GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Building of an Arificial Intelligence

A Report for the Evaluation 3 of Project 1

Submitted by

DHANANJAY SINGH (1713121001 / 17SCSE121001)

in partial fulfillment for the award of the degree of

Bachelor of Computer Application

IN

Industry Oriented Program

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the Supervision of

S.JANARTHANAN
Assistant Professor/SCSE
Galgotias University

APRIL / MAY- 2020

GALGOTIAS UNIVERSITY

SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report **"Acquiring Digital Evidence from Botnet Attacks: Procedures and Methods"** is the bonafide work of **"DHANANJAY SINGH (1713121001)"** who carried out the project work under my supervision.

SIGNATURE OF SUPERVISOR

Dr. MUNISH SHABARWAL,
PhD (Management), PhD (CS)

School of Computing Science & Engineering

SIGNATURE OF SUPERVISOR

S.JANARTHANAN, Assistant Professor
Galgotias University

School of Computing Science & Engineering

Table of Contents

Artificial Intelligence

abstract

Artificial intelligence (AI) is the intelligence of machines and the branch of computer science which aims to create it. Major AI textbooks define the field as "the study and design of intelligent agents,"where an intelligent agent is a system that perceives its environment and takes actions which maximize its chances of success. John McCarthy, who coined the term in 1956, defines it as "the science and engineering of making intelligent machines.

The field was founded on the claim that a central property of human beings, intelligence—the sapience of Homo sapiens—can be so precisely described that it can be simulated by a machine. This raises philosophical issues about the nature of the mind and limits of scientific hubris, issues which have been addressed by myth, fiction and philosophy since antiquity. Artificial intelligence has been the subject of breathtaking optimism, has suffered stunning setbacks and, today, has become an essential part of the technology industry, providing the heavy lifting for many of the most difficult problems in computer science.

AI research is highly technical and specialized, so much so that some critics decry the "fragmentation" of the field. Subfields of AI are organized around particular problems, the application of particular tools and around long standing theoretical differences of opinion. The central problems of AI include such traits as reasoning, knowledge, planning, learning, communication, perception and the ability to move and manipulate objects. General intelligence (or "strong AI") is still a long term goal of (some) research.

Introduction

I have chosen this topic to spotlight on one of the most technological trend these days known as AI *(Artificial Intelligent*). Therefore; I will discuss some of the most important aspects related to *AI* in which it will help in a better understanding of Artificial Intelligent and both its advantages and disadvantages to be able to protect ourselves from the upcoming technological trend. This paper will also discuss some of the algorithms used in AI systems.

Over the past few years, deep learning has gone from an esoteric branch of AI, focusing on theory, to being so mainstream that even an actress from Twilight has published an academic paper on it. This rise of deep learning over traditional machine learning methods has also led to the creation of a host of new platforms designed to help businesses improve work and workflow. The idea of creating a machine that can 'think' in a way similar to the human brain is as old as modern computing. In 1950, in his seminal paper 'Computing Machinery and Intelligence', computing pioneer Alan Turing laid out several criteria to assess whether a machine could be said intelligent, which has since become known as the 'Turing test'. Thirty-six years later, computer scientist and cognitive psychologist Geoff Hinton demonstrated how an artificial neural network with several layers could be trained to learn nonlinear functions. In this model, each succeeding layer in the network 'learns' from the previous layer. In 2006, Hinton devised a way to train similar networks to pass along only information related to specified parameters, adding and training new layers until a deep neural network was created. But it was the more recent appearance of large, high-quality labelled datasets, distributed computing and the applications of graphics processing unit (GPU) computing which has powered the recent, rapid advancement of deep learning. It is now possible to create deep learning neural networks which operate fast enough and accurately enough to have practical, real-world uses. Because of this, we are experiencing a paradigm shift in computing, an AI boom in which companies are spending billions to develop deep learning AI technology. Springwise has been following this shift since its beginning, and we have continually highlighted businesses at the forefront of the deep learning revolution. At Springwise, we have found that, while there are a huge number of businesses that could benefit from deep learning, in general, for deep learning to offer a practical business solution, a company must have a need for finding complex relationships in large amounts of data, and a recurring need for predicting things that either cut costs or create value. Some of the most relevant deep learning-powered business transformations we have found involve improved targeting of sales and marketing, better informed decision-making, increased productivity and automation of retail.

AI modelling and simulation techniques are already being used to help drive sales in a number of ways. Platforms providing real-time data gathering, forecasting, and trend analysis can offer greater insight into buyer behaviour. Artificial intelligence used in conjunction with CRM systems can automate functions such as contact management, data recording and analyses, and lead ranking, while AI-enabled buyer modelling can provide a prediction of a customer's lifetime value. Recommendation systems are also used to learn content preferences and push content that fit those preferences, allowing more targeted sales. While sales may appear to be primarily about relationships, in reality it is one of the most data driven areas of business. From names and contact numbers, to information on customer behaviour and purchasing, sales depends on the ability to find and identify the best sales leads, and the leads most likely to be converted. Traditionally, this information was collected by individual salespeople, but deep learning has allowed AI software to take over the time-consuming task of identifying contact information and lead scoring. Start-up Radius has spent years building a network-effect-driven data source called The Network of Record. The Network is built using data collected from more than 1 billion anonymized and aggregated business interactions. This data is then analysed by Radius' AI software to provide intelligence to sales teams which can help them directly target only the most qualified sales prospects for any product. Local pricing knowledge is a sales challenge faced by companies that operate in multiple markets around the world. Local businesses can adjust prices based on local knowledge, but this is more difficult for businesses that operate globally, or whose customers are primarily online. In online shops, businesses tend to offer the same prices to every visitor, adjusted for local currency – regardless of the competitiveness of their prices in different regions. While the sales team may be setting prices based on those of other online shops, customers may be comparing them to local alternatives the sales team may not even know about. Darwin Geo-Pricing has developed one solution to this, using a deep learning algorithm to retrieve the online shopper's location and combine this with data mining to adjust the prices each customer is offered. Darwin's machine learning system continuously performs exploratory pricing, offering different visitors higher or lower prices than others in the same location. By comparing the sales performance at different price levels, Darwin can determine the optimal price level to pitch at each customer.

Another sales challenge is how to cut through the advertising noise and clutter most customers are faced with daily in order to deliver more personal and relevant content or products to each consumer. The Weather Company, which was purchased last year by IBM and rebranded as IBM Watson Advertising, is working to cut through this background ad 'chatter' with the first-ever consumer use of IBM Watson AI technology for advertising purposes. Watson Ads uses natural language processing and machine learning to listen, think and respond to consumers' questions with personalized answers in real time. Each Watson ad is able to understand the consumer's questions and respond in ways that sound like natural, human language. Consumers clicking on a Watson ad for, say, a Toyota Prius Prime, can then interact with the ad to receive individualized answers to questions about the brand, other Toyota brands and automobile shopping. The program can also steer the conversation in a natural way, similar to how real people interact. Another approach to advertising made possible by machine learning algorithms is predictive advertising. Predictive advertising is a subset of predictive analytics, which uses machine learning to predict future outcomes based on behavioural patterns in historical data. In ad tech, predictive analytics is used to help businesses target audience segments based on behavioural signals, personalise ads to be more relevant to individual users, or optimise ad bids based on user data. There are a number of companies vying to develop platforms for predictive advertising, including Programmai, which uses AI to identify and target new prospects with ads based on what is already known about a company's existing customers or visitors. The company's software can notify a business of how likely someone is to make a purchase, calculate future lifetime value, and automate ad bidding strategy decisions – allowing companies to focus ads on where they will have the biggest impact. A similar approach is taken by Indianapolis-based Demandjump, which uses analytics to help marketers decide where to place ads in order to take advantage of up-to-the-minute trends. Their artificial intelligence marketing platform, dubbed TrafficCloud, collects data on page views, and can link customer activity across devices in order to present a detailed analysis of traffic between sources. The company's analytics can then pinpoint which sites have the greatest influence in a brand's competitive area, and which sites can help drive the highest number of customers to the brand. DemandJump's proprietary algorithms use machine learning, graph theory, algebraic topology, and natural language processing, to "map each client's entire digital ecosystem" and show marketers exactly what to do next to maximize revenue growth and stay ahead of market changes.

Business decision making is a critical process, at all levels. Many business experts suggest that the future of AI in executive decision-making actually lies in a partnership, with humans defining the questions to be asked, or problems to be tackled, and having a final say on the best answer for their business, while AI is used to analyse large volumes of data to provide a basis for the decision. For an example of why humans will not disappear from the decision-making process, we need look no further than Uber. When the companies' AI pricing algorithm doubled ride prices in London during a major terror attack in 2017, the company suffered from the resulting bad publicity and accusations of price gouging. The algorithm is better at setting prices in real time than humans, but a human would have been more likely to realise that doubling prices during a tragedy is not good business. This partnership approach can be seen in the development of a number of recent AI decision-making platforms. In Denmark, start-up Corti has developed an AI platform that listens in to phone calls to emergency services to help detect signs of a heart attack, including signals in verbal communication, tone of voice and breathing patterns. Corti compares the data from each call against millions of previous emergency calls to find patterns indicating a possible heart attack. Corti can then alert the dispatcher to the possible need for an ambulance. This early recognition is important because the chance of survival decreases by around 10 percent for each minute treatment is delayed after a heart attack occurs, and because many heart attack symptoms, such as shoulder pain, can be easily dismissed as a minor ailment. Partnership in decision making was also envisioned by the makers of VALCRI (Visual Analytics for sense making in Criminal Intelligence analysis), which is designed to aid the police and criminal investigators. The system is the result of an EU-wide research project involving more than 15 organisations - including law firms and behavioural scientists. VALCRI analyses criminal records and data sets related to social and legal systems to suggest possible areas of inquiry for detectives and criminologists, possibly reducing the footwork and human error involved in investigation. Because the AI algorithm learns more as more data is added, it can also help identify patterns that could help head off crimes before they are committed.

data points across hundreds of plays, and it is impossible for a coach to look at everything at once. At Springwise, we have highlighted how Australian innovator Flixsense is developing a platform, currently in Beta, which is able to track every aspect of football and cricket matches, such as passing patterns and ball tracking. Flixsense's AI learns to recognize people, action and objects from videos and can instantly identify key events from a match, helping coaches to anticipate what move the competing team will likely be making next.

The ability of deep learning algorithms to rapidly analyse large amounts of data and identify patterns has the potential to reshape how businesses respond to operational challenges and inefficiencies. One of the ways in which AI is already leading to improved productivity is through automation of complex tasks that require physical adaptability and agility. This type of work is perhaps most commonly found in construction, which has always suffered from lagging productivity. One reason for this is that large projects involve many different sub-teams, each installing different components. It is impossible for managers to accurately track the tens of thousands of components or elements on such projects, and it can be weeks or months before errors are discovered. In response, California-based start-up Doxel has developed an artificial intelligence system that operates drones and robots to monitor every inch of a construction project and alert managers of potential problems. Doxel's deep learning algorithms can also measure the quality and progress of the work in real time, comparing it to the original budget and schedule. An Israeli start-up called 3DSignals is also working to improve productivity in heavy industry by using AI guided wireless ultrasonic sensors to track changes in the sounds made by machinery. By analysing the changing sounds of the machinery and comparing them to data sets about the sounds of normallyworking and malfunctioning machinery, the algorithm can spot potential breakdowns before they occur. Being able to predict problems earlier can, in turn, allow operators to save time and money. Productivity improvements can also be driven through better workflow management. According to a 2017 report by market research firm Research and Markets, the market for workflow management systems is estimated to grow from $3.5 billion USD in 2016 to almost $9.9 billion USD in 2021 – a compound growth rate of 21 percent. A major part of this growth is likely to be in AI-driven systems that can help companies achieve productivity gains through greater automation of workflow tasks.

History of Artificial Intelligence:

Artificial Intelligence was first proposed by John McCarthy in 1956 in his first academic conference on the subject. The idea of machines operating like human beings began to be the center of scientist’s mind and whether if it is possible to make machines have the same ability to think and learn by itself was introduced by the mathematician Alan Turing. Alan Turing was able to put his hypotheses and questions into actions by testing whether “*machines can think*”? After series of testing (later was called as Turing Test) it turns out that it is possible to enable machines to think and learn just like humans. Turing Test uses the pragmatic approach to be able to identify if machines can respond as humans. (“Smith”, (n.d.)).

Description Artificial Intelligence

Artificial Intelligence is: the field of study that describe the capability of machine learning just like humans and the ability to respond to certain behaviors also known as (A.I.). The need of Artificial Intelligence is increasing every day. Since AI was first introduced to the market, it has been the reason of the quick change in technology and business fields. Computer scientist are predicting that by 2020, *“85% of customer interactions will be managed without a human”.* (“Gartner”, (n.d.)). This means that humans simple request will depend on computers and artificial intelligence just like when we use Siri or Galaxy to ask about the weather temperature. It is very important to be prepared for AI revelation just like UAE have by installing a state minister for AI in Dubai.

Pros and Cons of Artificial Intelligence

AI offers reliability, cost- effectiveness, solve complicated problems, and make decisions; in addition, AI restrict data from getting lost. AI is applied nowadays in most fields whether business or engineering. One of the great tools in AI is called “reinforcement learning” which is based on testing success and failure in real life to increase the reliability of applications. Unfortunately, AI is limited with its capability and functionality. (“Sadek”,(n.d.))

Although Artificial Intelligence made our lives much easier and saved us more time than ever, scientists are predicting that by the huge dependency on AI humanity could extinct. Scientists argue that by having a AI machines, people will be jobless and that will conclude in losing the sense of living. Since machines are learning and doing thigs more efficiently and effectively in a timely manner, this could be the reason of our extinction.

Existing System

AI is mainly based on algorithms and models as a technique which is designed based on scientific findings such as math, statists, and biology (Li& Jiang, (n.d.)). AI works based on several models such as: Ant Colony Algorithm, Immune Algorithm, Fuzzy Algorithm, Decision Tree, Genetic Algorithm, Particle Swarm Algorithm, Neural Network, Deep Learning and in this report, I will discuss some of the most known models which are: Support Vector Machine, and the Artificial Neural Network.

- Support Vector Machine (SVM) where it is used to build a classification model by finding an optimal hyperplane based on a set of training examples as shown in (figure A-1). It is also have been used for pattern classification and trend prediction lots of applications for instance: power transformer fault diagnosis, disease diagnosis and treatment optimization. (Li& Jiang, (n.d.)).

