

CSCE 580: Introduction to AI *CSCE 581: Trusted AI*

Lecture 7: Search Continued

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

14TH SEP 2023

Carolinian Creed: “I will practice personal and academic integrity.”

Credits: Copyrights of all material reused acknowledged

Organization of Lecture 7

- Introduction Segment
 - Recap of Lecture 6
- Main Segment
 - Problems: vacuum, sliding tile, N-queens
 - Search – uninformed
 - Analyzing search performance
 - Informed search
 - Quiz 1
- Concluding Segment
 - Course Project Discussion
 - About Next Lecture – Lecture 8
 - Ask me anything

Introduction Section

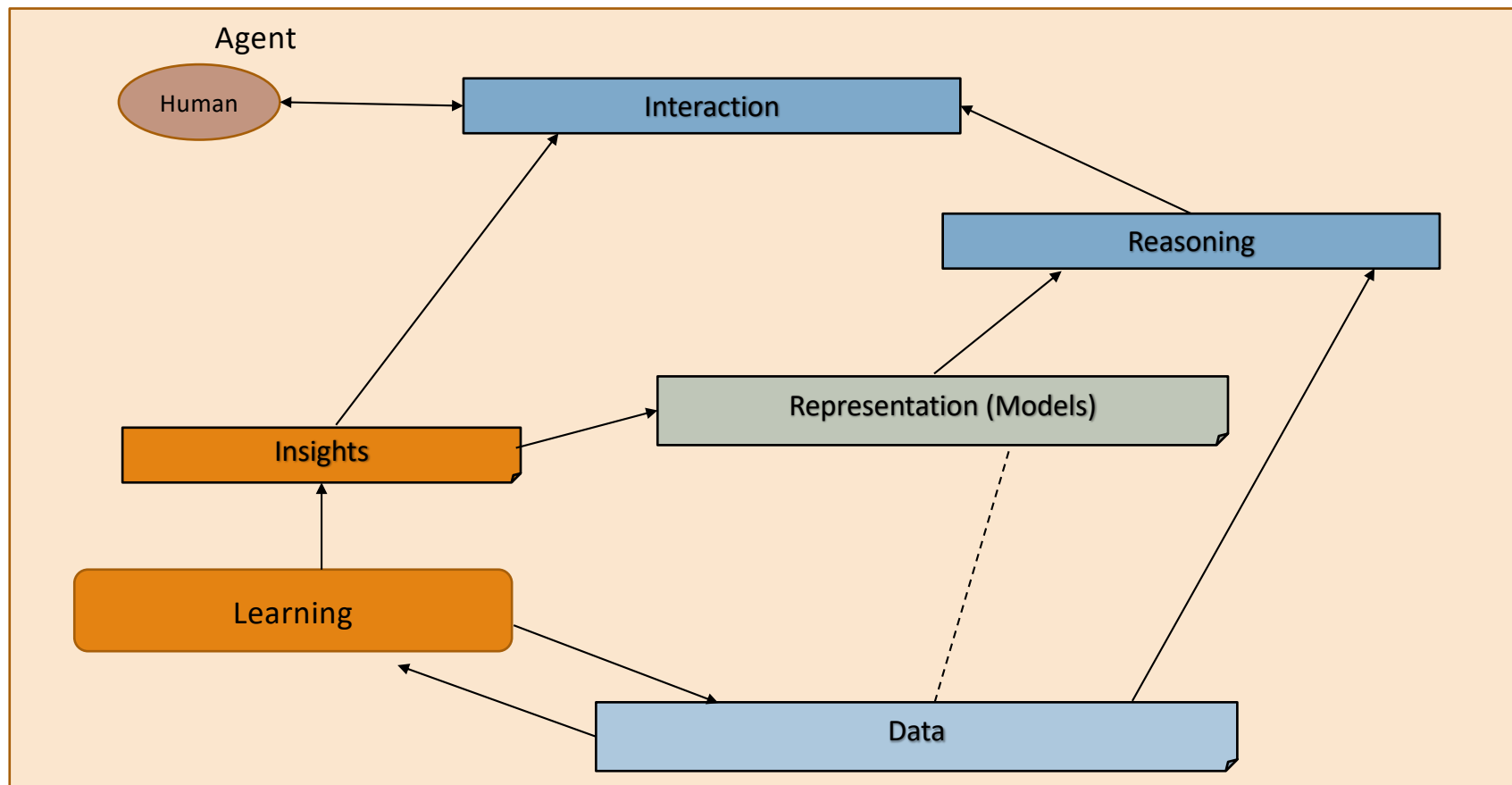
Recap of Lecture 6

- Problem solving agent – goal directed
- Problem formulation – abstraction, type of problems
- Search approach of problem solving

Intelligent Agent Model



Relationship Between Main AI Topics



Where We Are in the Course

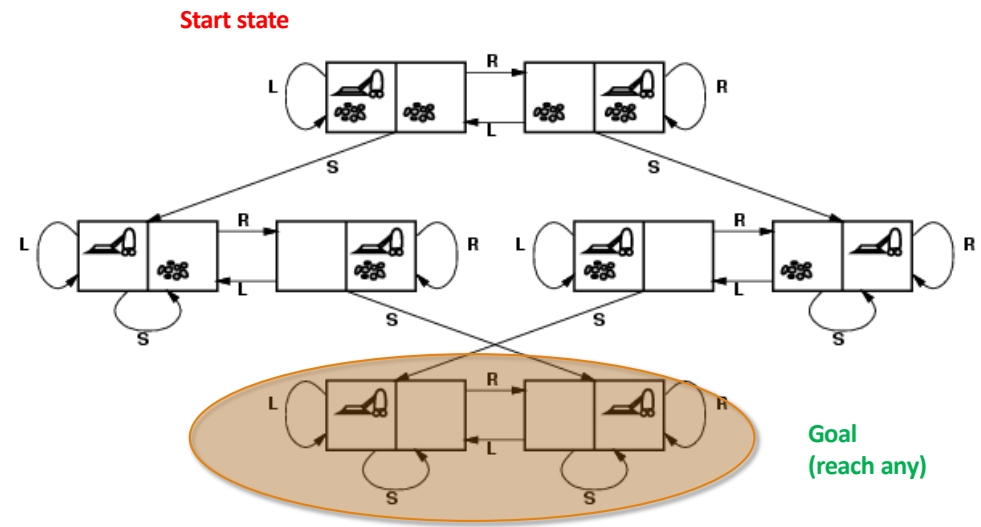
CSCE 580/ 581 – In This Course

- Week 1: Introduction, Aim: Chatbot / Intelligence Agent
- Weeks 2-3: Data: Formats, Representation and the Trust Problem
- Week 4-5: Search, Heuristics - Decision Making
- Week 6: Constraints, Optimization – Decision Making
- Week 7: Classical Machine Learning – Decision Making, Explanation
- Week 8: Machine Learning - Classification
- Week 9: Machine Learning - Classification – Trust Issues and Mitigation Methods
- Topic 10: Learning neural network, deep learning, Adversarial attacks
- Week 11: Large Language Models – Representation, Issues
- Topic 12: Markov Decision Processes, Hidden Markov models - Decision making
- Topic 13: Planning, Reinforcement Learning – Sequential decision making
- Week 14: AI for Real World: Tools, Emerging Standards and Laws; Safe AI/ Chatbots

Main Section

Example: Vacuum World

States	8 possible world states <i>(2room x 2dirt location x 2clean?)</i>
<ul style="list-style-type: none"> Initial state Goal state 	<ul style="list-style-type: none"> Any No dirt at all locations
Actions	Left, Right, Suck
<ul style="list-style-type: none"> Transition model Action cost 	<ul style="list-style-type: none"> Action transition (edges) 1



Adapted from:

1. Russell & Norvig, AI: A Modern Approach
2. Bart Selman's CS 4700 Course

Example: Sliding 8-tile Puzzle

States <ul style="list-style-type: none">• Initial state• Goal state	Location of tiles <ul style="list-style-type: none">• Any (given)• All numbers sorted, Empty tile in corner (given)
Actions <ul style="list-style-type: none">• Transition model• Action cost	move blank left, right, up, down <ul style="list-style-type: none">• Blank transition (edges)• 1

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

Adapted from:

1. Russell & Norvig, AI: A Modern Approach
2. Bart Selman's CS 4700 Course

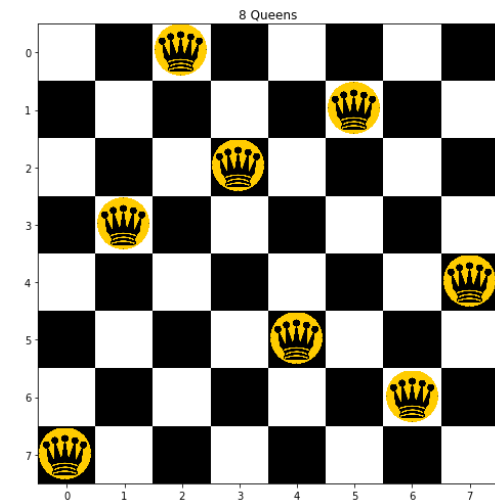
Exercise: N-Queen Puzzle

States

- Initial state
- Goal state

Actions

- Transition model
- Action cost

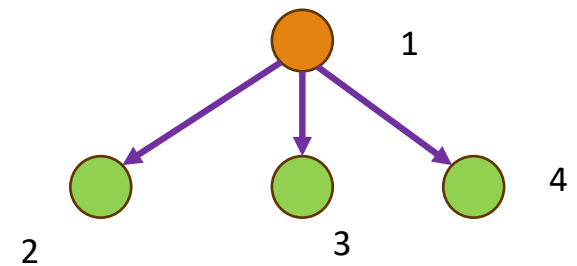


Adapted from:
Russell & Norvig, AI: A Modern Approach

Tree-search Algorithms

Basic idea: simulated exploration of state space by generating successors of already-explored states (a.k.a. ~ **expanding** states)

```
function TREE-SEARCH(problem, strategy) returns a solution, or failure
  initialize the search tree using the initial state of problem
  loop do
    if there are no candidates for expansion then return failure
    choose a leaf node for expansion according to strategy
    if the node contains a goal state then return the corresponding solution
    else expand the node and add the resulting nodes to the search tree
```



Adapted from:
1. Russell & Norvig, AI: A Modern Approach
2. Bart Selman's CS 4700 Course

Uninformed Search Strategies

Search strategies use only the information available in the problem definition. They do not use a measure of distance to goal (uninformed).

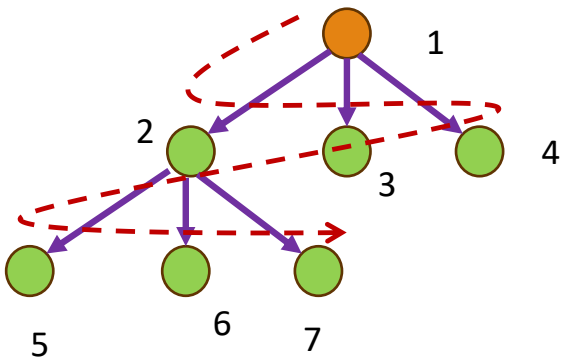
- Breadth-first search
- Uniform-cost search
- Depth-first search
- Depth-limited search
- Iterative deepening search
- Bidirectional search

Consideration: type of queue used for the **fringe of the search tree**
(collection of tree nodes that have been generated but not yet expanded)

Adapted from:

1. Russell & Norvig, AI: A Modern Approach
2. Bart Selman's CS 4700 Course

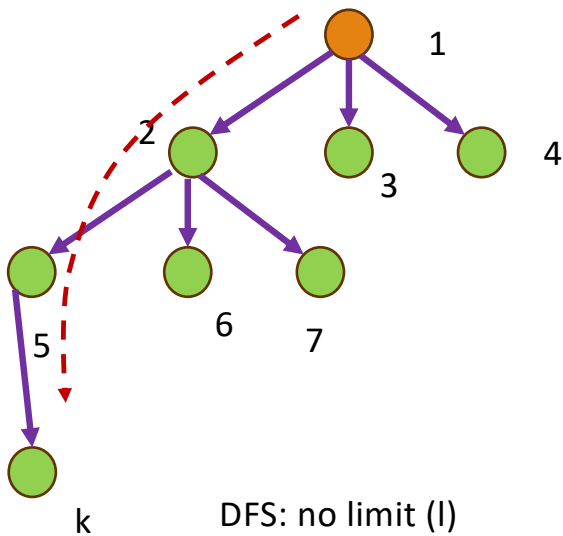
Breadth First Search (BFS)



```
function BREADTH-FIRST-SEARCH(problem) returns a solution node or failure  
  node  $\leftarrow$  NODE(problem.INITIAL)  
  if problem.IS-GOAL(node.STATE) then return node  
  frontier  $\leftarrow$  a FIFO queue, with node as an element  
  reached  $\leftarrow$  {problem.INITIAL}  
  while not IS-EMPTY(frontier) do  
    node  $\leftarrow$  POP(frontier)  
    for each child in EXPAND(problem, node) do  
      s  $\leftarrow$  child.STATE  
      if problem.IS-GOAL(s) then return child  
      if s is not in reached then  
        add s to reached  
        add child to frontier  
  return failure
```

Adapted from: Russell & Norvig, AI: A Modern Approach

Depth First Search (DFS) and Depth Limited Search (DLS)



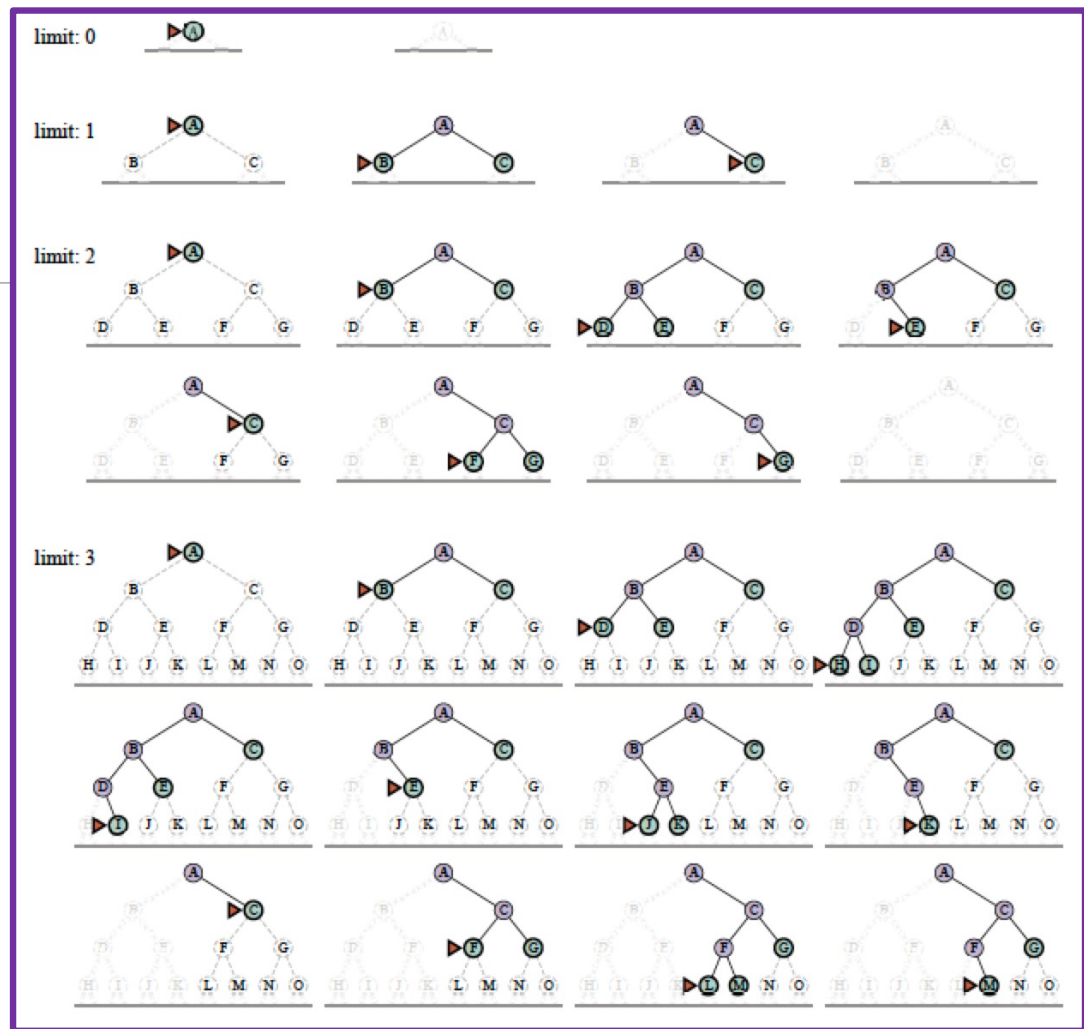
DFS: no limit (l)

