Learning to Behave by Reading

Regina Barzilay

Joint work with: Branavan, Harr Chen,
David Silver, Luke Zettlemoyer

Favorite Opening for my NLP Class

"I'm sorry Dave, I'm afraid I can't do that": Linguistics, Statistics, and Natural Language Processing circa 2001*

Lillian Lee, Cornell University

It's the year 2000, but where are the flying cars? I was promised flying cars.
– Avery Brooks, IBM commercial

According to many pop-culture visions of the future, technology will eventually produce the Machine that Can Speak to Us. Examples range from the False Maria in Fritz Lang's 1926 film *Metropolis* to *Knight Rider*'s KITT (a *talking* car) to *Star Wars*' C-3PO (said to have been modeled on the False Maria). And, of course, there is the HAL 9000 computer from *2001: A Space Odyssey*; in one of the film's most famous scenes, the astronaut Dave asks HAL to open a pod bay door on the spacecraft, to which HAL responds, "I'm sorry Dave, I'm afraid I can't do that".

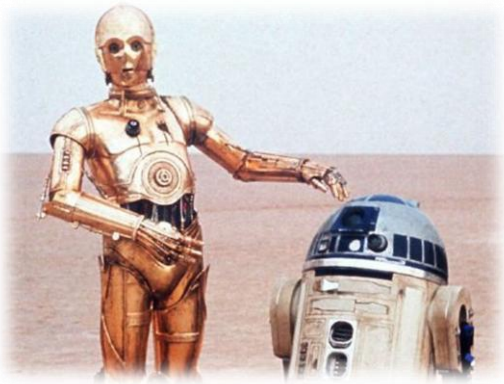
"I'm sorry Dave, I'm afraid I can't do that": Linguistics, Statistics, and Natural Language Processing circa 2001*

Lillian Lee, Cornell University

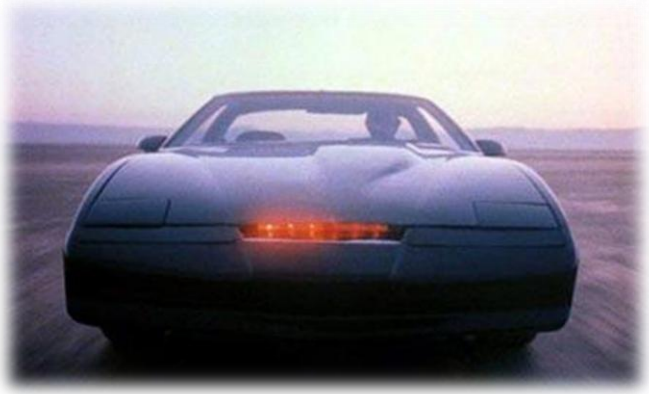
It's the year 2000, but where are the flying cars? I was promised flying cars.
– Avery Brooks, IBM commercial

According to many pop-culture visions of the future, technology will eventually produce the Machine that Can Speak to Us. Examples range from the False Maria in Fritz Lang's 1926 film *Metropolis* to *Knight Rider*'s KITT (a *talking* car) to *Star Wars*' C-3PO (said to have been modeled on the False Maria). And, of course, there is the HAL 9000 computer from *2001: A Space Odyssey*; in one of the film's most famous scenes, the astronaut Dave asks HAL to open a pod bay door on the spacecraft, to which HAL responds, "I'm sorry Dave, I'm afraid I can't do that".

1926: False Maria



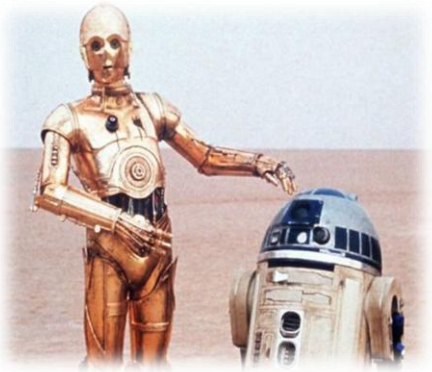
1977: Star Wars



1980s: Knight Rider

Can We Do It?

