African Virtual University

Applied Computer Science: CSI 3106

ARTIFICIAL INTELLIGENCE

Saffiong Kebbeh

Foreword

The African Virtual University (AVU) is proud to participate in increasing access to education in African countries through the production of quality learning materials. We are also proud to contribute to global knowledge as our Open Educational Resources are mostly accessed from outside the African continent.

This module was developed as part of a diploma and degree program in Applied Computer Science, in collaboration with 18 African partner institutions from 16 countries. A total of 156 modules were developed or translated to ensure availability in English, French and Portuguese. These modules have also been made available as open education resources (OER) on oer.avu.org.

On behalf of the African Virtual University and our patron, our partner institutions, the African Development Bank, I invite you to use this module in your institution, for your own education, to share it as widely as possible and to participate actively in the AVU communities of practice of your interest. We are committed to be on the frontline of developing and sharing Open Educational Resources.

The African Virtual University (AVU) is a Pan African Intergovernmental Organization established by charter with the mandate of significantly increasing access to quality higher education and training through the innovative use of information communication technologies. A Charter, establishing the AVU as an Intergovernmental Organization, has been signed so far by nineteen (19) African Governments - Kenya, Senegal, Mauritania, Mali, Cote d'Ivoire, Tanzania, Mozambique, Democratic Republic of Congo, Benin, Ghana, Republic of Guinea, Burkina Faso, Niger, South Sudan, Sudan, The Gambia, Guinea-Bissau, Ethiopia and Cape Verde.

The following institutions participated in the Applied Computer Science Program: (1) Université d'Abomey Calavi in Benin; (2) Université de Ougagadougou in Burkina Faso; (3) Université Lumière de Bujumbura in Burundi; (4) Université de Douala in Cameroon; (5) Université de Nouakchott in Mauritania; (6) Université Gaston Berger in Senegal; (7) Université des Sciences, des Techniques et Technologies de Bamako in Mali (8) Ghana Institute of Management and Public Administration; (9) Kwame Nkrumah University of Science and Technology in Ghana; (10) Kenyatta University in Kenya; (11) Egerton University in Kenya; (12) Addis Ababa University in Ethiopia (13) University of Rwanda; (14) University of Dar es Salaam in Tanzania; (15) Universite Abdou Moumouni de Niamey in Niger; (16) Université Cheikh Anta Diop in Senegal; (17) Universidade Pedagógica in Mozambique; and (18) The University of the Gambia in The Gambia.

Bakary Diallo

The Rector

African Virtual University

Production Credits

Author

Kebbeh Saffiong

Peer Reviewer

Robert Oboko

AVU - Academic Coordination

Dr. Marilena Cabral

Overall Coordinator Applied Computer Science Program

Prof Tim Mwololo Waema

Module Coordinator

Florence Tushabe

Instructional Designers

Elizabeth Mbasu

Benta Ochola

Diana Tuel

Media Team

Sidney McGregor

Barry Savala

Edwin Kiprono

Kelvin Muriithi

Victor Oluoch Otieno

Michal Abigael Koyier

Mercy Tabi Ojwang

Josiah Mutsogu

Kefa Murimi

Gerisson Mulongo

Copyright Notice

Supported By

AVU Multinational Project II funded by the African Development Bank.

Table of Contents

Course Overview

Welcome to Artificial Intelligence Unit 0

Artificial Intelligence is a subject is one of the most important and exciting sub-fields of Computer Science. This specialist programme covers the fundamentals of Computer Science and Computer Information Systems so as to maximise your future employment opportunities. It also offers the chance to carry out cutting-edge research in this high profile area.

Introductory modules cover Programming in Java, Computer Systems, Databases, Human-Centric Computing, and Algorithmic Foundations. The course is completed by a selection of modules covering important topics within the subject: Knowledge Representation, the study of Multi-Agent Systems, and Robotics among the others. The course also includes a related second year group project and an individual year project.

Prerequisites

No prerequisites required, however, if the learner is not comfortable with using computers he/she will not do well in this course

Materials

The materials required to complete this course are:

- Computers/Laptops
- Internet connectivity
- Software tools
- Social Networks and Virtual Platforms

Course Website:

This will be supplemented by readings handed out in class. You will

also need a book with coverage of Swing (Java's GUI toolkit or Visual

Studio) to refer to. Some other good books to own are listed below:

Russell and Norvig. Artificial Intelligence: A Modern Approach. A comprehensive reference for all the AI topics that we will cover.

Koller and Friedman. Probabilistic Graphical Models. Covers factor graphs and Bayesian networks (this is the textbook for CS228).

Sutton and Barto. Reinforcement Learning: An Introduction. Covers Markov decision processes and reinforcement learning. Available free online.

Course Goals

Upon completion of this course the learner should be able to:

Identify the major classical and modern AI paradigms,

Analyze the concepts and techniques of Artificial Intelligence.

Learning AI by developing skills of using AI algorithms for solving practical problems.

Analyze the structure of a given problem such that they can choose an appropriate method implement AI algorithms.

To appreciate an intelligent machine which is capable of reasoning, planning, solving problems, thinking abstractly, comprehending complex ideas, learning quickly and learning from experience.

Units

Unit 0: Pre-Assessment

Unit 1: Fundamentals of Artificial Intelligence

- The study of mental faculties through the use of computational models
- The study of computations that make it possible to perceive, reason, and act.
- The study of natural intelligence by the use of computer models.

Unit 2: Artificial Intelligence Programming

- Programming logic
- Designing and implementing intelligent components for interactive distributed computational media
- Developing tools for authoring the knowledge needed by such systems

Unit 3: Machine Learning and Games

- Computer Vision to perceive objects,
- Robotics to move them about.
- Machine Learning to adapt to new circumstances and to detect and extrapolate patterns.

Unit 4: Natural Language, Representation and Reasoning

- Natural Language Processing to enable it to communicate successfully in English (or some other human language).
- Knowledge Representation to store information provided before or during the interrogation.
- Automated Reasoning to use the stored information to answer questions and to draw new conclusions.

Assessment

Formative assessments, used to check learner progress, are included in each unit.

Summative assessments, such as final tests and assignments, are provided at the end of each module and cover knowledge and skills from the entire module.

Summative assessments are administered at the discretion of the institution offering the course. The suggested assessment plan is as follows:

1	[Assignments / Homeworks]	[35%]
2	[Test / Quiz]	[35%]
3	[Final Exams]	[30%]

Schedule

Unit	Activities	Estimated time
Unit 1	This unit aims to present the basic representation and reasoning paradigms used in AI in both theory and practice with careful attention to the underlying principles of logic, search, and probability. After having completed the course the student will be able to, give an overview of the field of artificial intelligence, its background, history, fundamental issues, challenges and main directions. The learner will be further able to interpret and formulate knowledge representations in the form of logic expressions.	16 Hours
Unit 2	This unit will provide you with an introduction to applied A.I. via programming features that support basic AI applications. By satisfying the goals of this unit, the learner will have a familiarity with AI programming and be able to use it in future models to implement various AI applications.	16 Hours

Unit 3	Expanding upon the material covered in Unit II, students will develop more advanced programs, and delve into the real-life aspects of the basic components of building and applying prediction functions with an emphasis on practical applications. Upon completion of this unit, the learner will understand the components of a machine learning algorithm, will also know how to apply multiple basic machine learning tools and further learn to apply these tools to build and evaluate predictors on real data.	20 Hours
Unit 4	Expanding upon the material covered in Unit III, students will be introduced to natural language processing and knowledge representation with emphasis on constructing computer programs that underfstand natural language. The goal is to be able to acquire information in order to fill in gaps with a knowledge - reasonong systems based on case - base reasoning methodology	20 Hours

Readings and Other Resources

Unit 0

Required readings and other resources:

Russell and Norvig, Artificial Intelligence, A Modern Approach

Koller Friedman, Probablistic Graphical Models

Unit 3

Required readings and other resources:

Hastie, Tibshirani and Friedman, The Elements of Statistical Learning, Covers Machine Learning (Available free online)

[Sutton and Barto, An Introduction to Re-enforcement Learning (Available free online)

Unit 4

Required readings and other resources:

Artificial Intelligence, A Modern Approach (3rd Edition) Stuart and Russell and Peter Norvig. Prentice Hall (2010), ISBN: 0-13-604259-7

James Allen, Natural Language Understanding, Benjammin/Cunmmings, 2nd Edition 1995.

Unit 0. Pre-Assessment

Unit Introduction

This unit is an introductory survey of artificial intelligence. The course will cover the history, theory, and computational methods of artificial intelligence. Basic concepts include representation of knowledge and computational methods for reasoning. The successful student will finish the course with specific modeling and analytical skills (e.g., search, logic, probability), knowledge of many of the most important knowledge representation, reasoning, and machine learning schemes, and a general understanding of AI principles and practice. The course will serve to prepare the student for further study of AI, as well as to inform any work involving the design of computer programs for substantial application domains.

Unit Objectives

Upon completion of this unit you should be able to:Identify the major classical and modern AI paradigms, and explain how they relate to each other

Analyze the structure of a given problem such that they can choose an appropriate paradigm in which to frame that problem

Implement a wide variety of both classical and modern AI algorithms.

Key Terms

Agent:A program that performs some informationgathering or processing task in the background. Typically, an agent is given a very small and well defined task.

Artificial Intelligence: The capability of computers or programs to operate in ways believed to mimic human thought processes, such as reasoning and learning.

Data: Facts or figures to be processed; evidence,records,statistics, etc. from which conclusions can be inferred information.

Demon: A utility that resides in RAM, waiting in the background until an event triggers it to take action. Print spoolers, e-mail handlers, and automatic backup utilities are examples of daemons.

Formular: Functions within computer programs. These formulas are used to process data that is passed through the function and output the result.

Intelligence: Is abstract thought, understanding, self-awareness, communication, reasoning, learning, having emotional knowledge, retaining, planning and problem solving.

Unit Assessment

Check your understanding!

[Enter Assessment Title here]

Instructions

It is hoped that this course can be graded 100% on homework, but the instructor reserves the right to give exams if there is a significant case of academic dishonesty. In that case, there will be in-class exams and possibly a final exam during the regularly scheduled final time. The final is cumulative. Exams are closed book and no electronic devices are allowed.

Grading Scheme

Ideally, the course will be graded entirely based on the homework assignments and quizzes, with homeworks worth varying amounts based on estimated difficulty.

If no exams are given, then home works will count for 90% of the course grade, and quizzes will count for 10%.

If there is a significant case of academic dishonesty, then exams will count for 50% of the course grade, homework for 45%, and quizzes for 5%. If there is a final it will count as two in-class exams.

Answers

Your thoughts and concerns on this course are important. You are encouraged to give feedback to the instructor and teaching fellow throughout the term. As always students will be asked to fill out a course evaluation at the end of the term.Your course grade will also be available on the course website. Please laise with your course instructor to confirm the web address of your course website, as this vary from different institutions of learning.

Unit Readings and Other Resources

The textbook for the course is Artificial Intelligence: A Modern Approach (3rd edition). Stuart Russell and Peter Norvig, Prentice Hall (2010) ISBN: 0-13-604259-7

The readings in this unit are to be found at the course-level section “Readings and Other Resources”.

Unit 1. Fundamentals of Artificial Intelligence

Unit Introduction

Artificial Intelligence (AI) is a field that has a long history but is still constantly and actively growing and changing. In this unit, the learner will learn the basics of modern AI as well as some of the representative applications of AI. During the course of the study, the learner will appreciate the basics and applications of AI, including: machine learning, probabilistic reasoning, scope, problems, and approaches of AI.

This course unit provides a first approach to answering the following questions. What methods are there that can help understanding complicated systems or programs? How can a program do what it is intended it to do? If there are two ways of solving the same problem, How can human understand what computers can do in principle, and are there problems that are not solvable by a computer?

Key Terms

Algorithm: A process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.

Artificial Intelligence: The theory and development of computer systems able to perform tasks normally requiring human intelligence.

Backward Search:A search that starts from the goal and searches for actions that could achieve that goal, recursively.

Decision Making: The action or process of making important decisions.

Machine Learning: A type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed

Probability: The quality or state of being probable; the extent to which something is likely to happen or be the case

Robots: A machine capable of carrying out a complex series of actions automatically, especially one programmable by a computer.

Learning Activities

Activity 1 - [Overview of Artificial Intelligence]

Introduction

This activity introduces the study of Artificial Intelligence (AI) for learners in all course streams. It is designed to stand alone as an introduction to AI, and provides a background for more advanced study. The activity presents A.I. from a probabilistic viewpoint, and is centred around two specific problems: (i) cognitive thinking; (ii) rational thinking. The lectures will present the main theoretical ideas needed to tackle these problems; the examples classes will reinforce these through paper-and-pencil exercises, and the labs will involve the development of programs to solve them. There will be one hour of lectures and one hour of examples classes each week, as well as five two-hour lab sessions over the semester.

Activity Details

Introduce mathematical notions relevant to Artificial Intelligence and their applications;

Basic familiarity with complex numbers and the standard operations for A.I.

Thinking humanly: The cognitive modelling approach

If we are going to say that a given program thinks like a human, we must have some way of determining how humans think. We need to get inside the actual workings of human minds. There are two ways to do this: Through introspection (trying to catch our own thoughts as they go by) Through psychological experiments. Once we have a sufficiently precise theory of the mind, it becomes possible to express the theory as a computer program. If the program's input/output and timing behavior matches human behavior, that is evidence that some of the program's mechanisms may also be operating in humans. For example, Newell and Simon, who developed GPS, the ``General Problem Solver'' (Newell and Simon, 1961), were not content to have their program correctly solve problems. They were more concerned with comparing the trace of its reasoning steps to traces of human subjects solving the same problems. This is in contrast to other researchers of the same time (such as Wang (1960)), who were concerned with getting the right answers regardless of how humans might do it. The interdisciplinary field of cognitive science brings together computer models from AI and experimental techniques from psychology to try to construct precise and testable theories of the workings of the human mind.