Cutoff: when result is cutoff due to l
(result may be there if l increased)

```
function DEPTH-LIMITED-SEARCH(problem,  $\ell$ ) returns a node or failure or cutoff
  frontier  $\leftarrow$  a LIFO queue (stack) with NODE(problem.INITIAL) as an element
  result  $\leftarrow$  failure
  while not IS-EMPTY(frontier) do
    node  $\leftarrow$  POP(frontier)
    if problem.IS-GOAL(node.STATE) then return node
    if DEPTH(node) >  $\ell$  then
      result  $\leftarrow$  cutoff
    else if not IS-CYCLE(node) do
      for each child in EXPAND(problem, node) do
        add child to frontier
  return result
```

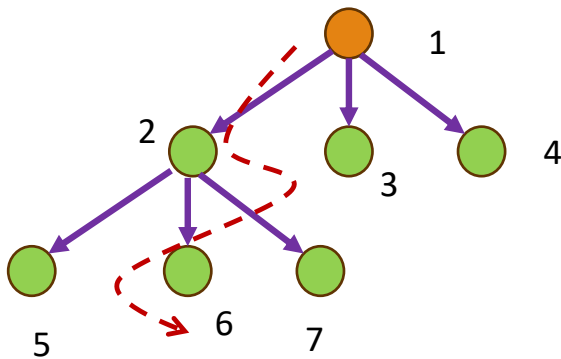
Adapted from: Russell & Norvig, AI: A Modern Approach

Illustration: DLS



Adapted from: Russell & Norvig, AI: A Modern Approach

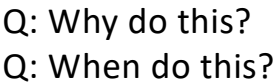
Best-First Search



```
function BEST-FIRST-SEARCH(problem, f) returns a solution node or failure
  node  $\leftarrow$  NODE(STATE=problem.INITIAL)
  frontier  $\leftarrow$  a priority queue ordered by f, with node as an element
  reached  $\leftarrow$  a lookup table, with one entry with key problem.INITIAL and value node
  while not IS-EMPTY(frontier) do
    node  $\leftarrow$  POP(frontier)
    if problem.IS-GOAL(node.STATE) then return node
    for each child in EXPAND(problem, node) do
      s  $\leftarrow$  child.STATE
      if s is not in reached or child.PATH-COST < reached[s].PATH-COST then
        reached[s]  $\leftarrow$  child
        add child to frontier
  return failure
```

```
function EXPAND(problem, node) yields nodes
  s  $\leftarrow$  node.STATE
  for each action in problem.ACTIONS(s) do
    s'  $\leftarrow$  problem.RESULT(s, action)
    cost  $\leftarrow$  node.PATH-COST + problem.ACTION-COST(s, action, s')
    yield NODE(STATE=s', PARENT=node, ACTION=action, PATH-COST=cost)
```