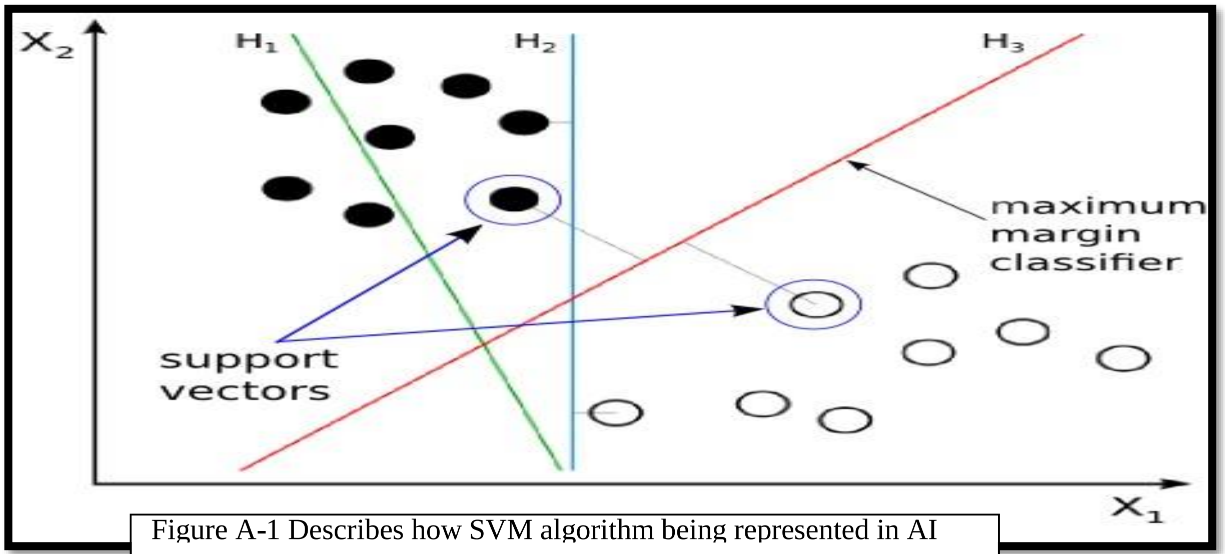


Figure A-1 Describes how SVM algorithm being represented in AI

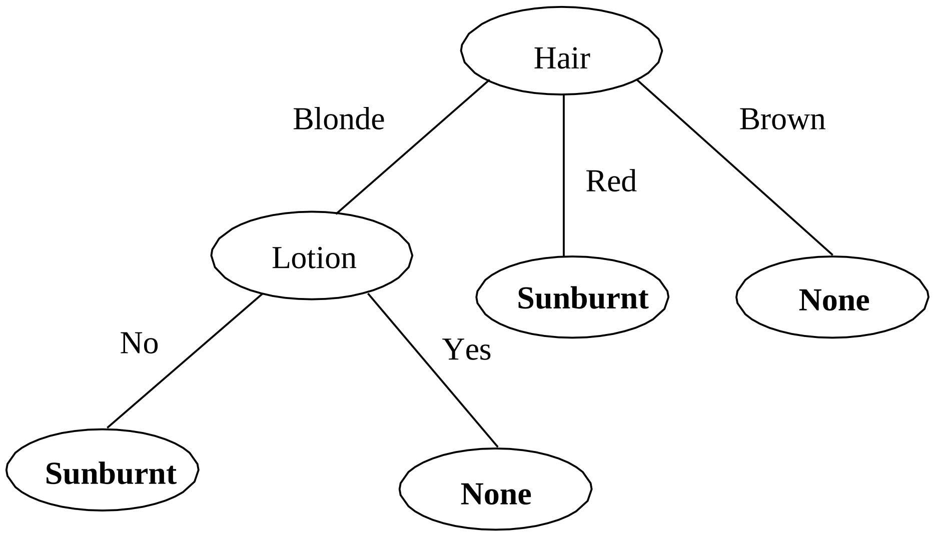
- Artificial Neural Network (ANN) is a representative model of understanding thoughts and behaviors in terms of physical connection between neurons. ANN has been used to solve variety of problems through enabling the machine to build mathematical models to be able to imitate natural activities from brains perspective as shown in (figure A- 2). By using this algorithm, the machine will be able to identify the solution of any problem just like human’s brain.

Proposed System

Machine Learning by Building Decision Trees Since the invention of computers a lot of efforts have been made in order to make them learn. If we can program a computer to learn – that is, to improve its performance through experience - the consequences will be far-reaching. For example, it will be possible to make a computer ordinate an optimal treatment for a new disease using its past experience from treatment of a series of related diseases. Providing learning capabilities to computers will lead to many new applications of computers. Furthermore, knowledge about automation of learning processes may give insight in humans' ability to learn. Unfortunately, we don't know yet how to make computers learn as well as humans. However, in recent years a series of algorithms has appeared which now makes it possible to successfully automate learning in some application areas. For example, one the most efficient algorithms for speech recognition are based on machine learning. Today the interest in machine learning is so great that it is the most active research area in artificial intelligence. The area may be divided into to sub areas, symbolic and non-symbolic machine learning. In symbolic learning the result of the learning process is represented as symbols, either in form of logical statements or as graph structures. In non-symbolic learning the result is represented as quantities, for example as weights in a neural network (a model of the human brain). In recent years the research in neural networks has been very intensive and remarkably good results have been achieved. This is especially true in connection with speech and image recognition. But research in symbolic machine learning has also been intensive. An important reason for this is that humans can understand the result of a learning process (in contrast to neural networks), which is significant if one should trust the result. The following two projects deal with symbolic machine learning and are both using socalled induction. Logical deduction is the process of learning from examples. The goal is to reach a general principle or conclusion from some given examples. Here is a simple example of induction. Suppose you see a set of letterboxes that are all red. By induction you may conclude that all letterboxes in the world are red (including letterboxes that you haven't seen). Decision trees are one of the most applied methods for leaning by induction. The general principle of decision trees is best illustrated through and example. Suppose that you don't know what causes people to be sunburnt. You observe a series of persons and register some of their features, among these whether or not they are sunburnt. The observations are given in the table blow.

Name	Hair	Height	Weight	Lotion	Effect
Sarah	Blonde	Average	Light	No	Sunburnt
Dana	Blonde	Tall	Medium	Yes	None
Alex	Brown	Short	Medium	Yes	None
Annie	Blond	Short	Medium	No	Sunburnt
Emily	Red	Average	Heavy	No	Sunburnt
Pete	Brown	Tall	Heavy	No	None
John	Brown	Medium	Heavy	No	None
Kate	Blonde	Small	Light	Yes	None

From this table we can construct the following decision tree.



This decision tree may be used to classify a given person, i.e., to predict whether or not he will get sunburnt. Start at the top of the tree and answer the question one after another, until you reach a leaf. The leaf stores the classification (Sunburnt or None).

In the present case the decision tree agrees with our intuition about factors that are decisive for getting surnburnt. For example, neither a person’s weight nor height plays a role.

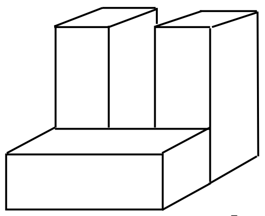
It is often possible to construct more than one decision tree that agrees with the observed data. However, not all of them may be equally well suited for making generalizations, i.e., to classify examples outside the set of examples used to build the tree. How do we build the best tree? Using the so-called *ID3 algorithm*, one of the most effective algorithms for induction, may solve this problem. The algorithm builds a tree while striving at as simple a tree as possible. The assumption is that a simple tree performs better than a complex tree when unknown data are to be classified. The simplification algorithm is based on *information theory.*

Machine Learning by Building Semantic Nets

One of the first successful machine learning algorithms was developed in 1972 by P. H. Winston. The algorithm is able to learn a concept by analyzing the differences that occur in a sequence of observations. For example, the concept of an “arch” can be learned as follows.

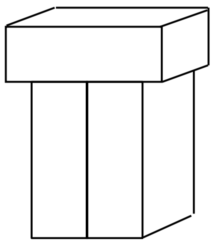
First the algorithm is presented for a series of examples of arrangements of blocks. Some of the arrangements represent an arch. These are called *positive examples*. Others do not represent an arch. They are called *neagtive examples.*

Suppose the following examples are presented to the algorithm:

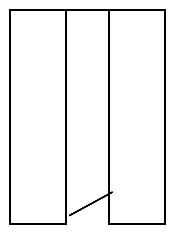


Non-arch

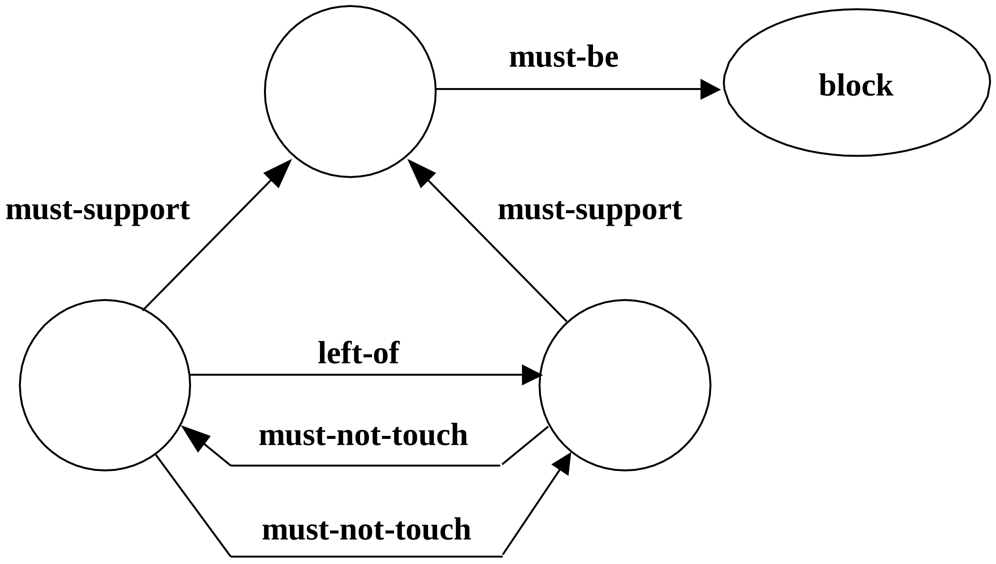
Non-arch



Arch



Then the algorithm will conclude that an arch consists of a block that is supported by two non-touching box-formed blocks. The conclusion is represented by a graph as the following (a so-called *semantic net*).



The goal of this project is to implement Winston’s original algorithm and test it on one or more example problems (including the arch learning problem).

Representation and use of *knowledge* plays a central role in artificial intelligence. Use of knowledge is a prerequisite for intelligent behavior.

Many formalisms for representing knowledge have been developed. Among these, *logic* is the most widely used. The logical formalism is often preferred since it makes it possible in a simple way to deduce new knowledge from existing knowledge. By this formalism it is possible to prove that a statement logically follows from a set of given statements.

Consider the following two statements

All humans are mortal.
Socrates is a human.

From these we can deduce that

Socrates is mortal.

The logical statements above may be expressed as follows in the logical formalism called *first-order predicate logic*.

$\forall x$: human(x) $\rightarrow$ mortal(x).
human(Socrates).

mortal(Socrates)

The first statement has the following meaning: for every x, if x is a human, then x is mortal.

A proof is found by applying one or more logical inference rules. For example, the proof above is found by the inference rule called *modus ponens*: given the following two statements

$P \rightarrow Q$,
P.

we may conclude

Q.

A breakthrough in automatic theorem proving was made in 1965 when J. A. Robinson [1] showed that all proofs in first-order predicate logic can be made by application of only one inference rule, *resolution* (a generalization of modus ponens).

Resolution is a method of proof by contradiction. In order to prove that a conclusion follows from a set of statements the conclusion is negated, and attempts are made two show that the negated conclusion contradicts with the other sentences. If it can be shown that the negated conclusion is inconsistent with the other sentences, then we can conclude that the original conclusion must be valid.

In this connection it is worth mentioning that resolution forms the basis of the logic programming language PROLOG – one of the most used languages for programming artificial intelligence systems.

Space does not allow a detailed description of first-order predicate logic and resolution. See the references below.

The goal of present project is an implementation of a program for automatic theorem proving. The program should be able to read a set of logical sentences, covert these to an internal form, called *clause form*, and prove that one of the statements logically follows from the other statements. The proof is made by resolution.

Dependent on the resources of the project group input and conversion may be included or excluded. The paper by Boyer and Moore [2] is a good starting point for the implementation of the proof part of the program.

Expert Systems

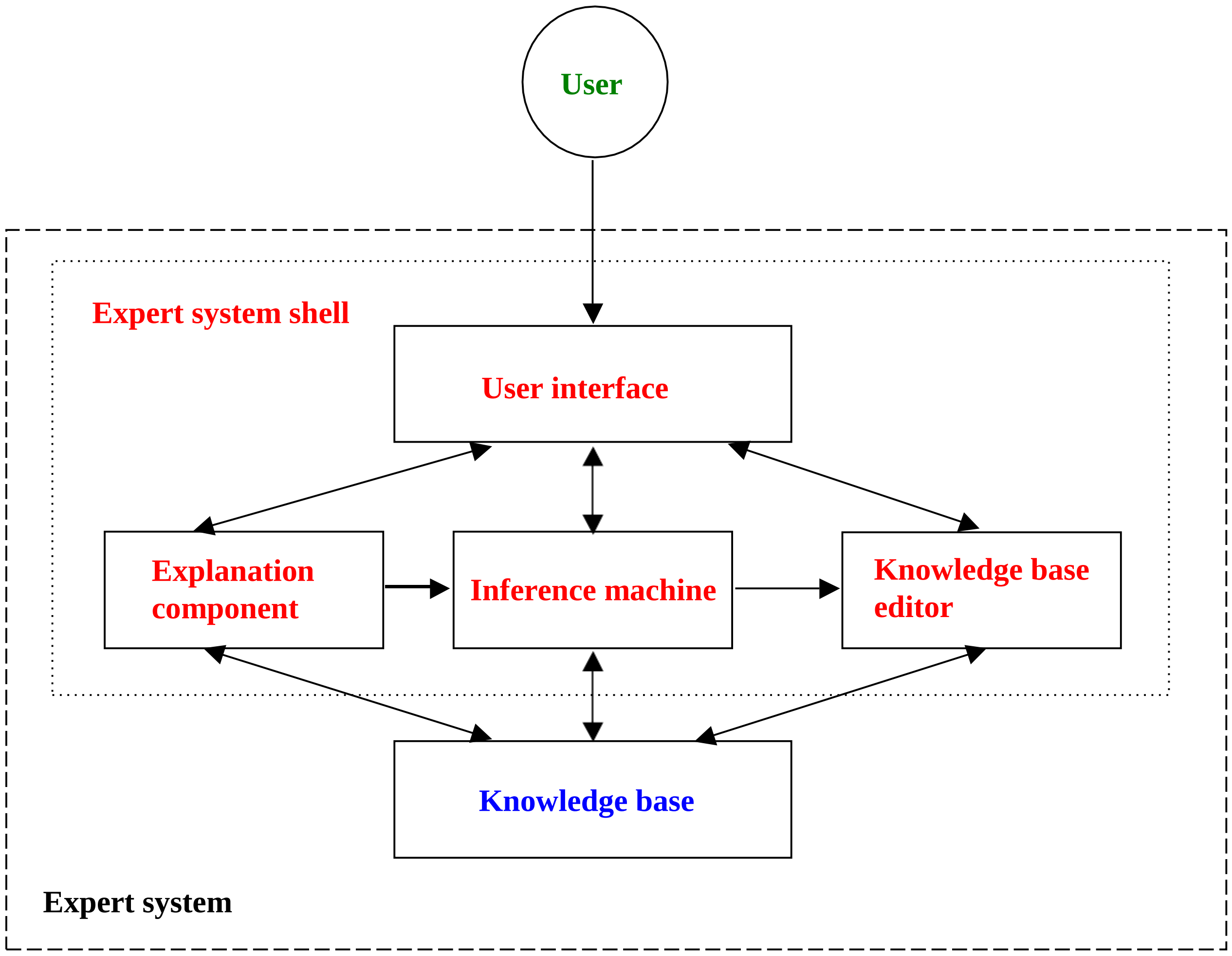
Human experts are able to solve problems at a high level because they exploit knowledge about their area of expertise. This fact provided the background for the design of programs with *expert-based* problem solving capabilities. An *expert system,* which such a program is often called, uses specific knowledge about an area in order to obtain competence comparable to that of a human expert. The specific knowledge may be obtained by interviewing one or more experts in the area in question.

The area “expert systems” is probably the sub-area of artificial intelligence that has achieved the greatest commercial success. Today expert systems are used in a large number of subject areas, ranging from medicine, chemistry, and geology to law, politics, and economics. Any area in which decisions are to be made is a potential application are for expert systems.

Below are listed examples of application areas [1].

1. Interpretation:	drawing hig-level conclusions based on data.
2. Prediction:	projecting possible outcomes.
3. Diagnosis:	determining the course of malfunction, disease, etc.
4. Design:	finding best configuration based on criteria.
5. Planning:	proposing a series of actions to achieve a goal.
6. Monitoring:	comparing observed behavior to the expected behavior.
7. Debugging and Repair:	prescribing and implementing remedies.
8. Instruction:	assisting students in learning.
9. Control:	governing the behavior of a system.

The figure on the next page shows the most import components of an expert system.



The *knowledge base* is the central component of an expert system. It stores knowledge in the form of facts and rules. Usually predicate logic is used for this purpose. The knowledge base may also store the *confidence level* that a fact or rule is true or valid.