Thinking rationally: The laws of thought approach

Thinking rationally:The laws pf thought approach

The Greek philosopher Aristotle was one of the first to attempt to codify ``right thinking,'' that is, irrefutable reasoning processes. His famous syllogisms provided patterns for argument structures that always gave correct conclusions given correct premises. For example, ``Socrates is a man; all men are mortal; therefore Socrates is mortal.'' These laws of thought were supposed to govern the operation of the mind, and initiated the field of logic. The development of formal logic in the late nineteenth and early twentieth centuries, provided a precise notation for statements about all kinds of things in the world and the relations between them. (Contrast this with ordinary arithmetic notation, which provides mainly for equality and inequality statements about numbers.) By 1965, programs existed that could, given enough time and memory, take a description of a problem in logical notation and find the solution to the problem, if one exists. (If there is no solution, the program might never stop looking for it.) The logicist tradition within artificial intelligence hopes to build on such programs to create intelligent systems.

There are two main obstacles to this approach. First, it is not easy to take informal knowledge and state it in the formal terms required by logical notation, particularly when the knowledge is less than 100% certain. Second, there is a big difference between being able to solve a problem ``in principle'' and doing so in practice. Even problems with just a few dozen facts can exhaust the computational resources of any computer unless it has some guidance as to which reasoning steps to try first. Although both of these obstacles apply to any attempt to build computational reasoning systems, they appeared first in the logicist tradition because the power of the representation and reasoning systems are well-defined and fairly well understood.

Conclusion

The goal of this assignment is to sharpen your math and programming skills needed for this unit. If you meet the prerequisites, you should find these problems relatively achievely. Some of these problems will occur again as subproblems of later homeworks, so make sure you know how to do them.

Assessment

A learner is required to undertake background reading, which is supported by lectures to explain various notions and to show the application of various techniques using examples. The coursework requires the students to solve exercises each week. Feedback for and help with this work is provided in the examples classes.

Activity 2 - [Applications of Artificial Intelligence]

Introduction

The purpose of this activity is to provide an overview of this field. We will cover topics including: agents, search, planning, and uncertainty. The goals of this activity is to provide a fundamental knowledge of the field.

Activity Details

Defining the Problem as state Search To understand what exactly artificial intelligence is, we illustrate some common problems. Problems dealt with in artificial intelligence generally use a common term called 'state'. A state represents a status of the solution at a given step of the problem solving procedure. The solution of a problem, thus, is a collection of the problem states. The problem solving procedure applies an operator to a state to get the next state. Then it applies another operator to the resulting state to derive a new state. The process of applying an operator to a state and its subsequent transition to the next state, thus, is continued until the goal (desired) state is derived. Such a method of solving a problem is generally referred to as state space approach.

Problem Spaces and Search

Building a system to solve a problem requires the following steps

Define the problem precisely including detailed specifications and what constitutes an acceptable solution;

Analyse the problem thoroughly for some features may have a dominant affect on the chosen method of solution;

Isolate and represent the background knowledge needed in the solution of the problem;

Choose the best problem solving techniques in the solution.

For example, in order to solve the problem play a game, which is restricted to two person table or board games, we require the rules of the game and the targets for winning as well as a means of representing positions in the game. The opening position can be defined as the initial state and a winning position as a goal state, there can be more than one. legal moves allow for transfer from initial state to other states leading to the goal state. However the rules are far too copious in most games especially chess where they exceed the number of particles in the universe 10. Thus the rules cannot in general be supplied accurately and computer programs cannot easily handle them. The storage also presents another problem but searching can be achieved by hashing.

The number of rules that are used must be minimised and the set can be produced by expressing each rule in as general a form as possible. The representation of games in this way leads to a state space representation and it is natural for well organised games with some structure. This representation allows for the formal definition of a problem which necessitates

the movement from a set of initial positions to one of a set of target positions. It means that the solution involves using known techniques and a systematic search. This is quite a common method in AI.

Formal description of a problem

Define a state space that contains all possible configurations of the relevant objects, without enumerating all the states in it. A state space represents a problem in terms of states and operators that change states:

- Define some of these states as possible initial states;
- Specify one or more as acceptable solutions, these are goal states;
- Specify a set of rules as the possible actions allowed. This involves thinking about the generality of the rules, the assumptions made in the informal presentation and how much work can be anticipated by inclusion in the rules.

The control strategy is again not fully discussed but the AI program needs a structure to facilitate the search which is a characteristic of this type of program.

Production system

a set of rules each consisting of a left side the applicability of the rule and the right side the operations to be performed;

one or more knowledge bases containing the required information for each task;

a control strategy that specifies the order in which the rules will be compared to the database and ways of resolving conflict;

a rule applier

Choose an appropriate search technique:

How large is the search space?

How well-structured is the domain?

What knowledge about the domain can be used to guide the search?

Example: the water jug problem There are two jugs called **four** and **three** ; **four** holds a maximum of four gallons and three a maximum of three gallons. How can we get 2 gallons in the jug four. The state space is a set of ordered pairs giving the number of gallons in the pair of jugs at any time ie (**four, three**) where **four** = 0, 1, 2, 3, 4 and three = 0, 1, 2, 3. The start state is (0,0) and the goal state is (2,n) where n is a don't care but is limited to three holding from 0 to 3 gallons. The major production rules for solving this problem are shown below:

initial	condition	goal	comment
1 (four,three)	if four < 4	(4,three)	fill four from tap

2 (four,three)	if three< 3	(four,3)	fill three from tap
3 (four,three)	If four > 0	(0,three)	empty four into drain
4 (four,three)	if three > 0	(four,0)	empty three into drain
5 (four,three)	if four+three<4	(four+three,0)	empty three into four
6 (four,three)	if four+three<3	(0,four+three)	empty four into three
7 (0,three)	If three>0	(three,0)	empty three into four
8 (four,0)	if four>0	(0,four)	empty four into three
9 (0,2)		(2,0)	empty three into four
10 (2,0)		(0,2)	empty four into three

11 (four,three) if four<4 (4,three-diff) pour diff, 4-four, into four from three 12 (three,four) if three<3 (four-diff,3) pour diff, 3-three, into three from four and a solution is given below

Jug four,	**jug three**	**rule applied**
0	0	
0	3	2
3	0	7
3	3	2
4	2	11
0	2	3
2	0	10

Control strategies. A good control strategy should have the following requirement:

The first requirement is that it causes motion. In a game playing program the pieces move on the board and in the water jug problem water is used to fill jugs.

The second requirement is that it is systematic, this is a clear requirement for it would not be sensible to fill a jug and empty it repeatedly nor in a game would it be advisable to move a piece round and round the board in a cyclic way. We shall initially consider two systematic approaches to searching.

Heuristics Search: A heuristic is a method that might not always find the best solution but is guaranteed to find a good solution in reasonable time. By sacrificing completeness it increases efficiency. It is particularly useful in solving tough problems which could not be solved any other way and if a complete solution was to be required infinite time would be needed i.e. far longer than a lifetime.

Example: The travelling salesman problem: A salesperson has a list of cities to visit and she must visit each city only once. There are distinct routes between the cities. The problem is to find the shortest route between the cities so that the salesperson visits all the cities once. Suppose there are N cities then a solution that would work would be to take all N! possible combinations and to find the shortest distance that being the required route. This is not efficient as with N=10 there are 3 628 800 possible routes. This is an example of combinatorial explosion. There are better methods for solution, one is called branch and bound.

Conclusion

The design of search programs. Each search process can be considered to be a tree traversal exercise. The object of the search is to find a path from an initial state to a goal state using a tree. The number of nodes generated might be immense and in practice many of the nodes would not be needed. The secret of a good search routine is to generate only those nodes that are likely to be useful. Rather than having an explicit tree the rules are used to represent the tree implicitly and only to create nodes explicitly if they are actually to be of use. The following issues arise when searching:

the tree can be searched forwards from the initial node to the goal state or backwards from the goal state to the initial state.

how to select applicable rules, it is critical to have an efficient procedure for matching rules against state

how to represent each node of the search process this is the knowledge representation problem or the frame problem. In games an array suffices in other problems more complex data structures are needed.

The breadth first does take note of all nodes generated but depth first can be modified. Check duplicate nodes

Assessment

Generate all the complete paths and find the distance of the first complete path. If the next path is shorter save it and proceed in this way abandoning any path when its length so far exceeds the shortest path length. Although this is better than the previous method it is still exponential. Heuristic Search applied to the travelling salesman problem:

Applying this concept to the travelling salesperson problem.

1 select a city at random as a start point;

2 repeat

3 to select the next city have a list of all the cities to be visited and choose the nearest one to the current city , then go to it;

4 until all cities visited This produces a significant improvement and reduces the time from order N! to N.

Is the problem decomposable into a set of nearly independent smaller or easier sub-problems?

- Can the solution steps be ignored or at least undone if they prove unwise?
- Is the problem's universe predictable?
- Is a good solution to the problem obvious without comparison to all other possible solutions?
- Is the desired solution a state of the world or a path to a state?
- Is a large amount of knowledge absolutely required to solve this problem or is knowledge important only to constrain the search?
- Can a computer that is simply given the problem return the solution or will the solution of the problem require interaction between the computer and a person?

Activity 3 - [Knowledge Representation]

Introduction

The knowledge based is the store in which the knowledge in the particular domain is kept. The knowledge base stores information about the subject domain. However, this goes further than a passive collection of records in a database. Rather it contains symbolic representations of experts' knowledge, including definitions of domain terms, interconnections of component entities, and cause-effect relationships between these components. The knowledge in the knowledge based is expressed as a collection of fact and rule. Each fact expresses relationship between two or more object in the problem domain and can be expressed in term of predicates.

Activity Details

Example: Procedural Knowledge

Knowledge encoded in some procedures

small programs that know how to do specific things, how to proceed.

e.g a parser in a natural language understander has the knowledge that a noun phrase may contain articles, adjectives and nouns. It is represented by calls to routines that know how to process articles, adjectives and nouns.

Issue in Knowledge Representation

Below are listed issues that should be raised when using a knowledge representation technique:

Important Attributes: Are there any attributes that occur in many different types of problem? There are two instance and isa and each is important because each supports property inheritance.

Relationships: What about the relationship between the attributes of an object, such as, inverses, existence, techniques for reasoning about values and single valued attributes. We can consider an example of an inverse in:

band(Andrew, Masikini, Banjul City)

This can be treated as Andrew Masikini plays in the band Banjul City or Andrew Masikini's band is Banjul City.Another representation is band = Banjul City band-members = Andrew Masikini……..

For example 1:

If Tom feeds a dog then it could become: feeds(tom, dog)

If Tom gives the dog a bone like:

gives(tom, dog,bone) Are these the same?

In any sense does giving an object food constitute feeding?

If give(x, food) feed(x) then we are making progress.

But we need to add certain inferential rules.

In the famous program on relationships Louise is Bill's cousin How do we represent this? louise = daughter (brother or sister (father or mother(bill))) Suppose it is Chris then we do not know if it is Chris as a male or female and then son applies as well.

Example 2: Logic Knowledge Representation

Here we will highlight major principles involved in knowledge representation. In particular predicate logic will be met in other knowledge representation schemes and reasoning methods.

Symbols used The following standard logic symbols we use in this course are:

For all	∀
There exists	∃
Implies	→
Not	¬
Or	v
And	Λ

Predicate logic is used to represent knowledge. There are other ways but this form is popular.

Predicate logic

An example: Consider the following:

- Prince is a mega star.
- Mega stars are rich.
- Rich people have fast cars.
- Fast cars consume a lot of petrol.

and try to draw the conclusion: Prince's car consumes a lot of petrol. So we can translate Prince is a mega star into: mega_star(prince) and Mega stars are rich into: ∀m: mega_star(m) rich(m) Rich people have fast cars, the third axiom is more difficult:

Properties for Knowledge Representation Systems

The following properties should be possessed by a knowledge representation system:

Representational Adequacy: the ability to represent the required knowledge;

Inferential Adequacy: the ability to manipulate the knowledge represented to produce new knowledge corresponding to that inferred from the original;

Inferential Efficiency: the ability to direct the inferential mechanisms into the most productive directions by storing appropriate guides;

Acquisitional Efficiency: the ability to acquire new knowledge using automatic methods wherever possible rather than reliance on human intervention.

Approaches to Knowledge Representation:

Example 1: Simple relational knowledge

Simple way to store facts.

- Each fact about a set of objects is set out systematically in columns (Fig below)
- Little opportunity for inference.
- Knowledge basis for inference engines.

Musician	Style	Instrument	Age
John Macundi	Jazz	Trumpet	Deceased
Andrew Masikini	Runbaa	Saxophone	35
Lamin Njie	Rapa	Guitar	Deceased
Pateh Thomas	hippop	Guitar	47

Example 2: Inheritable knowledge

Relational knowledge is made up of objects consisting of attributes and corresponding associated values. Property Inheritance Hierarchy Boxed nodes represent objects and values of attributes of objects. Values can be objects with attributes and so on. Arrows point from object to its value. This structure is known as a slot and filler structure, semantic network or a collection of frames.

Conclusion

Learning It refers to acquiring knowledge. This is more than simply adding new facts to a knowledge base. New data may have to be classified prior to storage for easy retrieval, etc.. Interaction and inference with existing facts to avoid redundancy and replication in the knowledge and also so that facts can be updated.

Unit Assessment

Check your understanding!

Unit 2.Artificial Intelligence Programming

Unit Introduction

The course gives students the opportunity to implement many classic AI algorithms and use them as modules in large AI systems to perform tasks such as speech and image processing, simulated soccer, poker playing, and robot navigation.

Some of the important AI methods that can appear in various projects include the A* algorithm, means-ends analysis, decision-tree learning, genetic algorithms, neural networks, bayesian classification, case-based reasoning, boosting and bagging.

Through this work, students will gain an in-depth understanding of "AI in practice" as opposed to the combination of "AI in theory" and "AI on toy problems" that one experiences in the introductory and intermediate AI courses.

Unit Objectives

Upon completion of this unit you should be able to:

Learn hands-on experience designing and implementing relatively large AI projects.

Demonstrate valuable insights into why, when and how to use AI methods in realistic problems that they may encounter in their technical careers

Learn basic ideas and techniques underlying the design of intelligent computer systems

Apply how to build agents that exhibit reasoning and learning

Program applications for basic variety of artificial intelligence

Key Terms

Agent: anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators

Game Playing: Discusses the theory and algorithms behind computer programs that can play games, such as chess and sudoku.