Source: Russell & Norvig, AI: A Modern Approach



the search, the two paths are joined (by the function JOIN_NODES) to form a solution. The first solution we get is not guaranteed to be the best; the function TERMINATED determines when to stop looking for new solutions.

```

function PROCEED(dir, problem, frontier, reached, solution) returns a solution
    // Expand node on frontier; check against the other frontier in reached2.
    // The variable “dir” is the direction: either F for forward or B for backward.
    node ← POP(frontier)
    for each child in EXPAND(problem, node) do
        s ← child.STATE
        if s not in reached or PATH-COST(child) < PATH-COST(reached[s]) then
            reached[s] ← child
            add child to frontier
            if s is in reached2 then
                solution2 ← JOIN-NODES(dir, child, reached2[s])
                if PATH-COST(solution2) < PATH-COST(solution) then
                    solution ← solution2
    return solution

```

Figure 3.14 Bidirectional best-first search keeps two frontiers and two tables of reached states. When a path in one frontier reaches a state that was also reached in the other half of the search, the two paths are joined (by the function JOIN-NODES) to form a solution. The first solution we get is not guaranteed to be the best; the function TERMINATED determines when to stop looking for new solutions.

Analyzing Search Performance

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes ¹	Yes ^{1,2}	No	No	Yes ¹	Yes ^{1,4}
Optimal cost?	Yes ³	Yes	No	No	Yes ³	Yes ^{3,4}
Time	$O(b^d)$	$O(b^{1+\lceil C^*/\epsilon \rceil})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lceil C^*/\epsilon \rceil})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$

Figure 3.15 Evaluation of search algorithms. b is the branching factor; m is the maximum depth of the search tree; d is the depth of the shallowest solution, or is m when there is no solution; ℓ is the depth limit. Superscript caveats are as follows: ¹ complete if b is finite, and the state space either has a solution or is finite. ² complete if all action costs are $\geq \epsilon > 0$; ³ cost-optimal if action costs are all identical; ⁴ if both directions are breadth-first or uniform-cost.

Coding Example

- N-Queens – code notebook
 - <https://github.com/biplav-s/course-ai-tai-f23/blob/main/sample-code/Class6-To-Class9-search.md>

Informed Search – Greedy best-first

Uses domain/problem specific hints to guide search

$$f(n) = h(n)$$

- f: estimated cost of best path via n to goal
- h: estimated cost to goal from n // h is also called heuristic function

Informed Search – A* search

Uses domain/problem specific hints to guide search

$$f(n) = g(n) + h(n)$$

- f: estimated cost of best path via n to goal
- g: cost of best path to n
- h: estimated cost to goal from n

Course Project

Project Discussion: What Problem Fascinates You ?

- Data
 - Water
 - Finance
 - ...
- Analytics
 - Search, Optimization, Learning, Planning, ...
- Application
 - Building chatbot
- Users
 - Diverse demographics
 - Diverse abilities
 - Multiple human languages

Project execution in sprints

- Sprint 1: (Sep 12 – Oct 5)
 - **Solving**: Choose a decision problem, identify data, work on solution methods
 - **Human interaction**: Develop a basic chatbot (no AI), no problem focus
- Sprint 2: (Oct 10 – Nov 9)
 - **Solving**: Evaluate your solution on problem
 - **Human interaction**: Integrated your choice of chatbot (rule-based or learning-based) and methods
- Sprint 3: (Nov 14 – 30)
 - **Evaluation**: Comparison of your solver chatbot with an LLM-based alternative, like ChatGPT

