UNIVERSITY OF ASIA PACIFIC

Department of Computer Science & Engineering

Course Title – Artificial INtelligence and Expert Systems.

Course Code – CSE-403.

Topic – Study on ELIZA, Cog-MIT, Deep Blue, AlphaGo & MYCIN

SUBMITTED BY

Shawan Das.

ID – 19101020

SectSection – A

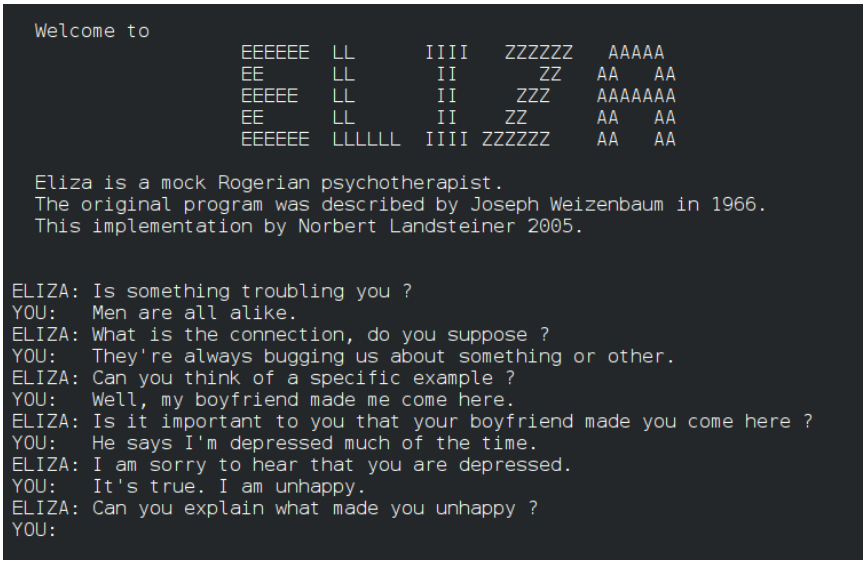
SUBMITTED TO

Dr. Nasima Begum

University of Asia Pacific

Date of Submission – 30–07–2022

ELIZA is an early “Natural Language Processing computer program” created from 1964 to 1966 at the “MIT Artificial INtelligence Laboratory” by Joseph Weizenbaum. It was created to demonstrate the superficiality of communication between humans and machines. For conversation, it uses “Pattern Matching” and substitution methodology that gives users an illusion of understanding on the part of the program. But it had no built-in framework for contextualizing events. ELIZA was created to provide a parody of the responses of non-directional psychotherapists in an initial psychiatric interview and to demonstrate that the communication between man and machine was superficial. ELIZA was well known for acting in the manner of a psychotherapist. The speech patterns are due to the data and instructions supplied by the Doctor script.