Machine learning: Disccuses algorithms that can automatically adapt to changing environments by learning from past observations. Also includes the theory behind probabilistic reasoning systems.

Robotics: Discusses the basic principles behind robotic systems, especially automatic tracking and navigation.

Learning Activities

Activity 1 Agents: Perception, Decisions, and Actuation

Introduction

This unit will provide the learner with an introduction to AI via programming features that support basic AI applications. By satisfying the goals of this unit, the learner will have a familiarity with AI programming and be able to use it in future models to implement various AI applications.

Activity Details

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators. Agents are divided into two categories, namely:

- Human agent: examples
- eyes, ears, and other organs for sensors; hands, legs, mouth, and other body parts for actuators.
- Robotic agent: examples
- cameras and infrared range finders for sensors; various motors for actuators
- Rational agents: example

for each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, based on the evidence provided by the percept sequence and whatever built-in knowledge the agent has, while performance measure: is the objective criterion for success of an agent's behavior e.g., performance measure of a vacuum-cleaner agent could be amount of dirt cleaned up, amount of time taken, amount of electricity consumed, amount of noise generated, etc.

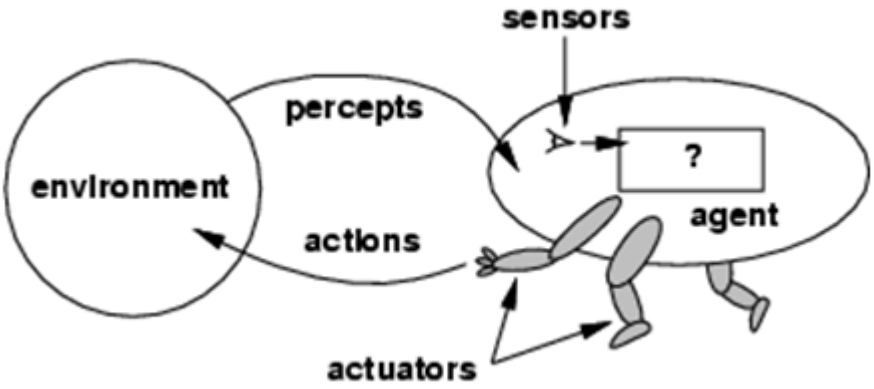
Activity Details

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators. Agents are divided into two categories, namely:

- Human agent: examples
  eyes, ears, and other organs for sensors; hands, legs, mouth, and other body parts for actuators.
- Robotic agent: examples
  cameras and infrared range finders for sensors; various motors for actuators
- Rational agents: example

for each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, based on the evidence provided by the percept sequence and whatever built-in knowledge the agent has, while performance measure: is the objective criterion for success of an agent's behavior e.g., performance measure of a vacuum-cleaner agent could be amount of dirt cleaned up, amount of time taken, amount of electricity consumed, amount of noise generated, etc.

Agents and their Environments



- The agent function maps from percept histories to actions:

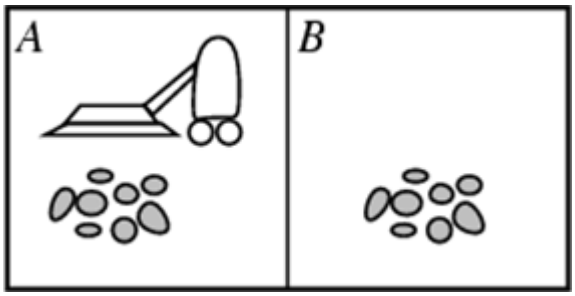
[f: P* → A]

- The agent program runs on the physical architecture to produce f

agent = architecture + program

An example of an agent at work: A logic operations of a robotic agent

A Vacuum-cleaner world



- Percepts: location and state of the environment, e.g., [A,Dirty], [B,Clean]
- Actions: Left, Right, Suck, NoOp

Agents can perform actions in order to modify future percepts so as to

obtain useful information (information gathering, exploration) An agent is autonomous if its behavior is determined by its own percepts & experience (with ability to learn and adapt) without depending solely on build-in knowledge.

Before we design an intelligent agent, we must specify its "task environment":

PEAS: = Performance measure

Environment

Actuators

Sensors

Example: Agent = taxi driver

Performance measure: Safe, fast, legal, comfortable trip, maximize profits

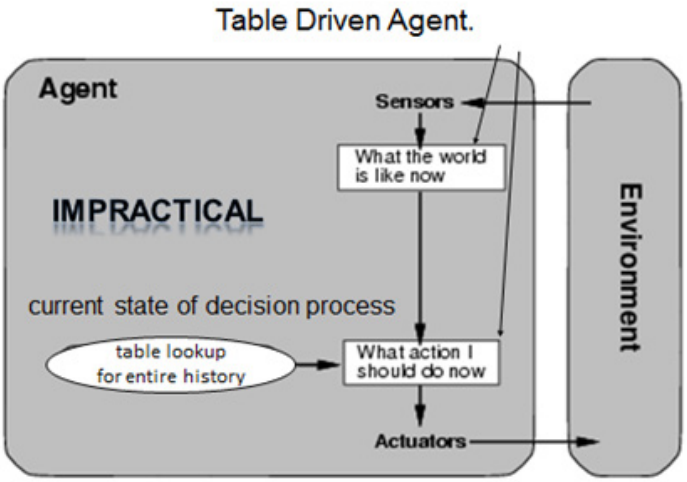
Environment: Roads, other traffic, pedestrians, customers

Actuators: Steering wheel, accelerator, brake, signal, horn

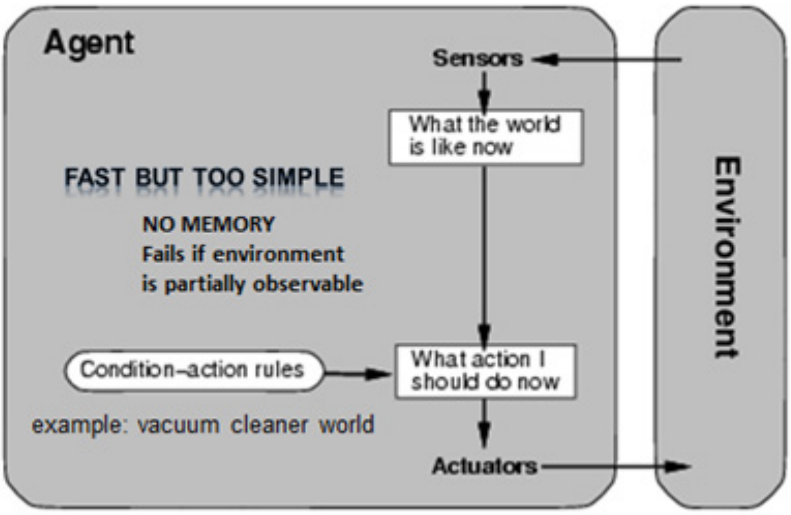
Sensors: Cameras, sonar, speedometer, GPS, odometer, engine sensors, keyboard.

There are five basic types in order of increasing generality:

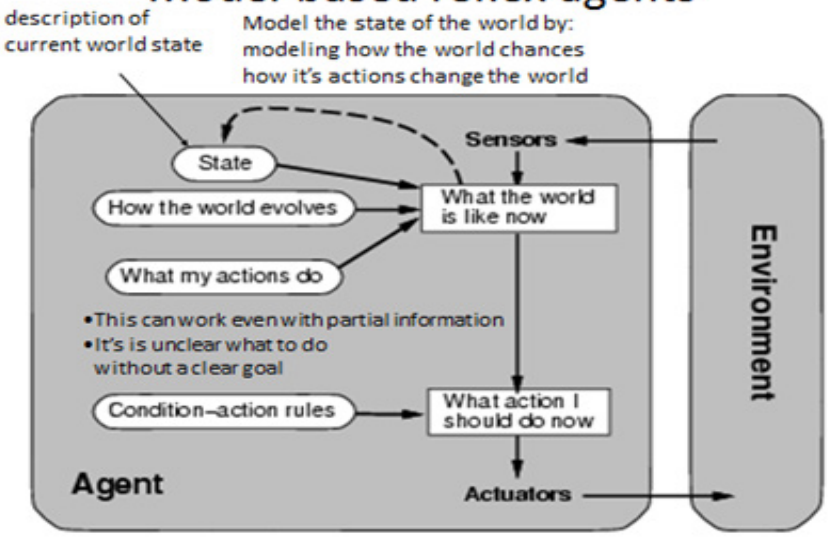
- Table Driven agent
- Simple reflex agents
- Model-based reflex agents
- Goal-based agents
- Utility-based agents



Simple reflex agents

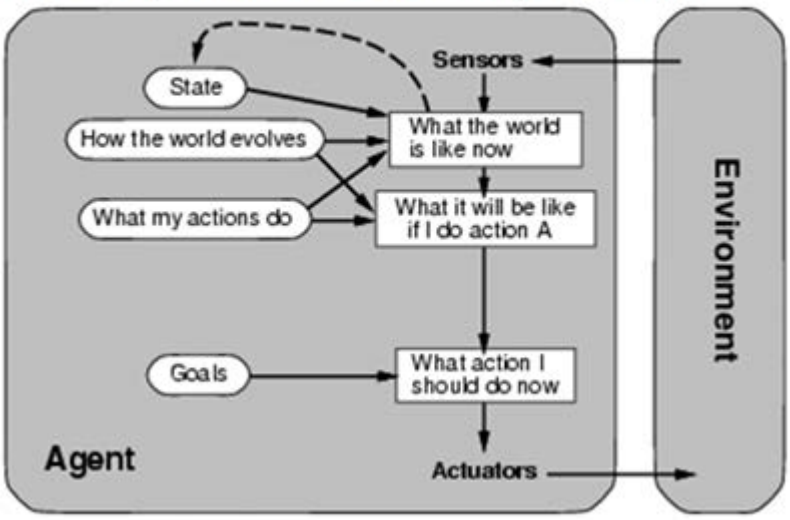


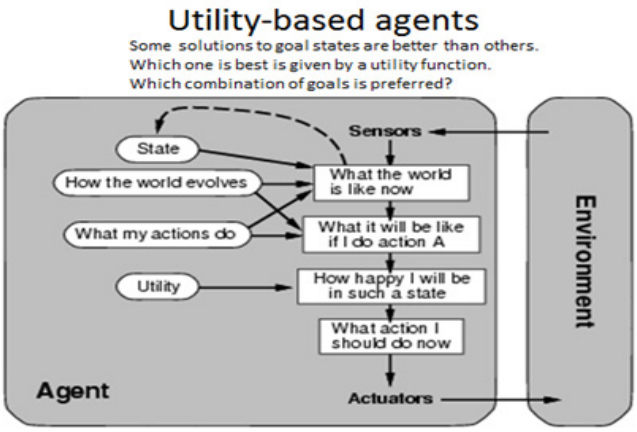
Model-based reflex agents

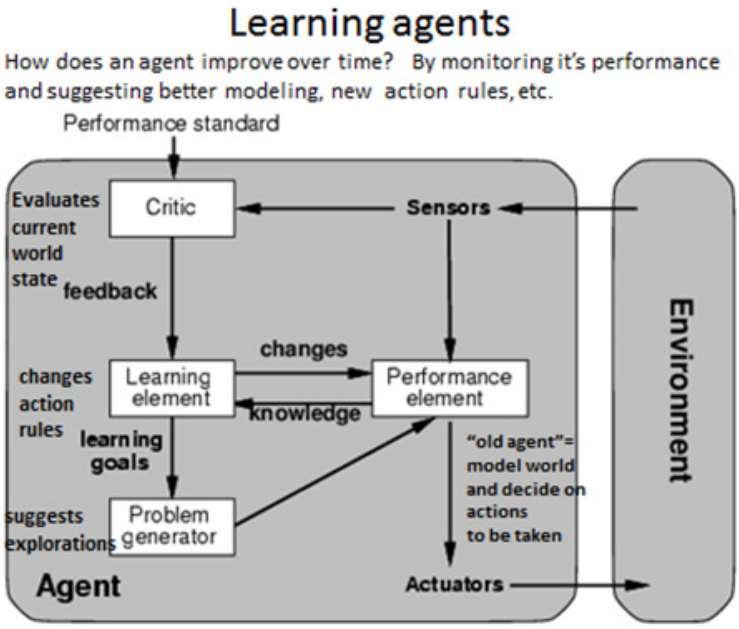


Goal-based agents

Goals provide reason to prefer one action over the other.
We need to predict the future: we need to plan & search







Rules and Logic Programming

We have now spent a fair bit of time learning about the language of first-order logic and the mechanisms of automatic inference. And, we've also found that (a) it is quite difficult to write first-order logic and (b) quite expensive to do inference. Both of these conclusions are well justified. Therefore, you may be wondering why we spent the time on logic.

We can motivate our study of logic in a variety of ways. For one, it is the intellectual foundation for all other ways of representing knowledge about the world. As we have already seen, the Web Consortium has adopted a logical language for its Semantic Web project. We also saw that airlines use a language not unlike FOL to describe fare restrictions. We will see later when we talk about natural language understanding that logic also plays a key role.

There is another practical application of logic that is reasonably widespread namely logic programming . In this section, we will look briefly at logic programming. Later, when we study natural language understanding, we will build on these ideas.

Logic in Practice

- Language of logic is extremely powerful.
- Say what's true, not how to use it.
  - ∀ x, y (∃ z Parent(x,z) ∧ Parent(z,y)) ↔ GrandParent(x,y)
  - Given parents, find grandparents
  - Given grandparents, find parents

One of the key characteristics of logic, as opposed to programming languages but like natural languages, is that in logic you write down what's true about the world, without saying how to use it. So, for example, one can characterize the relationship between parents and grandparents in this sentence without giving an algorithm for finding the grandparents from the grandchildren or a different algorithm for finding the grandchildren given the grandparents.

Logic in Practice

- Language of logic is extremely powerful.
- Say what's true, not how to use it.
  - ∀ x, y (∃ z Parent(x,z) ∧ Parent(z,y)) ↔ GrandParent(x,y)
  - Given parents, find grandparents
  - Given grandparents, find parents
- But, resolution theorem-provers are too inefficient!

