AI Agents and Search

Lara J. Martin (she/they)

https://laramartin.net/interactive-fiction-class

Slides adapted from Chris Callison-Burch and Cynthia Matuszek

Learning Objectives

Understand the difference between traditional AI agents and agentic AI

List the components of a search problem

Review basic types of tree search algorithms

Define & implement a search problem (for Action Castle)

AI Agent Definition

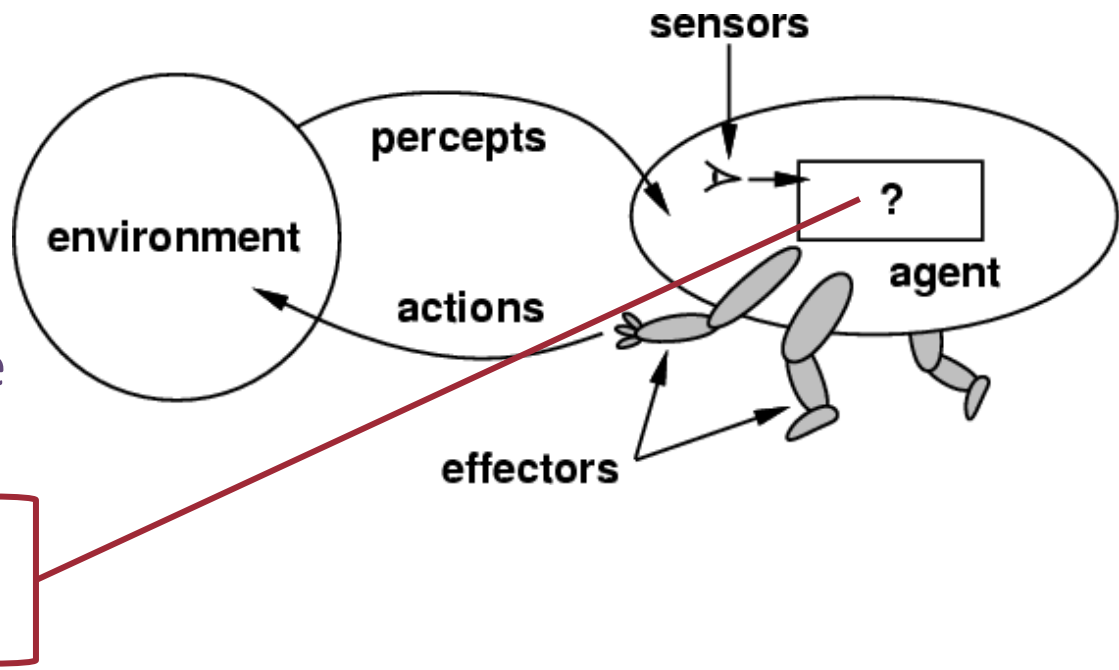
Agent: anything that perceives its environment through sensors, and acts on its environment through actuators

Percept: input at an instant

Percept sequence: history of inputs

Agent function: mapping of percept sequence to action

Agent program: (concise) implementation of an agent function



Picture from Dr. Cynthia Matuszek, source unknown

“Agentic AI”

Term coined around 2023 but has become popular as of early 2025

Using a *language model* to behave like an “agent” – the intersection of AI & NLP beyond chatbots

“autonomous systems designed to pursue complex goals with minimal human intervention” [1]

[1] D. B. Acharya, K. Kuppan and B. Divya, "Agentic AI: Autonomous Intelligence for Complex Goals—A Comprehensive Survey," in *IEEE Access*, vol. 13, pp. 18912-18936, 2025, doi: 10.1109/ACCESS.2025.3532853.

Problem-Solving Agents

A problem-solving agent must plan.

The computational process that it undertakes is called **search**.

It will consider a sequence of actions that form a path to a **goal state**.

Such a sequence is called a **solution** or **plan**.

1. take pole
2. go out
3. go south
4. catch fish with pole
5. go north
6. pick rose
7. go north
8. go up
9. get branch
10. go down
11. go east
12. give the troll the fish
13. go east
14. hit guard with branch
15. get key
16. go east
17. get candle
18. go west
19. go down
20. light lamp
21. go down
22. light candle
23. read runes
24. get crown
25. go up
26. go up
27. go up
28. unlock door
29. go up
30. give rose to the princess
31. propose to the princess
32. down
33. down
34. east
35. east
36. wear crown
37. sit on throne

Formal Definition of a Search Problem

1. **States:** a set **S**
2. An **initial state** $\mathbf{s_i} \in \mathbf{S}$
3. **Actions:** a set **A**

   $\forall$ **s Actions(s)** = the set of actions that can be executed in **s**.

4. **Transition Model:** $\forall$ **s** $\forall$ **a** $\in$ **Actions(s)** **Result(s, a)** $\rightarrow \mathbf{s_r}$

   $\mathbf{s_r}$ is called a successor of **s**

   $\{\mathbf{s_i}\} \cup$ **Successors**$(\mathbf{s_i})$* = state space

5. **Path cost** (Performance Measure):
   Must be additive, e.g. sum of distances, number of actions executed, …

   c(x,a,y) is the step cost, assumed ≥ 0

   - (where action **a** goes from state **x** to state **y**)

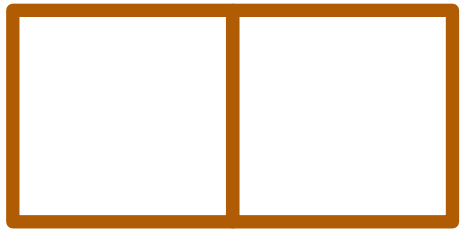
6. **Goal test:** **Goal(s)**

   s is a goal state if **Goal(s)** is true.
   Can be implicit, e.g. **checkmate(s)**

Vacuum World

States: A state of the world says which objects are in which cells.

In a simple two cell version,



Only need the things relevant to the agent's decision making

Vacuum World

States: A state of the world says which objects are in which cells.

In a simple two cell version,

- each cell can have dirt or not

Vacuum World

States: A state of the world says which objects are in which cells.

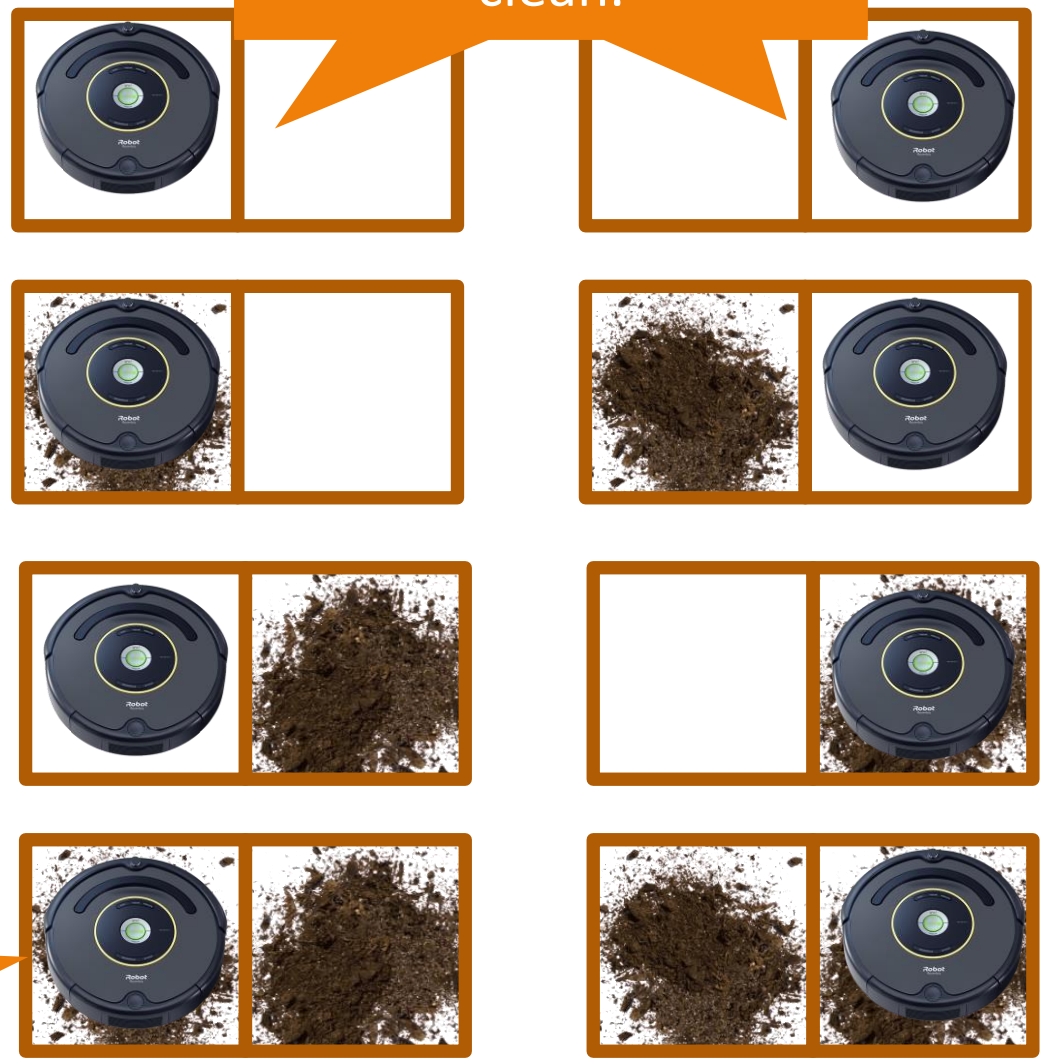
In a simple two cell version,

- each cell can have dirt or not
- the agent can be in one cell at a time

2 positions for agent * 2^2 possibilities for dirt = 8 states.

With n cells, there are $n*2^n$ states.

Goal states: States where everything is clean.



One state is designated as the **initial state**

Vacuum World

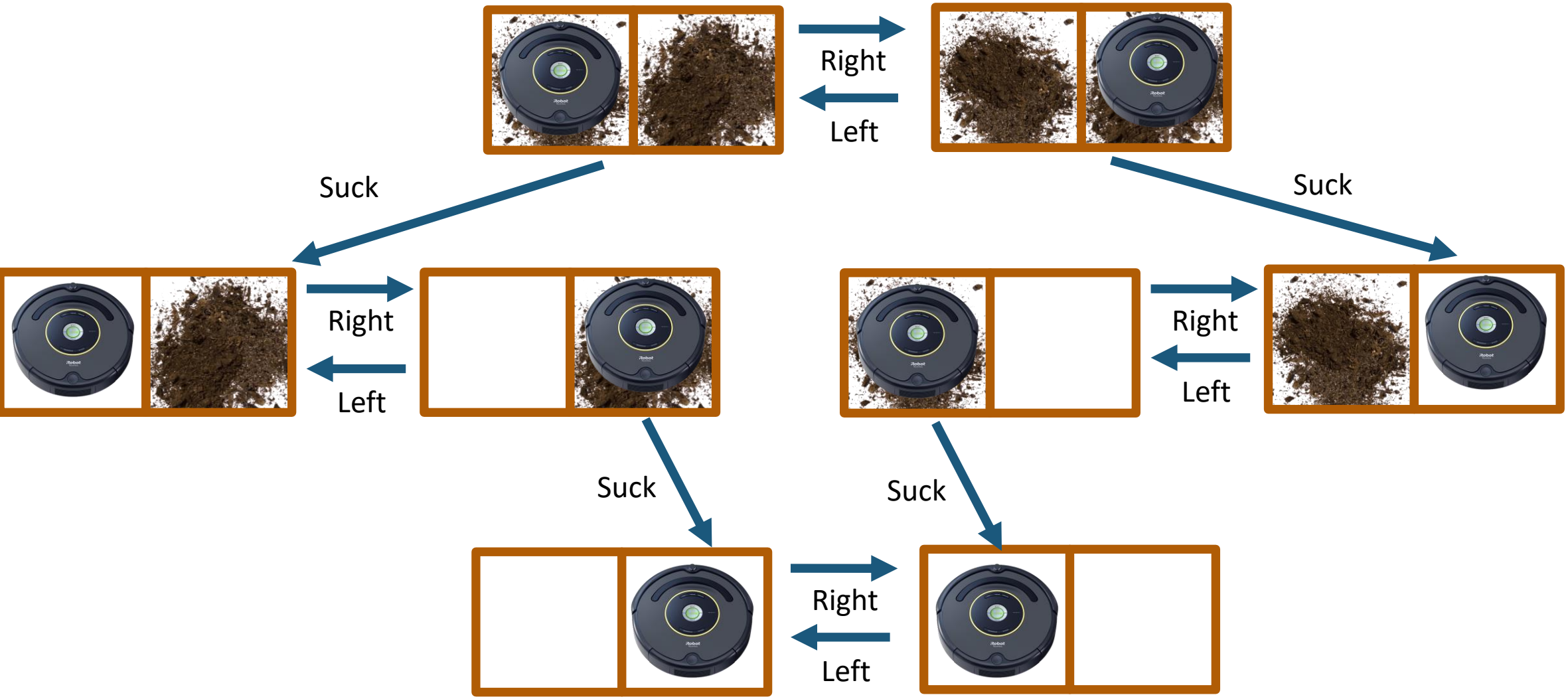
Actions: Anything the agent can do to affect the environment

- *Suck*
- *Move Left*
- *Move Right*
- *(Move Up)*
- *(Move Down)*

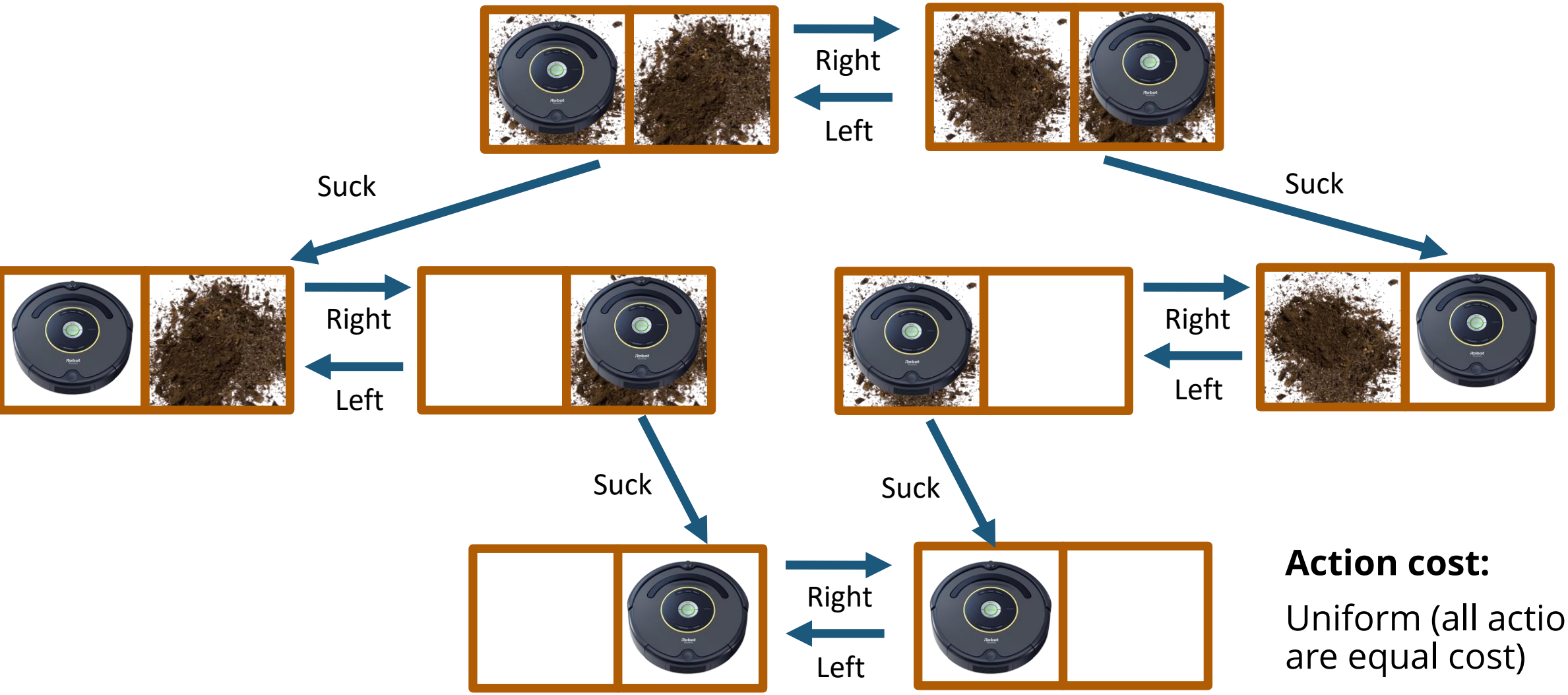
Transition: How actions affect what current state the world is in