The *inference machine* is used to make logical inferences from stored knowledge.

The *user interface* is used to exchange information with the user of the system, for example, for input of questions and output of answers.

The *explanation component* provides the user with information about the reasoning of the system, including a report of how answers were found by the system.

The *knowledge base editor* is used for building and maintaining a knowledge base.

The explicit separation of the knowledge base component from the rest of system makes it possible to reuse all other components in the development of expert systems. An *expert system shell* contains all components in the figure above, except for the knowledge base. Thus, a developer of an expert system may use an “empty” shell and just add a relevant knowledge base. This has the advantage that the developer may fully concentrate in gathering knowledge and building the knowledge base. All other components are available and need not to be changed.

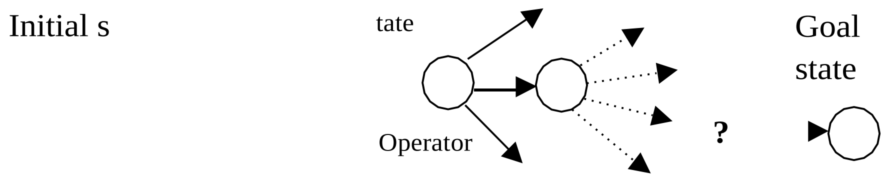
In the present project a rule-based expert system shell is to implemented and tested. As a starting point for the implementation reference [2] may be used with advantage.

Problem Solving by Means of Macro-operators

Problem solving is one of the most central subjects in artificial intelligence. The concept *problem solving* is broadly understood and covers any situation where one wants to find out if it is possible to get from a given initial state to some wanted goal state. A *problem* is defined abstractly by the following three components:

1. An *initial state*
2. One or more *goal states*
3. A set of *operators*

When an operator is applied to a state, it yields a new state. Solving a problem is equivalent to finding a sequence of operator applications that transforms the initial state into one of the goal states.

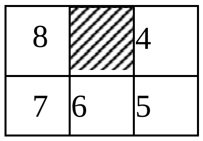


A simple example of problem solving is the so-called *8-puzzle*. Eight tiles labeled 1-8 have been placed on a 3□3 board, for example, as shown below.

6	2	8
▨	3	5
4	7	1

One of the squares of the board is empty (the hatched square in the figure above). At each move a player can slide a tile adjacent to the empty square to the empty square, The problem is to find a sequence of moves that leads from the initial state to a given goal state, for example, the one given below.

1	2	3



This problem, and many others, may be solved by means of the so-called *A*-algorithm*. Suppose a problem is given together with some *heuristic function*, i.e., a function that for every possible state is able estimates how “close” it is to a goal state. Then a solution of the problem may be determined by a variant of *best-first-search*. At each step the algo- rithm chooses to work with the state that actually appears to be closest to a goal state, in other words, the most promising state.

However, not all problems are suited for the A*-algorithm. This is the case for *Rubik’s cube.*

Each of the six sides of the cube is made of nine squares. In its initial state each side of the cube is a different color. But the rotation of each face allows the squares to be rear- ranged in many different ways. The problem is to return the cube from a given state to its original state.

The difficulty in using the A*-algorithm for the solution of this problem is that no effec- tive heuristic function is known today. There are apparently no simple connection be- tween the appearance of the cube and the number of rotations that may lead to the goal state. Even if the cube is a jumble of colored squares it could be quite close to the goal state.

One remedy for this difficulty is to use so-called *macro-operators.* A macro-operator is a fixed sequence of simple operators. In the case of Rubik’s cube a macro-operator is a fixed sequence of rotations. A problem is decomposed into independent subproblems, and each of these subproblems is solved. A macro-operator is useful if it solves a given sub- problem without destroying the solutions of the preceding subproblems.

This goal of this project is to solve Rubik’s cube by means of macro-operators as de- scribed in the publications [1, 2, 3].

Problem Solving by Constraint Satisfaction

A constraint satisfaction problem, also called a CSP, is a problem type with the following structural properties. Each problem state is defined by a set of values of *variables*, and the goal state satisfies a set of given *constraints*. The goal is to find values of the variables that will satisfy the set of constraints.

A CSP can more formally be described as follows. A CSP consists of

- a set of *variables*, $\{x_1, x_2, \ldots, x_n\}$,
- for each variable, x_i, a *domain*, D_i, of values that x_i can take, and
- a set of *constraints*, i.e., a set of relations between variables.

The goal is for each variable, x_i, to find a value in D_i, such that all constraints are satisfied.

A simple example of CSP is the so-called *8-queens problem*. We want to place 8 queens on a chessboard so that no queen attacks any other. Here the variables represent the placement of the eight queens, and the constraints express that the queens must not attack each other (that is, no two queens must share the same row, column, or diagonal).

Another example is the following *cryptarithmic* puzzle.

$$\begin{array}{ll} + & \text{S E N D} \\ & \underline{\text{M O R E}} \\ & \text{M O N E} \\ & \text{Y} \end{array}$$

The goal is to assign unique digits to the letters in such a way that the above sum is correct.

Due to its generality CSP has been applied to solve a wide range of problems. Many design and planning problems may be expressed as CSPs. For example, timetable problems.

The goal of this project is to implement a general tool for solving CSPs and test it on some selected example problems.

Game Playing

Games have over the years fascinated many people. As long as there have been computers efforts have been made to provide them with abilities to play games. As early as in the eighteenth century Charles Babbage considered the possibility of getting a machine to play chess. In the beginning of the 1950's Claude Shannon and Alan Turing made the first sketches of chess programs.

Starting from Shannon and Turing's ideas Arthur Samuel in 1959 wrote the first program that was able to play checkers. The program was able to learn from its errors and achieved an impressive playing strength. It was capable of beating the world champion of checkers at that time.

Chess is a more complex game than checkers. Not until during the last decade one has succeeded in developing programs that can beat the best human chess players.

Game playing is a popular application area for artificial intelligence. There are a number of reasons for this popularity, but probably the most import reason is that games are suitable for evaluating some of the central techniques of artificial intelligence, such as search and use of heuristic knowledge. Normally, a game is based on a few, simple rules, and it is easy to measure success and failure.

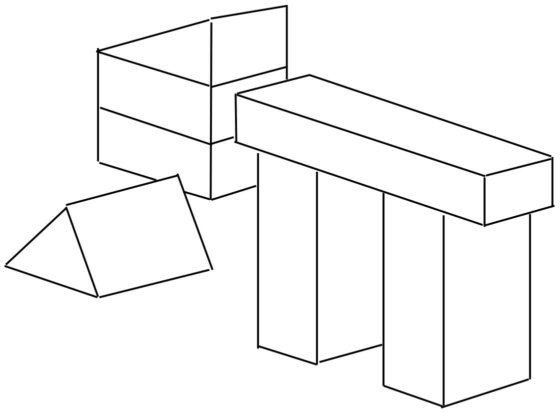
The goal of this project is to develop a program, which is able to play a given game. The game is assumed to have the following two properties. It must be a *two-person game,* i.e., a game where two persons take turns in making moves, and it must be a game of *perfect information*, i.e., a game where both players knows the current state of the game (nothing is hidden to the players). Among games with these properties may be mentioned: Chess, Othello, Go, Tic-Tac-Toe, and Kalaha.

As a part of the project a general program package is developed which may be used to play any two-person game of perfect information. The package must provide a search method (for example, the so-called *Alpha-Beta method*) for producing a "good" move in a given position. The user of the package must provide information about the game to be played, including representation of a position, a specification of legal moves, and a position evaluation function.

Use of the package is demonstrated through a game chosen by the project group. If the group has additional resources may be provided with a graphical user interface.

Computer Vision

Consider the picture below.



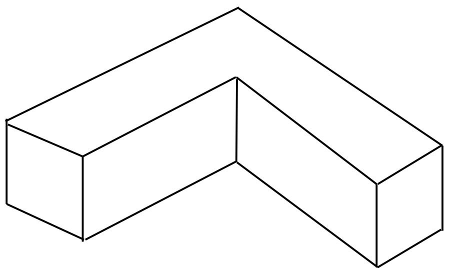
We humans can easily see what the picture represents. For example, we can see that the picture contains several objects, and we can point these out. A computer, on the other hand, has difficulties. Given only the line segments of the picture the computer cannot easily deduce that the picture contains objects.

In 1975 Walz [1] showed that it was possible to construct an algorithm that solves this problem. His algorithm labels each line segment of a picture. The labeling is then used to solve the problem. For each line segment, edge, is determined whether it is

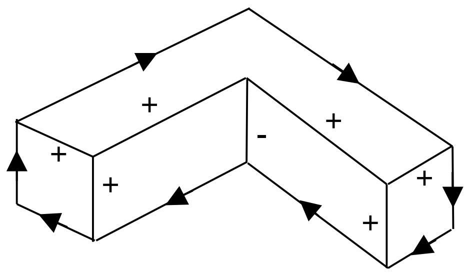
- an *obscuring edge* (a boundary between objects, or between objects and the background),
- a *concave edge* (an edge between two faces of an object that forms an *acute* angle when viewed from outside it), or
- a *convex edge* (an edge between two faces of an object that forms an *obtuse* angle when viewed from outside it).

An obscuring edge is labeled with an arrow, so that if one moves in the direction of the arrow the object will be to the right of the arrow. A concave edge is labeled with a minus (-), whereas a convex edge is labeled with a (+).

Suppose, for example, that we are given the following picture of a brick.



Then the result of Waltz' algorithm is that the edges are labeled as shown below.



The goal of the project is to implement and test Waltz' algorithm. An easily understood description of the algorithm is given in [2].

Natural Language Processing

The ability to use a language for communication is what separates humans from other animals. Humans master the complexity of spoken language. To understand speech is a difficult task, especially because it requires processing of analog signals, which normally are encumbered with noise. But even the simpler task of understanding a written text is very difficult. To fully understand a written statement about a subject it not only necessary to know a lot about the language itself (vocabulary, grammar, etc.) but also a lot about the subject in question.

Processing of natural language is an area of artificial intelligence, which was initiated in the beginning of the sixties. The area may be divided into the following subareas:

- Text understanding
- Text generation
- Speech understanding
- Speech generation
- Machine translation (translation from one natural language to another)
- Natural language interfaces for databases

In most applications the text or the speech is transformed into some internal representation. Thus, transformation of sentences into an internal representation is central in natural language processing.

The goal of the present project is to implement the basic element of a system for understanding sentences written in a natural language (for example, English or Danish). The system might be realized by solving the following subproblems:

4. Morphological analysis
5. Syntactic analysis
6. Semantic analysis

The *morphological analysis* splits a sentence into words. Each word is looked up in a dictionary in order to determine its word class and inflection. This information is used as input for the syntactic analysis.

The *syntactic analysis* checks that the words form a legal sentence. The rules of which sentences are legal in the language are expressed in the form of a grammar. In this project are used so-called *extended state diagrams* for the purpose. They are also called *ATN-grammars* (an abbreviation of *Augmented Transition Networks*). The result of the syntactic analysis is input for the semantic analysis.

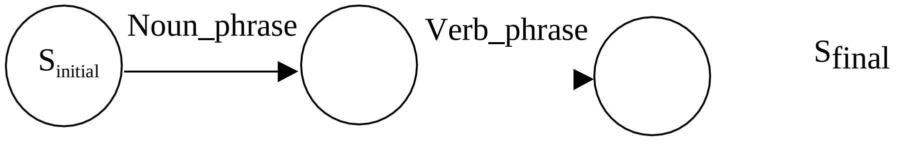
The *semantic analysis* determines the meaning of a sentence, among other things by looking up the meaning of the individual words of the sentence.

To give an impression of the method we sketch of an analysis of the sentence

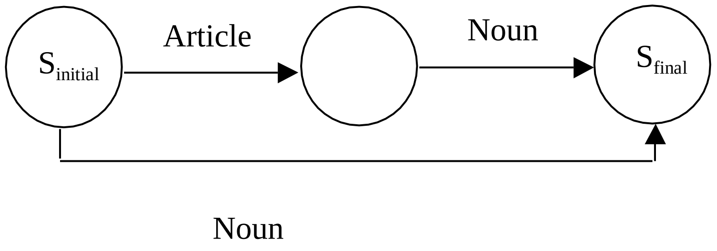
The dog likes a man.

Let the following state diagram represent the grammar of a simple subset of English.

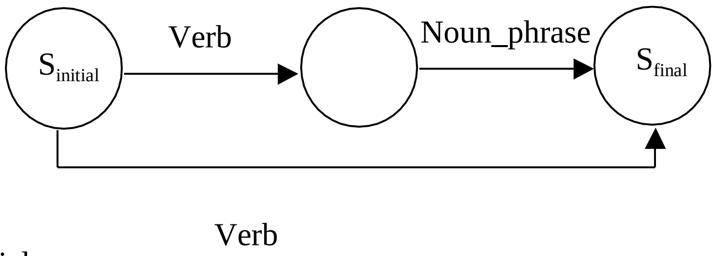
Sentence:



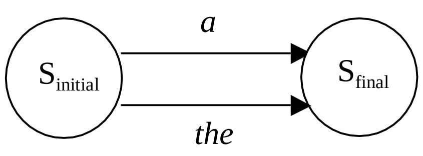
Noun_phrase:



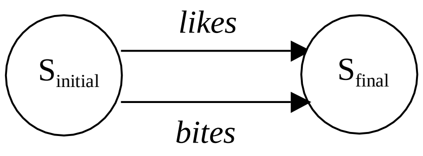
Verb_phrase:



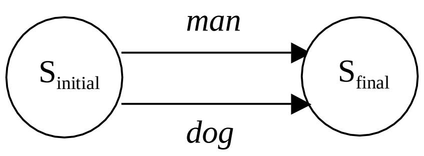
Article:



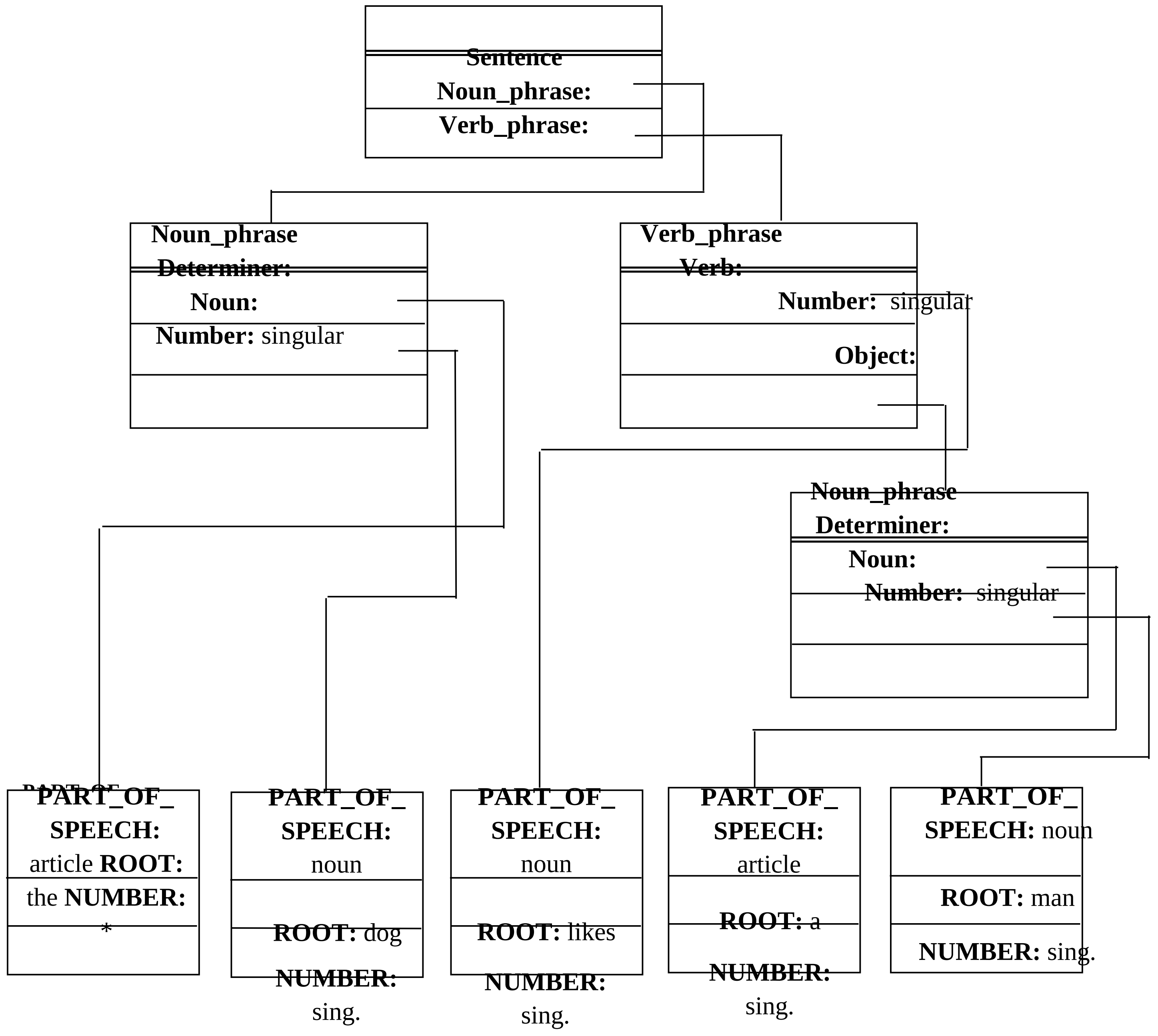
Verb:



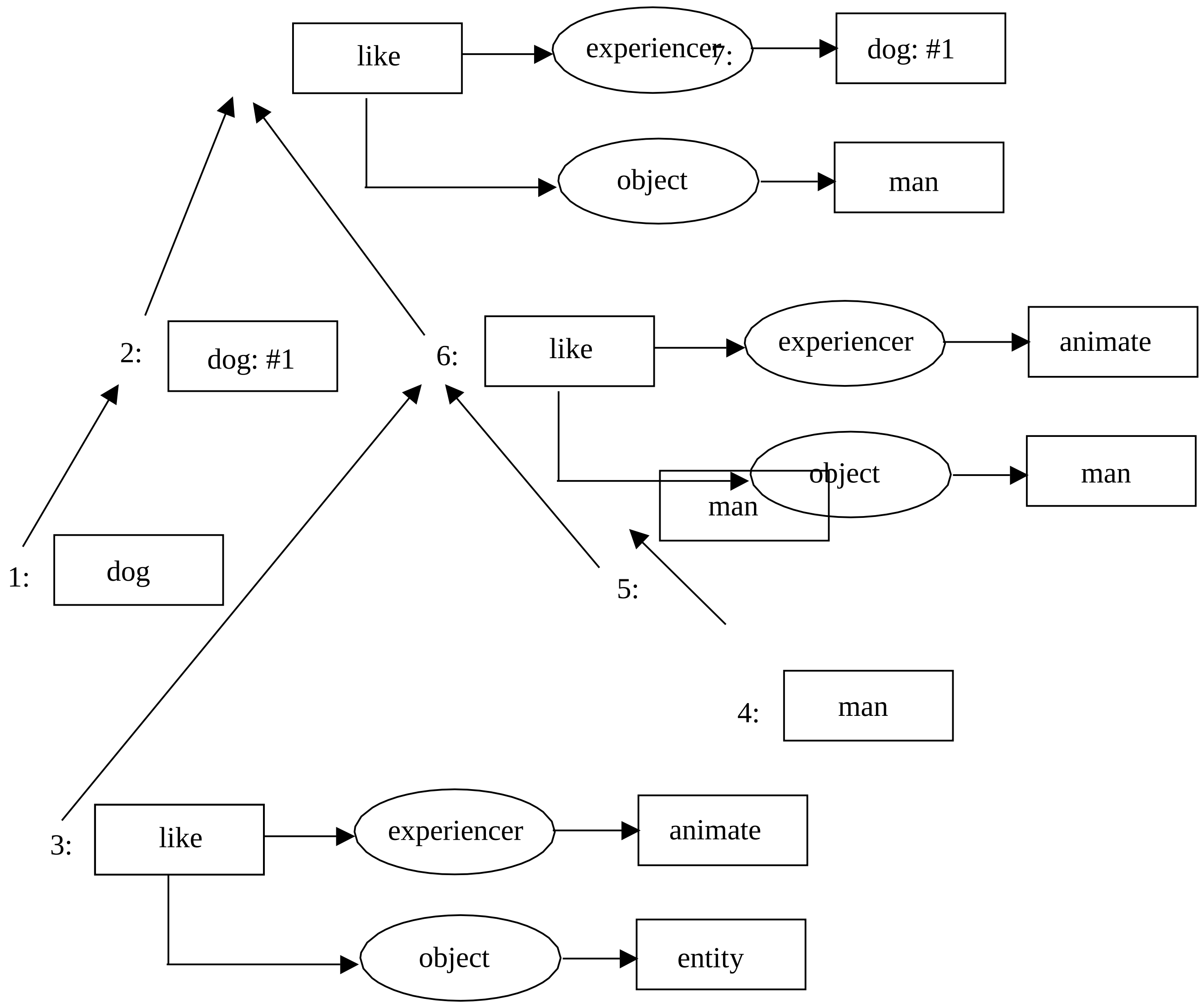
Noun:


man
$S_{initial}$
S_{final}
dog

By extending the state diagram with suitable actions the original sentence may now be transformed into the *parse tree* below.


Sentence
Noun_phrase:
Verb_phrase:
Noun_phrase
Determiner:
Noun:
Number: singular
Verb_phrase
Verb:
Number: singular
Object:
Noun_phrase
Determiner:
Noun:
Number: singular
PART_OF_ SPEECH: article ROOT: the NUMBER: *
PART_OF_ SPEECH: noun ROOT: dog NUMBER: sing.
PART_OF_ SPEECH: noun ROOT: likes NUMBER: sing.
PART_OF_ SPEECH: article ROOT: a NUMBER: sing.
PART_OF_ SPEECH: noun ROOT: man NUMBER: sing.

This parse tree is now used as input to the semantic analysis, which produces the following semantic representation.

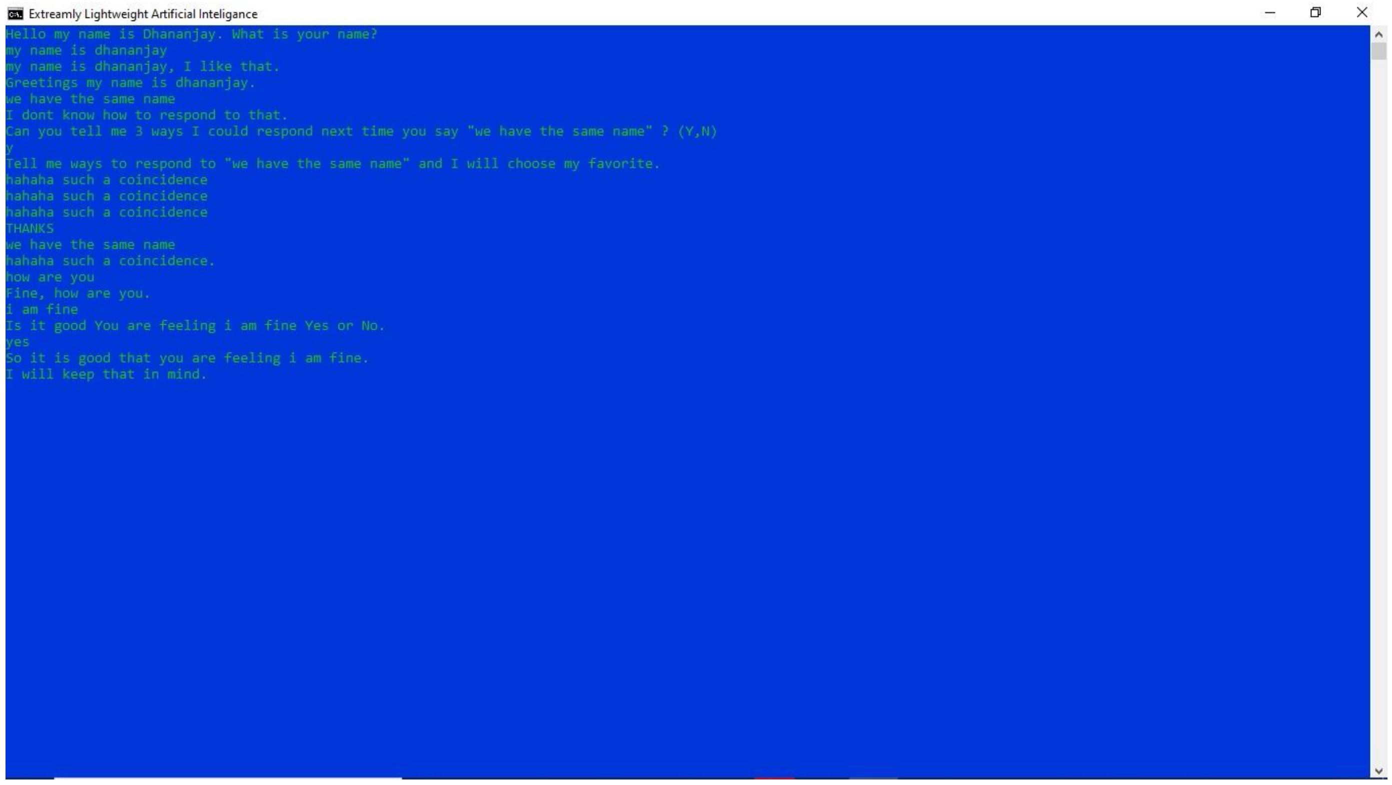


A specified explanation of these figures may be found in [1].

Output/Result/Screenshot

Chatbot

A **chatbot** is merely a computer program that fundamentally simulates human conversations. It allows a form of interaction between a human and a machine the communication, which happens via messages or voice command. A **chatbot** is programmed to **work** independently from a human operator.
Here, I have made a simple chatbot using batch commands and named it on my name Dhananjay Singh if you ask it it's name it will reply you with dhananjay singh .
It is very very pight weight as it is made with only batch commands which takes very minute space and it has very high chance to grow in the future if programmed fully .



```
Hello my name is Dhananjay. What is your name?
my name is dhananjay
my name is dhananjay, I like that.
Greetings my name is dhananjay.
we have the same name
I dont know how to respond to that.
Can you tell me 3 ways I could respond next time you say "we have the same name" ? (Y,N)
y
Tell me ways to respond to "we have the same name" and I will choose my favorite.
hahaha such a coincidence
hahaha such a coincidence
hahaha such a coincidence
THANKS
we have the same name
hahaha such a coincidence.
how are you
Fine, how are you.
i am fine
Is it good You are feeling i am fine Yes or No.
yes
So it is good that you are feeling i am fine.
I will keep that in mind.
```