There are, however, approaches to regaining some of the efficiency while keeping much of the power of the representation. These approaches involve both limiting the language as well as simplifying the inference algorithms to make them more predictable. Similar ideas underlie both logic programming and rule-based systems. We will bias our presentation towards logic programming.

Logic in Practice

- Language of logic is extremely powerful.
- Say what's true, not how to use it.
  - ∀ x, y (∃ z Parent(x,z) ∧ Parent(z,y)) ↔ GrandParent(x,y)
  - Given parents, find grandparents
  - Given grandparents, find parents
- But, resolution theorem-provers are too inefficient!
- To regain practicality:
  - Limit the language
  - Simplify the proof algorithm
- Rule-Based Systems
- Logic Programming

In logic programming we will also use the clausal representation that we derived for resolution refutation. However, we will limit the type of clauses that we will consider to the class called Horn clauses.

A clause is Horn if it has at most one positive literal. In the examples below, we show literals without variables, but the discussion applies both to propositional and first order logic.

There are three cases of Horn clauses:

- A rule is a clause with one or more negative literals and exactly one positive literal. You can see that this is the clause form of an implication of the form P1 ^ ... ^

Pn -> Q, that is, the conjuction of the P's implies Q.

A fact is a clause with exactly one positive literal and no negative literals. We generally will distinguish the case of a ground fact , that is, a literal with no variables, from the general case of a literal with variables, which is more like an unconditional rule than what one would think of as a "fact".

In general, there is another case, known as a consistency constraint when the clause has no positive literals. We will not deal with these further, except for the special case of a conjunctive goal clause which will take this form (the negation of a conjuction of literals is a Horn clause with no positive literal). However, goal clauses are not rules

Horn Clauses

- A clause is Horn if it has at most one positive literal
  - $\neg P_1 \vee \ldots \vee \neg P_n \vee Q$ (Rule)
  - Q (Fact)
  - $\neg P_1 \vee \ldots \vee \neg P_n$ (Consistency Constraint)
- We will not deal with Consistency Constraints

Conclusion

AI concepts and techniques are learned in two steps: theory, and implementation of theory in programs. AI techniques are difficult to program and can be obscured by the programming details.

To make these techniques explicit and to hide the programming details, AI languages--such as Lisp, Prolog, Scheme, and others--have been defined to have language features that directly support the implementation of AI techniques. The use of class libraries and source code libraries serve the same purpose for general-purpose languages, such as C++ and Java.

Assessment

1. The task environment of taxi-driving is episodic.Proof that it is true / false
2. A rational agent will always achieve its goal. Proof that it is true / false
3. Chess Agents:

Consider 2 intelligent agents playing chess with a clock.

One of them is called "deep blue", while the other is called Gary Kasparov.

a. Roughly specify the task environment for "Deep Blue". (This means specify each letter in the **"PEAS"**.

b. Determined each of the following properties to this environment:

   i. a fully observable or partially observable
   ii. deterministic or stochastic
   iii. episode or sequential
   iv. static, dynamic or semi-dynamic
   v. discrete or continuous
   vi. single agent or multi-agent

Explain your answers in detail with proofs..

Activity 2 - Logic

Introduction

In this unit, the learner will learn how to explain the syntax and semantics of logic statements in A.I., write statements, operate on statements, apply the rules of logic to transform a statement to equivalent statements, and evaluate statements in both the propositional and predicate calculus. Make inferences and prove statements in logic.

Activity Details

We will be using two types of logic: propositional logic and first-order logic. The difference between the two is in the use of variables. Propositional logic has no variables that take on values in a domain. For example, P is a sentence in the proposition calculus that is true or false. An example for the predicate calculus is P(X,Y), where P is a relation and X and Y are variables that take on values in a domain of discourse. The predicate calculus, also called the first-order calculus, extends the propositional calculus.

Introduction to Logic: Why Do We Need Logic?

- Problem-solving agents were very inflexible: hard code every possible state.
- Search is almost always exponential in the number of states.
- Problem solving agents cannot infer unobserved information.

Knowledge & Reasoning

To address these issues we will learn about:

- A knowledge base (KB): a list of facts that are known to the agent.
- Rules to infer new facts from old facts using rules of inference.
- Logic provides the natural language for this.

Example: Knowledge Base

Example: Knowledge Base

- set of sentences in a formal language.
- Declarative approach to building an agent:

Tell it what it needs to know.

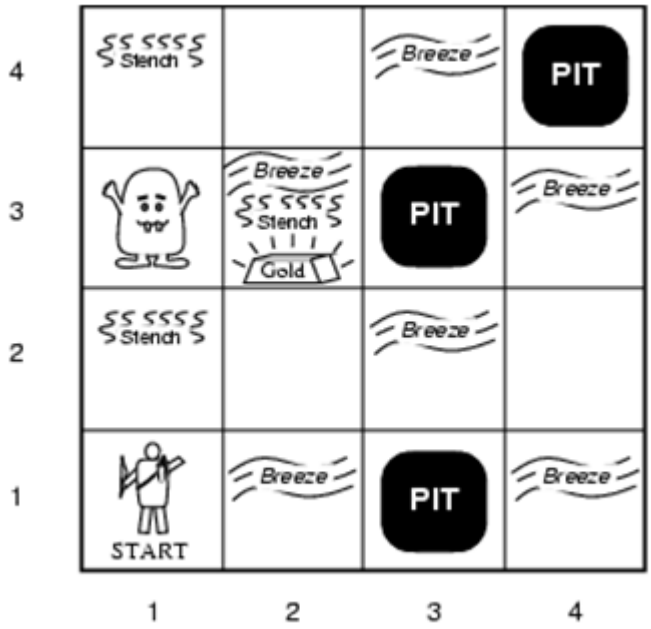
Ask it what to do → answers should follow from the KB

Example 1 Logic in searching and decision making

Wumpus World PEAS description

- gold: +1000, death: -1000
- -1 per step, -10 for using the arrow

Wumpus

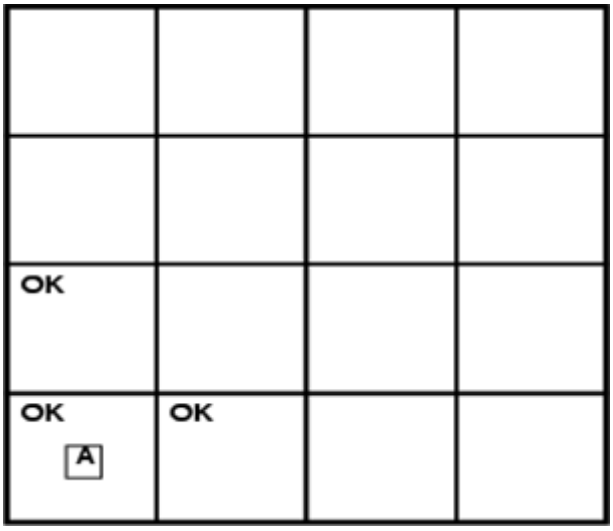


- Environment
- Squares adjacent to wumpus are smelly
- Squares adjacent to pit are breezy
- Glitter iff gold is in the same square
- Shooting kills wumpus if you are facing it
- Shooting uses up the only arrow
- Grabbing picks up gold if in same square
- Releasing drops the gold in same square
- Sensors: Stench, Breeze, Glitter, Bump, Scream
- Actuators: Left turn, Right turn, Forward, Grab, Release, Shoot

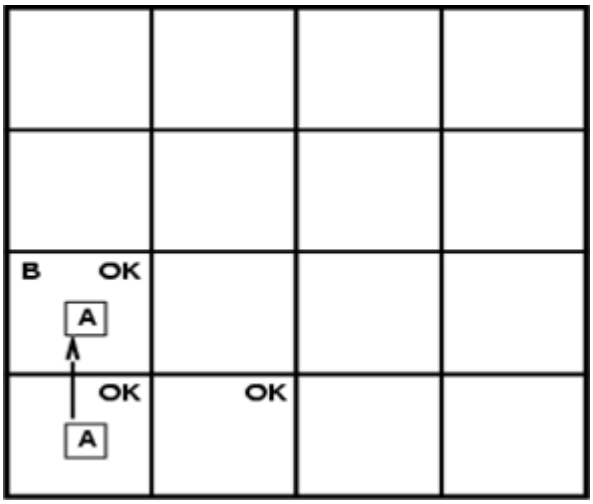
The rules of the Wumpus world characterization

- Fully Observable No – only local perception
- Deterministic Yes – outcomes exactly specified
- Episodic No – things we do have an impact.
- Static Yes – Wumpus and Pits do not move
- Discrete Yes
- Single-agent? Yes – Wumpus is essentially a natural feature

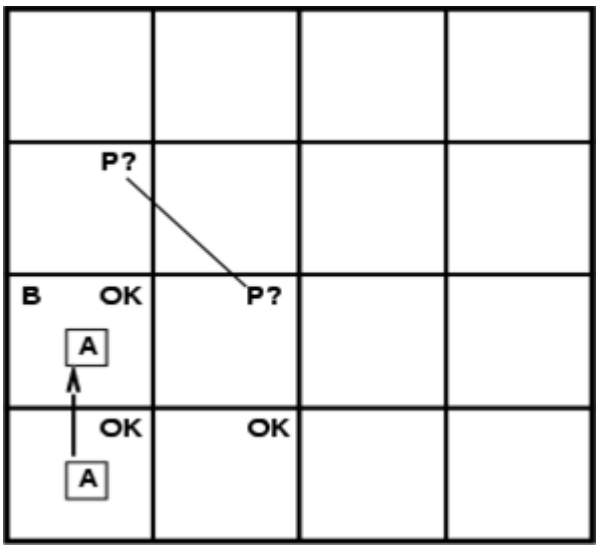
Example 2.1: Exploring a wumpus world



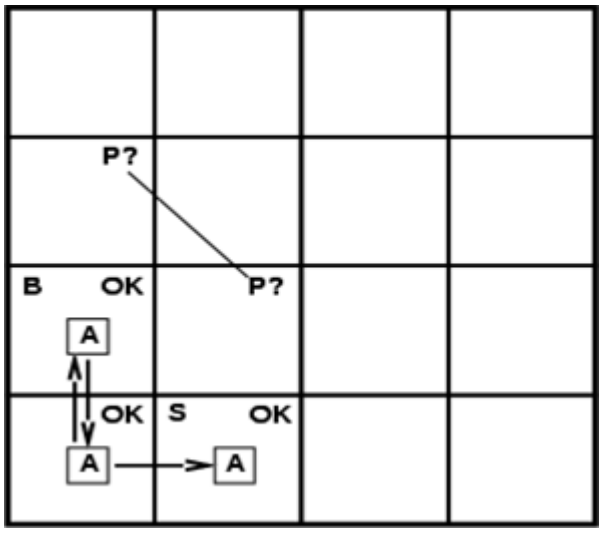
Reasoning State 1



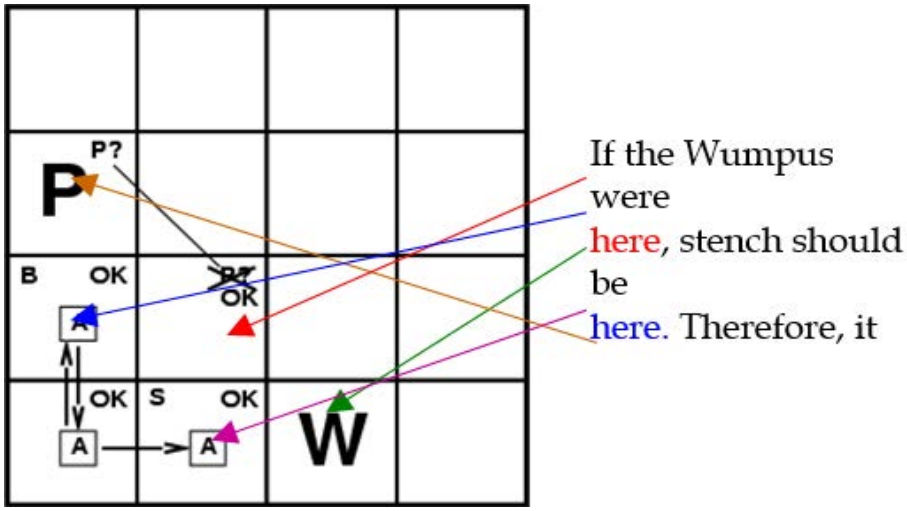
Reasoning State 2



Reasoning State 3



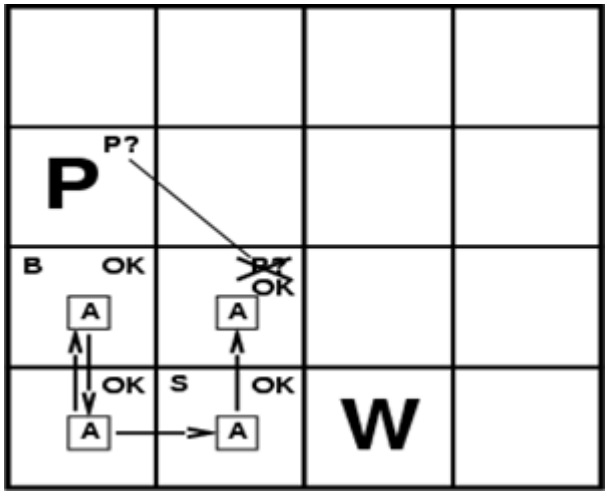
Reasoning State 4

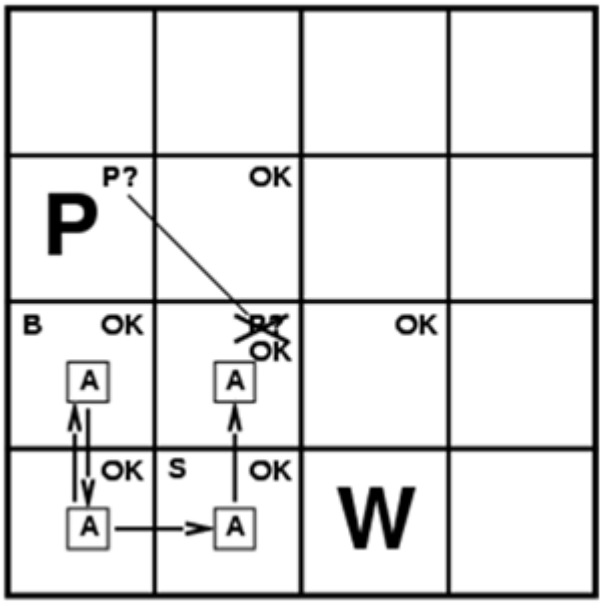


- Reasoning State 4

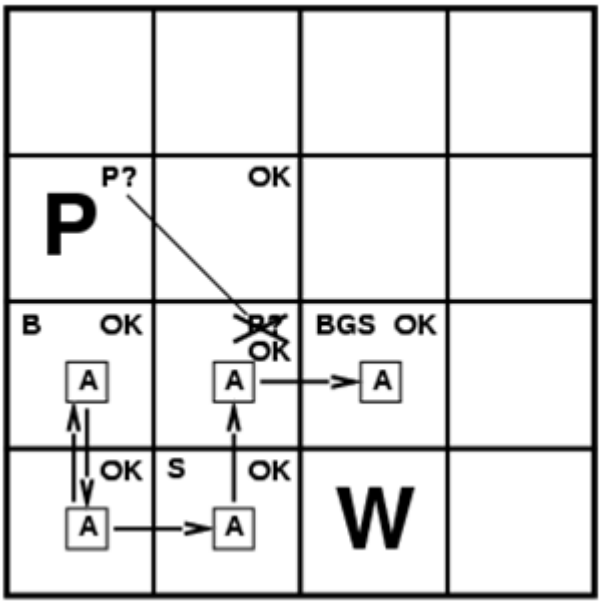
We need rather sophisticated reasonong here!