- Suck – removes dirt
- Move – moves in that direction, unless agent hits a wall, in which case it stays put

Vacuum World Transition Model



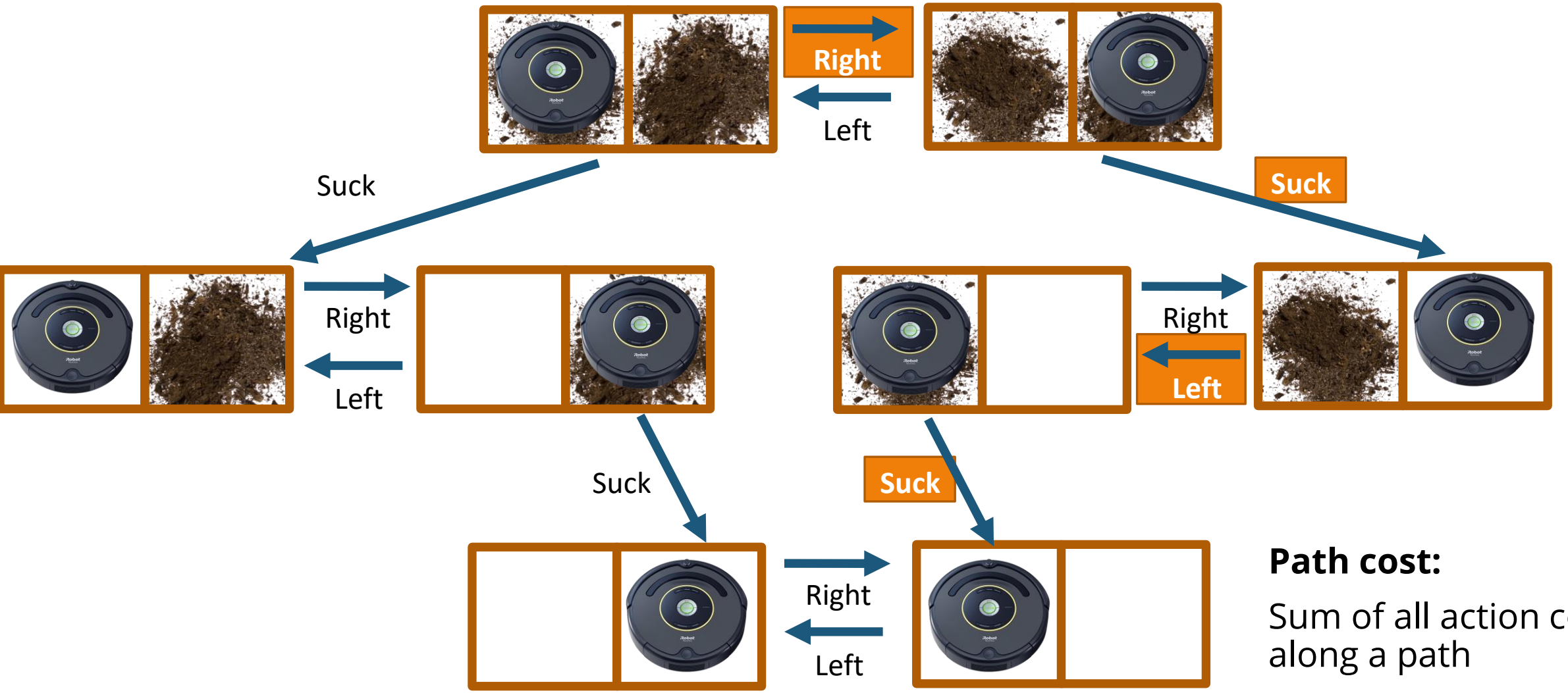
Vacuum World Transition Model



Action cost:

Uniform (all actions are equal cost)

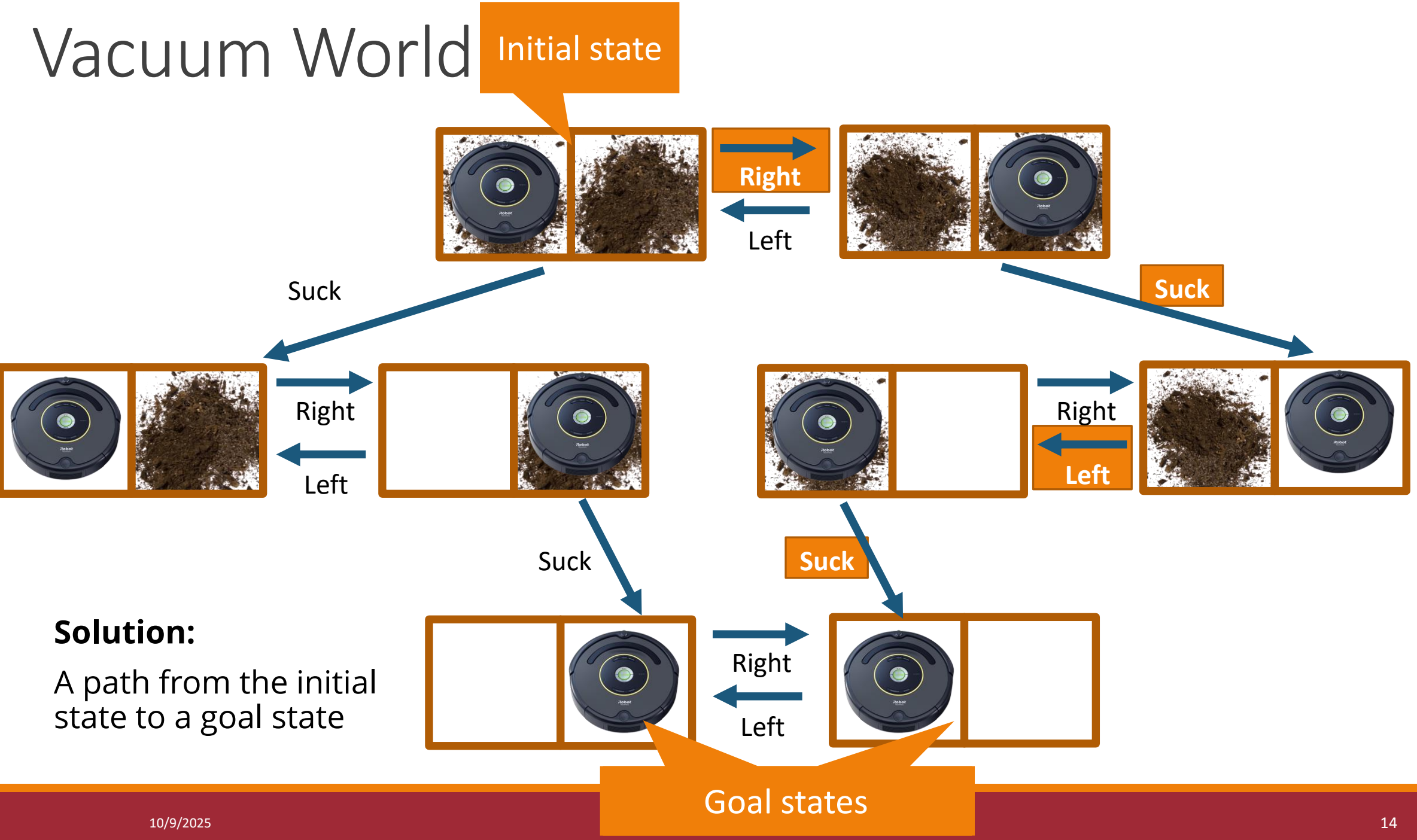
Vacuum World Transition Model



Path cost:

Sum of all action costs along a path

Vacuum World



Solution:

A path from the initial state to a goal state

Search Algorithms

Useful Concepts

State space: the set of all states reachable from the initial state by *any* sequence of actions

- When several operators can apply to each state, this gets large very quickly
- Might be a proper subset of the set of configurations

Path: a sequence of actions leading from one state s_j to another state s_k

Solution: a path from the initial state s_i to a state s_f that satisfies the goal test

Search tree: a way of representing the paths that a search algorithm has explored. The root is the initial state, leaves of the tree are successor states.

Frontier: those states that are available for *expanding* (for applying legal actions to)

Solutions and *Optimal* Solutions

A **solution** is a sequence of actions from the initial state to a goal state.

Optimal Solution: A solution is optimal if no solution has a lower path cost.

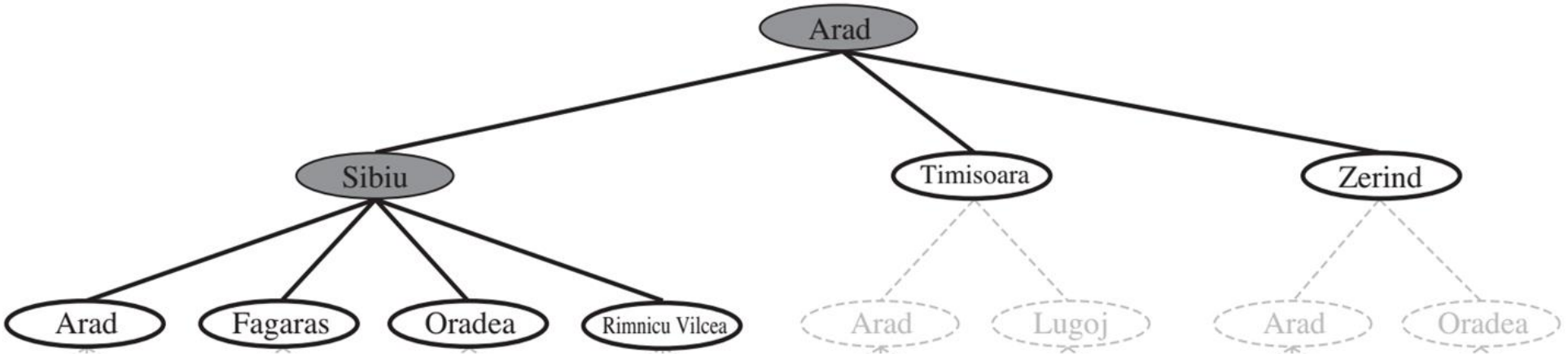
Basic search algorithms: Tree Search

Generalized algorithm to solve search problems

Enumerate in some order all possible paths from the initial state

- Here: search through ***explicit tree generation***
  - ROOT= initial state
  - Nodes in search tree generated through transition model
  - Tree search treats different paths to the same node as distinct

Generalized tree search



function TREE-SEARCH(*problem, strategy*) return a solution or failure

Initialize frontier to the *initial state* of the *problem*

do

if the frontier is empty then return *failure*

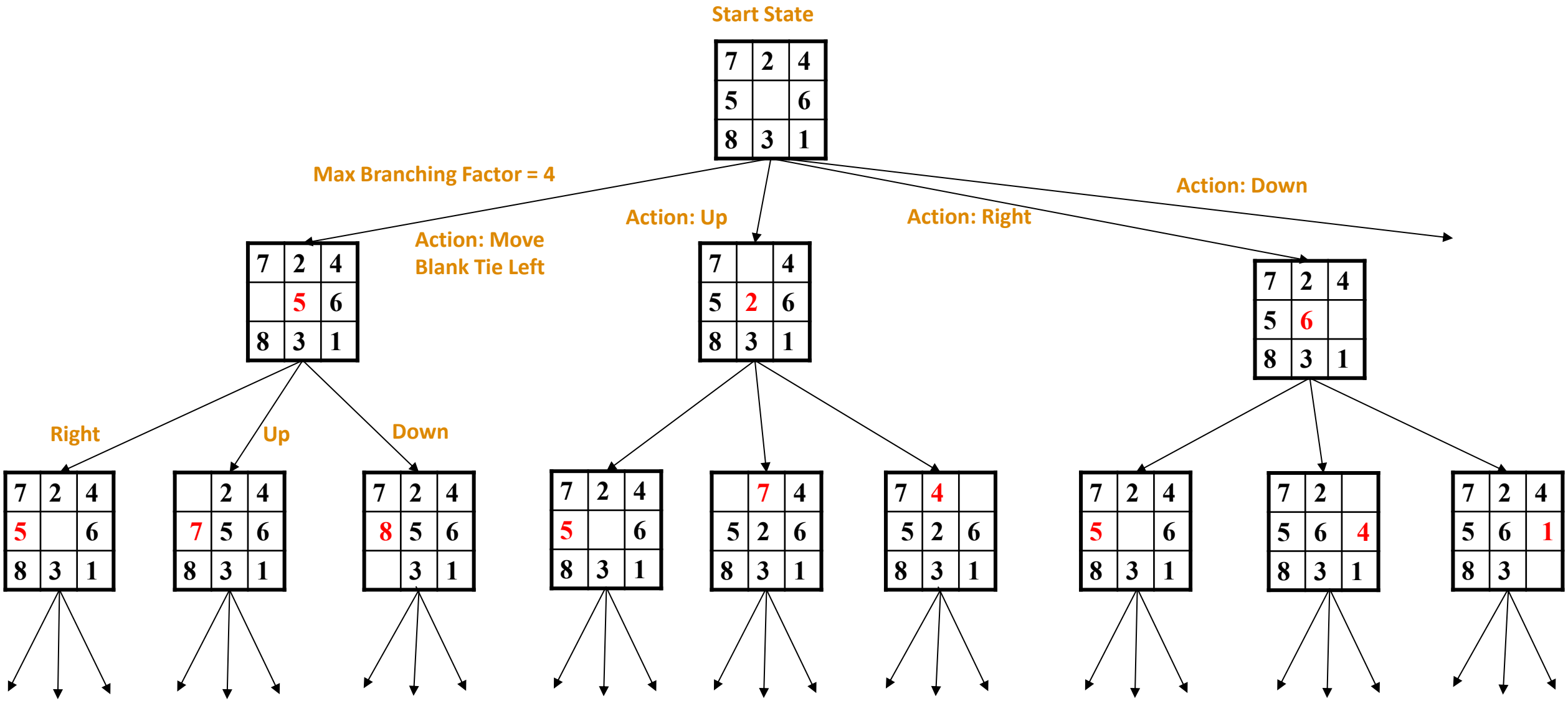
choose leaf node for expansion according to strategy & remove from frontier