Project Discussion: Dates and Deliverables

Project execution in sprints

- Sprint 1: (Sep 12 – Oct 5)
 - **Solving**: Choose a decision problem, identify data, work on solution methods
 - **Human interaction**: Develop a basic chatbot (no AI), no problem focus
- Sprint 2: (Oct 10 – Nov 9)
 - **Solving**: Evaluate your solution on problem
 - **Human interaction**: Integrated your choice of chatbot (rule-based or learning-based) and methods
- Sprint 3: (Nov 14 – 30)
 - **Evaluation**: Comparison of your solver chatbot with an LLM-based alternative, like ChatGPT

- Oct 12, 2023
 - Project checkpoint
 - In-class presentation
- Nov 30, 2023
 - Project report due
- Dec 5 / 7, 2023
 - In-class presentation

Skeleton: A Basic Chatbot

- Run in an infinite loop until the user wants to quit
- Handle any user response
 - User can quit by typing “Quit” or “quit” or just “q”
 - User can enter any other text and the program has to handle it. The program should write back what the user entered and say – “I do not know this information”.
- Handle known user query types // Depends on your project
 - “Tell me about N-queens”, “What is N ?”
 - “Solve for N=4?”
 - “Why is this a solution? ”
- Handle chitchat // Support at least 5, extensible from a file
 - “Hi” => “Hello”
 - ...
- *Store session details in a file*

Illustrative Project

1. **Title:** Solve and explain solving of n-queens puzzle
2. **Key idea:** Show students how a course project will look like
3. **Who will care when done:** students of the course, prospective AI students and teachers
4. **Data need:** n: the size of game; interaction
5. **Methods:** search
6. **Evaluation:** correctness of solution, quality of explanation, appropriateness of chat
7. **Users:** with and without AI background; with and without chess background
8. **Trust issue:** user may not believe in the solution, may find interaction offensive (why queens, not kings? ...)

Project Discussion: Illustration

1. Create a private Github repository called “CSCE58x-Fall2023-<studentname>-Repo”. Share with Instructor (biplav-s) and TA (kausik-l)
2. Create Google folder called “CSCE58x-Fall2023-<studentname>-SharedInfo”. Share with Instructor (prof.biplav@gmail.com) and TA (lakkarajukausik90@gmail.com)
3. Create a Google doc in your Google repo called “Project Plan” and have the following by next class (Sep 5, 2023)

1. **Title:** Solve and explain solving of n-queens puzzle
2. **Key idea:** Show students how a course project will look like
3. **Who will care when done:** students of the course, prospective AI students and teachers
4. **Data need:** n: the size of game; interaction
5. **Methods:** search
6. **Evaluation:** correctness of solution, quality of explanation, appropriateness of chat
7. **Users:** with and without AI background; with and without chess background
8. **Trust issue:** user may not believe in the solution, may find interaction offensive (why queens, not kings? ...)

Project Illustration: N-Queens

- Sprint 1: (Sep 12 – Oct 5)
 - **Solving**: Choose a decision problem, identify data, work on solution methods
 - Method 1: Random solution
 - Method 2: Search – BFS
 - Method 3: Search - ...
 - **Human interaction**: Develop a basic chatbot (no AI) as outlined
 - Deliverable
 - Code structure in Github
 - ./data
 - ./code
 - ./docs
 - ./test
 - Presentation: Make sprint presentation on Oct 12, 2023

Reference: Project Rubric

- **Project results – 60%**
 - Working system ? – 30%
 - Evaluation with results superior to baseline? – 20%
 - Considered related work? – 10%
- **Project efforts – 40%**
 - Project report – 20%
 - Project presentation (updates, final) – 20%
- **Bonus**
 - Challenge level of problem – 10%
 - Instructor discretion – 10%
- **Penalty**
 - Lack of timeliness as per announced policy (right) - up to 30%

Milestones and Penalties

- Oct 12, 2023
 - Project checkpoint
 - In-class presentation
 - **Penalty: presentation not ready by Oct 10, 2023 [-10%]**
- Nov 30, 2023
 - Project report due
 - **Project report not ready by date [-10%]**
- Dec 5 / 7, 2023
 - In-class presentation
 - **Project presentations not ready by Dec 4, 2023 [-10%]**

Lecture 7: Summary

- We talked about
 - Goal-directed problem solving agents
 - How to formulate problem formulations
 - Search concepts
 - Problems of controlled robot navigation, 8-tile
 - Search strategies

Concluding Section

Course Project

Project Discussion: What Problem Fascinates You ?