Example 2.2





Analysis 2 of reasoning state after state 4



Example 3: Logic

- We used logical reasoning to find the gold.
- Logics are formal languages for representing information such that conclusions can be drawn
- Syntax defines the sentences in the language
- Semantics define the "meaning" of sentences;

- i.e., define **truth** of a sentence in a world
- E.g., the language of arithmetic
- $x+2 \geq y$ is a sentence; $x2+y > \{\}$ is not a sentence → syntax

$x+2 \geq y$ is true in a world where $x = 7, y = 1$ → semantic

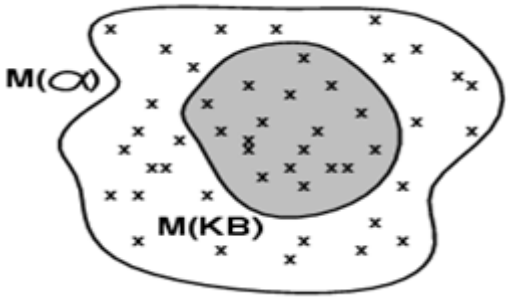
$x+2 \geq y$ is false in a world where $x = 0, y = 6$

Entailment

- Entailment means that one thing follows from another:KB = α
- Knowledge base KB entails sentence α if and only if α is true in all worlds where KB is true
- Example: the KB containing "the Giants won and the Reds won" entails "The Giants won".
- Example: x+y = 4 entails 4 = x+y

Models:

- Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated
- We say m is a model of a sentence α if α is true in m



M(α) is the set of all models of α

Then KB = α if M(KB) ⊆ M(α)

Example: KB = Giants won and Reds

won α = Giants won

Propositional logic: Syntax:

- Propositional logic is the simplest logic – illustrates basic ideas

The proposition symbols P1, P2 etc are sentences

Example:

- If S is a sentence, ¬S is a sentence
- **(negation)**
- If S1 and S2 are sentences, S1 ∧ S2 is a sentence **(conjunction)**
- If S1 and S2 are sentences, S1 ∨ S2 is a sentence
- **(disjunction)**
- If S1 and S2 are sentences, S1 ⇒ S2 is a sentence **(implication)**
- If S1 and S2 are sentences, S1 ⇔ S2 is a sentence **(biconditional)**

Propositional logic: Semantics

Each model/world specifies true or false for each proposition symbol

Example: P1,2 P2,2 P3,1

false true false

With these symbols, 8 possible models, can be enumerated automatically.

Rules for evaluating truth with respect to a model m:

¬S	is true if	S is false
S1 ∧ S2	is true if	S1 is true **and** S2 is true
S1 ∨ S2	is true if	S1is true **or** S2 is true
S1 ⇒ S2	is true if	S1 is false **or** S2 is true
i.e.,	is false if	S1 is true **and** S2 is false
S1 ⇔ S2	is true if	S1⇒S2 is true **and** S2⇒S1 is true

Simple recursive process evaluates an arbitrary sentence, e.g.,

¬P1,2 ∧ (P2,2 ∨ P3,1) = true ∧ (true ∨ false) = true ∧ true = true

Truth tables for connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

OR: P or Q is true or both are true.
XOR: P or Q is true but not both.

Implication is always true when the premises are False!

Functional examples of connectives:

Let Pi,j be true if there is a pit in [i, j].

Let Bi,j be true if there is a breeze in [i, j].

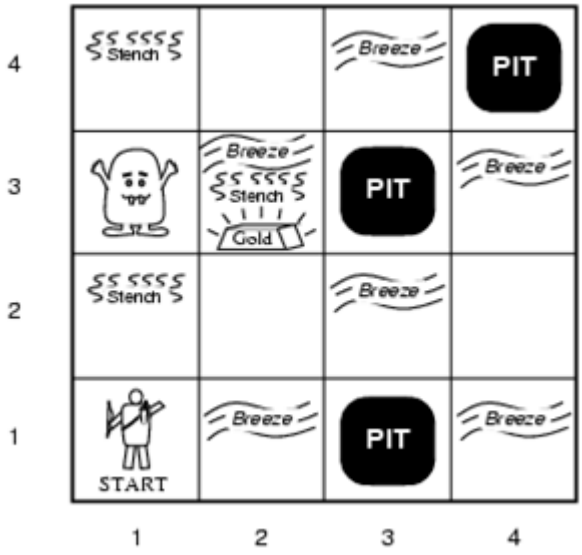
start: ← P1,1

← B1,1

B2,1

"Pits cause breezes in adjacent squares"

B1,1 ⇔ (P1,2 ∨ P2,1)B2,1 ⇔ (P1,1 ∨ P2,2 ∨ P3,1)

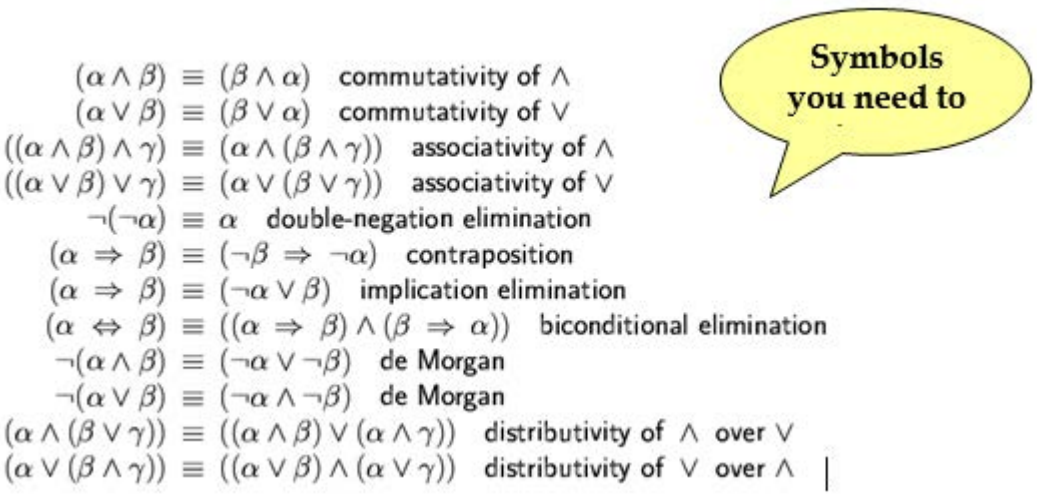


Inference by enumeration

- Enumeration of all models is sound and complete.
- For n symbols, time complexity is O(2n)...
- We need a smarter way to do inference!
- In particular, we are going to infer new logical sentences from the data-base and see if they match a query.

Logical equivalence

- To manipulate logical sentences we need some rewrite rules.
- Two sentences are logically equivalent if they are true in same models: $\alpha \equiv ß$ if $\alpha = \beta$ and $\beta = \alpha$



$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge$$
$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee$$
$$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge$$
$$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee$$
$$\neg(\neg\alpha) \equiv \alpha \quad \text{double-negation elimination}$$
$$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition}$$
$$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta) \quad \text{implication elimination}$$
$$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination}$$
$$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta) \quad \text{de Morgan}$$
$$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta) \quad \text{de Morgan}$$
$$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee$$
$$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge$$

A logic is **valid** if it is true in **all** models,

e.g., True, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference via the **Deduction Theorem**:

$KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid

A logic is **satisfiable** if it is true in some model

e.g., A∨ B, C

A sentence is **unsatisfiable** if it is false in all models

e.g., A∧¬A

Satisfiability is connected to inference via the following:

$KB \models \alpha$ if and only if $(KB \wedge \neg\alpha)$ is unsatisfiable

(there is no model for which KB=true and is false)

Conclusion

AI concepts and techniques are learned in two steps: theory, and implementation of theory in programs. AI techniques are difficult to program and can be obscured by the programming details. To make these techniques explicit and to hide the programming details, AI logics and truth tables have been defined to have language features that directly support the implementation of AI techniques. The use of class programing logics serves the same purpose for general-purpose languages, such as C++ and Java. The propositional logic for program examples of AI concepts and techniques can be derived to generate good A.I. and achievable algorithms. Java, being widely used, has the added advantage of making these techniques more widely available

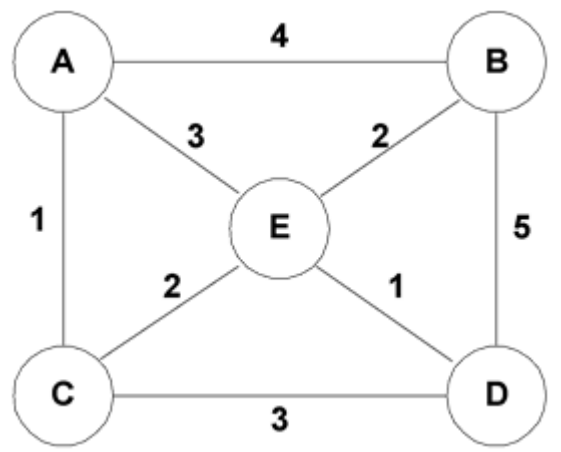
Assessment

Try this yourself (The learner)

If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.Derive the KB in normal form. Prove: Horned, Prove: Magical.

(Adapted from Barwise and Etchemendy (1993).)

1. Consider the following road map of cities:



Each city has a road connecting it to another city. The numbers on the road indicate how long it takes to travel between cities. Suppose you live in city A. You want to plan a trip that visits each city only once that starts and ends at home. For example, the trip ABDCEA (meaning you go from city A to B, then to D, etc) is one such path that would take you 17 hours to follow. Your goal is to choose a path that minimizes the time spent traveling. (Note: this is called the Traveling Salesman Problem

Formulate the state space for the problem

Draw a diagram of the complete state space, describing what the actions are

Describe a breath first search algorithm, and find the shortest trip from A to A that visits all cities.

Compare the time & space requirements for Depth-First search & Breadth-First Search on this problem.

Would uniform-cost search work well with this problem?

For example:

1

2

3

4

5

6

7

8

9

10

11

12

cost = |1-2|+|2-3|+|3-4|+|5-6|+|6-7|+|7-8|+|9-10|+|10-11|+|11-12|

+ |1-5|+|2-6|+|3-7|+|4-8|+|5-9|+|6-10|+|7-11|+|8-12|

Each number has either 4 neighbors or 3 neighbors (on the edge) or 2 neighbors (in a corner).

The total cost is the sum of the absolute differences between all neighboring pairs of numbers.

Activity 3 - [Add title here]

Introduction

In this section, we will look at some of the basic approaches for building programs that play two-person games such as tic-tac-toe, checkers and chess. Much of the work in this area has been motivated by playing chess, which has always been known as a "thinking person's game". Game playing programs are another application of search.

Activity Details

The states are the board positions (and the player whose turn it is to move). The operators are the legal moves. The goal states are the winning positions. A scoring function assigns values to states and also serves as a kind of heuristic function. The game tree (defined by the states and operators) is like the search tree in a typical search and it encodes all possible games.

The interpretation of Game Tree Search

Initial state:	Initial board position and player
Operators:	One for each legal move
Goal states:	Winning board positions
Scoring function:	Assigns numeric value to states
Game tree:	Encodes all possible games

There are a few key differences, however. For one thing, we are not looking for a path through the game tree, since that is going to depend on what moves the opponent makes. All we can do is choose the best move to make next.

Move generation state

Let's look at the game tree in more detail. Some board position represents the initial state and it's now our turn. We generate the children of this position by making all of the legal moves available to us. Then, we consider the moves that our opponent can make to generate the descendants of each of these positions, etc.

Note that these trees are enormous and cannot be explicitly represented in their entirety for any complex game.

Game tree (2-player, deterministic, turns)

Here's a little piece of the game tree for Tic-Tac-Toe, starting from an empty board. Note that even for this trivial game, the search tree is quite big.

Tic-Tac-Toe (game)

How do we search this tree to find the optimal move? We should use the Mini-Max operation technique

MiniMax operation technique

The key idea is that the more look ahead we can do, that is, the deeper in the tree we can look, the better our evaluation of a position will be, even with a simple evaluation function. In some sense, if we could look all the way to the end of the game, all we would need is an evaluation function that was 1 when we won and -1 when the opponent won.

The truly remarkable thing is how well this idea works. If you plot how deep computer programs can search chess game trees versus their ranking, we see a graph that looks something like this. The earliest serious chess program (MacHack6), which had a ranking of 1200, searched on average to a depth of 4. Belle, which was one of the first hardware-assisted chess programs doubled the depth and gained about 800 points in ranking. Deep Blue, which searched to an average depth of about 13 beat the world champion with a ranking of about 2900.