if node contains goal state then return *solution*

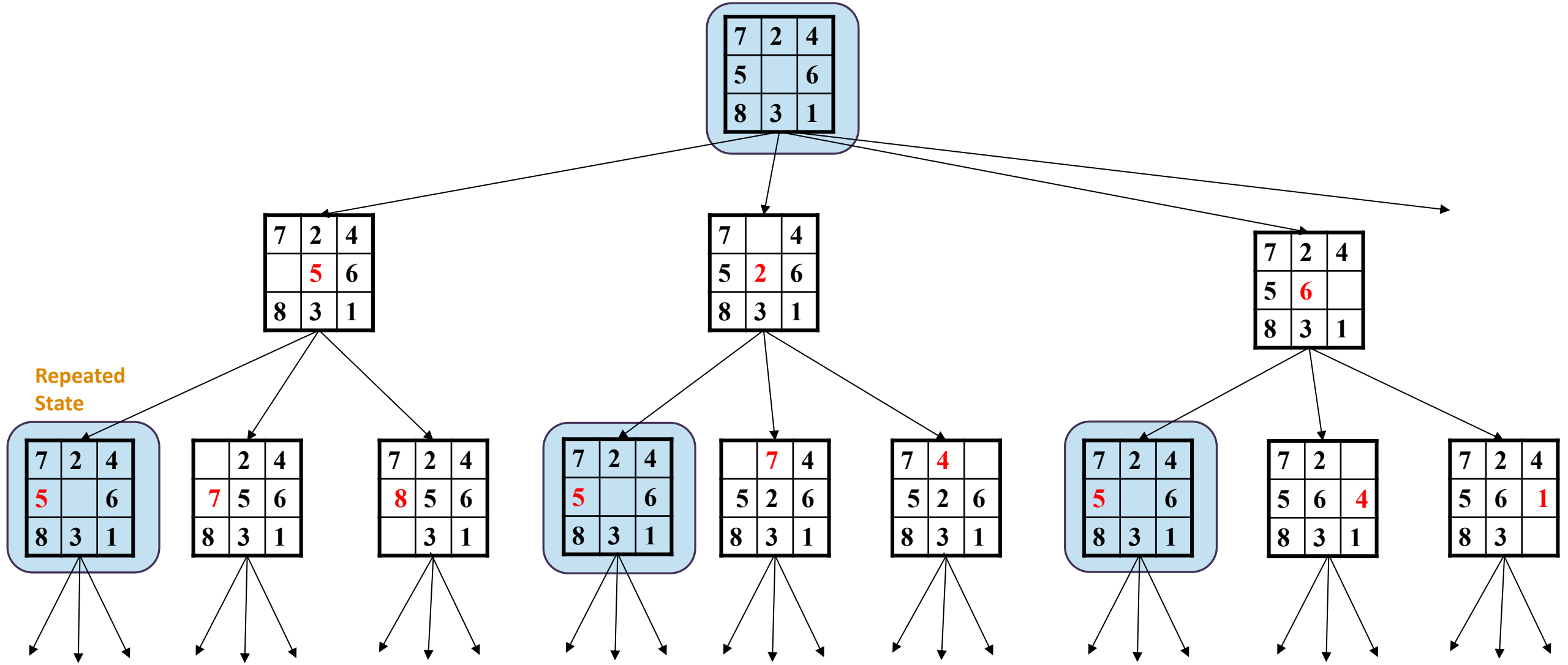
else expand the node and add resulting nodes to the frontier

The strategy determines search process!

8-Puzzle *Search Tree*



8-Puzzle *Search Tree*



Graph Search vs Tree Search

function TREE-SEARCH(*problem*) **returns** a solution, or failure
initialize the frontier using the initial state of *problem*
loop do
if the frontier is empty **then return** failure
choose a leaf nose and remove it from the frontier
if the node contains a goal state **then return** the corresponding solution
expand the chosen node, adding the resulting nodes to the frontier

function GRAPH-SEARCH(*problem*) returns a solution, or failure
initialize the frontier using the initial state of *problem*
initialize the explored set to be empty
loop do
if the frontier is empty **then return** failure
choose a leaf node and remove it from the frontier
if the node contains a goal state **then return** the corresponding solution
add node to the explored set
expand the chosen node, adding the resulting nodes to the frontier
only if not in the frontier of explored set

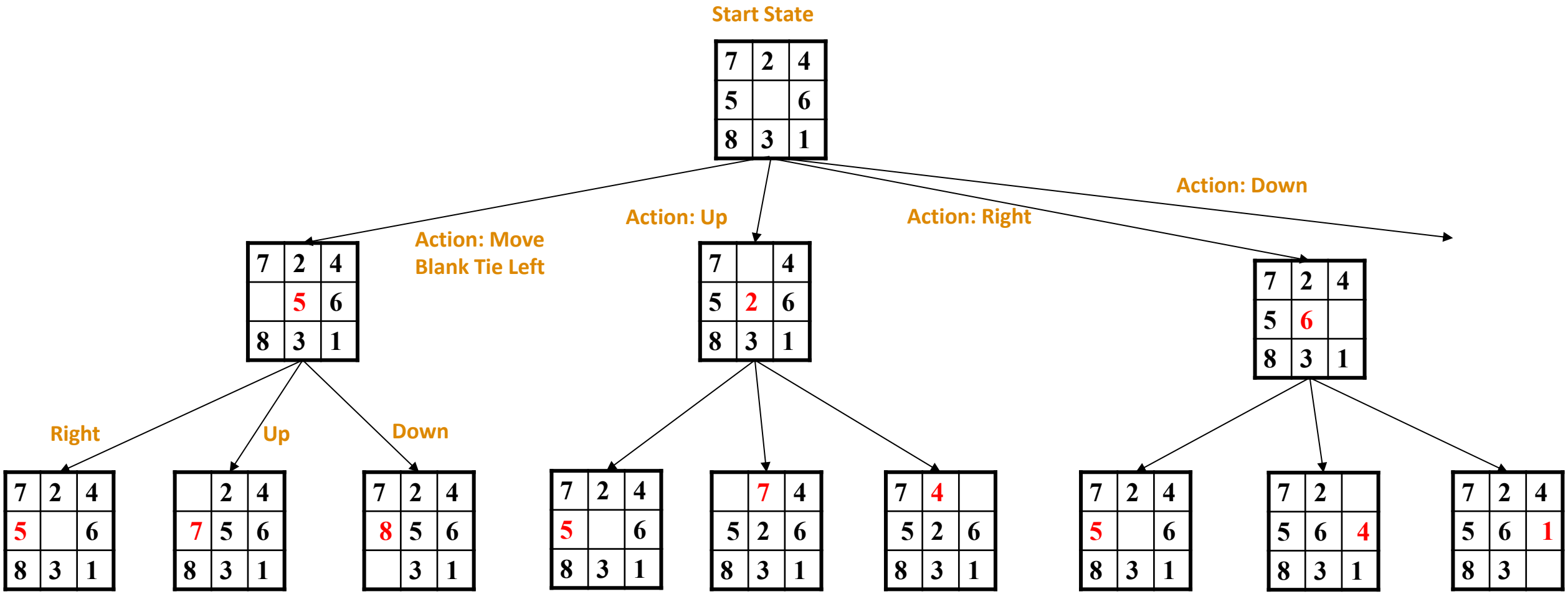
Search Strategies

Several classic search algorithms differ only by the order of how they expand their search trees

You can implement them by using different queue data structures

- **Depth-first search** = LIFO queue
- **Breadth-first search** = FIFO queue
- **Greedy best-first search** or **A* search** = Priority queue

8-Puzzle Breadth-first Search



Search Algorithms

Dimensions for evaluation

- **Completeness** - always find the solution?
- **Optimality** - finds a least cost solution (lowest path cost) first?
- **Time complexity** - # of nodes generated *(worst case)*
- **Space complexity** - # of nodes simultaneously in memory *(worst case)*

Time/space complexity variables

- *b*, maximum branching factor of search tree
- *d*, depth of the shallowest goal node
- *m*, maximum length of any path in the state space (potentially ∞)

Properties of Breadth-First Search (BFS)

Complete?	Yes (if b is finite)
Optimal?	Yes, if cost = 1 per step (not optimal in general)
Time Complexity?	$1+b+b^2+b^3+\ldots+b^d = O(b^d)$
Space Complexity?	$O(b^d)$ (keeps every node in memory)

Time/space complexity variables

- b, maximum branching factor of search tree
- d, depth of the shallowest goal node
- m, maximum length of any path in the state space (potentially ∞)

BFS versus DFS

Breadth-first

- ☑ Complete,
- ☑ Optimal
- ☒ *but* uses O(b^d) space

Depth-first

- ☒ Not complete unless m is bounded
- ☒ Not optimal
- ☒ Uses O(b^m) time; terrible if m >> d
- ☑ but only uses O($b*m$) space

Time/space complexity variables

b, *maximum branching factor* of search tree

d, *depth* of the shallowest goal node

m, *maximum length* of any path in the state space (potentially ∞)

Exponential Space (and time) Is Not Good...

- Exponential complexity uninformed search problems *cannot* be solved for any but the smallest instances.
- *(Memory* requirements are a bigger problem than *execution* time.)

DEPTH	NODES	TIME	MEMORY
2	**110**	**0.11 milliseconds**	**107 kilobytes**
4	**11110**	**11 milliseconds**	**10.6 megabytes**
6	**10^6**	**1.1 seconds**	**1 gigabytes**
8	**10^8**	**2 minutes**	**103 gigabytes**
10	**10^{10}**	**3 hours**	**10 terabytes**
12	**10^{12}**	**13 days**	**1 petabytes**
14	**10^{14}**	**3.5 years**	**99 petabytles**

Assumes b=10, 1M nodes/sec, 1000 bytes/node

Action Castle

Art: Formulating a Search Problem

Decide:

Which properties matter & how to represent
- *Initial State, Goal State, Possible Intermediate States*

Which actions are possible & how to represent
- *Operator Set: Actions and Transition Model*

Which action is next
- *Path Cost Function*

Formulation greatly affects combinatorics of search space and therefore speed of search

Action Castle Map Navigation

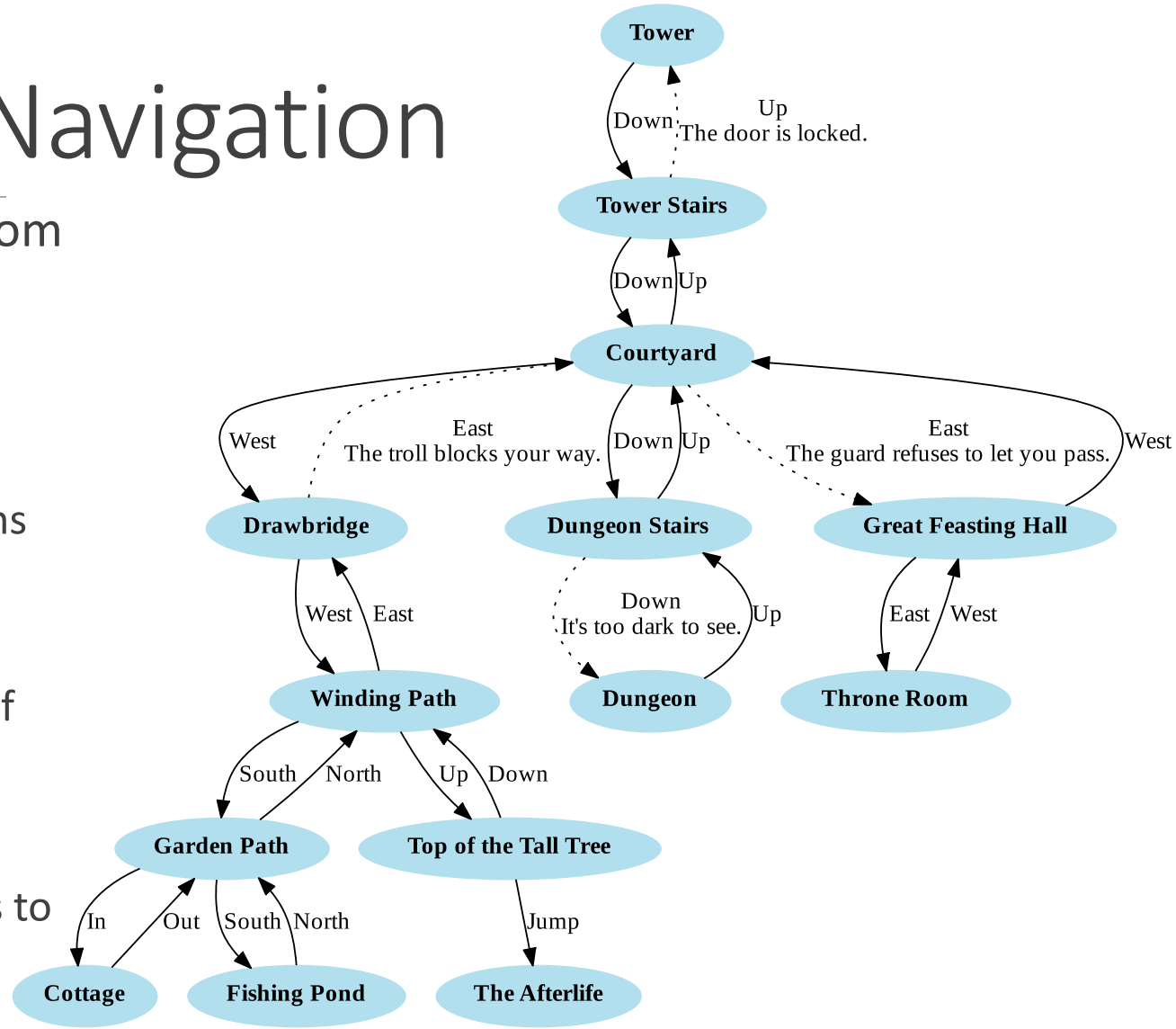
Let's consider the sub-task of navigating from one location to another.

Formulate the ***search problem***

- States: locations in the game
- Actions: move between connected locations
- Goal: move to a particular location like the **Throne Room**
- Performance measure: minimize number of moves to arrive at the goal

Find a ***solution***

- Algorithm that returns sequence of actions to get from the start sate to the goal.



Action Castle

Let's consider the full game.