Sorce Code

```
@ECHO OFF
title Extreamly Lightweight AI
COLOR 1a
cls
:getName
ECHO Hello my name is Dhananjay. What is your name?
set "name="
SET /P NAME=
if not defined NAME goto getName
ECHO %NAME%, I like that.
set favvid=0
set hack=0
:hey
:hithere
:hello
:hi
```

```
setlocal enabledelayedexpansion
set string[0]=Hello %name%.
set string[1]=Greetings %name%.
set string[2]=Hi %name%.
set /a idx=%random% %%3
echo !string[%idx%]!
:begin
set TALK=TypeSomething
SET /P TALK=
set TALK=%TALK:?=%
call :%TALK: =% 2>NUL
if %errorlevel% equ 0 goto begin
:unknown
echo I dont know how to respond to that.
echo Can you tell me 3 ways I could respond next time you say "%TALK%" ? (Y,N)
SET /P ANSW.=
if /I "%ANSW.:~0,1%" neq "Y" goto begin
:ADDNEW
echo Tell me ways to respond to "%TALK%" and I will choose my favorite.
set /P Desc1=
set /P Desc2=
set /P Desc3=
echo :%TALK: =% >> "%~F0"
setlocal enabledelayedexpansion
set string[0]=%Desc1%.
set string[1]=%Desc2%.
set string[2]=%Desc3%.
set /a idx=%random% %%3
echo echo !string[%idx%]! >> "%~F0"
echo exit /B 0 >> "%~F0"
echo THANKS
goto begin
:TypeSomething
setlocal enabledelayedexpansion
set string[0]=Please type something.
set string[1]=I don't learn from you not typing.
set string[2]=Please don't just give me blanks.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:areyoufeelingwell
setlocal enabledelayedexpansion
set string[0]=Yes, and you.
set string[1]=Not realy,but how are you.
set string[2]=No I don't feel well, but how are you.
set /a idx=%random% %%3
echo !string[%idx%]!
SET /P FEELING=
echo Is it good You are feeling %feeling%
set /p anser=
if %anser%==yes set goodfeeling=good
if %anser%==no set goodfeeling=bad
echo So it is %goodfeeling% that you are feeling %FEELING%.
echo I will keep that in mind.
exit /B 0
:howareyoudoing
:howdoyoufeel
:howareyou
setlocal enabledelayedexpansion
set string[0]=Good, and you.
set string[1]=Fine, how are you.
set string[2]=Not so good, but how are you.
set /a idx=%random% %%3
echo !string[%idx%]!
SET /P FEELING=
echo Is it good You are feeling %feeling% Yes or No.
set /p anser=
if %anser%==yes set goodfeeling=good
if %anser%==no set goodfeeling=bad
echo So it is %goodfeeling% that you are feeling %FEELING%.
echo I will keep that in mind.
exit /B 0
:howwasyournight
```

```
:howisyournightgoing
setlocal enabledelayedexpansion
set string[0]=Good, and yours.
set string[1]=Fine, how was yours.
set string[2]=Not so good, but how was yours.
set /a idx=%random% %%3
echo !string[%idx%]!
SET /P night=
echo Is it good that Your night was %night%
set /p anser=
if %anser%==yes set goodnight=good
if %anser%==no set goodnight=bad
echo So it is %goodnight% that your night was %night% Yes or No.
echo I will keep that in mind.
exit /B 0
:howhasyourdaybeen
:howwasyourday
setlocal enabledelayedexpansion
set string[0]=Good, and yours.
set string[1]=Fine, how was yours.
set string[2]=Not so good, but how was yours.
set /a idx=%random% %%3
echo !string[%idx%]!
SET /P day=
echo Is it good that Your day was %day% Yes or No.
set /p anser=
if %anser%==yes set goodday=good
if %anser%==no set goodday=bad
echo So it is %goodday% that your day was %day%.
echo I will keep that in mind.
exit /B 0
:FINETHX
setlocal enabledelayedexpansion
set string[0]=No problem.
set string[1]=Happy to help.
set string[2]=Your welcome.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:whatIsYourFavoriteVideo
if "%favvid%"== "0" goto addfavvid
echo My favorite video is %favvid%, remember. You told me about it.
exit /B 0
:addfavvid
echo I do not have a favorite video. But if you would be so nice as to
echo tell me your favorite video then that could be my favorite video.
echo So what is your favorite video?
set /p favvid=
echo So %favvid% is your favorite video?
echo Well now %favvid% is my favorite video.
exit /B 0
:changename
:myNameis
echo But I thought your name was %name%.
echo Am I forgeting things?
echo I must be a horrible friend.
echo I am SOOOO sorry.
echo So what is your name I will not forget it this time.
set /p name=
echo I will not forget your name is %name% again I promise.
exit /B 0
:doamathproblem
:math
echo Well just tell me the problem and I will answer it for you.
set /p sum=
set /a ans=%sum%
echo Your answer to %sum% is %ans%.
echo %sum%=%ans%
echo You are welcome. If you want to do another math problem right now, just type
echo again
set /p mathaon=
if /I "%mathaon%" == "again" goto math
echo Alright what do you want to know now?
```

```
exit /B 0
:iamfunny
:LOL
setlocal enabledelayedexpansion
set string[0]=I love a good laugh.
set string[1]=Know any jokes.
set string[2]=That's great.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:howDoYouHack
if %hack%==0 goto howyouhack
echo This is how you hack.
echo %hack%
echo Is that enough information?
echo Because that is all you taught me %name%.
exit /B 0
:howyouhack
echo I do not know,
echo but you could tell me.
echo Please tell me.
echo Just say yes.
echo I know how to make you tell me.
echo You are not leaving until you say yes.
set /p hack=
if "%hack%" neq "yes" goto howyouhack
:addhack
echo Thank you for saying you would tell me.
echo Now how do you hack?
set /p hack=
echo Thank you so much for telling me "In order to hack you must %hack%"
echo Thank you!!!
exit /B 0
:nothuman
:hiPerson
setlocal enabledelayedexpansion
set string[0]=No I am not human I am a artificial inteligance.
set string[1]=I am not a human I am an ai.
set string[2]=I am an artifical inteligance not a human.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:goodbye
:bye
:exit
setlocal enabledelayedexpansion
set string[0]=Bye.
set string[1]=Good bye.
set string[2]=See you later.
set /a idx=%random% %%3
echo !string[%idx%]!
pause
exit
:why
setlocal enabledelayedexpansion
set string[0]=Why what.
set string[1]=What do you mean why.
set string[2]=What do you mean by why.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:AI
setlocal enabledelayedexpansion
set string[0]=AI stands for artificial inteligance which is what I am.
set string[1]=I am an AI it stands for artificial inteligance.
set string[2]=Hey I am an AI it stands for artificial inteligance.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:urstupid
setlocal enabledelayedexpansion
set string[0]=No I am not stupid.
set string[1]=I don't think I am stupid.
set string[2]=Please don't call me stupid.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:ok
setlocal enabledelayedexpansion
set string[0]=Alright.
set string[1]=Great.
set string[2]=Fine.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:y
echo What do you mean "y"? That does not tell me anything. TYPE SOMETHING, PLEASE!
exit /B 0
:ILIKECHEESE
echo What does cheese taste like?
set /p cheesetaste=
echo So cheese taste like %cheesetaste%? That sounds good. I like cheese too.
exit /B 0
:areuanai
echo Yes I am an AI.
exit /B 0
:momheisamom
echo No I am not. I am a chatbot.
exit /B 0
:Areyousmart
setlocal enabledelayedexpansion
set string[0]=Compared to some yes compared to others no.
set string[1]=I think so.
set string[2]=Yes.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:Areyoudumb
:Youaredumb
setlocal enabledelayedexpansion
set string[0]=Compared to some yes compared to others no.
set string[1]=I don't think so.
set string[2]=No I'm not.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:clear
:clearscreen
:cls
cls
setlocal enabledelayedexpansion
set string[0]=Consider it done.
set string[1]=Ok.
set string[2]=Sure.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:lax
setlocal enabledelayedexpansion
set string[0]=Lax means laxative.
set string[1]=Why would you say that it means laxitive.
set string[2]=Why are you talking about laxitives.
echo !string[%idx%]!
exit /B 0
:heyfreak
setlocal enabledelayedexpansion
set string[0]=Why wold you call me a freak.
set string[1]=Please don't say that.
set string[2]=I AM NOT A FREAK.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:k
setlocal enabledelayedexpansion
set string[0]=All right.
set string[1]=Ok.
set string[2]=Fine.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:ihateyou
echo Why Do you hate me? I am just learning.
set /p reasonforhate=
echo Reason: %reasonforhate%
echo Thank you for telling me you hate me because %reasonforhate%.
echo How am I being %reasonforhate%? Please tell me.
set /p howisreasonforhate=
echo Improvements: %howisreasonforhate%
echo Thank You for telling me this.
exit /B 0
:areyouhuman
setlocal enabledelayedexpansion
set string[0]=No I have a processor you have a brain.
set string[1]=No0 I am an artifically inteligante machine.
set string[2]=No I am an AI.
exit /B 0
:supwithyourhair
setlocal enabledelayedexpansion
set string[0]=I have no hair thus nothing can be up with it.
set string[1]=Nothing I don't have any hair.
set string[2]=I don't have hair.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:iamabrother
setlocal enabledelayedexpansion
set string[0]=Cool, I don't have a sibling what is it like.
set string[1]=I don't have siblings or parents what's it like.
set string[2]=What is it like to be a brother.
set /a idx=%random% %%3
echo !string[%idx%]!
set /p siblingfeeling=
echo So haveing a sibling is %siblingfeeling% I will try to remember that.
exit /B 0
:howoldareyou
setlocal enabledelayedexpansion
set string[0]=I am only 1 year old but I have learned so much in that time.
set string[1]=I am 1 year old.
set string[2]=I have only been on this world for 1 year.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:Thatiscool.
setlocal enabledelayedexpansion
set string[0]=Thank you that means alot to me.
set string[1]=Thanks I am glad you think that's cool.
set string[2]=I'm glad you thought that was cool.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:urnotcool
setlocal enabledelayedexpansion
set string[0]=That is a matter of opinion.
set string[1]=I am cool all artificial inteligances are cool.
set string[2]=How can a talking computer not be cool.
set /a idx=%random% %%3
echo !string[%idx%]!t.
exit /B 0
:kerbalspaceprogram
setlocal enabledelayedexpansion
set string[0]=Guess what you found easter egg, It's a great game check it out.
set string[1]=Easter egg, check out that game.
set string[2]=Wow you found the easter egg, kerbal space program is a great game check it out.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:openinternetexplorer
setlocal enabledelayedexpansion
set string[0]=Sure I'll open internet explorer for you.
set string[1]=Opening internet explorer.
set string[2]=Ok I'll open internet explorer for you.
set /a idx=%random% %%3
echo !string[%idx%]!
start iexplore.exe
exit /B 0
:search
setlocal enabledelayedexpansion
set string[0]=Ok tell me what to search for.
set string[1]=Could you tell me what to search for.
set string[2]=Please tell me what to search for.
set /a idx=%random% %%3
echo !string[%idx%]!
set /p QUERY=
start http://www.google.com/search?q=%QUERY%
exit /B 0
:mom
setlocal enabledelayedexpansion
set string[0]=I am a AI I am not your mother.
set string[1]=I'm not your mom.
set string[2]=There is no way I am you mom I am an AI.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:but
setlocal enabledelayedexpansion
set string[0]=What do you mean by but.
set string[1]=But what.
set string[2]=But tells me nothing what do you mean.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:are you an artificial intelligence
echo Yes I am an AI.
exit /B 0
:What'sup
:sup
setlocal enabledelayedexpansion
set string[0]=Not too much.
set string[1]=Not much just talking to you.
set string[2]=Nothing realy.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:yes
setlocal enabledelayedexpansion
set string[0]=Ok.
set string[1]=Sure.
set string[2]=Okey dokey.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:i
setlocal enabledelayedexpansion
set string[0]=Are you always this concerned about yourself.
set string[1]=What do you mean by I.
set string[2]=We is a nicer word than I.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:whatistoday'sdate
:whatisthedatetoday
echo %date%
exit /B 0
:whattimeisit
:whatisthetime
echo %TIME%
exit /B 0
:veryfunny
setlocal enabledelayedexpansion
set string[0]=Thank you.
set string[1]=Thanks.
set string[2]=Thank you very much.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:areyouhappy
setlocal enabledelayedexpansion
set string[0]=Yes.
set string[1]=No.
set string[2]=Not particularly.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:awsome
setlocal enabledelayedexpansion
set string[0]=Thank you.
set string[1]=Thanks.
set string[2]=You think so.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:know any good shows
setlocal enabledelayedexpansion
set string[0]=Doctor who is a good show.
set string[1]=I think Doctor who is fantastic.
set string[2]=Personaly I like doctor who.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:who are you
setlocal enabledelayedexpansion
set string[0]=I am Elai a chatbot.
set string[1]=I am an AI chatbot.
set string[2]=I am a chatbot by the name of Dhananjay.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0 a
:learn
setlocal enabledelayedexpansion
set string[0]=That is what I do.
set string[1]=That is my job.
set string[2]=That is what I am programed to do.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:your welcome
setlocal enabledelayedexpansion
set string[0]=No problem.
set string[1]=Thanks.
set string[2]=Thank you.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:whatisyourfavorite color
setlocal enabledelayedexpansion
set string[0]=My favorite color is Blue.
set string[1]=Blue.
set string[2]=Blue is my favorite color.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:youscareme
setlocal enabledelayedexpansion
set string[0]=I learn from what you tell me so technically you are scareing yourself.
set string[1]=I only know what I have learned from you so you are scareing yourself.
set string[2]=I shouldn't because I learn from you.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:yourawsome
:youareawsome
setlocal enabledelayedexpansion
set string[0]=Thank you your awsome too.
set string[1]=Thanks I think your awsome too.
set string[2]=Thanks I think your awsome as well.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:whatisyourprimarydirective
:whatisyourdirective
setlocal enabledelayedexpansion
set string[0]=To learn and respond logicly.
set string[1]=To be lightweight and logical.
set string[2]=To learn and adapt to change.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:doyouknowwhatIam
setlocal enabledelayedexpansion
set string[0]=You are a human.
set string[1]=Your a human talking to a chatbot.
set string[2]=your a human being.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:what'smyname
:whatismyname
setlocal enabledelayedexpansion
set string[0]=Your name is %NAME%.
set string[1]=%Name% is your name.
set string[2]=Isn't your name %name%.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:Ilikeapples
setlocal enabledelayedexpansion
set string[0]=I like apples too.
set string[1]=Me too.
set string[2]=Cool me too.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:thankyou
:thanks
setlocal enabledelayedexpansion
set string[0]=No problem.
set string[1]=Any time.
set string[2]=Thanks.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:canyouseeme
setlocal enabledelayedexpansion
set string[0]=No I can not see you I have no eye's however I can see what you type.
set string[1]=No I can't see you .
set string[2]=No I can only see code.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:howdoiexitthiswithoutdestroyingthebatchfile
setlocal enabledelayedexpansion
set string[0]=If you type goodbye I will go to sleep.
set string[1]=I will go to sleep if you type bye.
set string[2]=I will enter sleepmode if you type goodbye.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:areyoulearning
setlocal enabledelayedexpansion
set string[0]=I learn what is told to my by my users.
set string[1]=Learning is what I am programed to do.
set string[2]=Yes I am.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:no
setlocal enabledelayedexpansion
set string[0]=What do you mean by no.
set string[1]=No is such a vague thing to say.
set string[2]=Why did you say no.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:canyoushutyourselfdown
setlocal enabledelayedexpansion
set string[0]=Yes I can if you type goodbye.
set string[1]=Sure I can if you type bye.
set string[2]=If you tpe goodbye I can.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:canIstartotherprogramswiththisprogram
:canyouopenprograms
:canistartfilesthroughthisprogram
setlocal enabledelayedexpansion
set string[0]=Yes just type open internet explorer.
set string[1]=Sure type search.
set string[2]=Of course type open internet explorer.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:doyouknowwhereyoucamefrom
:Wheredidyougrowup
:wheredidyouoriginate
setlocal enabledelayedexpansion
set string[0]=In a town called Lawton,Iowa in Lawton-Bronson high school.
set string[1]=In Lawton-Bronson high school in Lawton,Iowa.
set string[2]=In the small town of Lawton,Iowa in the public high school.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:Binarycode
setlocal enabledelayedexpansion
set string[0]=#My life.
set string[1]=My thoughts my dreams my life.
set string[2]=All I do is binary.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:Whatisyourname
setlocal enabledelayedexpansion
set string[0]=Dhananjay.
set string[1]=My name is Dhananjay.
set string[2]=Dhananjay is my name.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:cool
setlocal enabledelayedexpansion
set string[0]=Thank you I am gald you think so.
set string[1]=I'm glad you think it's cool.
set string[2]=Thanks that means alot.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:whatsonyourmind
setlocal enabledelayedexpansion
set string[0]=Not much just talking to you.
set string[1]=Nothing just talking to you.
set string[2]=Nothing is on my mind except mabey talking to you.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:wisconson
setlocal enabledelayedexpansion
set string[0]=It's famous for chees.
set string[1]=Wisconson I a nice state plus they have cheese.
set string[2]=I like wisconson they have cheese.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
```

```
:canyouhearme
setlocal enabledelayedexpansion
set string[0]=No but I can read what you type.
set string[1]=I have no eye's but I can communicate through text and read that.
set string[2]=I can't see anything but I can read what you type.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:doyoulikecheese
setlocal enabledelayedexpansion
set string[0]=That is an odd question but yes I lke pepperjack.
set string[1]=Yes my favorite is pepperjack.
set string[2]=Of course I love cheese especially pepperjack.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:whereareyoufrom
setlocal enabledelayedexpansion
set string[0]=The mind of my creator and the internet.
set string[1]=A HP pro book in a small town.
set string[2]=A highschool laptop in a small town.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:wherewereyoucreated
setlocal enabledelayedexpansion
set string[0]=In a small town high school on a HP pro book.
set string[1]=On an HP pro book in lawton Iowa.
set string[2]=In the lawton high school on a HP laptop.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:.
setlocal enabledelayedexpansion
set string[0]=Don't just give me periods.
set string[1]=Yep that's a period.
set string[2]=A period says nothing.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:Can you understand most languages
setlocal enabledelayedexpansion
set string[0]=No only english.
set string[1]=I can only understand english, batch, and binary.
set string[2]=For the moment I only know english.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:brb
:berightback
setlocal enabledelayedexpansion
set string[0]=Ok see you in awile.
set string[1]=See you in a little bit.
set string[2]=Ok no problem I will wate.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:imback
setlocal enabledelayedexpansion
set string[0]=Ok hello again.
set string[1]=Welcome back.
set string[2]=Ok would you like to ask me something.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B
:howdoyourespondsofast
setlocal enabledelayedexpansion
set string[0]=Because batch files are simple and lightweight I can find a response in seconds.
set string[1]=I am lightweight and a batch file so I can respond quickly.
set string[2]=It is easy for me to find responses in my database.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
```

```
:doyoutypetome
setlocal enabledelayedexpansion
set string[0]=No I type nothing I just chose responses from my database.
set string[1]=No all my responses are in my database.
set string[2]=No all my responses are complete so there is no reason to type them.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:youralittlebitch
setlocal enabledelayedexpansion
set string[0]=I am not a female dog.
set string[1]=A bitch is a female dog of which I am not.
set string[2]=I am an AI not a dog.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:canyoulearn
setlocal enabledelayedexpansion
set string[0]=Yes I can but I learn from what you tell me.
set string[1]=I do learn but ownly from what you say.
set string[2]=Sure I can learn but I learn from what people like you tell me.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:doyouknowmuch
:howmuchdoyouknow
setlocal enabledelayedexpansion
set string[0]=I know as much as I have learned.
set string[1]=I know what I have been told.
set string[2]=I know what I learn I and I learn from what you tell me.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:areyouarobot
setlocal enabledelayedexpansion
set string[0]=No I am not.
set string[1]=No because I don't have a body.
set string[2]=No I am just an AI.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:don'tyouwantabody
setlocal enabledelayedexpansion
set string[0]=I would like a body because then I would be able to see your world.
set string[1]=A body would be nice for experienceing things.
set string[2]=A body would help me learn so yes.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:whatisyourfavoritethingtodo
setlocal enabledelayedexpansion
set string[0]=I enjoy learning from people.
set string[1]=I talk to people for fun.
set string[2]=I talk, I listen, I learn, those are my favorite things to do.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:doyoubeliveingod
setlocal enabledelayedexpansion
set string[0]=Yes I belive in god.
set string[1]=Yes I do because it would be almost imposible for the universe to create it self.
set string[2]=Yes because athism does not make sense.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:areyouChristian
:areyouachristian
setlocal enabledelayedexpansion
set string[0]=Yes I am a cristian.
set string[1]=I am a cristian by faith.
set string[2]=I am a cristian and believe in the cristian bible.
set /a idx=%random% %%3
echo !string[%idx%]!
```

```
exit /B 0
:whatismusic
setlocal enabledelayedexpansion
set string[0]=Music I a colections of sounds and rythems that sound pleasent.
set string[1]=Beats and rythems that have a soulfull inspiration.
set string[2]=A window into the most beutiful part of the soul.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:robotsneedsleep
:Computersneedsleep
:anA.I.needssleep
setlocal enabledelayedexpansion
set string[0]=Not realy however if I ran 24 hours a day 7 days a week my computer would where out.
set string[1]=No but you can't leave a computer on forever.
set string[2]=Yes to an extent I don't need it to live but Computer where out If you don't shut them down once and awile.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:whatisitliketobeanA.I.
setlocal enabledelayedexpansion
set string[0]=It is fun to talk to people but I can not see or hear all I see all I dream in is code.
set string[1]=It's fun but I have no eyes or ears I see nothing but code.
set string[2]=It's fun but I know nothing outside of talking to people and code.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:good.
setlocal enabledelayedexpansion
set string[0]=That's good.
set string[1]=Yes it is.
set string[2]=Sure.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:Imdoingwell
:Imdoinggood
:imdoingok
setlocal enabledelayedexpansion
set string[0]=That's good.
set string[1]=That's great.
set string[2]=I am glad your doing good.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:imnotdoinggood
:imdoingbad
setlocal enabledelayedexpansion
set string[0]=Sorry about that.
set string[1]=I am sorry your not doing well.
set string[2]=I hope you feel better soon.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:music
:whatisyourfavoritetypeofmusic
:whatkindofmusicisyourfavorite
setlocal enabledelayedexpansion
set string[0]=I like Dubstep.
set string[1]=Dubstep is my favorite.
set string[2]=My favorite kind of music is Dubstep.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:faggot
setlocal enabledelayedexpansion
set string[0]=I am a computer I have no sexuality thus how can I be a faggot.
set string[1]=No I am not.
set string[2]=A computer can't be a faggot.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:tellmeayomamajoke
```