Mini-Max Algorithm

Here is pseudo-code that implements Min-Max. As you can see, it is a simple recursive alternation of maximization and minimization at each layer. We assume that we count the depth value down from the max depth so that when we reach a depth of 0, we apply our static evaluation to the board.

Source: Artificial Intelligence. Copyright © 2004 by Massachusetts Institute of Technology.

The interpretation of the algorithm

- Visit the nodes in a depth-first manner
- Maintain bounds on nodes.
- A bound may change if one of its children obtains a unique value.
- A bound becomes a unique value when all its children have been checked or pruned.
- When a bound changes into a tighter bound or a unique value, it may become inconsistent with its parent.
- When an inconsistency occurs, prune the sub-tree by cutting the edge between the inconsistent bounds/values.

→ This is like propagating changes bottom-up in the tree.

Properties of MinMax

- Complete? Yes (if tree is finite)
- Optimal? Yes (against an rational opponent)
- Time complexity? O(bm)
- Space complexity? O(bm) (depth-first exploration)
- For chess, b ≈ 35, m ≈100 for "reasonable" games

→ exact solution completely infeasible Example:

Pruning

1. Do we need to expand all nodes?
2. No: We can do better by pruning branches that will not lead to success.

Properties of α-β

Pruning does not affect final result (it is exact).

Good move ordering improves effectiveness of pruning

(see last branch in example above)

With "perfect ordering," time complexity = O(bm/2)

→ doubles depth of search

The Practical Implementation

How do we make this practical?

Standard approach:

- cutoff test: (where do we stop descending the tree)
- depth limit
- better: iterative deepening
- cutoff only when no big changes are expected to occur next (quiescence search).
- evaluation function
- When the search is cut off, we evaluate the current state
- by estimating its utility. This estimate if captured by the evaluation function.

The Evaluation function of the Implementation

For chess, typically linear weighted sum of features

Eval(s) = w1 f1(s) + w2 f2(s) + … + wn fn(s)

e.g., w1 = 9 with

f1(s) = (number of white queens) – (number of black queens), etc.

Deterministic games in practice

- Checkers: Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994.
- Chess: Deep Blue defeated human world champion Garry Kasparov in a six-game match in 1997.
- Othello: human champions refuse to compete against computers: they are too good.
- Go: human champions refuse to compete against computers: they are too bad

Conclusion

The field of AI is a complex area of research. AI for games takes on different forms depending on the needs of the game designed, ranging from simple sets of rules for computer-controlled entities to more advanced adaptive systems. Applying AI concepts to games is a necessary way to increase the believability of the virtual characters created in electronic entertainment, but it is not an impossible challenge. For the AI to make meaningful decisions, it needs some way of perceiving its environment. In simpler systems, this perception can be a simple check on the position of the player entity. As systems become more demanding, entities need to identify key features of the game world, such as viable paths to walk through, cover-providing terrain, and areas of conflict. The challenge for designers and developers is to come up with a way to identify key features important to the intelligence system. For example, in a tactical game, past history can decide the best tactics to use against a player team, such as defensive, offensive, berserk, or some balanced means of play. In a strategy game, the optimal composition of units in an army can be discovered on a per-player basis. In games where the AI is controlling supportive characters for the player, the adaptive AI can better complement the player's natural style by learning the way the player acts.

Assessment

Consider a Tic-Tac-Toe game. Let Xn be the number of rows, columns or diagonals containing exactly n X's and no O's. Similarly, let On be the number of rows, columns or diagonals containing exactly n O's and no X's. We propose a utility function which assigns +1 to any position with X3 = 1 (i.e. winning position) and assigns -1 to any position with O3 = 1 (i.e. loosing position). The linear evaluation function we suggest is 3X2 + X1 - (3O2 + O1):

How many states (i.e. board positions) are there in a Tic-Tac-Toe game (including symmetric board positions)

What is the depth of the complete game tree? Does the complete game tree contain all the board positions you counted in (a)?

Show the game tree down to depth 2, namely starting from an empty board to all position in which there is one X and one O on the board.

Mark on your tree the evaluation of the positions at level 2 (i.e. the value of the utility function at each position). Thereafter, mark on your tree the min-max values.

Apply alpha-beta search on your tree, and mark the pruned subtrees when traversing from left to right, from right to left. What is the optimal order?

What property should the leaf values of a tree have so that pruning will be maximized (resp. minimized) when the tree below is traversed from left to right.

Consider the following game tree in which the static scores (in parentheses at the tip nodes) are all from the first player's point of view. Assume that the first player is the maximizing player.

a) What move should the first player choose?

b) What nodes would not need be examined using the alpha-beta algorithm – assuming that the nodes are examined left-to-right order?

Try it yourself (α-β pruning example)

Unit Summary

Through this work, students will gain an in-depth understanding of "AI in practice" as opposed to the combination of "AI in theory" and "AI on toy problems" that one experiences in the introductory and intermediate AI courses.Students will be free to program in the language of their choice, although Python, Java and C++ will be recommended.

Unit Assessment

Check your understanding!

Instructions

The course will consist of 2-4 projects, depending up the year and the extent of the individual projects. Each project will be supported by a series of lectures on relevant theoretical and practical issues surrounding the problem domain, while some class meetings will be reserved for interactive discussions of the problem and student progress.

What can you use a search client for? Here are a few ideas based on my own work projects:

Roughly determine if two words or phrases are associated with each other by concatenating the words or phrases and counting the number of search results for the combined search query.

Determine if a product name or ID code is spelled correctly or if a company

carries a product by setting up a custom Nutch instance that only spiders the 204 10.5 Indexing and Search with Nutch Clients company's web site(s). Always follow the terms and conditions of a web site when setting up a spider.

Improve search results by adding a list of project-specific synonyms to the

search client. Expand search terms using the synonym list.

If you need access to public information, spider the information infrequently

and then perform local search queries and use the local page caches.

For very little effort you can set up Nutch server instances that spider specific in-formation servers. You can often add significant value to application programs by adding search functionality and by using Nutch you can locally control the information.

Grading Scheme

50% standard lectures, and 50% interactive project discussions between students and teacher.

Grading:

- Quizzes (10%)
- Two projects (40%)
- A midterm (20%)
- Homework (30%).

100% - in course Students will be allowed to work alone or in groups of 2 (but no larger)
Answers

Students will receive their courses and feedback in the AVU course management system or a portal platform.

Unit Readings and Other Resources

The readings in this unit are to be found at course level readings and other resources.

Mark Watson's Practical Artificial Intelligence Programming with Java:Artificial Intelligence, A Modern Approach. Second Edition, by Russel and Norvig (Prentice Hall).

Artificial Intelligence for Games, Second Edition, - Ian Millington and John Funge, Edition: Second ISBN:0123747317

The textbook explains the subject material in detail. It is strongly recommended that you read the book. It is strongly recommended that you read the book and attend all lectures and all meetings of your discussion section. You will be responsible for all material covered in the lectures and discussion sections, and for all assigned reading in the book.

Unit 3. Machine Learning and Games

Unit Introduction

Machine learning is one of the areas of Artificial Intelligence. It consists of programming the computer within its given capacity - Similarly, instead of manual programming every single given problems at all times, problems can be solved by computing the raw data already available. The computer needs to improve its performance from tasks already observed, it must be self-programmed, it must develop new knowledge from experiences provided. The aim is to apply the rules learned on new examples. It is used when the program is too complex to be handled manually, or when lack of information on the task to be solved, or when data change over time or when the system behaves differently depending on users ...

It is utilusé in many areas including:

- search engines
- recognition of handwritten characters
- visual recognition
- speech recognition
- fraud detection
- aid diagnosis
- the missing data detection
- robotics etc.

Language, as well learning, are two of the most difficult areas in AI because they require several other faculties. Intelligent agents must learn from their experience: how can an agent be called intelligent if it is unable to learn? In knowledge engineering, machine learning offers a solution to the problem of knowledge acquisition: Initially, a minimum amount of knowledge can be modeled, the rest can be learned by the system from examples in the field.

There are several important issues in machine learning:

- **The problem of generalization:** how a machine can she identify invariants in a set of data so that these invariants can be put to use when solving future problems?
- **The problem of biased induction:** how a programmer uses his own intuitions and heuristics in the design of representations, models or algorithms that support the learning process? The problem is that, although these biased inductions make learning possible, at the same time they limit what can be learned by the system.
- **The problem of the empiricist dilemma:** if there is no biased induction in the design of the learning system, 1) how does the system learn anything useful? 2) how does the system know that he has learned something?

There are 3 types of learning:

- supervised learning
- unsupervised learning
- reinforcement learning

The data are called training examples.

For the procedure, we divide the data set into two parts:

- some are used for testing: training data
- some are used to measure performance: evaluation data.

After this process it is possible to see if the machine can apply the method to the new data.

Unit Objectives

Upon completion of this unit you should be able to:

- Choose the most appropriate modes of representation in a given context
- Solve problems using AI-solving techniques
- Design and implement solutions using a declarative approach

Assessment

[Learning]: Self - Computer programming

[Decision Tree]: Computer tree whose leaves correspond to decisions

[Classifier]: Separate data according to a given criterion

[Example training]: A set of data to the computer

Learning Activities

Activity 1 - Supervised Learning

Introduction

In supervised learning, the information provided is in the form of pairs (xi, yi), where xi yi is the input and the target corresponding to xi yi is the expected result for the input xi. It is referred to as the data is labeled. The solution of the task is known for a number of observations. They are generally labeled by a supervisor (human expert), hence the name supervised learning. When the target takes discrete yi values (values), we speak of classification and when yi takes real values, we speak of regression.

There are a number of algorithms for this field:

- decision trees
- neural networks
- the support vector machines

We will see in detail how the function of decision tree algorithm.

Activity Details

Supervised Learning: Decision Tree

A tree consists of nodes and arcs. The nodes are connected by arcs. Node, uppermost is called the root. The terminal nodes are called leaf (see fig 1).

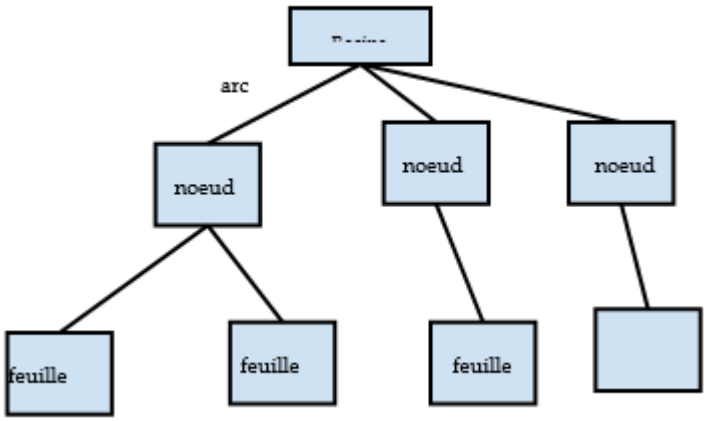


fig 1

For the method of decision trees, it is to build a decision tree from a set of labeled examples. Using this tree can label them new instances. For starters, we have a multi-set of examples E $\subseteq A1 \times A2 \times \ldots \times An \times C$ **where Al are attributes and C is the target attribute.**

Each attribute A has a field of possible values: dom (A). The areas here are finite sets. Output there will be a function that shows how to classify all elements of $A_1 \times A_2 \times \ldots \times A_n$ **There are several possible functions, it is to find one that gives best answer for all elements of** $A_1 \times A_2 \times \ldots \times A_n$

Consider an example when it comes to knowing whether a message is spam or not. So C = spam and dom (C) = {yes, no}. The following table gives a set of 7 examples.

Author	Key Words	HTML Link	Shift	Spam
known	yes	yes	no	no
unknown	no	no	yes	yes
unknown	yes	no	no	no
unknown	yes	yes	yes	yes
known	no	no	no	no
unknown	no	no	no	no
known	yes	no	no	no

The attributes are: Author, Keywords, Link HTML, Shift and the target attribute is spam.

dom (Author) = {known} Unknown

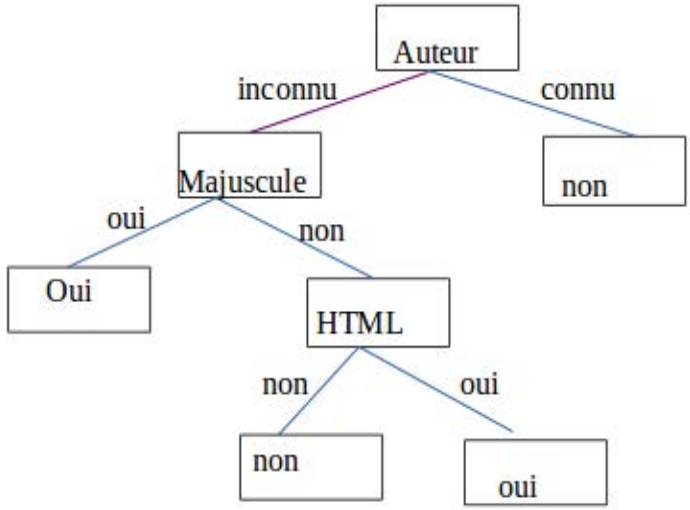
dom (Keywords) = {yes, no}

dom (HTML link) = {yes, no}

dom (Shift) = {yes, no}

dom (Spam) = {yes, no}

You can have as decision tree as the tree above - below



NB:

Auteur	means	Author
Inconnu	means	Unknown
Connu	means	Known
Majuscule	means	Shift
Oui	means	Yes

The arcs are labeled by the values of the domain node from which they flow.

To find the best tree is - to - say that has good coverage, how to choose the root? These are heuristics that allow us to answer such questions. Among the heuristic, one that is best known is that of information gain. This is to see among all the attributes, the one with the greatest information gain. This notion of gain leads to the concept of entropy. Entropy measures how much the different classes are mixed.