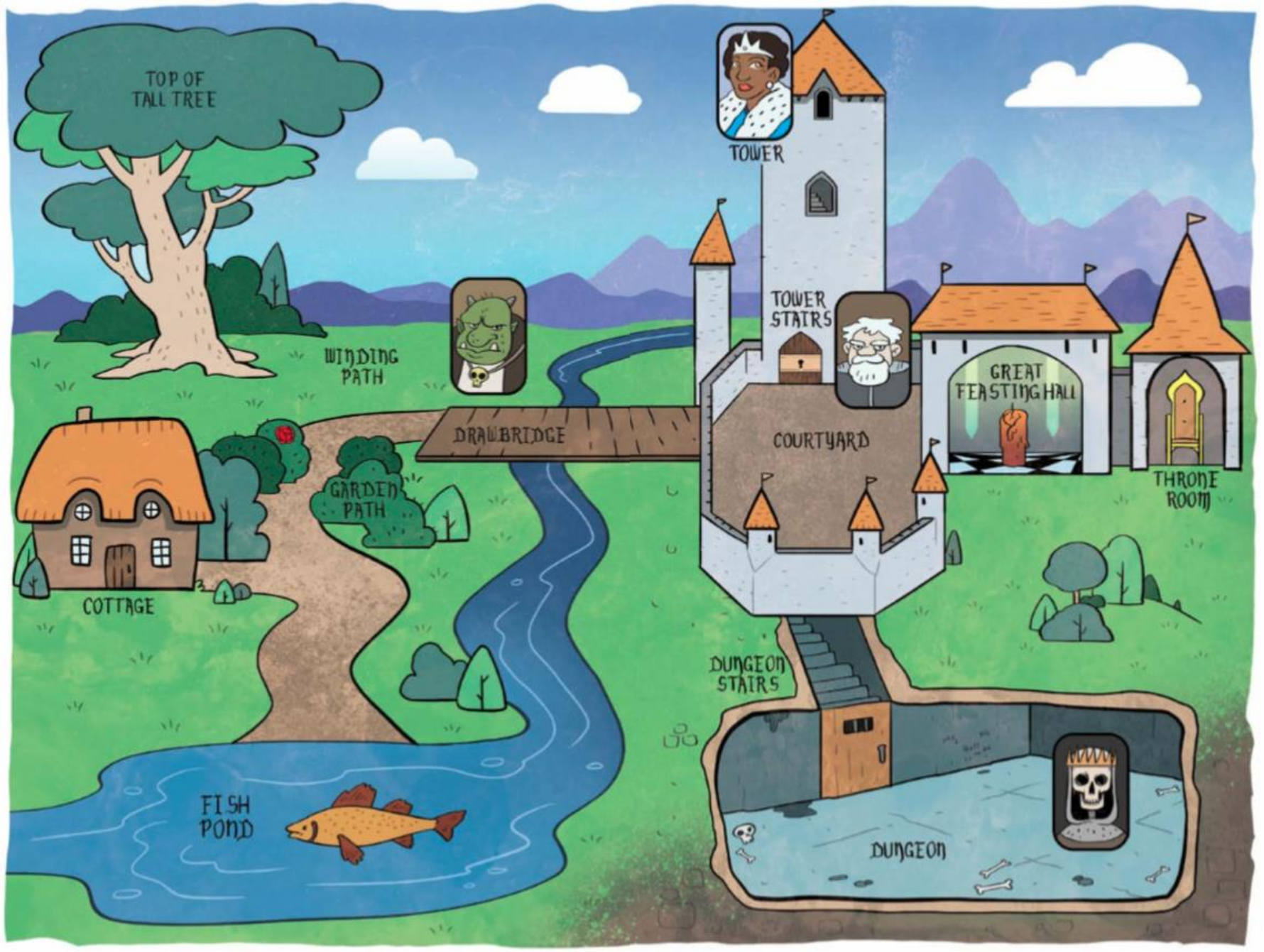
Actions

Start State

Transitions

State Space

Goal test



Actions

Go

- Move to a location

Get

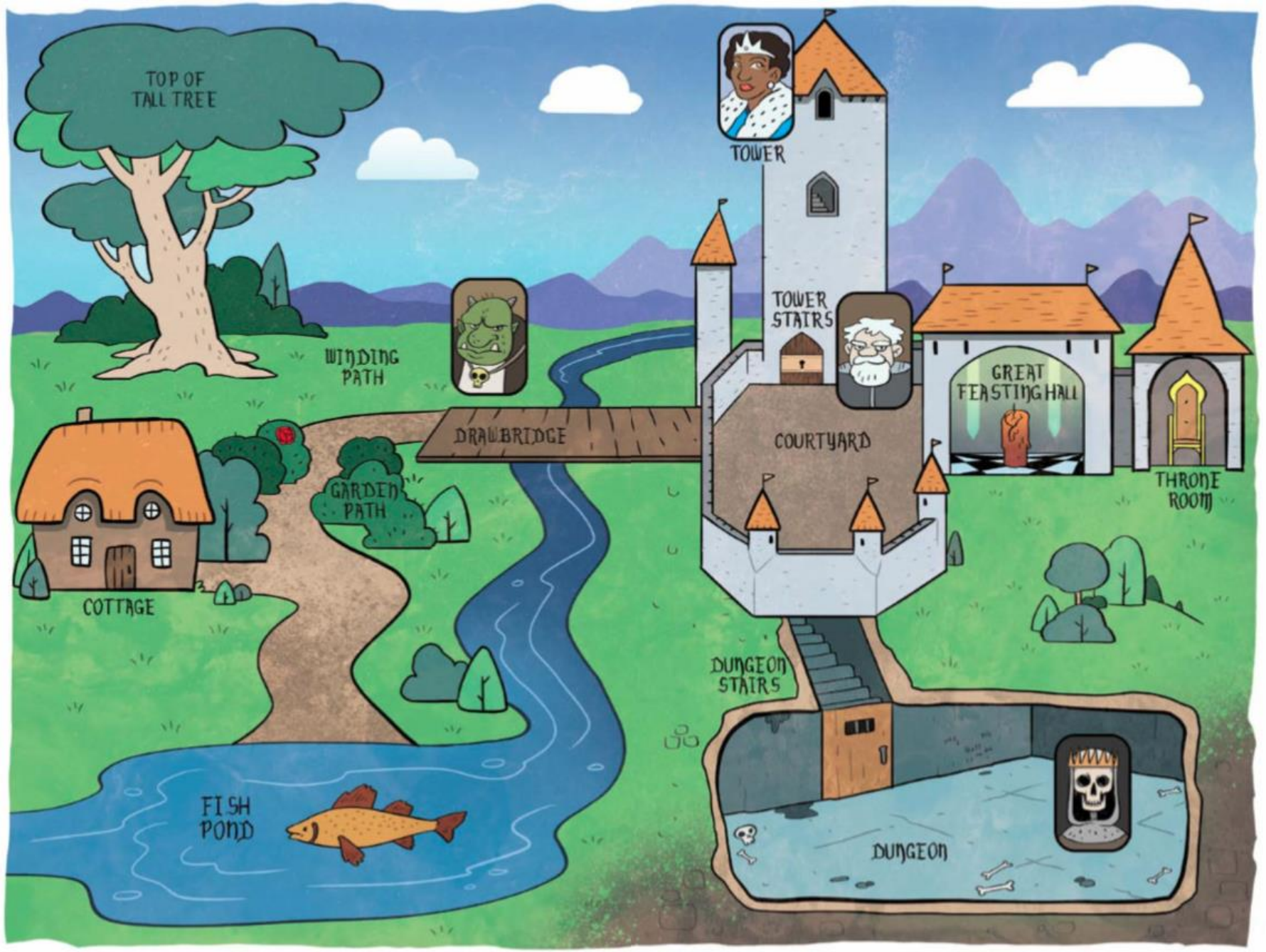
- Add an item to inventory

Special

- Perform a special action with an item like “Catch fish with pole”

~~**Drop**~~

- ~~Leave an item in current location~~



State Info

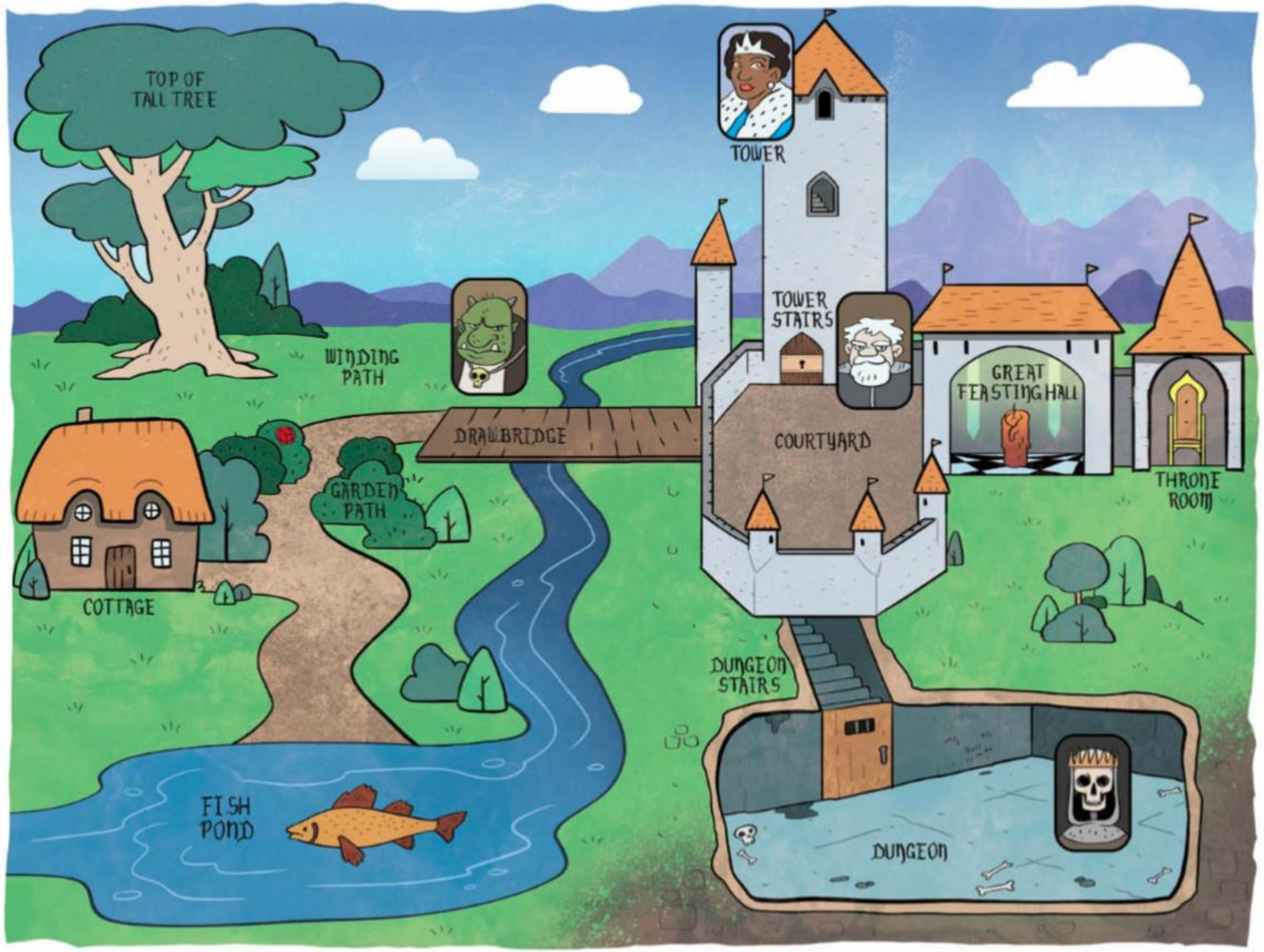
Location of Player

Items in their inventory

Location of all items / NPCs

Blocks like

- Troll guarding bridge,
- Locked door to tower,
- Guard barring entry to castle



In-Class Activity

https://laramartin.net/interactive-fiction-class/in_class_activities/search/action-castle-search.html

BFS has been implemented for you. You will be defining the action space, the state space, and the goal test.

This knowledge check is worth 2 points since it's bigger.

```python
def BFS(game, goal_conditions):
    command_sequence = []
    if goal_test(game, goal_conditions): return c

    frontier = queue.Queue()
    frontier.put((game, command_sequence))

    visited = dict()
    visited[get_state(game)] = True

    while not frontier.empty():
        (current_game, command_sequence) = frontier.get()
        current_state = get_state(current_game)
        parser = Parser(current_game)
        available_actions = get_available_actions(current_game)

        for command in available_actions:
            # Clone the current game with its state
            new_game = copy.deepcopy(current_game)
            # Apply the command to it to get the resulting state
            parser = Parser(new_game)
            parser.parse_command(command)
            new_state = get_state(new_game)
            # Update the sequence of actions that we took to get to the resulting state
            new_command_sequence = copy.copy(command_sequence)
            new_command_sequence.append(command)
            if not new_state in visited:
                visited[new_state] = True
                if goal_test(new_game, goal_conditions):
                    frontier.put((new_game, new_command_sequence))
    # Return None to indicate there is no solution.
    return None
```

The frontier tracks order of unexpanded search nodes. Here we're using a FIFO queue

The visited dictionary prevents us from revising states.

TODO: implement get_state()

get_available_actions() to return all commands that could be used here.

TODO: implement get_available_actions()

The parser can execute this command to get the resulting state.

Check to see if this state satisfies the goal test, if so, return the command sequence that got us here.

TODO: implement goal_test()

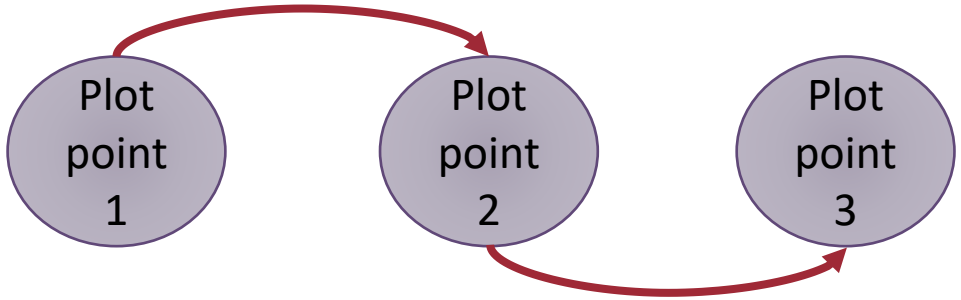
```python
def BFS(game, goal_conditions):
  command_sequence = []
  if goal_test(game, goal_conditions): return command_sequence

  frontier = queue.Queue()
  frontier.put((game, command_sequence))

  visited = dict()
  visited[get_state(game)] = True

  while not frontier.empty():
    (current_game, command_sequence) = frontier.get
    current_state = get_state(current_game)
    parser = Parser(current_game)
    available_actions = get_available_actions(current_game)

    for command in available_actions:
      # Clone the current game with its state
      new_game = copy.deepcopy(current_game)
      # Apply the command to it to get the resulting state
      parser = Parser(new_game)
      parser.parse_command(command)
      new_state = get_state(new_game)
      # Update the sequence of actions that we took to get to the resulting state
      new_command_sequence = copy.copy(command_sequence)
      new_command_sequence.append(command)
      if not new_state in visited:
        visited[new_state] = True
        if goal_test(new_game, goal_conditions):
        frontier.put((new_game, new_command_sequence))
  # Return None to indicate there is no solution.
  return None
```

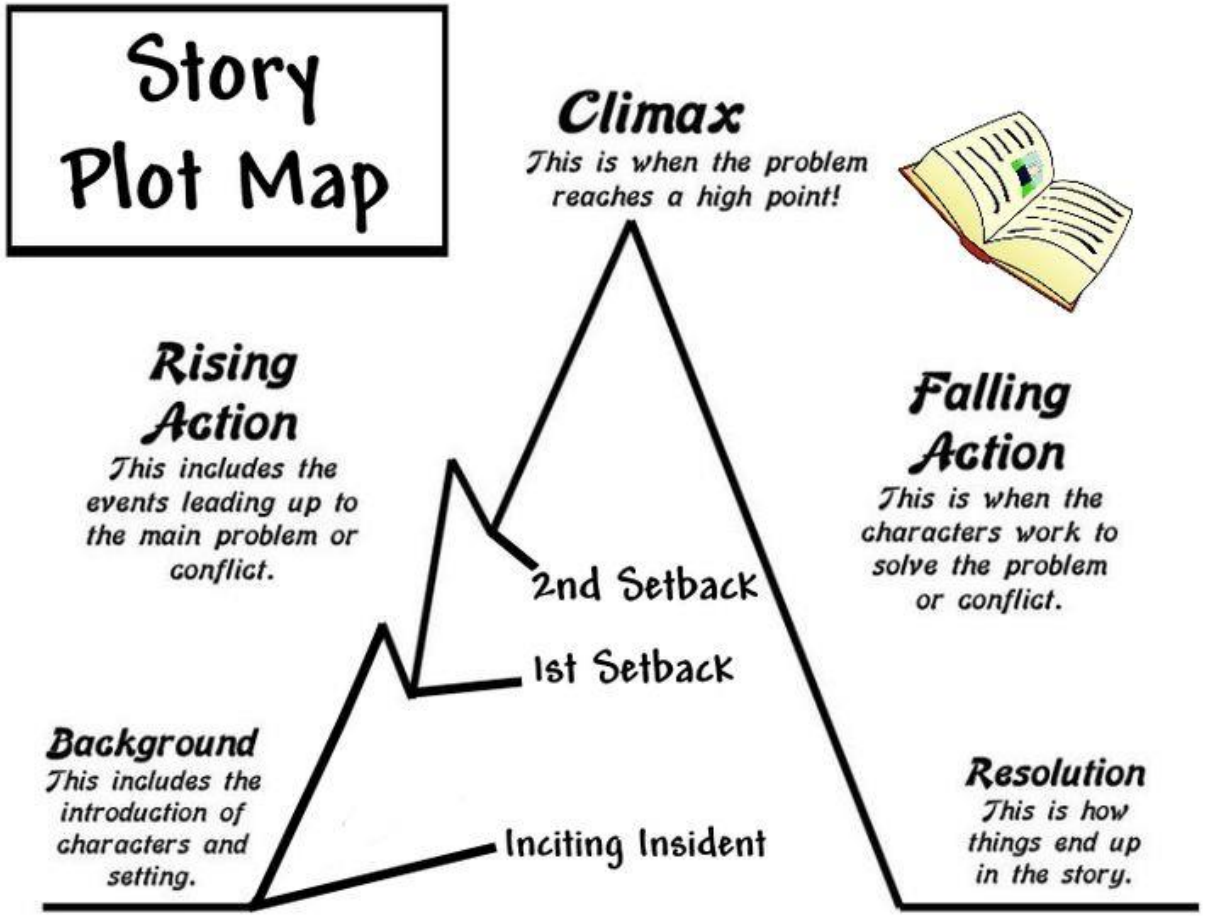
To be used, a key in the dictionary get_state() must return an immutable object

Review:
Scripts, Procedures, and Plots…oh my!

