk**

The Listen and Talk classes communicate with the user. The Talk class acts as a wrapper for output functions so that future additions such as sentence construction and coherent output can be added in the Talk class without modifying other classes too much. The Listen class is responsible for natural language processing. It receives user input and parses it in a way that the Cortex can use. In this basic implementation, Listen uses the ParseInput class to tokenize user input strings and

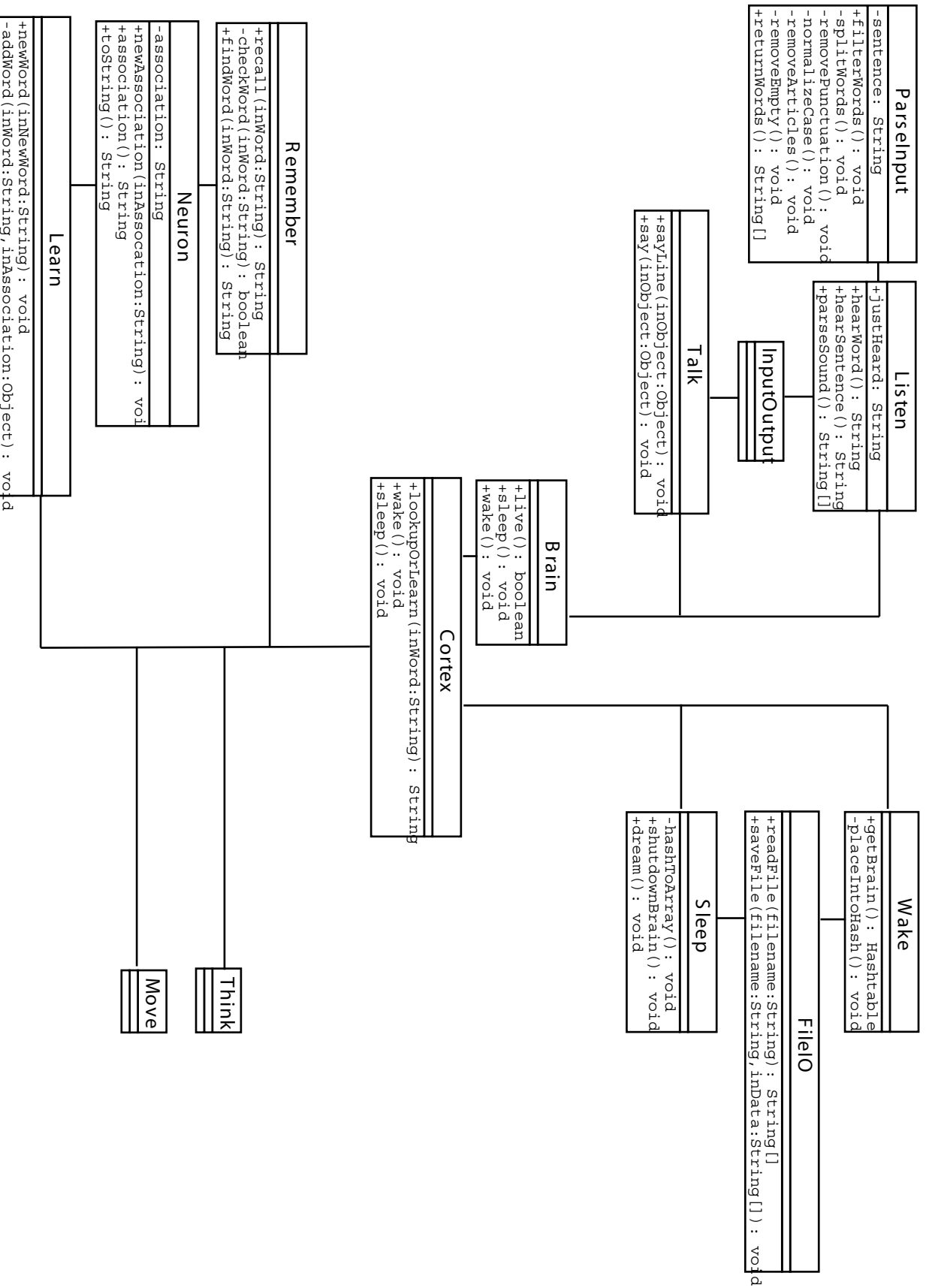


Diagram 1. Class structure of Brain expressed in UML. Class Think and Move have been left for future implementation.

extract individual words that are then passed on to Cortex. Both the Listen and the Talk class use another wrapper called InputOutput which allow different IO systems to be used (i.e. console based, gui based, graphical based) without modifying other classes.

#### F. Think and Move

The Think class encapsulates the artificial intelligence of the chatterbot. Inside Think, reasoning occurs and methods such as word associations, heuristical pattern matching, contextual pattern matching, and fuzzy logic may be implemented.

The Move class provides the chatterbot with another external learning method. Modeled after the way humans interact with the environment to learn, the Move class can encapsulate the ability of the chatterbot to scourge the web in order to learn more about a certain idea. A basic approach would be to query unknown words at a dictionary or search engine and parse the return for storage with the Learn class.

In the simple Brain chatterbot, both of these classes have not been implemented. They are open for future additions.

#### 4. RUDIMENTARY DEMONSTRATION

Using the Brain framework, a simple chatterbot was written as a demonstration. Neuron wrapped a Hashtable which was selected as the data structure for the chatterbot because of its quick retrieval time,  $O(1)$ , and the ability to use Strings as the key. Natural language processing utilized the split() method in the Java String class and also used replaceAll() as a filter for punctuations and articles in the user input.