1926: False Maria



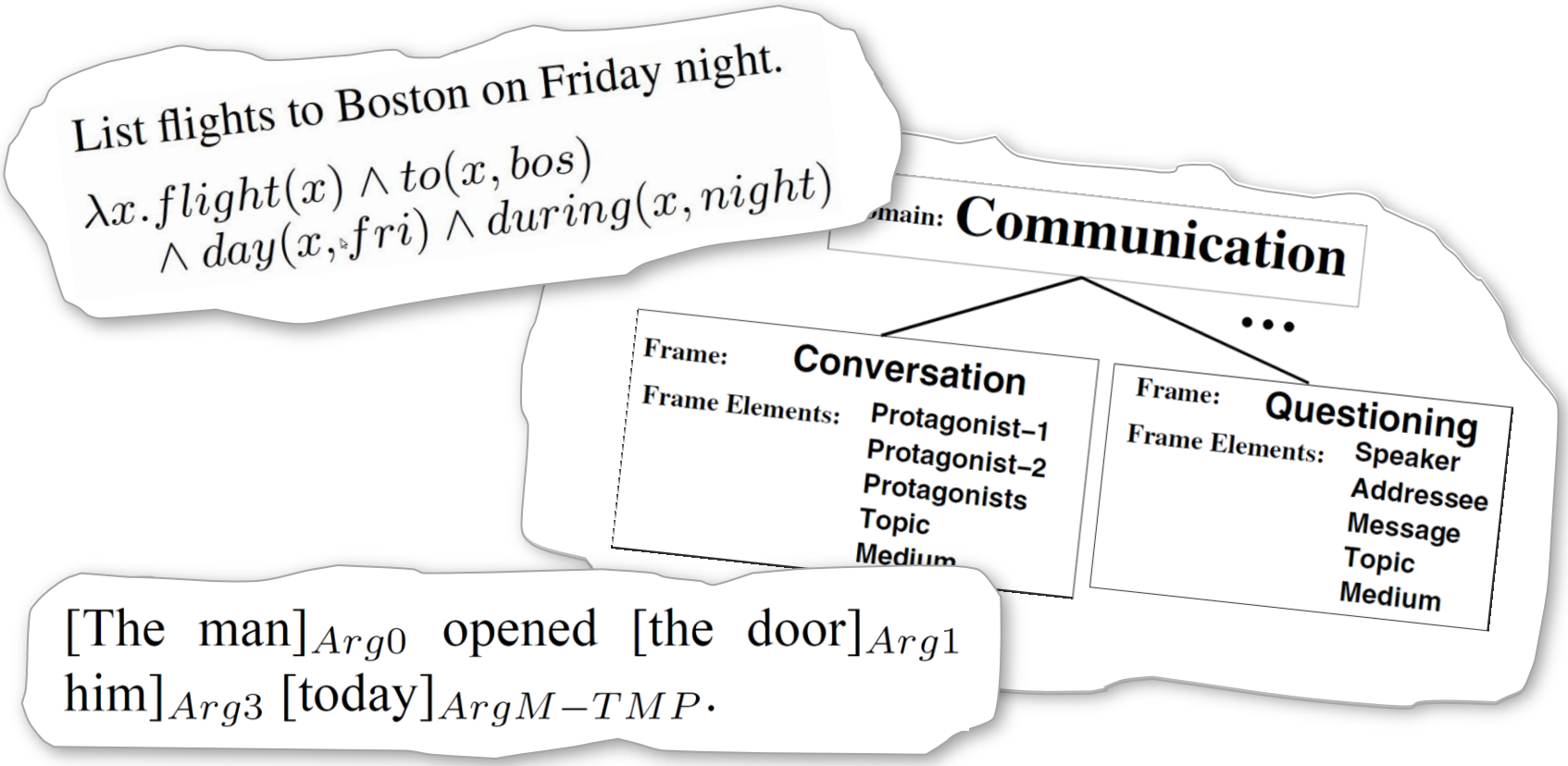
1977: Star Wars

Objective: **Select actions based on information from language and control feedback**

Challenge: **Ground language in world dynamics**

Semantic Interpretation: Traditional Approach

*Map text into an **abstract representation***



(Typical papers on semantics)

Semantic Interpretation: Our Approach

*Map text to **control actions***

Text

Build your city
on grassland
with a river
running through
it if possible.