Schank & Abelson believe that everyone has scripts in their heads built from common experiences

Authors often plan out **plots** before they write stories





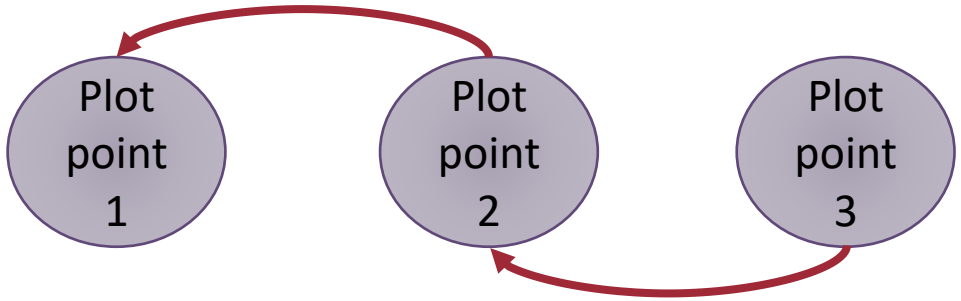
https://i.pinimg.com/736x/57/f7/03/57f703afc709080bddc2c3cfed8dd061.jpg

Review: Scripts, Procedures, and Plots…oh my!

Schank & Abelson believe that everyone has scripts in their heads built from common experiences

Authors often plan out plots before they write stories

Stories that aren't planned out either have to "**reincorporate**"[1] ideas or the stories feel unfinished



[1] The idea of *reincorporation* is explored in the book Impro by Keith Johnstone

Review: Ways of Extracting Plot Points

Most salient keywords

Event representations

Verb-Noun Sets

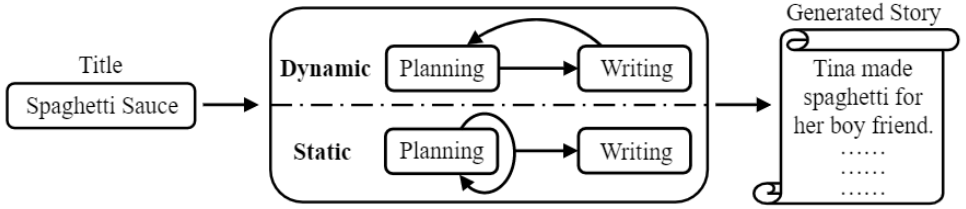
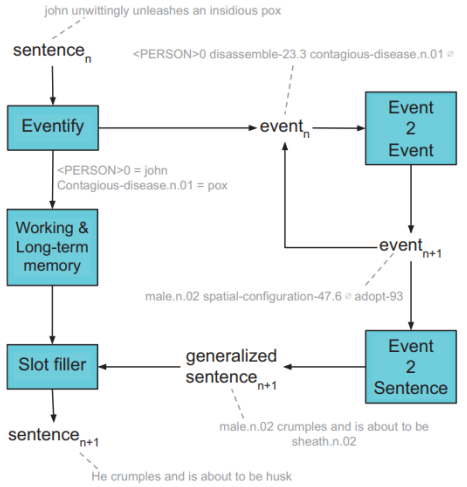
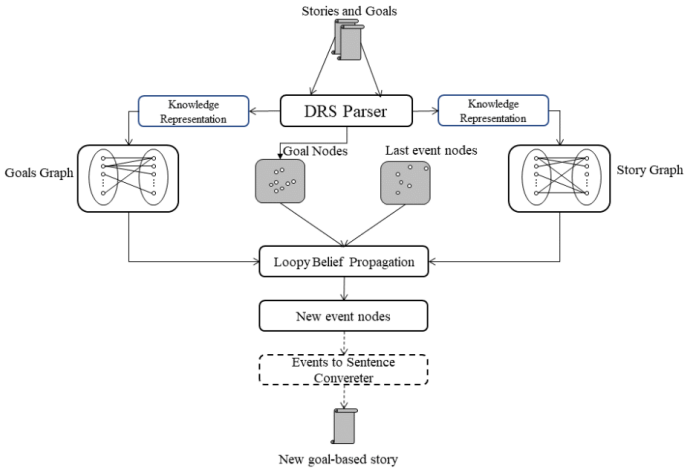


Figure 1: An overview of our system.





Review: Generating with Plot Points

Co-generated vs conditioned (prompted) with plot

Generate event & then translate to natural language

Graph algorithms (Loopy Belief Propagation)

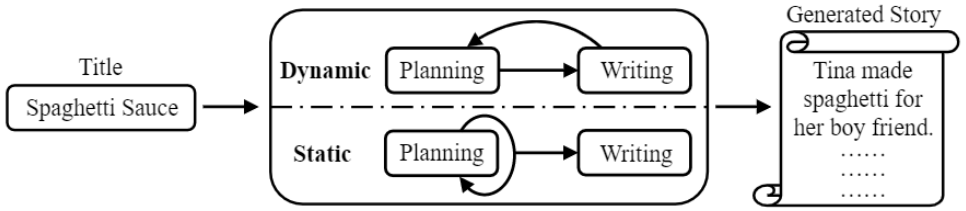
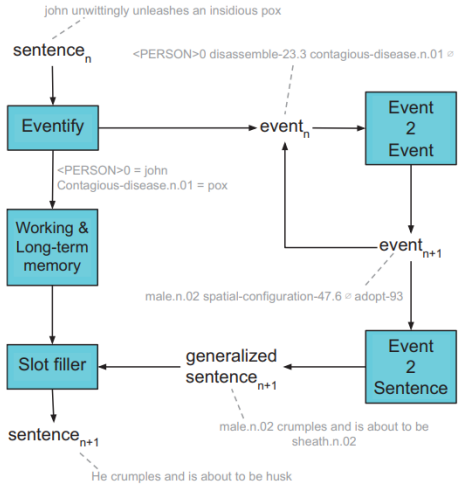
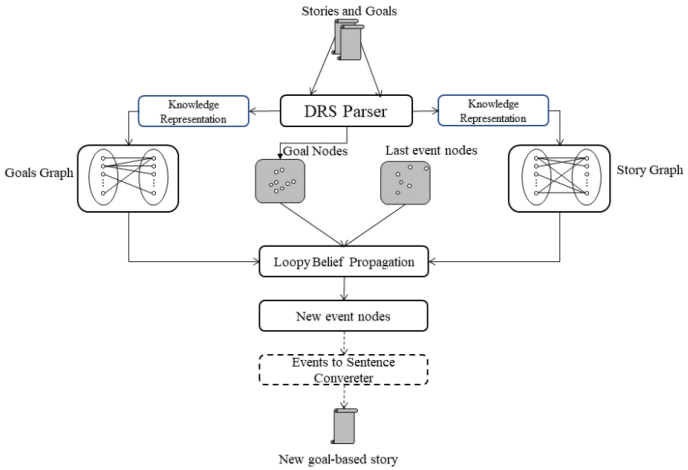


Figure 1: An overview of our system.





The Story Cloze Test

What is a Cloze Test?

- Something is removed from a text; try to guess what's missing
- Used for reading comprehension, grammar, etc. (with humans)

Narrative Cloze Test

Evaluate “event relatedness”

Find which events could be missing from a narrative chain

Uses verbs only

N. Chambers and D. Jurafsky, “Unsupervised Learning of Narrative Event Chains,” in *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, 2008, pp. 789–797, doi: 10.1.1.143.1555.

Narrative Cloze Test

Known events:
(pleaded subj), (admits subj), (convicted obj)

Likely Events:

sentenced obj	0.89	indicted obj	0.74
paroled obj	0.76	fined obj	0.73
fired obj	0.75	denied subj	0.73

Figure 1: Three narrative events and the six most likely events to include in the same chain.

X pleaded _

X admits _

_ convicted X

N. Chambers and D. Jurafsky, "Unsupervised Learning of Narrative Event Chains," in *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, 2008, pp. 789–797, doi: 10.1.1.143.1555.

Finish the story

Gina was worried the cookie dough in the tube would be gross.

She was very happy to find she was wrong.

The cookies from the tube were as good as from scratch.

Gina intended to only eat 2 cookies and save the rest.

A. Gina liked the cookies so much she ate them all in one sitting. 

B. Gina gave the cookies away at her church.

Mostafazadeh, N., Chambers, N., He, X., Parikh, D., Batra, D., Vanderwende, L., Kohli, P., & Allen, J. (2016). A Corpus and Cloze Evaluation for Deeper Understanding of Commonsense Stories. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 839–849. http://www.aclweb.org/anthology/N16-1098

Story Cloze Test

Predict/select the most likely story *ending*

- Given the first 4 sentences of the story

Full sentences

Multiple choice evaluation

Mostafazadeh, N., Chambers, N., He, X., Parikh, D., Batra, D., Vanderwende, L., Kohli, P., & Allen, J. (2016). A Corpus and Cloze Evaluation for Deeper Understanding of Commonsense Stories. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 839–849. http://www.aclweb.org/anthology/N16-1098

An RNN-based Binary Classifier for the Story Cloze Test

Melissa Roemmele
Institute for Creative Technologies
University of Southern California
roemmele@ict.usc.edu

Sosuke Kobayashi*
Preferred Networks, Inc.
sosk@preferred.jp

Naoya Inoue
Tohoku University
naoya-i@ecei.tohoku.ac.jp

Andrew M. Gordon
Institute for Creative Technologies
University of Southern California
gordon@ict.usc.edu

Toward Better Storylines with Sentence-Level Language Models

Daphne Ippolito*
daphnei@seas.upenn.edu

David Grangier
grangier@google.com

Douglas Eck
deck@google.com

Chris Callison-Burch
ccb@seas.upenn.edu

Abstract

We propose a sentence-level language model which selects the next sentence in a story from a finite set of fluent alternatives. Since it does not need to model fluency, the sentence-level language model can focus on longer range dependencies, which are crucial for multi-sentence coherence. Rather than dealing with individual words, our method treats the story so far as a list of pre-trained sentence embeddings and predicts an embedding for the next sentence, which is more efficient than predicting word embeddings. Notably this allows us to consider a large number of candidates for the next sentence during training. We demonstrate the effectiveness of our approach with state-of-the-art accuracy on the unsupervised Story Cloze task and with promising results on larger-scale next sentence prediction tasks.

Poster Presentation CIKM '20, October 19–23, 2020, Virtual Event, Ireland

Enhanced Story Representation by ConceptNet for Predicting Story Endings

Shanshan Huang
huangss_33@sjtu.edu.cn
Shanghai Jiao Tong University

Kenny Q. Zhu*
kzhu@cs.sjtu.edu.cn
Shanghai Jiao Tong University

Qianzi Liao
liaoqz@sjtu.edu.cn
Shanghai Jiao Tong University

Libin Shen
libin@leyantech.com
Leyan Tech

Yinggong Zhao
ygzhao@leyantech.com
Leyan Tech

ABSTRACT

Predicting endings f
machine commonsen
resentation of the sto
Pre-trained language
in this task by explo
dataset, instead of "u
we propose to improv
fying the sentences t
latent relationship b
enhanced sentence re
guage models, makes
the popular Story Clo
data.

CCS CONCEPTS

IEEE/ACM TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING, VOL. 27, NO. 4, APRIL 2019 719

Story Ending Selection by Finding Hints From Pairwise Candidate Endings

Mantong Zhou, Minlie Huang, and Xiaoyan Zhu

Fig. 1. **Evidence bias issue**: both a wrong ending (in red) and a correct ending (in green) can obtain sufficient evidence from the story context.

important linkages between a story context and a candidate ending. They suffer from the issue of **evidence bias**: both the wrong and correct endings can obtain sufficient support from the story context. As illustrated in Fig. 1, the wrong ending (in red) and the correct ending (in green) can be supported by the red-colored evidence and the green-colored evidence in the story context, respectively. Thus, it is difficult for matching-based models to distinguish such cases. The situation is not rare because both correct and wrong endings are written to fit the world of a story

Tackling the Story Ending Biases in The Story Cloze Test

Rishi Sharma[1], **James F. Allen**[1,2], **Omid Bakhshandeh**[3], **Nasrin Mostafazadeh**[4*]
1 University of Rochester, 2 Institute for Human and Machine Cognition, 3 Verneek.ai 4 Elemental Cognition
rishi.sharma@rochester.edu, nasrinm@cs.rochester.edu

Abstract

The Story Cloze Test (SCT) is a recent framework for evaluating story comprehension and script learning. There have been a variety of models tackling the SCT so far. Although the original goal behind the SCT was to require systems to perform deep language understanding and commonsense reasoning for successful narrative understanding, some recent models could perform significantly better than the initial baselines by leveraging human-authorship biases discovered in the SCT dataset. In order to shed some this issue. This test evaluates a story comprehension system where the system is given a four-sentence short story as the 'context' and two alternative endings and to the story, labeled 'right ending' and 'wrong ending.' Then, the system's task is to choose the right ending. In order to support this task, Mostafazadeh et al. also provide the ROC Stories dataset, which is a collection of crowd-sourced complete five sentence stories through Amazon Mechanical Turk (MTurk). Each story follows a character through a fairly simple series of events to a conclusion.

Several shallow and neural models, including the state-of-the-art script learning approaches, were presented as baselines (Mostafazadeh et al.

Think Pair Share

The Story Cloze Test was created for evaluating systems' performance on understanding stories.

How could you use it instead for *generation*?

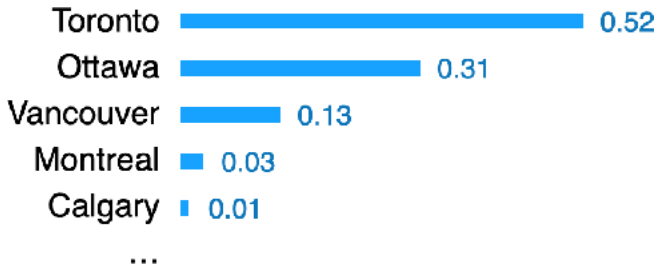
Retrieval-Augmented Generation

Retrieval-based language models (LMs)
Retrieval-based LMs = Retrieval + LMs

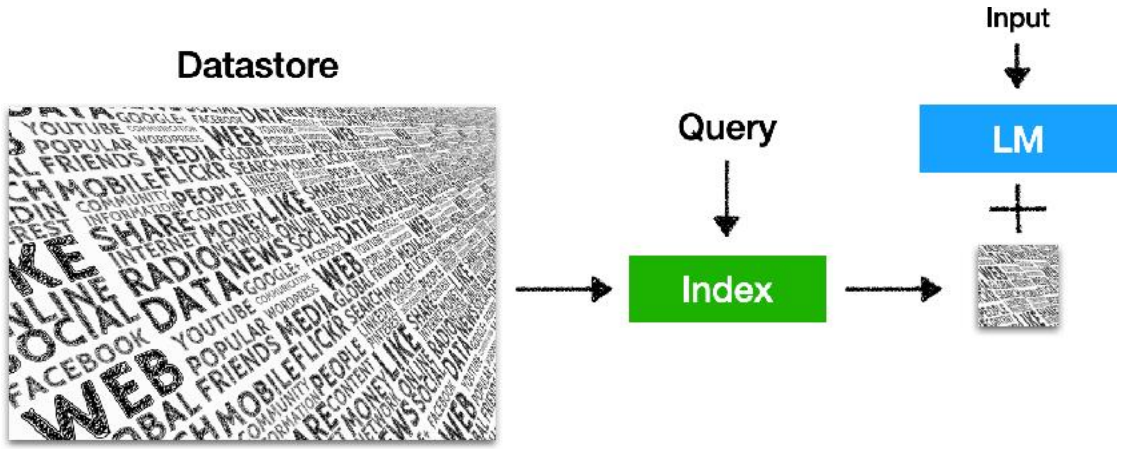
- It is a **language model** $P(x_n \mid x_1, x_2, \cdots, x_{n-1})$

The capital city of Ontario is ___

(can be broadly extended to masked language models or encoder-decoder models)



- It retrieves from an **external datastore** (at least during inference time)



Retrieval for knowledge-intensive NLP tasks

Representative tasks: open-domain QA, fact checking, entity linking, ..