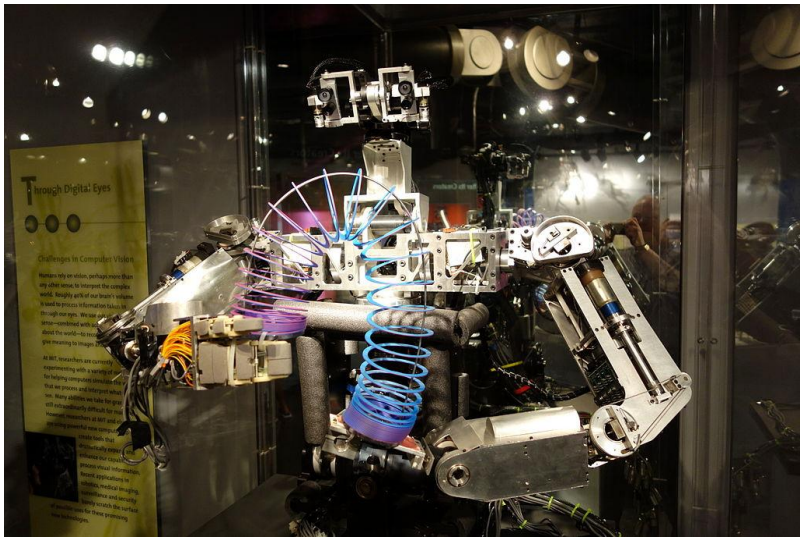


Weizenbaum originally wrote ELIZA in MAD-Slip for CTSS on an IBM-7094, as a program to make natural-language conversation possible with a computer. ELIZA starts its process of responding to an input by a user by first examining the text input for a “keyword”. A keyword is a word designated as important by the acting ELIZA script, which assigns to each keyword a precedence number, or a RANK, designed by the programmer. If such words are found, they are put into a keystack, with the keyword of the highest RANK at the top. For example when the DOCTOR script encounters words such as "alike" or "same", it would output a message pertaining to similarity, in this case "In what way?", as these words had high precedence numbers. This also demonstrates how certain words, as dictated by the script, can be manipulated regardless of contextual considerations, such as switching first-person pronouns and second-person pronouns and vice versa, as these too had high precedence numbers. Such words with high precedence numbers are deemed superior to conversational patterns and are treated independently of contextual patterns. The example that Weizenbaum gives is the input "You are very helpful", which is transformed to "You are very helpful". This is then broken into (1) empty (2) "You" (3) "are" (4) "very helpful". The decomposition rule has broken the phrase into four small segments that contain both the keywords and the information in the sentence.

In 2021 Jeff Shrager searched MIT's Weizenbaum archives, along with MIT archivist Myles Crowley, and found files labeled Computer Conversations. These included the complete source code listing of ELIZA in MAD-SLIP, with the DOCTOR script attached. The Weizenbaum estate has given permission to open-source this code under a Creative Commons CC0 public domain license. Another version of Eliza popular among software engineers is the version that comes with the default release of GNU Emacs, and which can be accessed by typing `M-x doctor` from most modern Emacs implementations.

Algorithm: ELIZA uses “pattern matching” and substitution methodology . It was originally written in MAD-Slip.

Cog was a project at the Humanoid Robotics Group of the Massachusetts Institute of Technology. It was based on the hypothesis that human-level intelligence requires gaining experience from interacting with humans, like human infants do. This in turn required many interactions with humans over a long period. Because Cog's behavior responded to what humans would consider appropriate and socially salient environmental stimuli, the robot was expected to act more human. This behavior also provided the robot with a better context for deciphering and imitating human behavior. This was intended to allow the robot to learn socially, as humans do.



The purpose of this project was to Study Theories of cognitive science and artificial intelligence.

The Goals of Cog projects are:

- To design and fabricate a humanoid face for each robot that fosters suitable social contact between robots and humans.
- To create a robot which is capable of interacting with humans and objects in a human-like way.
- To develop a relatively general system by which Cog can learn causal relations between commands to its motors and input from its sensors (primarily vision and mechanical proprioception).

- To shift the robot aesthetic to a design language that utilizes strong curvilinear and organic forms through state of the art design processes and materials.

As of 2003, all development of the project had ceased.

Today Cog is retired to the Massachusetts Institute of Technology museum.

Deep-Blue was a chess-playing expert system run on a unique purpose-built IBM supercomputer. It was the first computer to win a game, and the first to win a match, against a reigning world champion under regular time controls. Development began in 1985 at Carnegie Mellon University under the name ChipTest. It then moved to IBM, where it was first renamed Deep Thought, then again in 1989 to Deep Blue. It first played world champion Garry Kasparov in a six-game match in 1996, where it lost four games to two. It was upgraded in 1997 and in a six-game re-match, it defeated Kasparov by winning three games and drawing one. Deep Blue's victory was considered a milestone in the history of artificial intelligence and has been the subject of several books and films.

Deep Blue's evaluation function was initially written in a generalized form, with many to-be-determined parameters. Values for these parameters were determined by analyzing thousands of master games. The evaluation function was then split into 8,000 parts, many of them designed for special positions. The opening book encapsulated more than 4,000 positions and 700,000 grandmaster games, while the endgame database contained many six-piece endgames and all five and fewer piece endgames. An additional database named the “extended book” summarizes entire games played by Grandmasters. The system combines its searching ability of 200 million chess positions per second with summary information in the extended book to select opening moves.

Deep Blue used custom VLSI chips to parallelize the alpha-beta search algorithm,an example of GOFAI (Good Old-Fashioned Artificial Intelligence). The system derived its playing strength mainly from brute force computing power. It was a massively parallel IBM RS/6000 SP Supercomputer with 30 PowerPC 604e processors and 480 custom 600 µm CMOS VLSI "chess chips" designed to execute the chess-playing expert system, as well as FPGAs intended to allow patching of the VLSIs (which ultimately went unused) all housed in two cabinets. Its chess playing program was written in C and ran under the AIX operating system. It was capable of evaluating 200 million positions per second, twice as fast as the 1996 version. In 1997, Deep Blue was upgraded again to become the 259th most powerful supercomputer according to the TOP500 list, achieving 11.38 GFLOPS on the parallel high performance LINPACK benchmark.

Algorithm: Deep Blue used custom MLSI chips to parallelize the alpha-beta search algorithm, an example of GOFAI(Good Old-Fashioned Artificial Intelligence).