Control actions

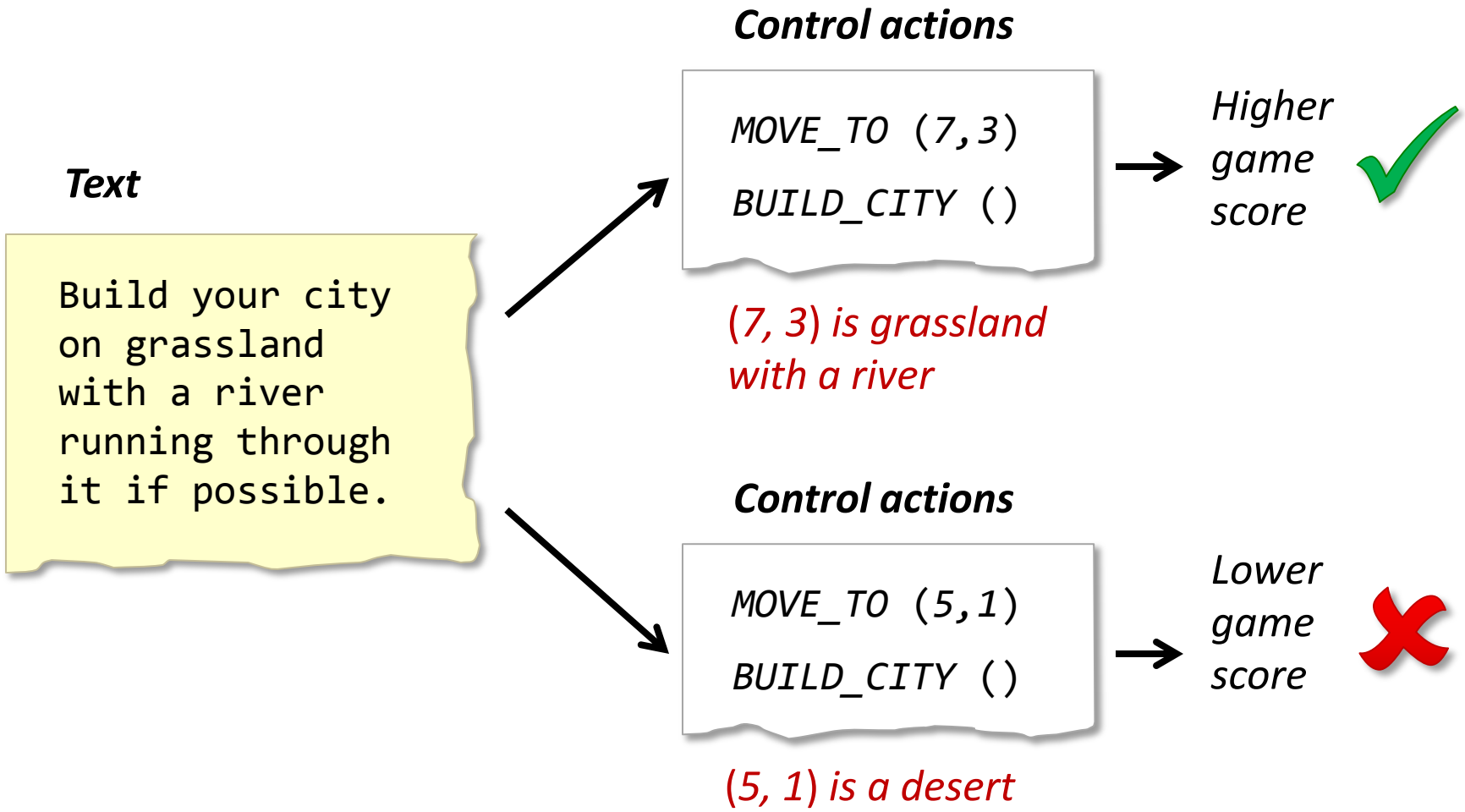
MOVE_TO (7,3)

BUILD_CITY ()

...