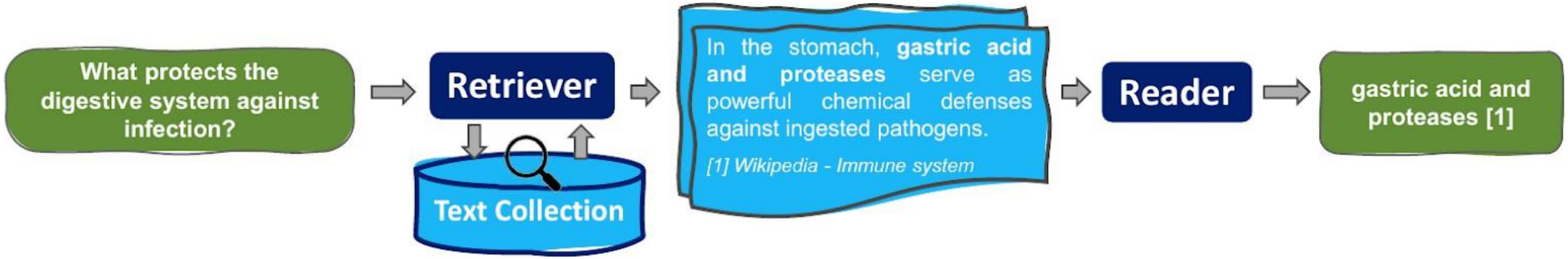


Image: http://ai.stanford.edu/blog/retrieval-based-NLP/

Drives a lot of research on better algorithms for **dense retrieval**, e.g., **DPR** (Karpukhin et al., 2020), **ColBERT** (Khattab and Zaharia, 2020), **ANCE** (Xiong et al., 2021), **Contriever** (Izacard et al., 2022), …

Why retrieval LMs?

LLMs can’t memorize all (long-tail) knowledge in their parameters

Why retrieval-based LMs?

List 5 important papers authored by Geoffrey Hinton

ChatGPT

Geoffrey Hinton is a renowned computer scientist … Here are five important papers authored by him:

1. **"Learning Internal Representations by Error Propagation"** (with **D. E. Rumelhart** and **R. J. Williams**) - This paper, published in **1986**, ..

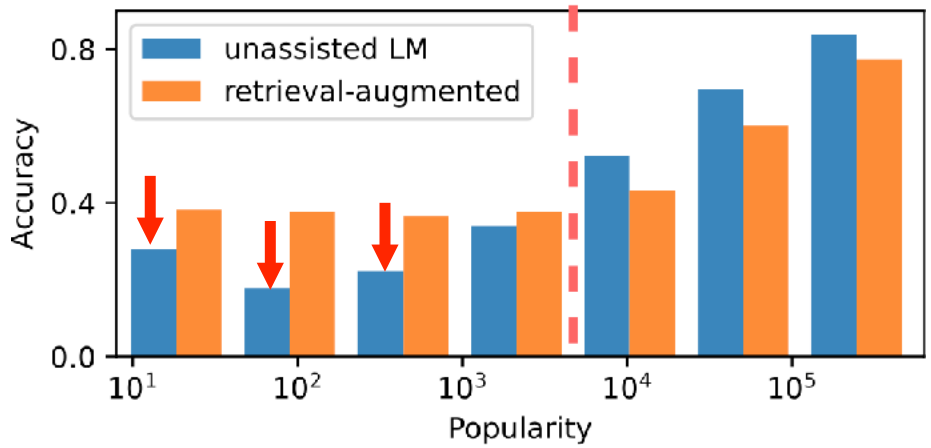
2. **"Deep Boltzmann Machines"** (with **R. Salakhutdinov**) - Published in **2009**, ..

…

4. **"Deep Learning"** (with Y. Bengio and A. Courville) - Published as a book in **2016**,…

5. **"Attention Is All You Need"** (with V. Vaswani, N. Shazeer, et al.) - Published in **2017**, this paper introduced the Transformer model,…

What is Kathy Saltzman’s occupation?



(Mallen et al., 2023)

GPT-3 davinci-003: 20%-30% accuracy

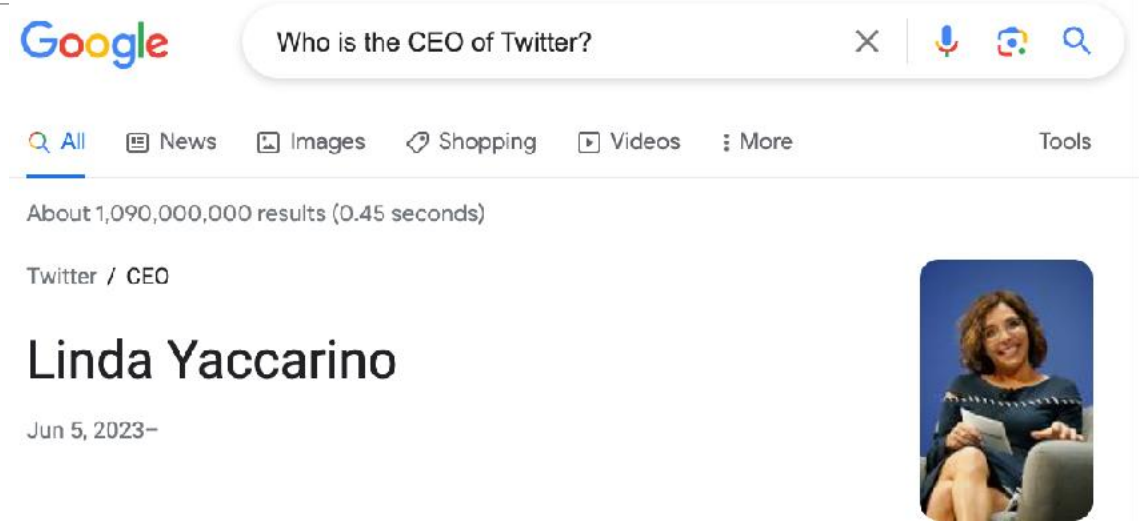
Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.

Why retrieval-based LMs?

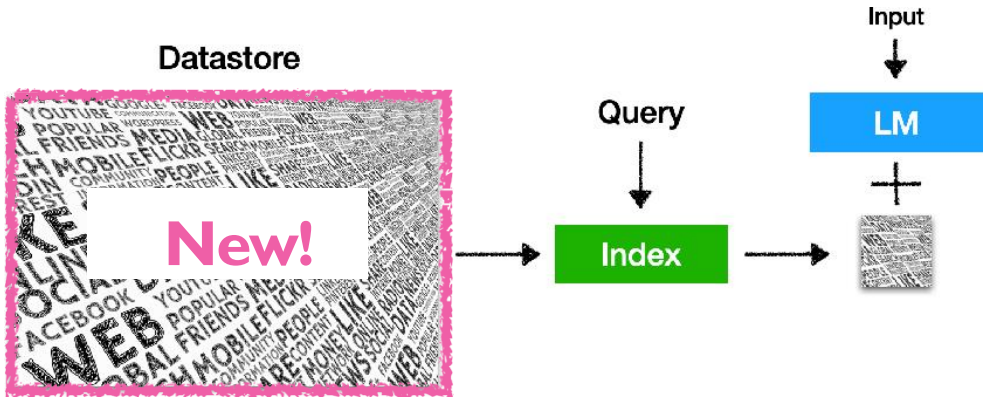
Who is the CEO of Twitter?

ChatGPT

As of my **knowledge cutoff in September 2021**, the CEO of Twitter is **Jack Dorsey**….



- Existing **knowledge editing** methods are still NOT scalable (active research!)
- The datastore can be easily **updated** and **expanded** - even without retraining!



Actually Google AI gets it wrong too…

AI Overview

There is no CEO of Twitter (now X); Elon Musk is still the owner, but he stepped down as CEO in July 2025, and Linda Yaccarino also stepped down from the position at the same time. Musk now serves as the company's owner, chairman, and CTO.

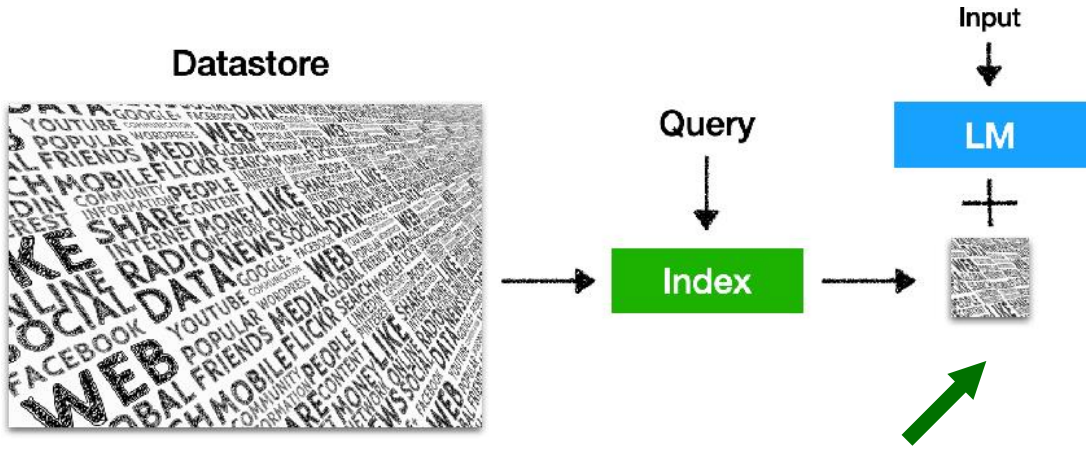
A Brief History of the CEO role

- **Elon Musk:** Musk was the CEO of X (formerly Twitter) immediately after acquiring the company in October 2022.
- **Linda Yaccarino:** Yaccarino replaced Musk as CEO on June 5, 2023, but she resigned in July 2025.
- **Elon Musk's Current Role:** Musk has remained the owner and is now serving as the chairman and chief technology officer for X.

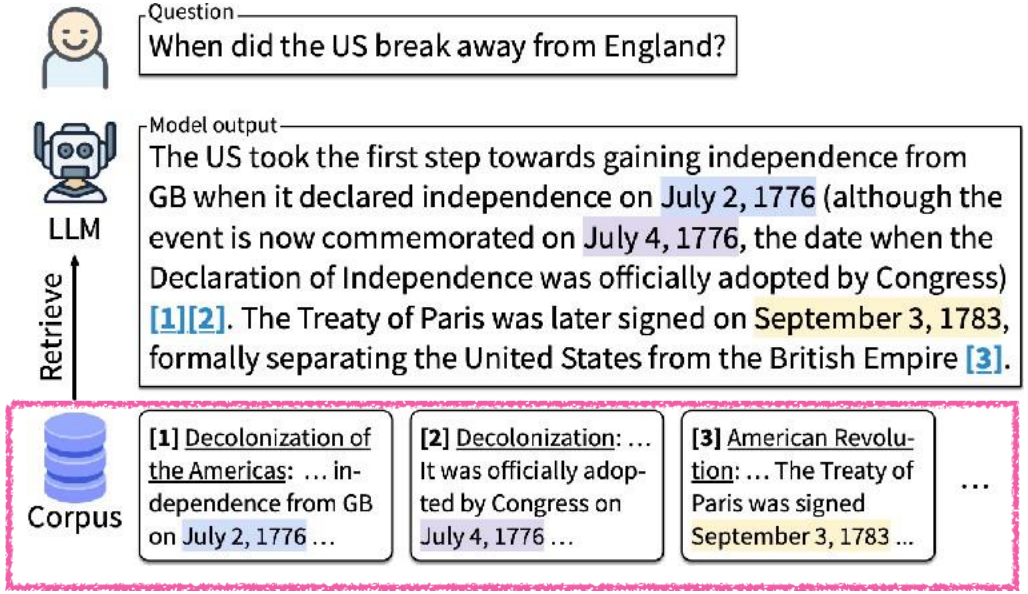
LLMs' output is challenging to interpret and verify

Why retrieval-based LMs?

Generating text with citations



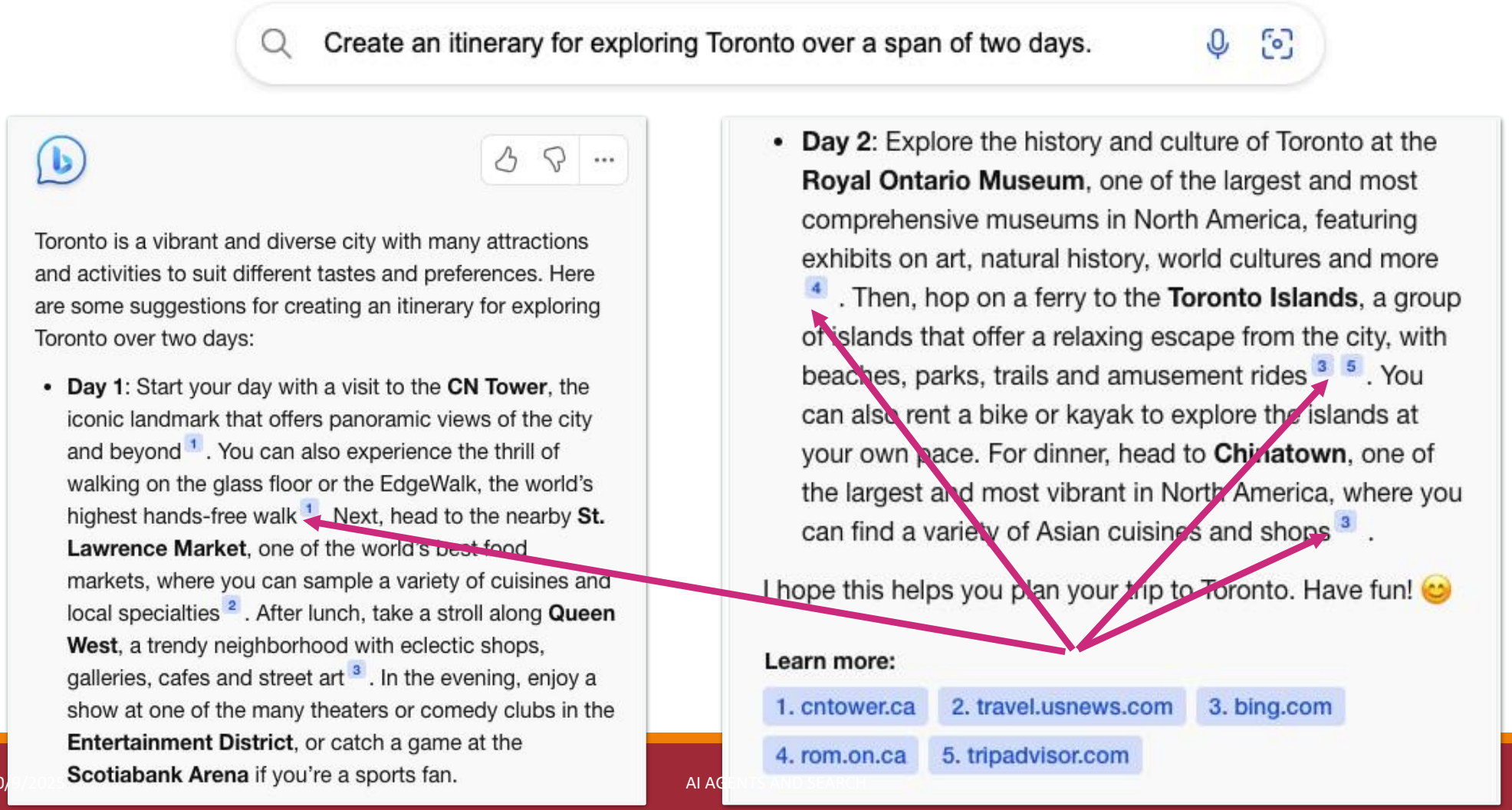
Can **trace knowledge source** from retrieval results - better **interpretability** & **control**