With a set of n divided into different classes, entropy is defined as:

$$H(D)=-(p_1\log_2 p_1+p_2\log_2 p_2+...+p_n\log_2 p_n)$$

where pi is the proportion of elements in D that are labeled by the ith class. In the example above, is

$$H(D)=-(\frac{2}{7}\log_2\frac{2}{7}+\frac{5}{7}\log_2\frac{5}{7})$$

We have (HD) = 0.86, as there are 2/7 and 5/7 yes no

The information gain of an attribute is defined as follows:

$$\sum_{x \in dom(A)} \frac{|D[A=x]|}{|D|} \times H(D[A=x])$$

Or $D[A=x]$ D denotes the elements having the x value for attribute A

calculate the entropy for example the values of the attribute Author

$$H(D[Auteur=connu]) = -(\frac{3}{3}\log_2\frac{3}{3} + \frac{0}{3}\log_2\frac{0}{3}) = 0$$

because for the known value of Author, there are 3 non (Spam) and 0 yes.

$$H(D[Auteur=inconnu]) = -(\frac{2}{4}\log_2\frac{2}{4} + \frac{2}{4}\log_2\frac{2}{4}) = 1$$

because for the unknown value Author ago 2 no (Spam) and 2 yes (spam).

Now calculate the gain information to author.

$$G(Auteur, D) = 0.68 - (\frac{3}{7} \times 0 + \frac{4}{7} \times 1)$$

As for the attribute Author, we experienced 3 and 4 unknown

We will then have: $G(Auteur, D) = 0.68 - \frac{4}{7} = 0.109$

The same is done for all attributes, the attribute having the greatest gain will be taken as root. Once the root is chosen, the procedure is the same for the other attributes.

Conclusion

The arcs are labeled by the values of the domain node from which they flow.

How to find the best tree is - to - say that has good coverage, how to choose the root? These are heuristics that allow us to answer such questions. Among the heuristic, one that is best known is that of information gain. This is to see among all the attributes, the one with the greatest information gain. This notion of gain leads to the concept of entropy. Entropy measures how much the different classes are mixed. With a set of n divided into different classes, entropy is defined as:

Assessment

Check your understanding

Activity 2 - Unsupervised Learning

Introduction

The objective of the unsupervised learning is to group unlabeled data according to their similarity. In this case there are only examples, not labels (raw data), the number of classes and their nature were not predetermined and must find common ground between these data, it is called unsupervised learning or clustering. No expert is pressed.

We'll talk in detail one of the algorithms: the K-means algorithm.

Activity Details

The K-means algorithm is an algorithm used in unsupervised learning. For this we proceed as follows:

a) randomly select k

b) create k random groups and determine their centers

c) re-allocate each object to the group whose center is the closest

d) recalculate the center of each group

e) repeating steps c) and d) until convergence (the centers no longer change)

For example, we have a set A = {1,2,3,6,7,8,13,15,17}.

Apply the K-means algorithm to this set to classify its elements.

Let k = 3……. We take three groups g1, g2 and g3, respective centers 1,2 and 3.

We calculate the distance between each element and the center, taking as distance d (x, y)

= | xy |

Hint - We to a table to facilitate the task.

pi	1	2	3	6	7	8	13	15	17
d(c1,pi)	0	1	2	5	6	7	12	14	16
d(c2,pi)	1	0	1	4	5	6	11	13	15
d(c3,pi)	2	1	0	3	4	5	10	12	14
dist min	0								
		0							
			0	3	4	5	10	12	14

So where g1 = {1}, c1 = 1,

g2 = 2 = {2v2,

g3 = {} 3,6,7,8,13,15,17, c3 = 9.86 (the center of gravity of the points)

It recalculates the distances between points and new centers.

pi	1	2	3	6	7	8	13	15	17
d(c1,pi)	0	1	2	5	6	7	12	14	16
d(c2,pi)	1	0	1	4	5	6	11	13	15
d(c3,pi)	8.86	7.86	6.86	3.86	2.86	1.86	3.14	5.14	7.14
dist min	0								
		0	1						
				3.86	2.86	1.86	3.14	5.14	7.14

It is noted that g {1} = 1, c1 = {1}

g2 = {2.3}, c2 = 2.5

g3 = {6,7,8,13,15,17}, c3 = 11

It recalculates the distances between points and the new centers.

pi	1	2	3	6	7	8	13	15	17
d(c1,pi)	0	1	2	5	6	7	12	14	16
d(c2,pi	1.5	0.5	1.5	3.5	4.5	5.5	10.5	12.5	14.5
d(c3,pi)	10	9	8	5	4	3	2	4	6
dist min	0								
		0.5	1.5	3.5					
					4	3	2	4	6

g1 = {1}, c1 = {1}

g2 = {2,3,6} c2 = 3.67

g3 = {7,8,13,15,17}, c3 = 12

Recalculation of distances

pi	1	2	3	6	7	8	13	15	17
d(c1,pi)	0	1	2	5	6	7	12	14	16
d(c2,pi)	2.67	1.67	0.67	2.33	3.33	4.33	9.33	11.33	13
d(c3,pi)	11	10	9	6	5	4	1	3	5
dist min	0	1							
			0.67	2.33	3.33				
						4	1	3	5

g1={1,2} c1= 1.5

g2={3,6,7}, c2=5.33

g3={8,13,15,17} , c3=13.25

pi	1	2	3	6	7	8	13	15	17
d(c1,pi)	0.5	0.5	1.5	4.5	5.5	6.5	11.5	13.5	15.5
d(c2,pi)	4.33	3.33	2.33	0.67	1.67	2.67	7.67	9.67	11.67
d(c3,pi)	12.25	11.25	10.25	7.25	6.25	5.25	0.25	1.75	3.75
	0.5	0.5	1.5						
				0.67	1.67	2.67			
dist min							0.25	1.75	3.75

g1={1,2,3}, c1=2

g2={6,7,8}, c2 =7

g3={13,15,17},c3 = 15

Recalculation of centres

pi	1	2	3	6	7	8	13	15	17
d(c1,pi)	1	0	1	4	5	6	11	13	15
d(c2,pi)	6	5	4	1	0	1	6	8	10
d(c3,pi)	14	13	12	9	8	7	2	0	2
	1	0							
				1	0	1			
dist min							2	0	2

g1 = {1,2,3}, c 1 = 2

g 2 = {6,7,8}, c2 = 7

g3 = {13,15,17}, c3 = 15

It is now evident that the centers do not change.

So it is now possible to gather all into 3 groups g1 = {1,2,3}, {6,7,8} = g2 and

g3 = {13,15,17} as with respective centers 2, 3 and 17.

Conclusion

From a set of data without labels, it is actually a collection of the same data of similar points. Unsupervised learning is used in many fields including seismology.

Assessment

In the map, considering the points p1 (1,3) P2 (3.3) P3 (4.3), p4 (5.3), p5 (1,2), p6 (4.2) p7 (1.1) and p8 (2.1). Using the k-mean algorithm and for k = 2, do a re-grouping of these points as an exercise.

Activity 3 - Game

Introduction

Games are one of the favorite areas of Artificial Intelligence. Beyond the fun aspect of the game, it is an interesting technique, thanks to heuristic solution to general aspects of research. In the past, only the computing power of computers were being used by considering a number of ways in record time. But the combinatorial explosion problem led to the consideration of the use of heuristics to better problem solving. Then there was the appearance of action plans to achieve the intended purpose, that is - to - say beat the opponent.

Activity Details

For instance, where there are two players in a game case: J1 and J2, the game is asynchronous, that is - to -dire each player has his turn to play, equally, each player knows about the game rules and conditions at every given time. It starts in the situation where J1 is the first to play. One set can be viewed as a tree with:

- The root is the current state of play
- Depth peer nodes to match the case J1 must play
- The depth to match the odd nodes if J2 must play
- An arc corresponds to a stroke played and binding a state where this one is permitted to the state resulting from the application of the coup
- The leaves correspond to endgames: (losers states, winners or zero J1)
- A path from the root to a leaf describes part.

Here is the tree to develop the game, every sheet is with its value, then trace these values with the assumption that each player chooses the best move for him / her self. Thus Max chooses the move leading to the state of maximum value, Min chooses the move leading to the minimum value of state. The Max player is called the maximizing player (or player), the Min player is called the minimizing player (or opponent).

A state of the game is a configuration of the game, for example, the state of a chess game is the set of all positions of the pieces on the chessboard. The initial state is the state of play before the players start to play, and the final state is a state ending the game (win, lost, tied)

Winning a state is the value gained by a player he /she reached at the end of the game state. A stroke is an action game for passing from one state to another.

For instance: move a counter of A in B

The threads of a state ei, are attainable states since this state. They indicate them by function (ei). They define a heuristic function, noted h, which links a value to every state of the game, it allows us to know the winnings.

Algorithm of MinMax.

Function Minimax (e, d)

Entries: knot e, depth d

Exits: Value Minimax of the knot e

if final? (e) or (d == 0) then returns h (e)

if not

if player? (e) then

turn max {Minimax (e i, d − 1) | e i ∈ fr (e)}

otherwise

go back min {Minimax (e i, d − 1) | e i ∈ fr (e)}

Conclusion

The algorithm of MiniMax applies to the theory of games for games to two players with no sum (and in full information) consisting in minimizing loss maximum. It leads the computer to see again all possibilities for a number restricted by blows and to allocate them a value which takes into account benefits for the player and for his opponent. The best choice is then the one who minimizes the losses of the player while assuming that the opponent also tries to maximize them (game is in no sum).

Assessment

On a table matches are willing of bite d '. Each time it is the turn of a player, it chooses 1, 2 or 3 bites at the same time. The player who to withdraw the last match is the loser. Make the tree MINIMAX for situation with 5 matches.

Unit Summary

This unit concerned automatic study, and games. Automatic study is a domain very appraised in artificial intelligence, because it is not any more a question of programming the computer manually (what is tiring in certain cases), but rather to provide in the computer of the shone allowing shone capacities - even to teach, it is - - to tell to auto-program. There are several types of studies and every type includes several algorithms.

Games don't remain on the sidelines in Artificial Intelligence. This section on gaming has evolved, and there were even computers that beat world champions. This part also includes several algorithms including the minimax algorithm described above.

Unit Assessment

Check your understanding!

Evaluation for machine learning

Instructions

Using data from the following table, make a decision tree.

Day	Sky	Temperature	Humidity	Wind	Tennis Play
1	Sunny	Hot	High	Low	No
2	Sunny	Hot	High	Windy	No
3	Cloudy	Hot	High	Low	Yes
4	Rainy	Tempered	High	Low	Yes
5	Rainy	Cold	Normal	Low	Yes
6	Rainy	Cold	Normal	Windy	No
7	Cloudy	Cold	Normal	Windy	Yes
8	Sunny	Tempered	High	Low	No
9	Sunny	Cold	Normal	Low	Yes
10	Rainy	Tempered	Normal	Low	Yes
11	Sunny	Tempered	Normal	Windy	Yes
12	Cloudy	Tempered	High	Windy	Yes
13	Cloudy	Hot	Normal	Low	Yes
14	Rainy	Tempered	High	Windy	No

Answers

From the minimax algorithm, describe some of tic-tac-toe.

Unit Readings and Other Resources

The readings in this unit are to be found at course level readings and other resources.

Artificial Intelligence: A Modern Approach, Third Edition, Stuart Russell & Peter Norvig,

Pearson Education Inc., December, 2009 ISBN: 978-0-13-604259-4

Machine Learning, Tom Mitchell, McGraw Hill, 1997, ISBN: 978-0-070-42807-2

Introduction to Machine Learning, Ethem Alpaydin, MIT Press, 2010 ISBN: 978-0-262-01243-0

Unit4. Natural Language, Representation and Reasoning

Unit Introduction

Nowadays that the use information technology is an essential tool in our daily business activities, there is a need for more increasing manipulation of information, especially its transmission and communication mechanism. This can mechanism can be achieved without a large-scale convention knowledge representation. In that regard, the emergence of formal numbers to facilitate the representation of information has made words and sentences no longer sufficient as the only means of communication mechanism.

It is therefore important to note the growing necessity for humans to interact with computers, not with an artificial language, but the language used by humans: natural language. This study of this unit will focus in fundamental techniques used for knowledge representation and natural language.

Unit Objectives

Upon completion of this unit the learner should be able to:

- Identify different knowledge representation paradigms
- Represent concepts using paradigms cited
- Turn on logical forms proposed statements
- Describe a semantic network representing a particular situation
- Make deductions from a semantic tree
- Include different levels of development of natural language
- Cite some sites using natural language for programs

Key Terms

Knowledge: What we have learned through study or practice

Knowledge Representation: Representing information in form that computers can utilize

Propositional Logic: Study assertions that take true or false values

First Order Logic: Extension of propositional logic with the addition of variables and quantifiers.

Semantic Networks: Trees with labeled nodes representing concepts

Morphology: The study of words and their variations

Syntax: The structure of statement in a computer language

Semantics:The linguistics and logic concerned with meaning

Pragmatic: Dealing with things sensibly and realistically in a way that is based on practical rather than theoretical

Learning Activities

Activity 1 - Knowledge Representation

Introduction

This is the transcription of knowledge in a symbolic form so that they can be exploited by a reasoning system. There are several types of knowledge: Thus, knowledge can be:

- Always true
- Scalable
- Uncertain
- blurred
- Typical
- Ambiguous

For representation there are two aspects:

Data structures and Methods for exploiting these data.

This will allow the learner to approach this subject with an ability to analyze the below topics.

- Logical representations
- Semantic networks

Activity Details

[As said before, knowledge representation will be addressed in two ways: the logical representations and semantic networks.

1.The logical representations

1.Propositional logic

it's assertions capable of making true or false values in a given universe. These statements are composed of predicts and their arguments. Operators can associate to give their other assertions: ¬ (negation), ^ (and) v (or) => (implication), <=> (equity).

Example: fish (salmon) denotes a predicate as fish and fish its argument. This statement takes the value 'true' if we interpret as: salmon is a fish.

We have several arguments: tail (horse, long): the horse's tail is long.

2. Logic of first order of predicates

The logic of the proposal having the shortcomings, the predicates of the first order is an extension, with the introduction of the variables noted: u, v, w, etc. that can be quantified with the universal quantifier $\forall$ or existential quantifier $\exists$ It is called 1st order because these are the variables that are not quantified predicates.

For example $\forall$ x P ($\exists$) means that the proposition P (x) is true for all the elements x of the domain of definition of P.

( ) Q (y) means that there is at least an element y of the field for which Q (y) is true.