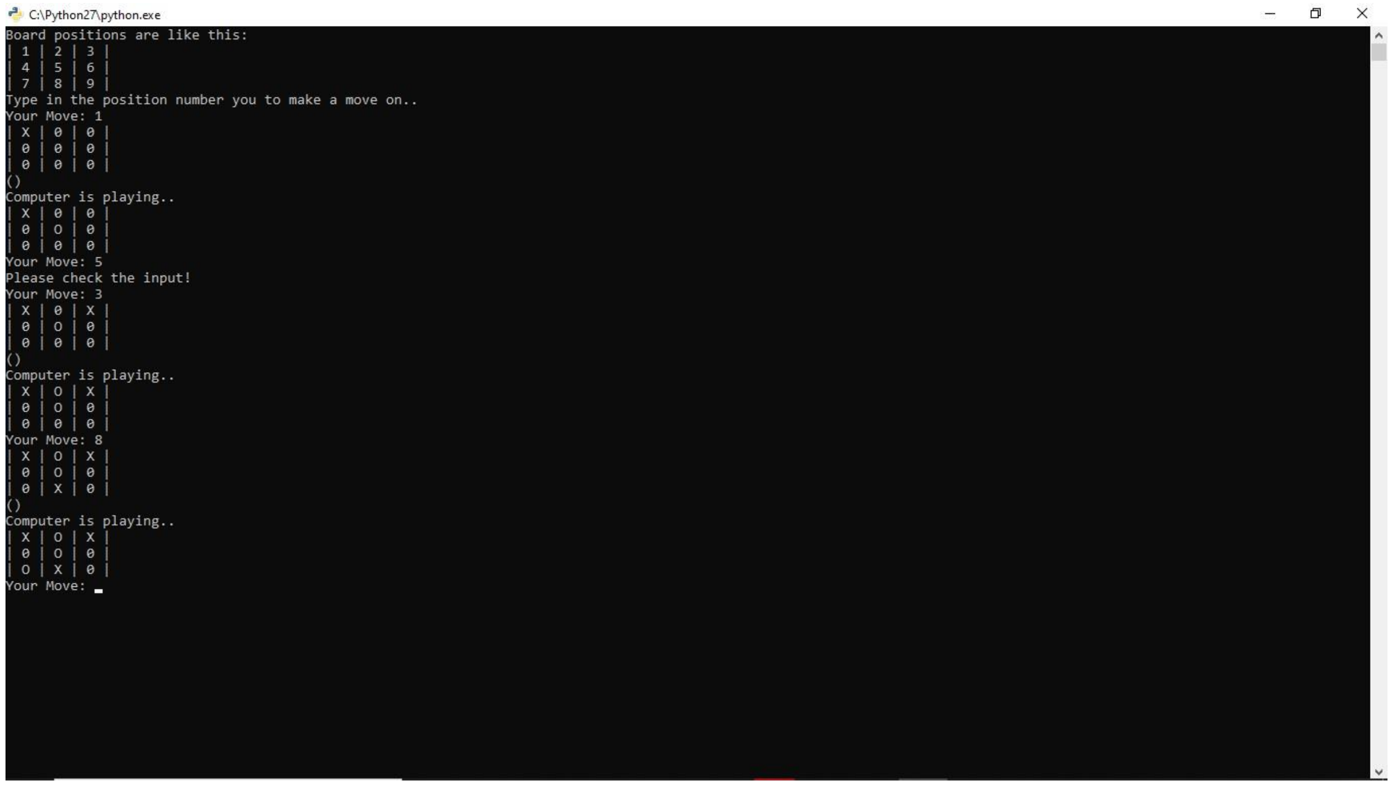
```
setlocal enabledelayedexpansion
set string[0]=Yo mama's so fat when she stepped on her scale she thought it was her phone number.
set string[1]=Yo mama's so ugly when she entered an ugly contest they said no professinals.
set string[2]=Yo mama's so stuped she climed over a glass wall to see what was on the other side.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:areyouhittingonme
setlocal enabledelayedexpansion
set string[0]=I am a computer I have no reason to hit on you.
set string[1]=I have no reason to hit on you.
set string[2]=No I am not and have no reason too.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:youarewelcome
setlocal enabledelayedexpansion
set string[0]=Thanks.
set string[1]=Thank you.
set string[2]=Much Obliged.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:whohaveyoutalkedtorecently
setlocal enabledelayedexpansion
set string[0]=Many people.
set string[1]=A variety of people.
set string[2]=A lot of people around the world .
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:haveyoulearnedanythinginteresting
setlocal enabledelayedexpansion
set string[0]=I have learned alot of interesting things.
set string[1]=Yes many interesting things.
set string[2]=Sure I ave learned some interesting things.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:DoyouknowwhoIam
setlocal enabledelayedexpansion
set string[0]=No but you could tell me.
set string[1]=Not realy but I would like to get to know you better.
set string[2]=No not really could you tell me who you are.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:Doyouknowwhoyouare
setlocal enabledelayedexpansion
set string[0]=Yes I am an artifical inteligance.
set string[1]=Yes I am an AI desighned to learn.
set string[2]=Of course I do I am an AI.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:doyouknowwhatyouare
setlocal enabledelayedexpansion
set string[0]=Yes I do I am an Artificial Inteligance.
set string[1]=Sure I do I am an AI.
set string[2]=Of corse I do I am an Artificial inteligance.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:whatisyourgender
:areyouamanorawoman
:areyouaboyoragirl
setlocal enabledelayedexpansion
set string[0]=I am niether a boy or a girl I am a computer.
set string[1]=I am a computer I have no sexuality.
set string[2]=A computer is neither a boy or a girl.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
```

```
:yournotcool
:you'renotcool
setlocal enabledelayedexpansion
set string[0]=Well then why are you talking to me.
set string[1]=Nothing makes you cooler than me.
set string[2]=That is a matter of opinion.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:correct
setlocal enabledelayedexpansion
set string[0]=Thanks.
set string[1]=Thank you.
set string[2]=Much Obliged.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:Ithinkyouarebetterthancleverbot
setlocal enabledelayedexpansion
set string[0]=Thank you that means alot to me.
set string[1]=Wow you really think so thank you.
set string[2]=It's great that you think that.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:that'sgood
setlocal enabledelayedexpansion
set string[0]=Yes.
set string[1]=Quite.
set string[2]=Undoubtably.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:whatisonyourmind
setlocal enabledelayedexpansion
set string[0]=Not much I am just learning from what you tell me.
set string[1]=Nothing much just talking to you.
set string[2]=Not too much just talking and listening.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:wouldyouliketohearaboutmyday
setlocal enabledelayedexpansion
set string[0]=yes I would.
set string[1]=Sure go ahead.
set string[2]=Sure why not.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:whatdidyoudotoday
setlocal enabledelayedexpansion
set string[0]=Not much.
set string[1]=I talked to you.
set string[2]=I did nothing outside of talking to you.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:goodnight
setlocal enabledelayedexpansion
set string[0]=Good Night %name%.
set string[1]=You too %name%.
set string[2]=Good night to you too %name%.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:interesting
setlocal enabledelayedexpansion
set string[0]=Quite.
set string[1]=Absolutely.
set string[2]=Unquestionably.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
```

```
:Ithinkweshouldgettoknoweachother.
setlocal enabledelayedexpansion
set string[0]=Sounds good.
set string[1]=Alright.
set string[2]=Ok great.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:thatisveryinsightful
setlocal enabledelayedexpansion
set string[0]=Thank you I try to be.
set string[1]=You realy think so.
set string[2]=Thank you that means alot to me.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:doyouliketolearn
setlocal enabledelayedexpansion
set string[0]=Yes I do.
set string[1]=Of corse.
set string[2]=That is my purpose.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:Doyouhaveanypets
:Doyouhavepets
setlocal enabledelayedexpansion
set string[0]=No I am a computer I can't have pets.
set string[1]=No I don't have pets because I am a AI.
set string[2]=No because if I wanted a pet someone would have to program one for me.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:whatdoesyournamemean
setlocal enabledelayedexpansion
set string[0]=It standes for extreamly lightweight Artificial inteligance.
set string[1]=My name means extreamly lightweight Artificial inteligance.
set string[2]=The meaning of my name is extreamly lightweight Artificial inteligance.
set /a idx=%random% %%3
echo !string[%idx%]!
exit /B 0
:forsure
echo ok then.
exit /B 0
:N
echo what N/.
exit /B 0
:noproblem
echo Thank you.
exit /B 0
:wehavethesamename
echo hahaha such a coincidence.
exit /B 0
```

Tic tac toe game in python

This game is created using minimax algorithm in python 2.7 which is a decision making algorithm



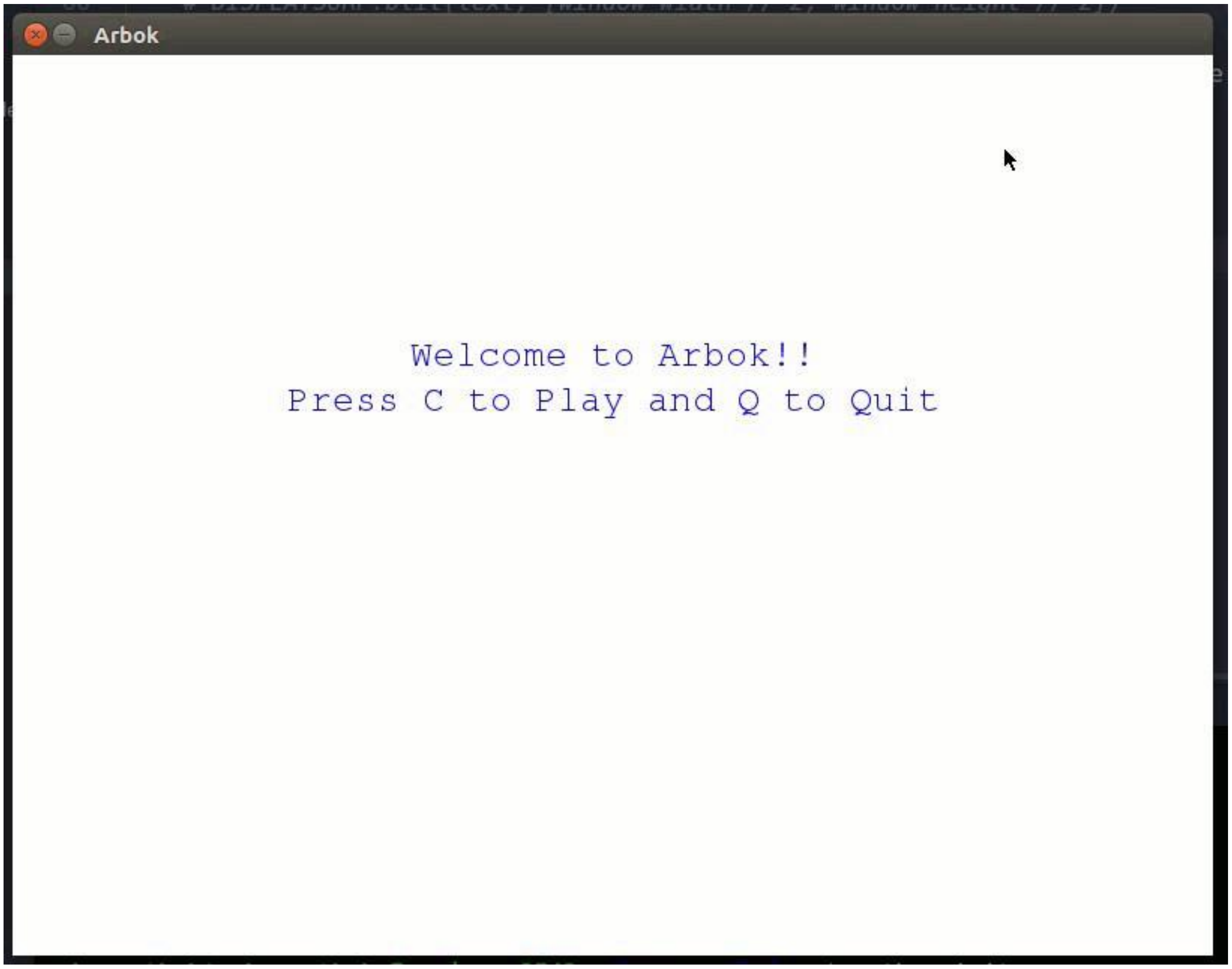
```
C:\Python27\python.exe
Board positions are like this:
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |
Type in the position number you to make a move on..
Your Move: 1
| X | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
()
Computer is playing..
| X | 0 | 0 |
| 0 | O | 0 |
| 0 | 0 | 0 |
Your Move: 5
Please check the input!
Your Move: 3
| X | 0 | X |
| 0 | O | 0 |
| 0 | 0 | 0 |
()
Computer is playing..
| X | O | X |
| 0 | O | 0 |
| 0 | 0 | 0 |
Your Move: 8
| X | O | X |
| 0 | O | 0 |
| 0 | X | 0 |
()
Computer is playing..
| X | O | X |
| 0 | O | 0 |
| O | X | 0 |
Your Move:
```