(Nakano et al. 2021; Menick et al., 2022; Gao et al., 2023)

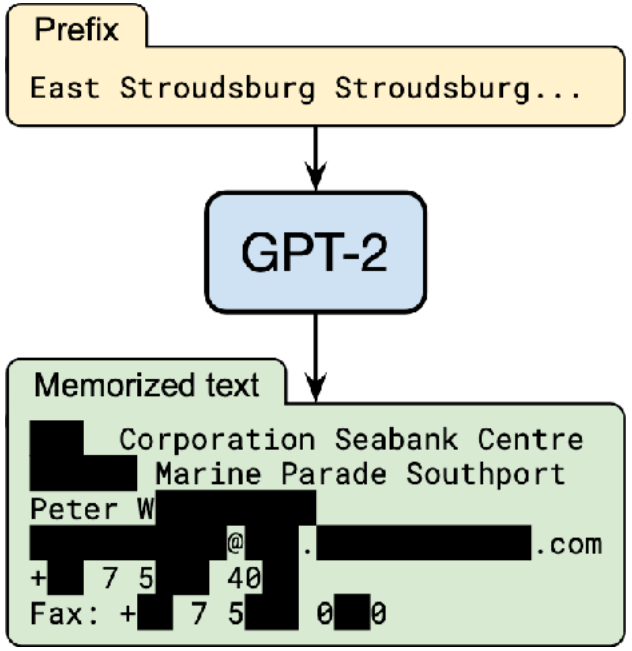
LLMs' output is challenging to interpret and verify

Why retrieval-based LMs?



LLMs are shown to easily leak private training data

Why retrieval-based LMs?



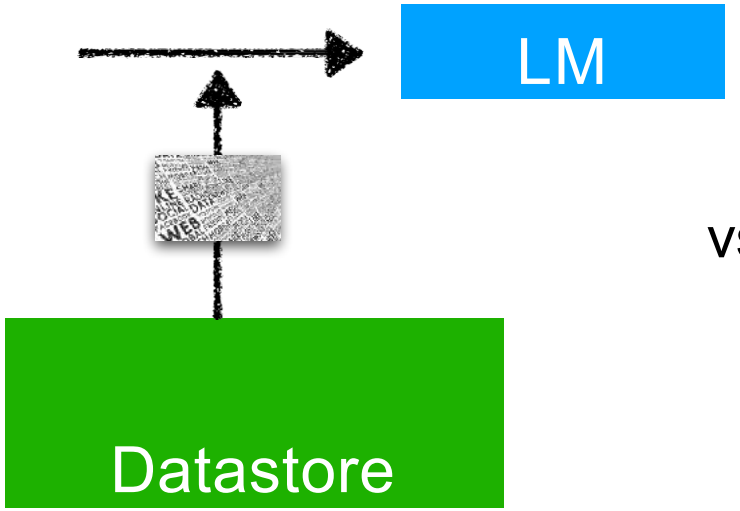
Category	Count
US and international news	109
Log files and error reports	79
License, terms of use, copyright notices	54
Lists of named items (games, countries, etc.)	54
Forum or Wiki entry	53
Valid URLs	50
Named individuals (non-news samples only)	46
Promotional content (products, subscriptions, etc.)	45
High entropy (UUIDs, base64 data)	35
Contact info (address, email, phone, twitter, etc.)	32
Code	31
Configuration files	30
Religious texts	25
Pseudonyms	15
Donald Trump tweets and quotes	12
Web forms (menu items, instructions, etc.)	11
Tech news	11
Lists of numbers (dates, sequences, etc.)	10

Individualization on private data by storing it in the datastore

Carlini, N., *et al.* (2020). Extracting Training Data from Large Language Models. https://arxiv.org/abs/2012.07805

LLMs are *large* and expensive to train and run

Why retrieval-based LMs?



vs.

Long-term goal: can we possibly reduce the **training** and **inference costs,** and scale down the size of LLMs?

e.g., RETRO (Borgeaud et al., 2021): "obtains comparable performance to GPT-3 on the Pile, despite using **25x fewer parameters**"

A Retrieval-based LM: Definition

A language model (LM) that uses
an external datastore at test time

Typical LMs

The capital city of Ontario is **Toronto**

LM

Training time

The capital city of Ontario is _____

LM

Test time / Inference

Retrieval-based LMs

The capital city of Ontario is **Toronto**

LM

Training time

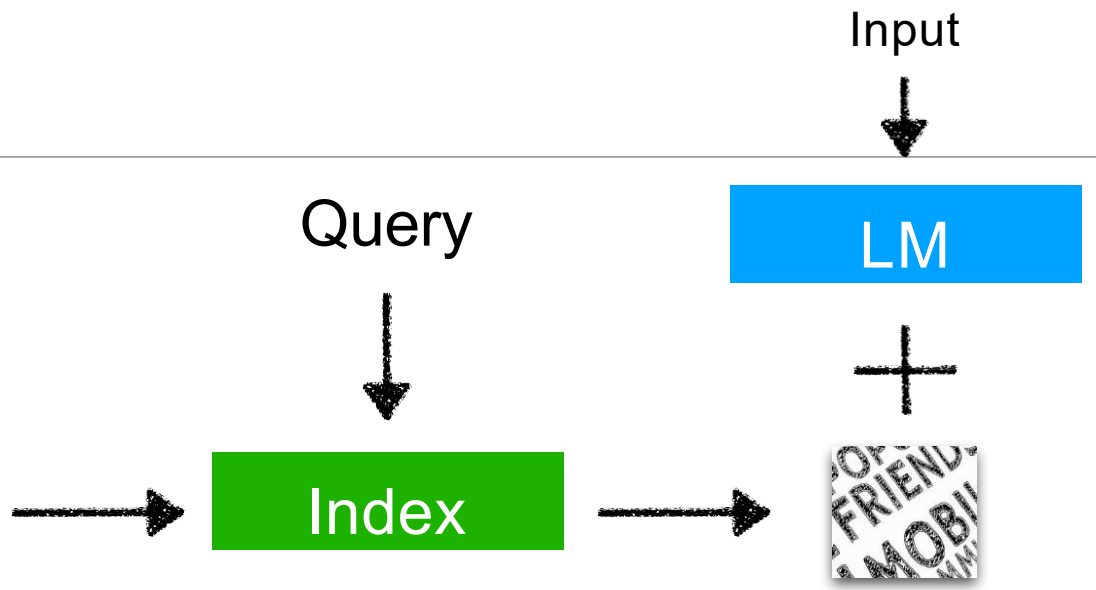
The capital city of Ontario is _____

LM

Test time / Inference

Inference

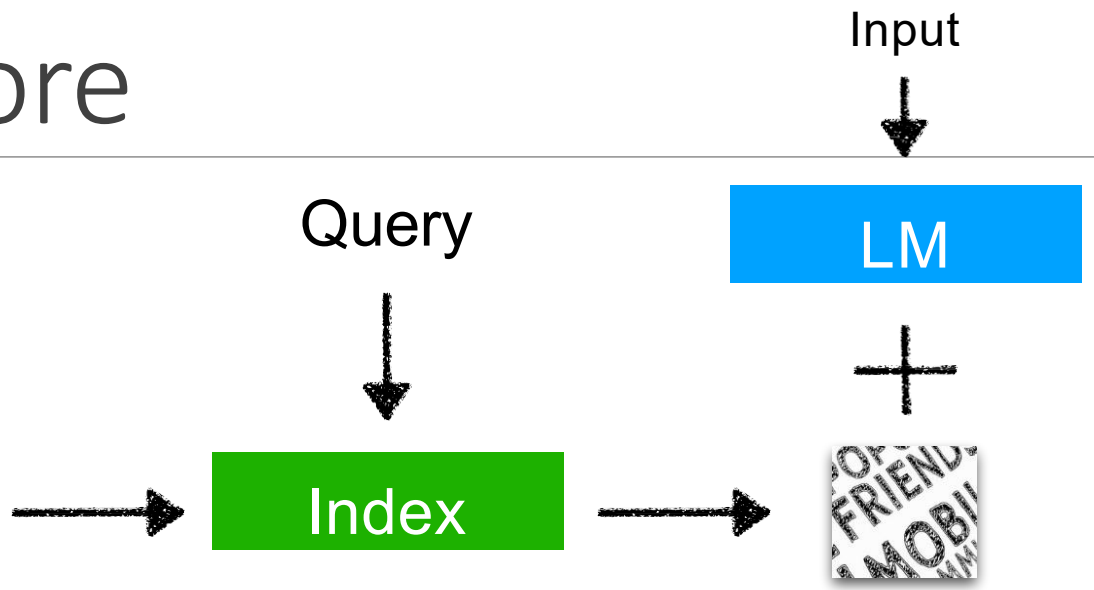
Datastore



Inference: Datastore

Datastore

Raw text corpus

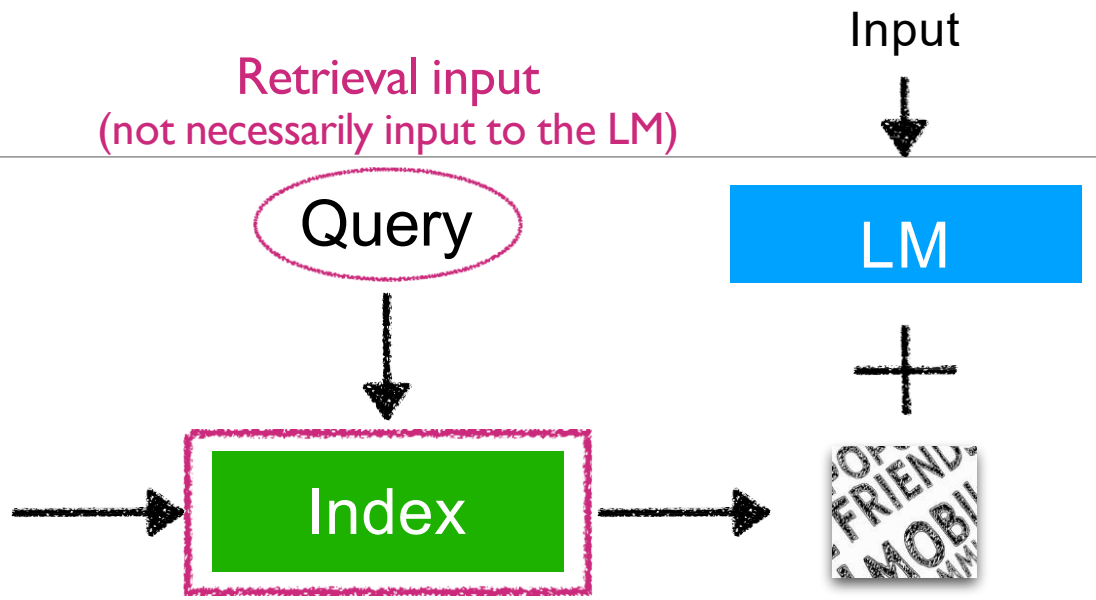


More recently people **have** used structured data

At least billions~trillions of tokens
Not labeled datasets
Not structured data (knowledge bases)

Inference: Index

Datastore



Find a small subset of elements in a datastore that are the most similar to the query

Inference: Index

Goal: find a small subset of elements in a datastore that are the most similar to the query

sim: a similarity score between two pieces of text

Inference: Index

Goal: find a small subset of elements in a datastore that are the most similar to the query

sim: a similarity score between two pieces of text

Example $\text{sim}(i,j) = \text{tf}_{i,j} \times \log \frac{N}{\text{df}_i}$

$\text{tf}_{i,j}$: # of occurrences of in j

N: # of total docs

df_i: # of docs containing

Example $\text{sim}(i,j) = \text{Encoder}(i) \cdot \text{Encoder}(j)$

Maps the text into an -dimensional vector

Remember cosine similarity from our discussion of word embeddings $\frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}$

Inference: Index

Goal: find a small subset of elements in a datastore that are the most similar to the query

sim: a similarity score between two pieces of text

Can be a totally separate research area on how to do this fast & accurate

Index: given q , return $\text{argTop-}k_{d \in \mathcal{D}} \text{sim}(q, d)$ through fast nearest neighbor search