- Data
 - Water
 - Finance
 - ...
- Analytics
 - Search, Optimization, Learning, Planning, ...
- Application
 - Building chatbot
- Users
 - Diverse demographics
 - Diverse abilities
 - Multiple human languages

Project execution in sprints

- Sprint 1: (Sep 12 – Oct 5)
 - **Solving**: Choose a decision problem, identify data, work on solution methods
 - **Human interaction**: Develop a basic chatbot (no AI), no problem focus
- Sprint 2: (Oct 10 – Nov 9)
 - **Solving**: Evaluate your solution on problem
 - **Human interaction**: Integrated your choice of chatbot (rule-based or learning-based) and methods
- Sprint 3: (Nov 14 – 30)
 - **Evaluation**: Comparison of your solver chatbot with an LLM-based alternative, like ChatGPT

Project Discussion: Dates and Deliverables

Project execution in sprints

- Sprint 1: (Sep 12 – Oct 5)
 - **Solving**: Choose a decision problem, identify data, work on solution methods
 - **Human interaction**: Develop a basic chatbot (no AI), no problem focus
- Sprint 2: (Oct 10 – Nov 9)
 - **Solving**: Evaluate your solution on problem
 - **Human interaction**: Integrated your choice of chatbot (rule-based or learning-based) and methods
- Sprint 3: (Nov 14 – 30)
 - **Evaluation**: Comparison of your solver chatbot with an LLM-based alternative, like ChatGPT

- Oct 12, 2023
 - Project checkpoint
 - In-class presentation
- Nov 30, 2023
 - Project report due
- Dec 5 / 7, 2023
 - In-class presentation

Project Discussion: Illustration

1. Create a private Github repository called “CSCE58x-Fall2023-<studentname>-Repo”. Share with Instructor (biplav-s) and TA (kausik-l)
2. Create Google folder called “CSCE58x-Fall2023-<studentname>-SharedInfo”. Share with Instructor (prof.biplav@gmail.com) and TA (lakkarajukausik90@gmail.com)
3. Create a Google doc in your Google repo called “Project Plan” and have the following by next class (Sep 5, 2023)

1. **Title:** Solve and explain solving of n-queens puzzle
2. **Key idea:** Show students how a course project will look like
3. **Who will care when done:** students of the course, prospective AI students and teachers
4. **Data need:** n: the size of game; interaction
5. **Methods:** search
6. **Evaluation:** correctness of solution, quality of explanation, appropriateness of chat
7. **Users:** with and without AI background; with and without chess background
8. **Trust issue:** user may not believe in the solution, may find interaction offensive (why queens, not kings? ...)

Reference: Project Rubric

- **Project results – 60%**
 - Working system ? – 30%
 - Evaluation with results superior to baseline? – 20%
 - Considered related work? – 10%
- **Project efforts – 40%**
 - Project report – 20%
 - Project presentation (updates, final) – 20%
- **Bonus**
 - Challenge level of problem – 10%
 - Instructor discretion – 10%
- **Penalty**
 - Lack of timeliness as per announced policy (right) - up to 30%

Milestones and Penalties

- Oct 12, 2023
 - Project checkpoint
 - In-class presentation
 - **Penalty: presentation not ready by Oct 10, 2023 [-10%]**
- Nov 30, 2023
 - Project report due
 - **Project report not ready by date [-10%]**
- Dec 5 / 7, 2023
 - In-class presentation
 - **Project presentations not ready by Dec 4, 2023 [-10%]**

About Next Lecture – Lecture 8

Lecture 8: Searching for Problem Solving

- Informed Search
 - Heuristics for efficient search
- Class 9: Local search
- Class 10: Adversarial games and search