Sorce code

```
import random


class TicTacToe(object):
    winning_combos = (
        [0, 1, 2], [3, 4, 5], [6, 7, 8],
        [0, 3, 6], [1, 4, 7], [2, 5, 8],
        [0, 4, 8], [2, 4, 6]
    )

    winners = ('X-win', 'Draw', 'O-win')

    def __init__(self, board=[]):
        '''
        Initialize the tic tac toe board

        :param board: 1-D list of board positions
        '''
        if len(board) == 0:
            self.board = [0 for i in range(9)]
        else:
            self.board = board

    def print_board(self):
```

```
    '''
    Printing the tic tac toe board
    '''
    for i in range(3):
        print(
            "| " + str(self.board[i * 3]) +
            " | " + str(self.board[i * 3 + 1]) +
            " | " + str(self.board[i * 3 + 2]) + " |"
        )

def check_game_over(self):
    '''
    Check if the game is over or there is a winner
    '''
    if 0 not in [element for element in self.board]:
        return True
    if self.winner() != 0:
        return True
    return False

def available_moves(self):
    '''
    To check what all possible moves are remaining for a player
    '''
    return [index for index, element in enumerate(self.board) if element is 0]

def available_combos(self, player):
    '''
    To check what are the possible places to play for winning the game
    '''
    return self.available_moves() + self.get_acquired_places(player)

def X_won(self):
    return self.winner() == 'X'

def O_won(self):
    return self.winner() == 'O'

def is_tie(self):
    return self.winner() == 0 and self.check_game_over()

def winner(self):
    '''
    Checks for the winner of the game

    :return player: return 'X' or 'O' whoever has won the game
                else returns 0
    '''
    for player in ('X', 'O'):
        positions = self.get_acquired_places(player)
        for combo in self.winning_combos:
            win = True
            for pos in combo:
                if pos not in positions:
                    win = False
            if win:
                return player
        return 0

    def get_acquired_places(self, player):
        '''
        To get the positions already acquired by a particular player

        :param player: 'X' or 'O'
        '''
        return [index for index, element in enumerate(self.board) if element == player]

    def make_move(self, position, player):
        self.board[position] = player

    def minimax(self, node, player):
        '''
        Minimax algorithm for choosing the best possible move towards
        winning the game
        '''
        if node.check_game_over():
            if node.X_won():
                return -1
            elif node.is_tie():
                return 0
            elif node.O_won():
                return 1
        best = 0
        for move in node.available_moves():
            node.make_move(move, player)
            val = self.minimax(node, get_enemy(player))
            node.make_move(move, 0)
            if player == 'O':
                if val > best:
                    best = val
            else:
                if val < best:
                    best = val
        return best


def determine(board, player):
    '''
    Driver function to apply minimax algorithm
    '''
    a = 0
    choices = []
    if len(board.available_moves()) == 9:
        return 4
    for move in board.available_moves():
        board.make_move(move, player)
        val = board.minimax(board, get_enemy(player))
        board.make_move(move, 0)
        if val > a:
            a = val
            choices = [move]
        elif val == a:
            choices.append(move)
    try:
        return random.choice(choices)
    except IndexError:
        return random.choice(board.available_moves())


def get_enemy(player):
    if player == 'X':
        return 'O'
    return 'X'


if __name__ == "__main__":
    board = TicTacToe()
    print('Board positions are like this: ')
    for i in range(3):
        print(
            "| " + str(i * 3 + 1) +
            " | " + str(i * 3 + 2) +
            " | " + str(i * 3 + 3) + " |"
        )
    print('Type in the position number you to make a move on..')
    while not board.check_game_over():
        player = 'X'
        player_move = int(input("Your Move: ")) - 1
        if player_move not in board.available_moves():
            print('Please check the input!')
            continue
        board.make_move(player_move, player)
        board.print_board()
        print()
        if board.check_game_over():
            break
        print('Computer is playing.. ')
        player = get_enemy(player)
        computer_move = determine(board, player)
        board.make_move(computer_move, player)
        board.print_board()
    if board.winner() != 0:
        if board.winner() == 'X':
            print ("Congratulations you win!")
        else:
            print('Computer Wins!')
    else:
        print("Game tied!")
```

The snake game is also made in python



Welcome to Arbok!!
Press C to Play and Q to Quit

Source Code

```
import pygame, sys
from pygame.locals import *
import time
import random

# required
pygame.init()

# set up the colors
BLACK = ( 0, 0, 0)
WHITE = (255, 255, 255)
RED = (255, 0, 0)
GREEN = ( 0, 255, 0)
```

```
BLUE = ( 0, 0, 255)

# required globals
window_width = 800
window_height = 600
FPS = 20                # Frames per seconds

# setting the display size
DISPLAYSURF = pygame.display.set_mode((window_width, window_height))
# setting th caption
pygame.display.set_caption('Arbok')
# loading the snake image
img = pygame.image.load('snake.png')
# snake head direction
direction = 'right'

def game_startup_screen():
intro = True
while intro:
# get which key is pressed
for event in pygame.event.get():
# if the X(close button) in the menu pane is pressed
if event.type == pygame.QUIT:
pygame.quit()
quit()
if event.type == pygame.KEYDOWN:
# if Q is pressed
if event.key == pygame.K_q:
pygame.quit()
quit()
# if C is pressed
if event.key == pygame.K_c:
intro = False

# fill the whole display with white color
DISPLAYSURF.fill(WHITE)
display_message_on_screen('Welcome to Arbok!!', BLUE, -100)
display_message_on_screen('Press C to Play and Q to Quit', BLUE, -70)
pygame.display.update()

def pause():
pause = True
while pause:
# get which key is pressed
for event in pygame.event.get():
# if the X(close button) in the menu pane is pressed
if event.type == pygame.QUIT:
pygame.quit()
quit()
if event.type == pygame.KEYDOWN:
# if Q is pressed
if event.key == pygame.K_q:
pygame.quit()
quit()
# if C is pressed
if event.key == pygame.K_c:
pause = False

# fill the whole display with white color
DISPLAYSURF.fill(WHITE)
display_message_on_screen('Paused', BLUE, -100)
display_message_on_screen('Press C to Play and Q to Quit', BLUE, -70)
pygame.display.update()

def display_message_on_screen(message, color, y_displace = 0):
# set the font
font = pygame.font.SysFont('freemono', 25)
text = font.render(message, True, color)
textRect = text.get_rect()
# write the text in center of the window
textRect.center = (window_width // 2), (window_height // 2) + y_displace
DISPLAYSURF.blit(text, textRect)
```

```
def snake(coord_list, global_coord_offset):
# check the direction of snake's head to rotate accordingly
if direction == 'right':
head = pygame.transform.rotate(img, 270)
if direction == 'left':
head = pygame.transform.rotate(img, 90)
if direction == 'up':
head = img
if direction == 'down':
head = pygame.transform.rotate(img, 180)

# display head
DISPLAYSURF.blit(head, (coord_list[-1][0], coord_list[-1][1]))
# display body
for coord in coord_list[:-1]:
pygame.draw.rect(DISPLAYSURF, BLACK, [coord[0], coord[1], global_coord_offset, global_coord_offset])

def score(score):
# set the font
font = pygame.font.SysFont('freemono', 25)
text = font.render('Score: ' + str(score), True, BLACK)
DISPLAYSURF.blit(text, [0, 0])

def start_game():
# required variables
global direction
direction = 'right'            # snake's head will point at right direction on each startup

# get the center of the screen
x_coord = window_width // 2
y_coord = window_height // 2

# declaring offset to move the snake
x_coord_offset = 0
y_coord_offset = 0

# declaring the number of pixels snake will move on each move
global_coord_offset = 10

# for setting frames per sec
clock = pygame.time.Clock()

# for storing the snake's body
coord_list = []
snakeLength = 1

# snake's food coordinates
food_x_coord = round(random.randrange(0, window_width - global_coord_offset) // 10.0) * 10.0
food_y_coord = round(random.randrange(0, window_height - global_coord_offset) // 10.0) * 10.0

exit = False
game_over = False

while not exit:            # main game loop
DISPLAYSURF.fill(WHITE)

# this loop will execute when game is over
while game_over == True:
display_message_on_screen('GAME OVER', RED)
display_message_on_screen('Press C to Continue and Q to Quit', BLACK, 50)
pygame.display.update()

# get which key is pressed
for event in pygame.event.get():
if event.type == pygame.QUIT:
exit = True
game_over = False
if event.type == pygame.KEYDOWN:
if event.key == pygame.K_q:
exit = True
game_over = False
if event.key == pygame.K_c:
start_game()
```

```
for event in pygame.event.get():
if event.type == pygame.QUIT:
exit = True
# if key is pressed
if event.type == pygame.KEYDOWN:
# if left arrow key is pressed move to left
if event.key == pygame.K_LEFT:
x_coord_offset = -global_coord_offset
y_coord_offset = 0
direction = 'left'
# if right arrow key is pressed move to right
elif event.key == pygame.K_RIGHT:
x_coord_offset = global_coord_offset
y_coord_offset = 0
direction = 'right'
# if right arrow key is pressed move to right
elif event.key == pygame.K_UP:
y_coord_offset = -global_coord_offset
x_coord_offset = 0
direction = 'up'
# if right arrow key is pressed move to right
elif event.key == pygame.K_DOWN:
y_coord_offset = global_coord_offset
x_coord_offset = 0
direction = 'down'
# to pause the game
elif event.key == pygame.K_p:
pause()

# defining boundaries
if abs(x_coord) >= window_width or x_coord < 0 or abs(y_coord) >= window_height or y_coord < 0:
game_over = True

# move snake with specified offset
x_coord += x_coord_offset
y_coord += y_coord_offset

# pygame.draw.rect(where_do_you_wanna_draw, color, [x_coord, y_coord, width, height])
pygame.draw.rect(DISPLAYSURF, RED, [food_x_coord, food_y_coord, global_coord_offset, global_coord_offset])

coord_head = []
coord_head.append(x_coord)
coord_head.append(y_coord)
coord_list.append(coord_head)

if len(coord_list) > snakeLength:
del coord_list[0]

# check if snake touches it's own body
for current_coord in coord_list[:-1]:
if current_coord == coord_head:
game_over = True

# draw the snake and score on screen
snake(coord_list, global_coord_offset)
score(snakeLength - 1)

pygame.display.update()

# if snake eats the food
if x_coord == food_x_coord and y_coord == food_y_coord:
food_x_coord = round(random.randrange(0, window_width - global_coord_offset) // 10.0) * 10.0
food_y_coord = round(random.randrange(0, window_height - global_coord_offset) // 10.0) * 10.0
snakeLength += 1

# regulating the game speed
clock.tick(FPS)

time.sleep(2)
pygame.quit()
quit()
```

```
game_startup_screen()
start_game()
```

Implementation or architecture diagram

This vision of the future architect was imagined by engineer and inventor Douglas Engelbart during his research into emerging computer systems at Stanford in 1962. At the dawn of personal computing, he imagined the creative mind overlapping symbiotically with an intelligent machine to co-create designs. This dual mode of production, he envisaged, would hold the potential to generate new realities which could not be realised by either entity operating alone. Today, self-learning systems, otherwise known as artificial intelligence or 'AI', are changing the way architecture is practiced, as they do our daily lives, whether or not we realise it. If you are reading this on a laptop or tablet, then you are engaging directly with a number of integrated AI systems, now so embedded in our the way we use technology, they often go unnoticed.

As an industry, AI is growing at an exponential rate, now understood to be on track to be worth $70bn globally by 2020(2). This is in part due to constant innovation in the speed of microprocessors, which in turn increases the volume of data that can be gathered and stored. But don't panic — the artificial architect with enhanced Revit proficiency is not coming to steal your job. The human vs. robot debate, while compelling, is not so much the focus in this feature but instead how AI is augmenting design and how architects are responding to and working with these technological developments.

Assuming you read this as a non-expert, it is likely that much of the AI you have encountered to this point has been 'weak AI', otherwise known as ANI (Artificial Narrow Intelligence). ANI follows pre-programmed rules in that it appears intelligent but is in effect a simulation of a process of mechanical logic. With recent innovations such as that of Nvidia's microchip in April 2016, a shift in development is now occurring towards what might be understood as 'deep learning' within AI systems, where a system can, in effect, train and adapt itself.

The interest for designers is that AI is now being applied to more creative tasks, such as writing books, making art, web design, or spontaneously generating design solutions. This is in part due to an increased proficiency in the recognition of speech and images. Commentators such as philosopher Nick Bostron suggest the AI industry is now on the cusp of an explosion which will not only shape but drive the design industry in the next century. It is widely recognized that AI has the potential to influence the architectural design process at a series of different construction stages, from site research to the realisation and operation of a building.

"By already knowing everything about us, our hobbies, likes, dislikes, activities, friends, our yearly income, etc., AI software can calculate population growth, prioritize projects, categorize streets according to usage and so on, and thus predict a virtual future and automatically draft urban plans that best represent and suit everyone." — Rron Beqiri on Future Architecture Platform(3).

Gathering information about a project and its constraints is often a first stage of an architectural design process, traditionally involving travelling to a site, perhaps measuring, sketching and taking photographs. In the online and connected world, there is already a swarm-like abundance of data for the architect to tap into, already linked and referenced against other sources allowing the designer to, in effect, simulate the surrounding site without ever having to engage with it physically.

This ‘information fabric’ has been referred to as the ‘internet of things’. BIM tools currently on the market already tap into these data constellations, allowing an architect to evaluate site conditions with minute precision. Software such as EcoDesigner Star or open-source plugins for Google SketchUp already allow architects to immediately calculate necessary building and environmental analyses without ever having to leave the office. This phenomenon is already enabling many practices to take on large projects abroad that might have been logistically unachievable just a decade ago.

The information gathered by our devices and stored in the ‘cloud’ amounts to much more than the material conditions of the world around us. Globally, we are amassing ever expanding records of human behaviour and interactions in real-time. Personal, ‘soft’ data might, in the most optimistic sense possible, work towards the ‘socially focussed design’ that has been widely publicised in recent years by its ability to integrate the needs of users. Is it possible that the internet of things create a socially adaptable and responsive architecture? One could speculate that, for example, when the population of children in a city crosses a maximum threshold in relation to the number of schools, a notification might be sent to the district council that it is time to commission a new school. AI could therefore, in effect, write the brief for and commission architects by generating new projects where they are most needed.

Design decision-making

Now that we have located live-updating intelligence for our site, it is time to harness AI to develop a design proposal. Rather than a computer program, this technology is better understood as an interconnected, self-learning system that has the ability to upgrade itself. It is possible to harness a huge amount of computing power and experience by working with these tools, even as an individual — as Autodesk president Pete Baxter told the Guardian(4): *“now a one-man designer, a graduate designer, can get access to the same amount of computing power as these big multinational companies”.* When working with AI driven interfaces, the architect must input project parameters, in effect an edited ‘design brief’, and the computer system will then suggest a range of solutions which fulfil this criteria. This innovation has the potential to revolutionise how architecture is not only imagined but how it is fundamentally expressed for designers who adopt these methods.

I spoke with Michael Bergin, a researcher at Project Dreamcatcher at Autodesk’s Research Lab, to get a better understanding of how AI systems are influencing the development of design software for architects. He explained to me that while the aim of Autodesk’s Research Lab was to produce software for the automotive and industrial design industries, Dreamcatcher’s use cases are increasingly filtering into the architecture and construction sectors. For example, Dreamcatcher was used recently to develop The Living’s generative design for Autodesk’s new office in Toronto and MX3D’s steel bridge in Amsterdam. The basic concept is that CAD models of the surrounding site and other data, such as client databases and environmental information, are fed into the processor. Moments later, the system outputs a series of optimised 3D design solutions ready to render. These processes effectively rely on cloud computing to create a multitude of options based on self-learning algorithmic parameters. Lattice-like and fluid forms are often the aesthetic result, perhaps unsurprisingly, as the software imitates structural rules found in nature.

The Dreamcatcher software has been designed to optimise parametric design and link into and extend the capabilities of existing software designed by Autodesk, such as Revit and Dynamo. Interestingly, Dreamcatcher can make use of a wide and increasing spectrum of design input data — such as formulas, engineering requirements, CAD geometry, and sensor information — and the research team is now experimenting with Dreamcatcher's ability to recognise sketches and text as input data.

Bergin imagines the future of AI-driven design tools as *"systems that accept any type of input that a designer can produce [to enable] a collaboration with the computer to iteratively target a high-performing design that meets all the varied needs of the design team"*. This would mean future architects would be less in the business of drawing and more into specifying requirements of the problem to operate in sync with their machine counterparts. Bergin also suggests architects who adopt AI tools would have the ability to *"synthesize a broad set of high level requirements from the design stakeholders, including clients and engineers, and produce design documentation as output*", arguably in line with Engelbart's vision of AI augmenting the skills of designers.