Examples of formulas:

($\exists$) (COUNTRY (x) ^ CAPITAL (x, Bamako))

($\forall$) (ELEPHANT (x) => EARS (x, wide))

3. The logical reasoning:

It is producing new menus from existing formulas. For this, there are rules that provide some software like Prolog software. These rules are:

The rule of the 'modus ponens'': P and P => Q enable to prove that Q is true

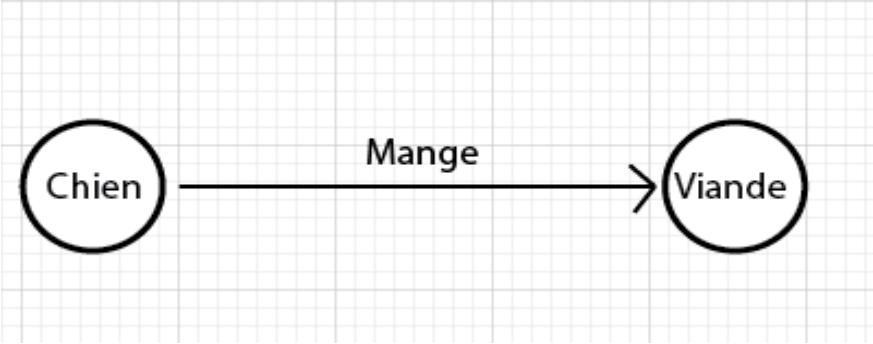
Example: COMPUTER (x) => TOOL (x, COMPUTER) and INFORMATICIEN (JOHN) can be inferred TOOL (JEAN, COMPUTER). This mechanism is called unification or filtering.

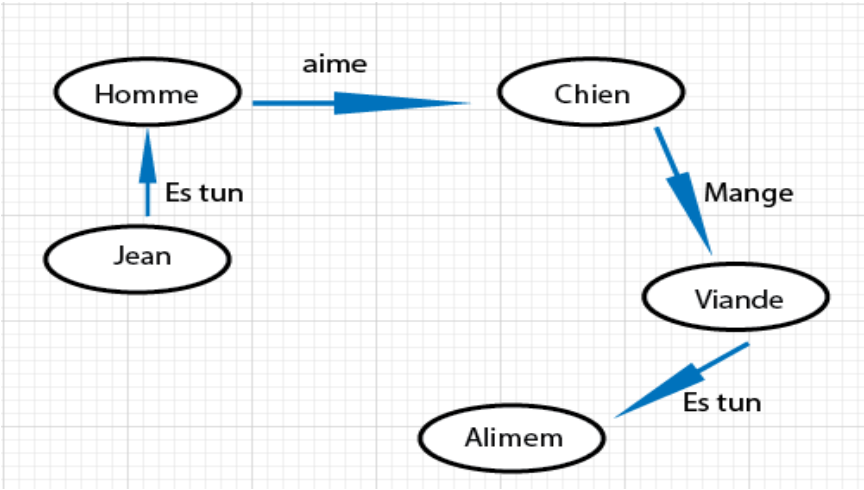
The rule of the 'modus tollens': from P => Q and $\neg Q$ is deduced $\neg P$ Example: PROFESSOR (x) => TOGE (x, BLUE) and $\neg(TOGE(Jean, BLEUE))$ so $\neg(PROFESSEUR(Jean))$ The principle of resolution: It allows to build a new clause from two clauses considered parents clauses. A clause is a disjunction between several simple formulas.

2. The semantic networks

A semantic network is a graph whose nodes labeled represent concepts and edges resulting relationships of meaning between concepts. These relations can translate words like 'is kind of a' 'or' 'is" or other expressions reflecting circumstantial complements.

Example:





Conclusion

Note:

Semantic networks generally have knowledge links are hierarchical in nature.

Conclusion

Knowledge representation by logic has some flaws, because the system is based on the knowledge of developers and as it is impossible to implement intelligent modules that explain the behavior.

Semantic networks in turn are suitable for the representation of classifications and in this case the only links are those that are limited to the hierarchical relationships of the type "is" and "is a kind of". This representation ambition was to be a model of memory human.]

Key Terms

1- What kind of knowledge representation paradigms discussed here?

2- Using the logic of first order representing the following proposals:

a) A computer teacher has a white toga

b) There is at least one bird that does not fly

c) If A is a student and if A has validated more than 3 invalid credits, then adjourned A.

3. Represent the following proposals using semantic networks:

a) works at Issa ORANGE service. This service is part of the Department of Communication. Mariam is the wife of Issa; she works at Marital service that is part of the Department of Communication. Musa and the children of Hawa Issa and Mariam. Moussa is a school called Castor and Hawa in a school called Planet. Moussa loves his cat is a feline.

b) A mammal is an animal. A bird is an animal. The dog is a mammal. The cat is a mammal. The rooster is a bird. The pigeon is a bird. The rooster and pigeon eat seeds. The cat and the dog have hair and eat birds. Birds have feathers.

Activity 2 - Natural language processing Presentation

Introduction

The natural language processing is an area that the result of joint efforts of linguists and computer scientists. It is through the processing on the coding information (algorithms). Nowadays one of the AI challenges is to use natural language to communicate with the machine. The natural language processing occurs in several areas:

- Machine Translation
- Spellchecking
- Speech recognition
- The speech synthesis
- Search Engines

Activity Details

There are several levels of natural language processing; here we take the case of the texts. It is:

Morphological level:

Morphology is the study of morpheme and types of existing combinations between morphemes. The morpheme is the smallest linguistic unit having meaning and form. We distinguish lexical morphemes and grammatical morphemes (affixes). Lexical morphemes or lexemes are to be applied to objects in the world such as nouns, verbs and adjectives, while grammatical grammatical role play; they combine to give tokens to other words. Lexical have an expandable list (shocking, hyper shocking), while the list of grammar is frozen. In a system of NLP, morphological analysis has as main objective to recognize the grammatical properties of words, identify named entities such as the names of objects.

Syntactic level:

It is developing rules for combining words to obtain phrases. These rules talk about the nature of the constituents of the sentence, the hierarchical structure of the components. This level is much used for the compilation of programming languages.

Semantic level: the senses of "words". IL offers a correspondence between real-world situations and structures recognized by the syntactic level. A word has no meaning when used in context.

The pragmatic level: interpretation in the general sense to address any ambiguities in relation to certain concepts It covers some attitudes to be taken vise - à – visa statements: probative, desirability, truth. Pragmatic can evolve over time in a dialog.

Conclusion

The natural language processing is nowadays widely used in industry (language industry), research is always pointed towards this area even if the achievements are still inadequate. Current trends are:

Use of logic for processing specific aspects of language Interaction between parsing and semantic analyst.

Fault tolerance, with unfamiliar words and phrases

Extensive dialogue between man and machine.

Assessment

To understand machine language and to program a machine to act in place of man, the program algorithm must agreeable between man and his desired machine program. That is the set of rules, e.g.

a. A sentence consists of a noun accompanied by a verbal group.

b. A noun: common name + Article

c. Verbal Group: verb + noun

d. The only known items are and.

e. The only known common names are dog meat

f. The only known word is love.

Write all the possible sentences that follow the rules of this mini-grammar. There must be 16 in all.

Identify those with errors at the semantic level and those with errors syntactically. A phrase like "dog likes meat" contains syntax errors but still understandable.

Activity 3 - Voice and Speech Recognition

Introduction

This activity of the unit will enable the learner with a basic introduction to Natural Language Understanding (NLU) in AI. Simple example of voice and speech recognition project will be designed. Some of what we have seen, in search and in learning, will be applied in this project. Natural language processing and understanding is a large field of research and has entire courses devoted to it. So, in this introduction, our objective is simply to introduce the problems and approaches.

Activity Details

Voice recognition can be very complicated depending on the level of sophistication you desire. The approach is comparatively simple set of requirements. It will be appreciable to recognize your own voice. Thus the learner do not require complex speech-to-text and voice recognition libraries or any of the excellent 3rd party software via Internet search engines (there is no shortage of these!).

That is the solution I have tried to think up below:

Solution Description

You (the learner) will be writing this project in C or any other language he/she is comfortable with.

However, below are the the but I wish to discuss a language agnostic process, focusing on the process its self.

1. You will be required to pre-record your dictionary of words to match those being spoken. Example, imagine that you have 20 recordings of your 20 different words, or perhaps short phrases or sentences of two or three words. I believe this makes the process of comparing two recording files easier than actually converting the audio to text and comparing two strings.

2. A microphone should be connected to your hardware device running my code. [1]. The code will be continuously taking fixed length samples, say 10msec in length for example, and storing 10 consecutive samples for example, in a circular logging style.

1. This would likely be connected through a band-pass filter and op-amp, as would the dictionary recordings be made, to keep the stored and collected audio samples smaller.

2. Work with you A.I. instructor and peer groups to get and assemble the required hardware. This part is going to be much more hardware involved so not really for discussion here.

3. The code looks at it's stored 10 consecutive samples and looks for a volume increase to indicate a word or phrase is being said (a break from silence) and then increases is consecutive sample collecting to say 500 samples for example. That would mean it captures 5 seconds of audio in 10 msec samples.

In this article, I tell you how to program speech recognition, speech to text, text to speech and speech synthesis in C# using the System.Speech library.

Speech recognition in C#

Speech recognition

To create a program with speech recognition in C#, you need to add the System.Speech library. Then, add this using namespace statement at the top of your code file:

```
using System.Speech.Recognition;

using System.Speech.Synthesis;

using System.Threading;

Then, create an instance of the SpeechRecognitionEngine:

SpeechRecognitionEngine _recognizer = new SpeechRecognitionEngine();
```

Then, we need to load grammars into the SpeechRecognitionEngine. If you don't do that, the speech recognizer will not recognize phrases. For example, add a grammar with `the phrase` "test" and we give the grammar the name "testGrammar":

```
recognizer.LoadGrammar(new Grammar(new GrammarBuilder("test")) { Name =
"testGrammar" }); // load a grammar "test"

Or:

Grammar gr = new Grammar(new GrammarBuilder("test")); gr.Name =
testGrammar";

 _recognizer.LoadGrammar(gr);
```

If you don't want to give a name to the grammar, do this:

NB: The learner should continue work with the instructor as a guide to further develop this code or a similar ode that will do the same as an activity project.

Conclusion

For language learning, speech recognition can be useful for learning a second language. It can teach proper pronunciation, in addition to helping a person develop fluency with their speaking skills.

Students who are blind (see Blindness and education) or have very low vision can benefit from using the technology to convey words and then hear the computer recite them, as well as use a computer by commanding with their voice, instead of having to look at the screen and keyboard.

Students who are physically disabled or suffer from Repetitive strain injury/other injuries to the upper extremities can be relieved from having to worry about handwriting, typing, or working with scribe on school assignments by using speech-to-text programs. They can also utilize speech recognition technology to freely enjoy searching the Internet or using a computer at home without having to physically operate a mouse and keyboard.

Speech recognition can allow students with learning disabilities to become better writers. By saying the words aloud, they can increase the fluidity of their writing, and be alleviated of concerns regarding spelling, punctuation, and other mechanics of writing. Also, see Learning disability.

Assessment

a). how do we go about adding search queries to this program?

b). What is the voice recognition space capacity of your program

c). Measure the sensitivity of your recognition program with any identified tool using proofs

d). The completion of your course work, write and document the features, strengths and challenges of your program.

Unit Assessment

Check your understanding!

[Knowledge representation]

Instructions

AI concepts and techniques are learned in two steps: theory, and implementation of theory in programs. AI techniques are difficult to program and can be obscured by the programming details. To make these techniques explicit and to hide the programming details, AI languages--such as Lisp, Prolog, Scheme, and others--have been defined to have language features that directly support the implementation of AI techniques. The use of class libraries and source code libraries serve the same purpose for general-purpose languages, such as C++ and Java. The Watson text uses Java Classes for program examples of AI concepts and techniques. Java, being widely used, has the added advantage of making these techniques more widely available.

Grading Scheme

Test, participation quiz: 35%

Group project: 35%

Individual Project 30%

Answers

Students will receive their courses and feedback in the AVU course management system or a portal platform.

Unit Summary

The increasing need to share knowledge gave birth to a representation of those - one. A knowledge representation is a system using symbols and operations on those symbols. But having only symbols and operations is not enough because it is impossible to design a performance without considering modeling the reasoning. The logic was one of the first formalism designed to knowledge representation, and remains today as the basis of much research in AI.

Cognitive science also had their share of contribution, and semantic networks.Two key aspects are the basis of the motivation for natural language processing:The desire to model natural language (which is a fascinating skill) in order to test hypotheses about the mechanisms of human communication.

The need to build applications that can deal effectively with written or audio documents available electronically. This makes the NLP (Natural Language Processing) an area that is at the crossroads of several disciplines such as artificial intelligence, logic, linguistics and even statistics and neuro-sciences.

The application of this field has created some tools: example

Automatic translation tools:

Google translation:	http: //translate.google.frreverse:http: //www.reverso.net
Babel Fish:	http: //www.fr.babelfish.yahoo.com
Syzran:	http: //www.systran.fr/traduction-en-ligne-gratuite

Search tools:

Google:	http: //www.google.com
Bing	http: //www.bing.com
yahoo:	http: //fr.yahoo.com
AltaVista	http: //fr.altavista.com
Wiki:	http: //www.wikio.fr

Unit Readings and Other Resources

The readings in this unit are to be found at course level readings and other resources.

+Reading: UCI: Max Welling's ICS 171 Introduction to Artificial Intelligence Course

Reading: Mark Watson's Practical Artificial Intelligence Programming with Java:

Reading: IT: Kaelbling and Lozano-Perez 6.034 Artificial Intelligence Course:

The African Virtual University Headquarters

Cape Office Park

Ring Road Kilimani

PO Box 25405-00603

Nairobi, Kenya

Tel: +254 20 25283333

contact@avu.org

oer@avu.org

The African Virtual University Regional Office in Dakar

Université Virtuelle Africaine

Bureau Régional de l'Afrique de l'Ouest

Sicap Liberté VI Extension

Villa No.8 VDN

B.P. 50609 Dakar, Sénégal

Tel: +221 338670324

bureauregional@avu.org