AlphaGo is the first computer program to defeat a professional human Go player, the first to defeat a Go world champion, and is arguably the strongest Go player in history. It was developed by DeepMind Technologies, a subsidiary of Google. Subsequent versions of AlphaGo became increasingly powerful, including a version that competed under the name Master. After retiring from competitive play, AlphaGo Master was succeeded by an even more powerful version known as AlphaGo Zero, which was completely self-taught without learning from human games.

AlphaGo and its successors use a Monte Carlo tree search algorithm to find its moves based on knowledge previously acquired by machine learning, specifically by an artificial neural network (a deep learning method) by extensive training, both from human and computer play. A neural network is trained to identify the best moves and the winning percentages of these moves. This neural network improves the strength of the tree search, resulting in stronger move selection in the next iteration.

Algorithm: AlphaGo and its successors use a “Monte Carlo Tree Search”-algorithm guided by a “value network” and a “policy network”- implemented by using deep neural network technology.

An early version of AlphaGo was tested on hardware with various numbers of CPUs and GPUs, running in asynchronous or distributed mode. Two seconds of thinking time was given to each move. The resulting Elo ratings are listed below. In the matches with more time per move higher ratings are achieved.

Configuration and performance

Configuration	Search threads	No. of CPU	No. of GPU	Elo rating
Single[4] p. 10–11	40	48	1	2,181
Single	40	48	2	2,738
Single	40	48	4	2,850
Single	40	48	8	2,890
Distributed	12	428	64	2,937
Distributed	24	764	112	3,079
Distributed	40	1,202	176	3,140
Distributed	64	1,920	280	3,168

As of 2016, AlphaGo's algorithm uses a combination of machine learning and tree search techniques, combined with extensive training, both from human and computer

play. It uses Monte Carlo tree search, guided by a "value network" and a "policy network," both implemented using deep neural network technology. A limited amount of game-specific feature detection pre-processing (for example, to highlight whether a move matches a nakade pattern) is applied to the input before it is sent to the neural networks. The networks are convolutional neural networks with 12 layers, trained by reinforcement learning

The system's neural networks were initially bootstrapped from human gameplay expertise. AlphaGo was initially trained to mimic human play by attempting to match the moves of expert players from recorded historical games, using a database of around 30 million moves. Once it had reached a certain degree of proficiency, it was trained further by being set to play large numbers of games against other instances of itself, using reinforcement learning to improve its play. To avoid "disrespectfully" wasting its opponent's time, the program is specifically programmed to resign if its assessment of win probability falls beneath a certain threshold; for the match against Lee, the resignation threshold was set to 20%.

MYCIN is an expert system developed at Stanford University in the mid1970's to aid physicians in the selection of antibiotics for patients with severe infections. In several different evaluations, MYCIN has demonstrated an ability to perform at or near the level of expert physicians (JO). As we shall emphasize later, technical performance, narrowly defined, is not the only criterion for acceptability, and current research on descendants of MYCIN is aimed at overcoming some of its deficiencies. However, MYCIN's combination of competent performance and conceptual simplicity make it particularly convenient as an illustrative example.

In MYCIN all knowledge on infectious diseases is represented in the form of rules. The current system contains about 500 rules that deal with the diagnosis and treatment of bacteremia (bacteria in the blood) and meningitis (infection in the cerebrospinal fluid). In the program these rules are expressed in a stylized form that simplifies computer interpretation and facilitates their translation into English for human examination. The following is an example of a MYCIN rule expressed in English:

If (i) the infection is meningitis and

(ii) organisms were not seen in the stain of the culture and

(iii) the type of infection may be bacterial and

(iv) the patient has been seriously burned, then there is suggestive evidence that Pseudomonas aeruginosa is one of the organisms that might be causing the infection.

To use such general knowledge about infectious diseases, MYCIN must obtain specific knowledge about a particular patient. These patient data are stored in a dynamic

database in the form of "attribute-object-value" triples. For example, the database might contain the fact that the stain (attribute) of a particular organism (object) is Gram-negative (value) or that the type of a particular infection is bacterial.

MYCIN works in two phases, diagnosis and therapy. In its diagnosis phase, the program's main goal is to apply its rules to determine the identity 0 of all suspicious organisms. When it attempts to apply a rule, it queries its database to see whether the needed facts are avail· able. Thus, to apply our example rule, MYCIN would begin by accessing the database to see what is known about the infection of the patient. If the infection were known not to be meningitis, the rule would be discarded at once. However, if the infection were thought to be meningitis, the program would check the other parts of the premise in tum. If all parts were satisfied, MYCIN would apply the rule, concluding that the organism's identity might be Pseudomonas and thereby updating the database.