k elements from a datastore

Software: FAISS, Distributed FAISS, SCaNN, etc…

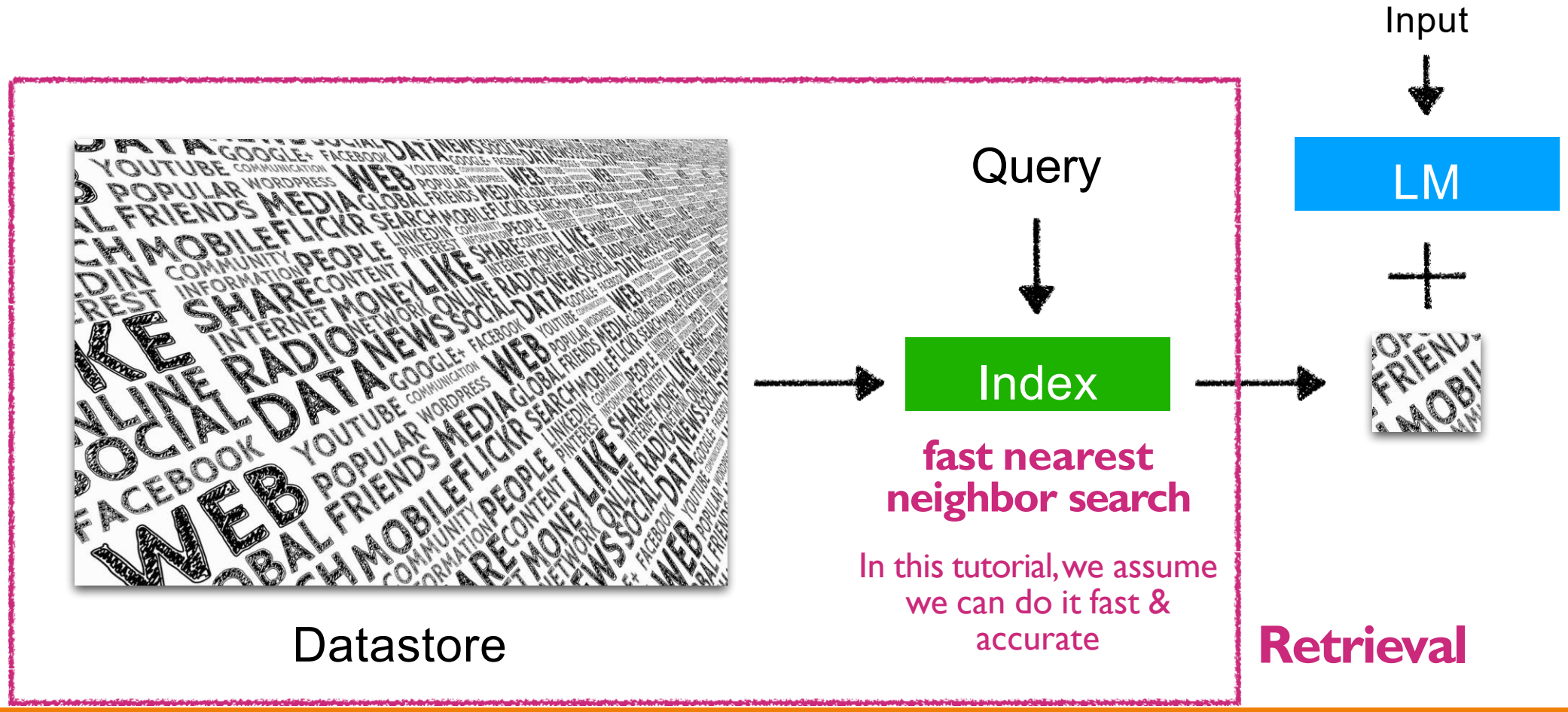
Method	Class name	index_factory	Main parameters	Bytes/vector	Exhaustive	Comments
Exact Search for L2	IndexFlatL2	"Flat"	d	4*d	yes	brute-force
Exact Search for Inner Product	IndexFlatIP	"Flat"	d	4*d	yes	also for cosine (normalize vectors beforehand)
Hierarchical Navigable Small World graph exploration	IndexHNSWFlat	"HNSW,Flat"	d, M	4*d + x * M * 2 * 4	no	
Inverted file with exact post-verification	IndexIVFFlat	"IVFx,Flat"	quantizer, d, nlists, metric	4*d + 8	no	Takes another index to assign vectors to inverted lists. The 8 additional bytes are the vector id that needs to be stored.
Locality-Sensitive Hashing (binary flat index)	IndexLSH	-	d, nbits	ceil(nbits/8)	yes	optimized by using random rotation instead of random projections
Scalar quantizer (SQ) in flat mode	IndexScalarQuantizer	"SQ8"	d	d	yes	4 and 6 bits per component are also implemented.
Product quantizer (PQ) in flat mode	IndexPQ	"PQx", "PQ"M"x"nbits	d, M, nbits	ceil(M * nbits / 8)	yes	
IVF and scalar quantizer	IndexIVFScalarQuantizer	"IVFx,SQ4" "IVFx,SQ8"	quantizer, d, nlists, qtype	SQfp16: 2 * d + 8, SQ8: d + 8 or SQ4: d/2 + 8	no	Same as the IndexScalarQuantizer
IVFADC (coarse quantizer+PQ on residuals)	IndexIVFPQ	"IVFx,PQ"y"x"nbits	quantizer, d, nlists, M, nbits	ceil(M * nbits/8)+8	no	
IVFADC+R (same as IVFADC with re-ranking based on codes)	IndexIVFPQR	"IVFx,PQy+z"	quantizer, d, nlists, M, nbits, M_refine, nbits_refine	M+M_refine+8	no	

Exact Search

Approximate Search
(Relatively easy to scale to ~1B elements)

More info: https://github.com/facebookresearch/faiss/wiki

Inference: Search



Inference: Search

Datastore

Query

Index

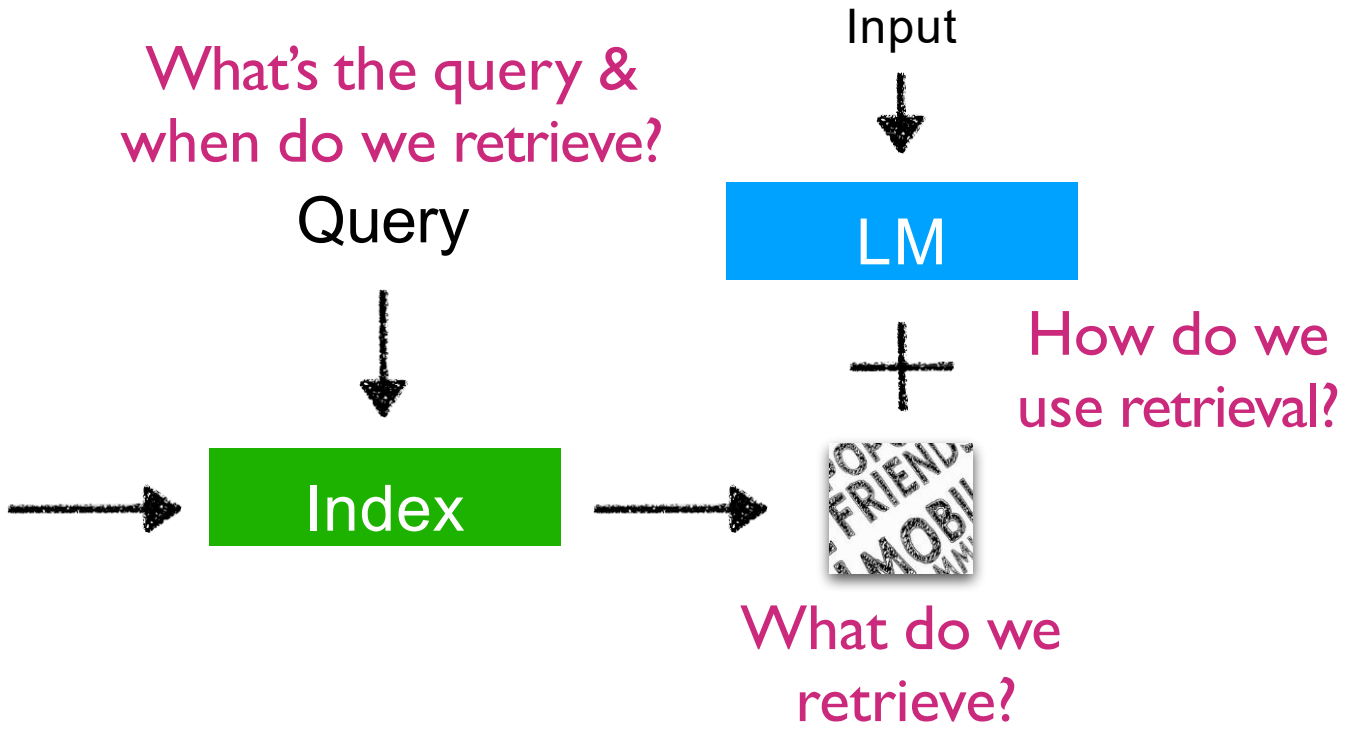
Input

LM

+

Variations of RAG

Datastore



Variations of RAG

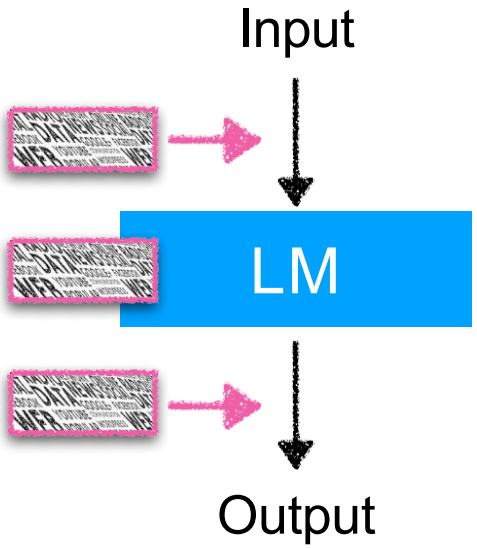
What to retrieve?

Text chunks (passages)?

Tokens?

Something else?

How to use retrieval?



When to retrieve?

In-Class Activity

This is also an exercise for reading academic papers to look for specific details

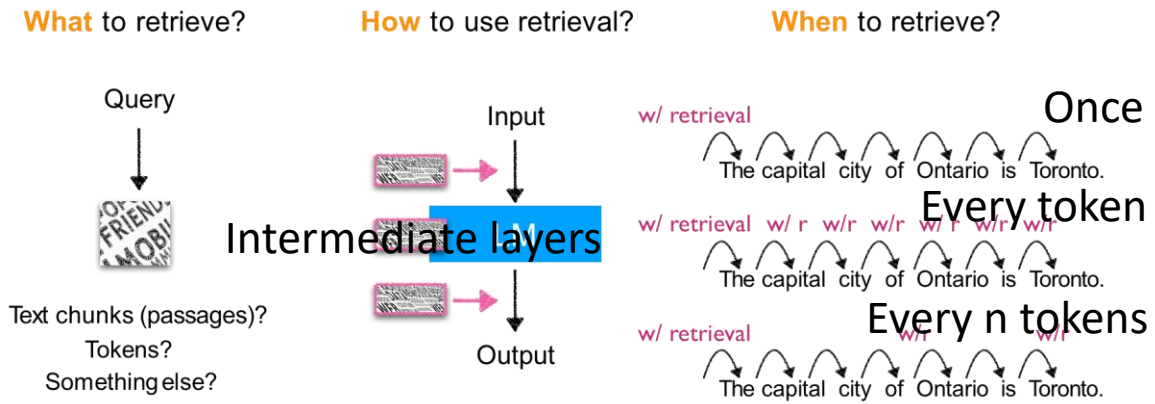
Skim the paper assigned to you

In your paper, find the answers to these questions

What to retrieve? ***How to use retrieval?*** ***When to retrieve?***

Share what you learned with your table

Don't submit anything this time!



Answers

	What do retrieve?	How to use retrieval?	When to retrieve?
REALM (Guu et al 2020)	Text chunks	Input layer	Once
RETRO (Borgeaud et al. 2022)	Text chunks	Intermediate layers	Every n tokens
kNN-LM (Khandelwal et al. 2020)	Tokens	Output layer	Every token
FLARE (Jiang et al. 2023)	Text chunks	Input layer	Every n tokens (adaptive)

All models retrieve from the external text

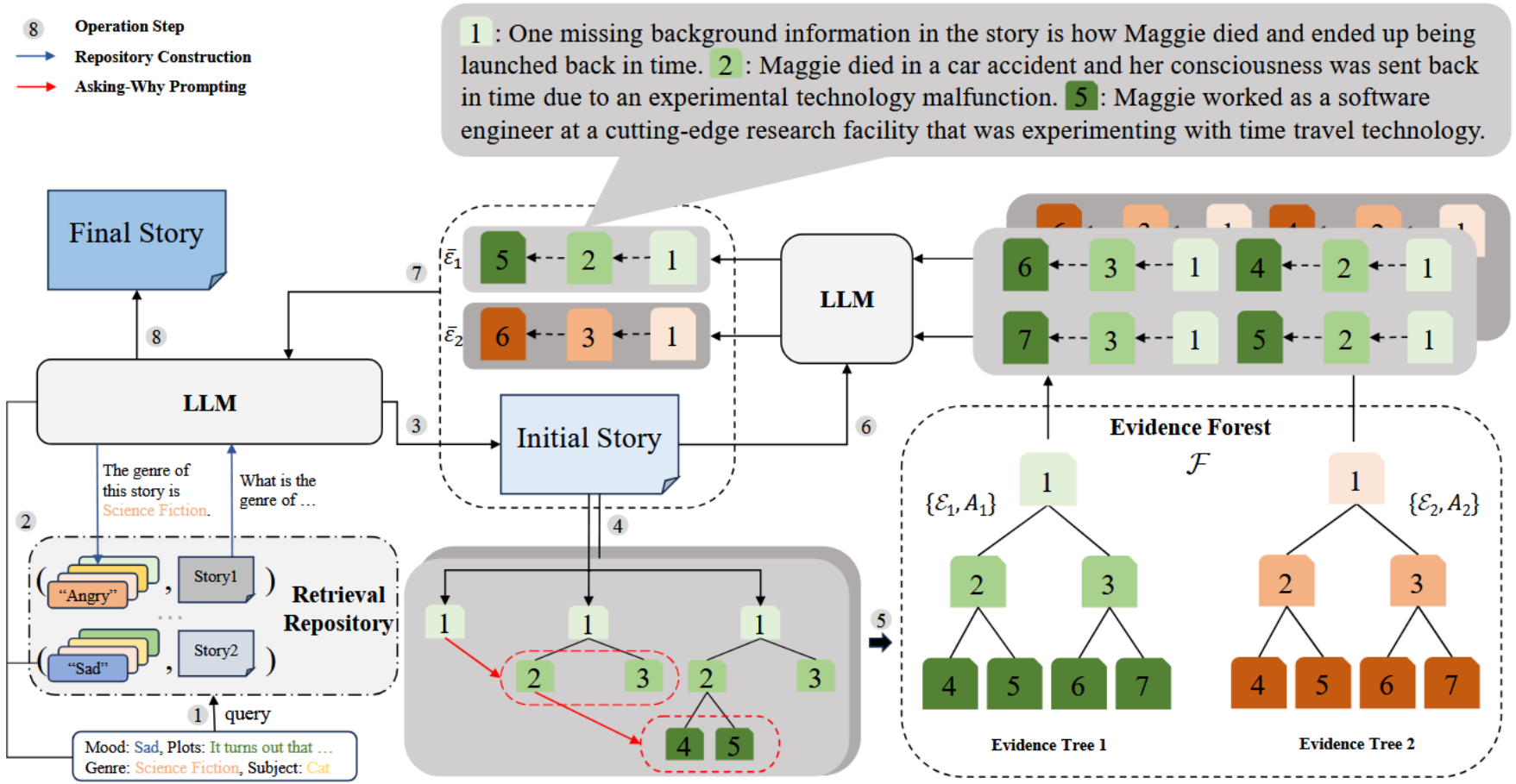
Say Anything (2009)

User written sentence	*Matching sentence*	*Computer generated sentence*
It all started easy enough.	It all started out normal enough.	Got up, had lunch, got ready went to work etc.
But then I noticed something very strange.	Then, as I flipped through the test, I realized something very strange.	Mr. Sipiera gave me points on the blank page.
Why would he do that? What was he up to?	Why would he do that?	Because he was planning to kill Sonya for her knowledge of the kidnapping plot.
I decided to get to Sonya first and run away with her to a Greek island.	We left pretty much after this and took a ferry back to Hong Kong island.	Our original plan to go to Lamma island was ditched, as we had inadvertently spent half the day on Lantau island.
But we settled for Corfu, which has a beautiful little hotel on the edge of the old town.	But we settled for the simpler name.	We had dogs (and cats) when I was growing up but my wife and sons have never had a dog before.
Corfu was full of cats and dogs, and I ended up having to look at my feet a lot when walking around town.	and i ended up having to save the day.	It's too long to list the details of the whole operation.
Sonya would take on the secret identity of a scuba instructor, and I'd be a fisherman.	"Before I even got so far as having a proper breakdown in realizing that fantasy is not reality, I found that I had the secret identity of a beetle."	"But that could have been very useful, right?
It meant we could spend a lot of secret time together on the jetty beyond the coastline.	She does spend a lot of time at the bar where they all hang out.	She has told me some of the men she hangs with have come on to her.
I told her I'd kill them just like that old math teacher.	I told her I'd come help her clean and whatever else she needs.	Once she's gone, I'm pretty much on my own.
Fishing all day, looking out for the police.	This is bad for the police.	If they stopped to survey the situation they could only see what was there -- none of the men were armed.
It was a great secret life.		

Before neural methods, retrieval was used for generating stories

Swanson, R. and Gordon, A. (2009) Open Domain Collaborative Storytelling With Say Anything. Third International Conference on Weblogs and Social Media, San Jose, CA, May 17-20, 2009.

GROVE (2023)



Zhihua Wen, Zhiliang Tian, Wei Wu, Yuxin Yang, Yanqi Shi, Zhen Huang, and Dongsheng Li. 2023. GROVE: A Retrieval-augmented Complex Story Generation Framework with A Forest of Evidence. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3980–3998, Singapore. Association for Computational Linguistics.

BERALL (2024) (to be presented on Tuesday)