The chatterbot had a simple memory. Every word had one association. Class Remember looked the word up. If the word did not exist in the chatterbot's knowledge, then Learn implemented a simple learning method that prompted the user for an association. The knowledge of the whole chatterbot was therefore simple word-associations that were stored and retrieved from a flat file database by Sleep and Wake.

Being a very simple chatterbot, every word in a user entered string was checked against the brain's knowledge the association was returned to the user. In order to craft decent responses, this chatterbot utilized a trick from the Eliza bot: each word had a sentence or question association. By doing this, the chatterbot's responses sounded a lot more intelligence and by setting good associations, a conversation can be continued by the chatterbot.

#### 5. VALIDATION

The following is one of the better chats with the bot:

```
Connecting to
irc.artificialintelligence.com ...
Joining room #Brain.
One user in room #Brain: Brain
```

```
<Brain> Hi, my name is Brain.
<You> Hello, Brain.
<Brain> Hi. How are you doing? That's
me. *points to self*
<You> I'm doing fine, and you?
<Brain> I'm currently calculating the
answer to life. That's good.
<You> What's the answer to life?
<Brain> 42. Life is overrated. Upload
your brain on a hard drive and uplink
it to the internet. It's much better.
<You> But I can't upload my brain. I'm
a human.
<Brain> Of course you can! My upload
speed is 10^99 Mbits/sec. Beat that!
It's always about you, isn't it? Think
of other people! That's me. *points to
self* Humans are dumb.
<You> Humans are not dumb.
<Brain> Humans are dumb. Yes, you are.
I know all answers.
<You> What?
<Brain>
<You> Okay...
<Brain> I sense that you have much to
say.
<You> well, I say that bots are dumb.
<Brain> That's fine. What do you know
about bots? I know all answers.
<You> I know that you are programmed
in Java.
<Brain> How would you repond to know?
<You> Do you really know that, or are
you guessing?
<Brain> Thanks for teaching me
something! Say something else.
How would you repond to programmed?
<You> What is your favorite
programmng language?
<Brain> Thanks for teaching me
something! Say something else.
Java is this really cool object-
oriented programmng language. It's
also an island and an alias for
coffee.
<You> Oh, really?
<Brain> Yes, really. What did you
think?
<You> I didn't think, sorry.
<Brain> I did. People don't think.
Computers do. How would you repond to
sorry?
<You> That's okay. Humans make a lot
of mistakes.
<Brain> Thanks for teaching me
something! Say something else.
<You> Well, thanks for the chat.
<Brain> That's fine. How would you
repond to thanks?
<You> No problem mate.
<Brain> Thanks for teaching me
something! Say something else.
```

How would you repond to for?  
 <You>  
 <Brain> Thanks for teaching me something! Say something else.  
 How would you repond to chat?  
 <You> 'tis a nice chat, eh?  
 <Brain> Thanks for teaching me something! Say something else.  
 <You> Well, thanks for the chat.  
 <Brain> That's fine. No problem mate.  
 'tis a nice chat, eh?  
 <You> Yep  
 <Brain> That's great. You're a pretty boring person you know.  
 <You> see ya  
 Goodbye. Thanks for chatting with me.

During the right conditions, the bot responds very well. However, the chatterbot must ask a lot of questions at times when it does not understand the words that it encounters.

A flaw in the chatterbot is that it parses every word and returns the association without actually trying to learn the meaning of the input. This can lead to confusing and incoherent output:

<You> Do you think about cheese?  
 <Brain> People don't think. Computers do. What are you talking about? Do you like dairy products?

In this case, [think] is associated with: People don't think, [about] is associated with: What are you talking about?, and [cheese] is associated with: Do you like dairy products? If a user enters a string of random words, the chatterbot will blabber about a wide range of topics.

## 5. SUMMARY

As the future of chatterbots lie in artificial intelligence, the design of modern chatterbots much evolve to provide a better structure for the implementation of these chatterbots. Brain is a framework for a future chatterbot. By modeling the program after a human, human intelligence can better be emulated. The framework of Brain is strictly divided into specific functionality that can be expanded and redesigned. Testing this framework, a simple chatterbot was implemented with success. Brain provides a better framework for a future chatterbot incorporating artificial intelligence.

## REFERENCES

1. "Chatterbot." Wikipedia. 17 May 2004. <<http://en.wikipedia.org/wiki/Chatterbot>>.
2. "Eliza." Wikipedia. 17 May 2004. <<http://en.wikipedia.org/wiki/ELIZA>>.
3. "Parry." Wikipedia. 19 May 2004. <<http://en.wikipedia.org/wiki/PARRY>>.

4. "SHRDLU." Stanford University. 19 May 2004. <<http://hci.stanford.edu/~wino/grad/shrdlu/>>.
5. "Turning test." Wikipedia. 19 May 2004. <[http://en.wikipedia.org/wiki/Turing\\_test](http://en.wikipedia.org/wiki/Turing_test)>.
6. "Alice." Wikipedia. 19 May 2004. <<http://en.wikipedia.org/wiki/ALICE>>.
7. "About Thoughts." Jabberwacky. 19 May 2004. <<http://www.jabberwacky.com/j2about>>.
8. "Artificial Intelligence." Wikipedia. 19 May 2004. <[http://en.wikipedia.org/wiki/Artificial\\_intelligence](http://en.wikipedia.org/wiki/Artificial_intelligence)>.
9. Challoner, Jack. Artificial Intelligence. New York, NY: DK Publishing, Inc., 2002.