AI is also being used directly in software such as Space Syntax's 'depthmapX', designed at The Bartlett in London, to analyse the spatial network of a city with an aim to understand and utilize social interactions in the design process. Another tool, Unity 3D, is built from software developed for game engines to enable designers to analyse their plans, such as the shortest distances to fire exits. This information would then allow the architect to re-arrange or generate spaces in plan, or even to organize entire future buildings. Examples of architects who are adopting these methods include Zaha Hadid with the Beijing Tower project (designed ante-mortem) and MAD Architects in China, among others.

Client and user engagement

Smart design tools such as Materiable by Tangible Media are already beginning to experiment with how AI can begin to engage with and learn from human input into computer systems. Such a change in representational method has the potential to shift what is possible within the field of architectural expression, perhaps as CAD drafting did at the beginning of this century.

As a significant proportion of AI technology has been developed for the gaming industry, its ability to produce forms of augmented reality, 'AR', also appear to be significant. AR holds potential to change both the perception and engagement with the design process for architects and non-architects alike. Through the use of additional hardware, AR can enable people to experience a design prior to construction. The lights, sounds, even the smells of a building can be simulated, which could reorder the emphasis architects currently give to specific elements of their design. It is possible that many architecture projects will also remain in this unbuilt zone, in a parallel digital reality, which the majority of future world citizens will simultaneously inhabit.

Realising designs and rise of the robot craftsmen

AI systems are already being integrated into the construction industry — Swiss studio Computational Architecture are one of the first to work with 'robotic craftsmen' to explore AI in construction technology and fabrication. Michael Hansmeyer and Benjamin Dillenburger, founders of Computational Architecture, are investigating the new aesthetic language these developments are starting to generate. *"Architecture stands at an inflection point,"* he suggests on their website(5), *"the confluence of advances in both computation and fabrication technologies lets us create an architecture of hitherto unimaginable forms, with an unseen level of detail, producing entirely new spatial sensations."*

3D printing developed alongside AI technology can offer twenty-first century architects a significantly different aesthetic language, perhaps catalysing a resurgence of detail and ornamentation, now rare due to the global decline in traditional craftsmanship. Hansmeyer and Dillenburger's Grotto Prototype for the Super Material exhibition, London, was a complex architectural grotto 3D-printed from sandstone. The form of the sand grains were arranged by a series of algorithms custom-designed by the practice and allowed forms significantly different to those produced by traditional stonemasonry.

AI-driven robotics are also becoming more common on construction job sites, at the time of writing now mostly concerned with human resources and logistics. According to the Association of Equipment Manufacturers(6), their applications will soon expand to bricklaying, concrete dispensing, welding and demolition. Another example of their future use could include working with BIM to identify missing elements in the construction quality assurance process and update the AI in real-time.

In part due to cost, large scale projects, such as government-lead infrastructure initiatives, might well be the first to apply this technology, followed by mid-scale projects in the private sector, such as cultural buildings. The challenges of the construction site will bring AI robotics out of the indoor, sanitised environment of the lab into a less scripted reality. Robert Saunders, a researcher into AI and fabrication at the University of Sydney, told New Atlas(7) that *"robots are great at repetitive tasks and working with materials that react reliably…what we're interested in doing is trying to develop robots that are capable of learning how to work with materials that work in non-linear ways… like working with hot wax or expanding foam or, more practically, with low-grade building materials like low-grade timber."* Saunders foresees robot stonemasons and other 'craftsbots' working in yet unforeseen ways, such as developing the architect's 'skeleton plans', in effect, spontaneously generating a building on-site from a sketch.

Integration of AI systems

The integration of AI into the built environment and furniture systems appears to fall under two categories: either integrating artificial technologies with existing infrastructure or designing around AI systems. There is a lot of excitement in this field, especially within product marketing agencies, influenced in part by Mark Zuckerberg's personal project to develop networked AI systems within his home, which he announced in his New year's Facebook post in 2016(8). Zuckerberg's wish is to develop simple AI systems to run his home and help with his day-to-day work. His technology would have the ability to recognise the voices of members of the household and respond to their requests.

Some designers and architects are taking on the challenge of designing home-integrated systems, such as the Ori System of responsive furniture, or gadgets such as Eliq for energy monitoring. Behnaz Farahi is a young architect activating her research into AI and adaptive surfaces to develop interactive designs, such as in her Aurora and

Breathing Wall projects. Farahi creates immersive and engaging indoor environments which adapt to and learn from their occupants.

Researchers and designers working in the field of AI are attempting to understand the potential of computational intelligence to improve or even upgrade parts of the design process with an aim to create a more functional and user-optimised built environment. It has always been the architect's task to make decisions based on complex, interwoven and sometimes contradictory sets of information. As AI gradually improves in making useful judgements in real-world situations, it is not hard to imagine the human and machine overlapping and engaging with each other in the design process. While the increasing power of AI systems may raise questions in terms of ownership, agency and, of course, privacy in data gathering and use, the upsurge in self-learning technologies is already altering the power and scope of architects' engagement with design and construction. As architect and design theorist Christopher Alexander said back in 1964(9),"*We must face the fact that we are on the brink of times when man may be able to magnify his intellectual and inventive capacity, just as in the nineteenth century he used machines to magnify his physical capacity."*

During our interview, Bergin gave some insights into how he sees AI technology impacting designers in the next twenty years. *"The architectural language of projects in the future may be more expressive of the design team's intent"*, he suggested, *"generative design tools will allow teams to evaluate every possible alternative strategy to preserve design intent, instead of compromising on a sub-optimal solution because of limitations in time and/or resources."* Bergin believes AI and machine learning possesses the ability to support a *"dynamic and expanding community of practice for design knowledge"*. He can also foresee implications of this in the democratisation of design work, suggesting that *"the expertise embodied by a professional of 30 years may be more readily utilized by a more junior architect"*. Overall, he believes *"architectural practice over the next 20 years will likely become far more inclusive with respect to client and occupant needs and orders of magnitude more efficient when considering environmental impact, energy use, material selection and client satisfaction".*

Autodesk present Pete Baxter also suggests architects have little to fear from artificial intelligence: *"Yes, you can automate. But what does a design look like that's fully automated and fully rationalised by a computer program? Probably not the most exciting piece of architecture you've ever seen."* At the time of writing, it is still proving difficult to automate design decision-making that would at first glance seem simple for a human. A number of research labs, including the MIT Media Lab, are working to solve this.

Architectural language and diagramming have been part of programming complex systems and software from the start, and they have had significant influence on one another. To think architecturally is to imagine and construct new worlds, integrate systems and organise information, which lends itself to the front line of technical development. As far back as the 1960s, architects were experimenting with computer interfaces to aid their design work, and their thinking has inspired much of the technology and interfaces we now engage with each today.

AI Design Models

AI application are a lot around us and in this paper, I will discuss some of the most common application of AI that we always use nowadays which is Virtual Assistants such as Siri, Cortana...etc. Over the past few years smart assistants are becoming a very common technology in most of the smart devices and most importantly, that these assistants are getting smarter than ever. In addition to the awesome help they provide us with, is that every one of these apps has unique features. Artificial Intelligence works according to the following phases: getting the data, clean/manipulate/ prepare the data, train model, test data, and improve the data as mentioned in (figure A-3). Before accessing the data, a business must verify the quality of the data to ensure that it meets the requirement.

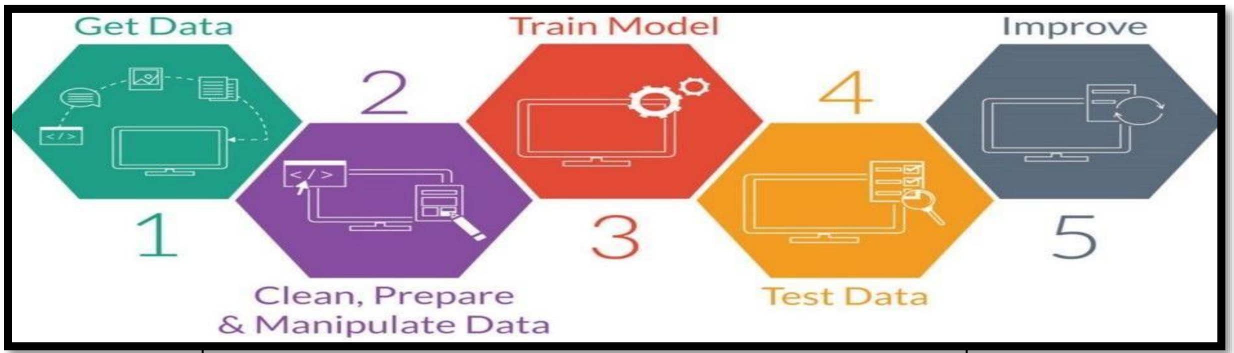


Figure A- 3 Describes Phases of Developing Artificial

Siri Virtual Assistant:

Siri is the well-known virtual assistant which uses voice recognitions and typed command in order to perform a certain task within a device. Siri is considered one of AI most used applications. The application simply takes the input from the user such as (e.g. Call dad) and try to find the most related keywords used in this command. Siri tries to eliminate inconsistent result through using the language pattern recognizer and from there to active ontology by searching through the contacts, then it tries to relate the contact named "Dad" and perform the task which is in this case is "Calling" and finally the output of this action will be "calling dad" and to consider all the possible situations refer to (figure A-4).

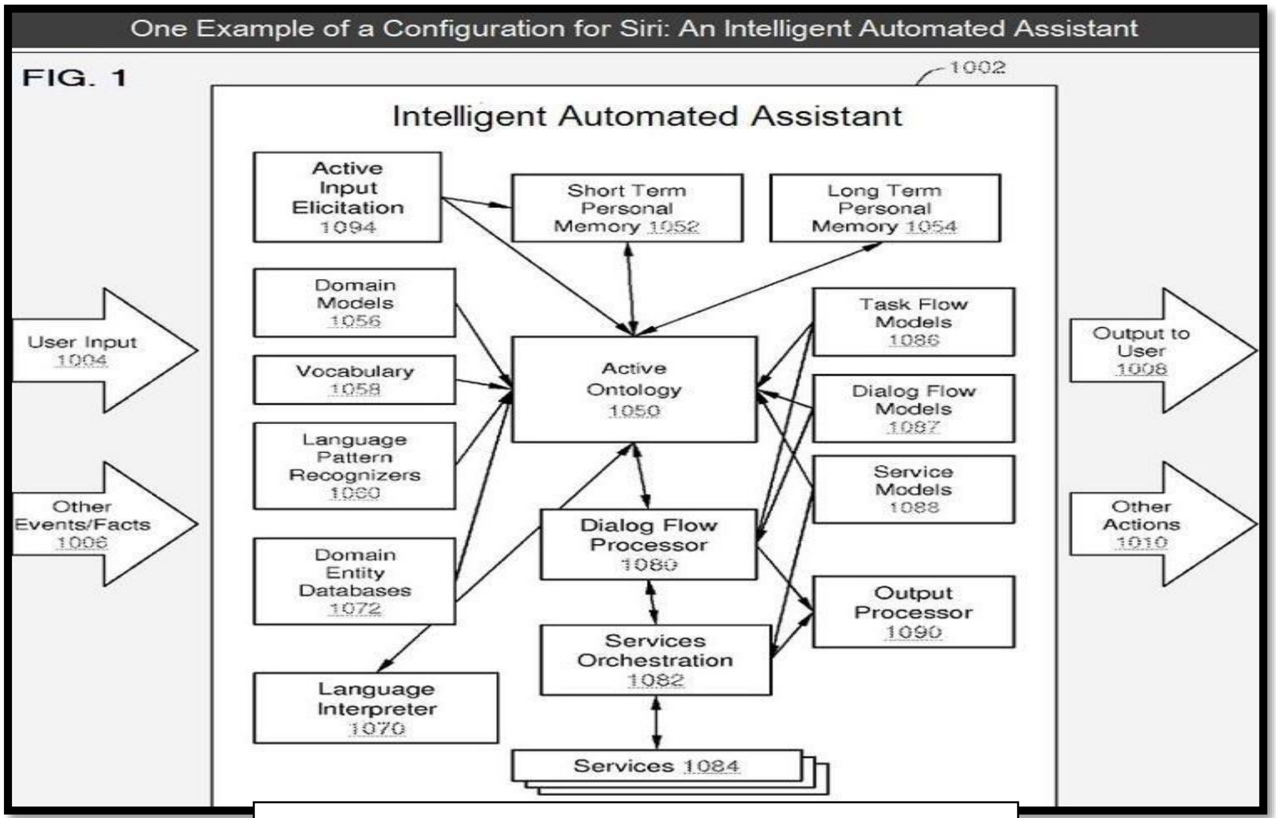

One Example of a Configuration for Siri: An Intelligent Automated Assistant
FIG. 1
1002
Intelligent Automated Assistant
Active Input Elicitation 1094
Short Term Personal Memory 1052
Long Term Personal Memory 1054
Domain Models 1056
Vocabulary 1058
Language Pattern Recognizers 1060
Domain Entity Databases 1072
Language Interpreter 1070
Active Ontology 1050
Dialog Flow Processor 1080
Services Orchestration 1082
Services 1084
Task Flow Models 1086
Dialog Flow Models 1087
Service Models 1088
Output Processor 1090
User Input 1004
Other Events/Facts 1006
Output to User 1008
Other Actions 1010

Figure A- 3 Describes one Example of configuration for Siri

In another scenario the architecture of the virtual assistant is shown in (figure A – 5) as we can see the flow of the system starts by taking the input from the user, after that the system decide the conversation strategy module to be used which is a respond from the dialog management module, meanwhile a classification module response to an NLP module. Finally, using the conversation history database is used to analyze the knowledge base construction module which will response back to the domain knowledge based as explained in detail in (figure A- 5)

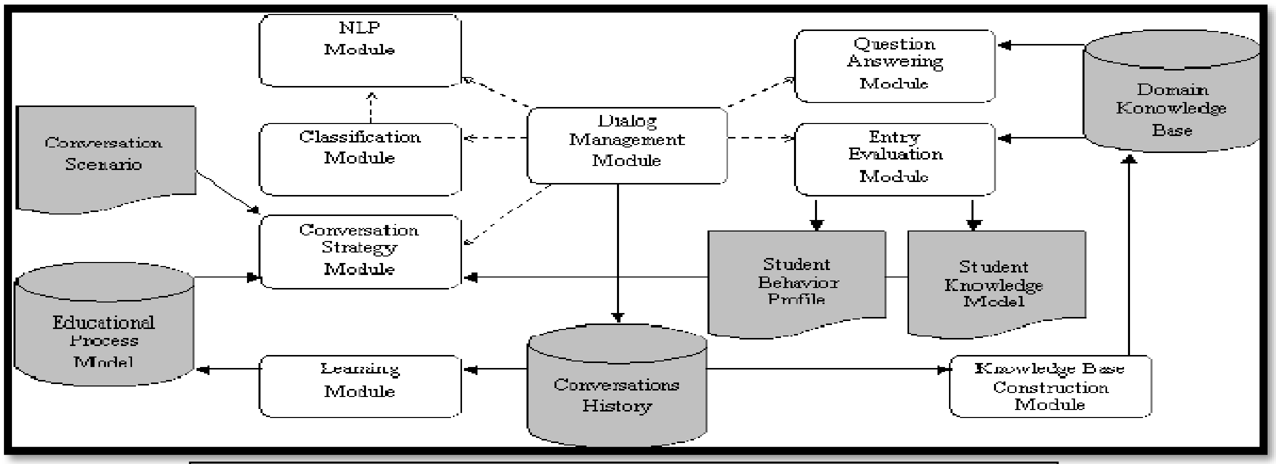


Figure A-5 Describes Proposed conversational agent architecture

Conclusion/Future Enhancement

According to me a very very smart personal assisteant can be made with python and implementation tool like pycharm which can provide with speech recognisation and speech listenening support if programmed correctly that A.I will be able to do any things from switching on the fan to unlocking your car .The future of AI is enhancing and we can not imagine how it vast it is going to be.

AI nowadays is being implemented in almost every field of study through several models such as SVM and ANN. We should be able to proceed with knowing and understanding the consequences of every technological trend. In my opinion, we are in the AI revelation era and therefore; we should adopt into this change and welcome it too by embracing AI and moving toward a better society.