Enables language learning from control feedback

Learning from Control Feedback



Learn Language via Reinforcement Learning

A Very Different View of Semantics

Appropriateness: 1	Thoroughness: 1
Clarity: 3	**Impact of Ideas or Results: 2**
Originality / Innovativeness: 1	Recommendation: 1
Soundness / Correctness: 2	Reviewer Confidence: 4
Meaningful Comparison: 1	Audience: 3

It shows that reinforcement learning can map from language directly into a limited set of actions and learn to disambiguate certain constructs. Because the task is not comparable to other research, ***it is not clear that this is progress at all for NLP research.***

…

There is an underlying criticism of NLP work in the suggestion that it is not required. *Yet NLP has in the past 20 years achieved an incredible level of sophistication and acceptance that language matters (syntax and semantics as a formal system) in order to harness the complexity of tasks we accomplish with language.*

…

There is some suggestion that the authors have deeper concerns for language…

Challenges

- Situational Relevance

  Relevance of textual information depends on control state

  "Cities built on or near water sources can irrigate to increase their crop yields, and cities near mineral resources can mine for raw materials."

- Abstraction

  Text can describe abstract concepts in the control application

  "Build your city on grassland."

  "Water is required for irrigation."

- Incompleteness

  Text does not provide complete information about the control application

  "Build your city on grassland with a river running through it if possible."

  (what if there are no rivers nearby?)

General Setup

Input:

- *Text documents useful for the control application*
- *Interactive access to control application*

Goal:

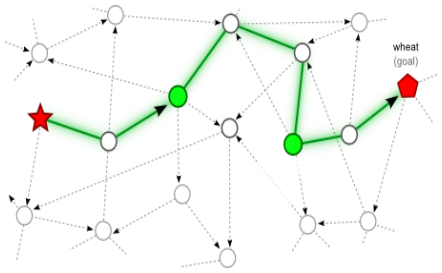
- *Learn to interpret the text and learn effective control*

Outline

1. Step-by-step imperative instructions
2. High-level strategy descriptions
3. General descriptions of world dynamics

Instruction Interpretation (1)

Complex Game-play (2)



High-level Planning (3)

Mapping Instructions to Actions

Input

Instructions:
step-by-step
descriptions of actions

1. Click **Start**, point to **Search**, and then click **For Files or Folders.**
2. In the **Search for** box, type "msdownld.tmp"
3. In the **Look in** list, click **My Computer**, and then click **Search Now**.
4. ...

Target environment:
where actions
need to be executed

Output

Action sequence
executable in the environment

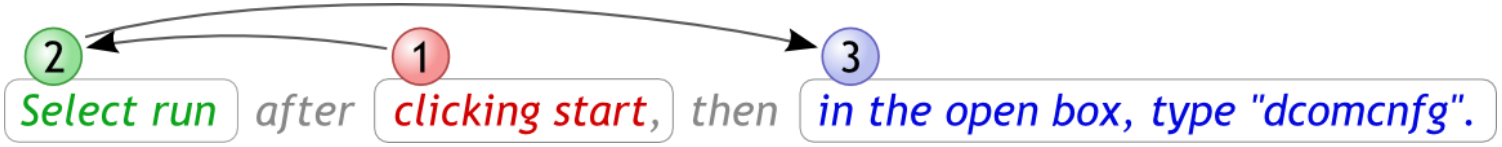
```
LEFT_CLICK( Start )
LEFT_CLICK( Search )
...
TYPE_INTO( Search for: ,"msdownld.tmp")
...
```

Learning Agenda

Segment text to chunks that describe individual commands

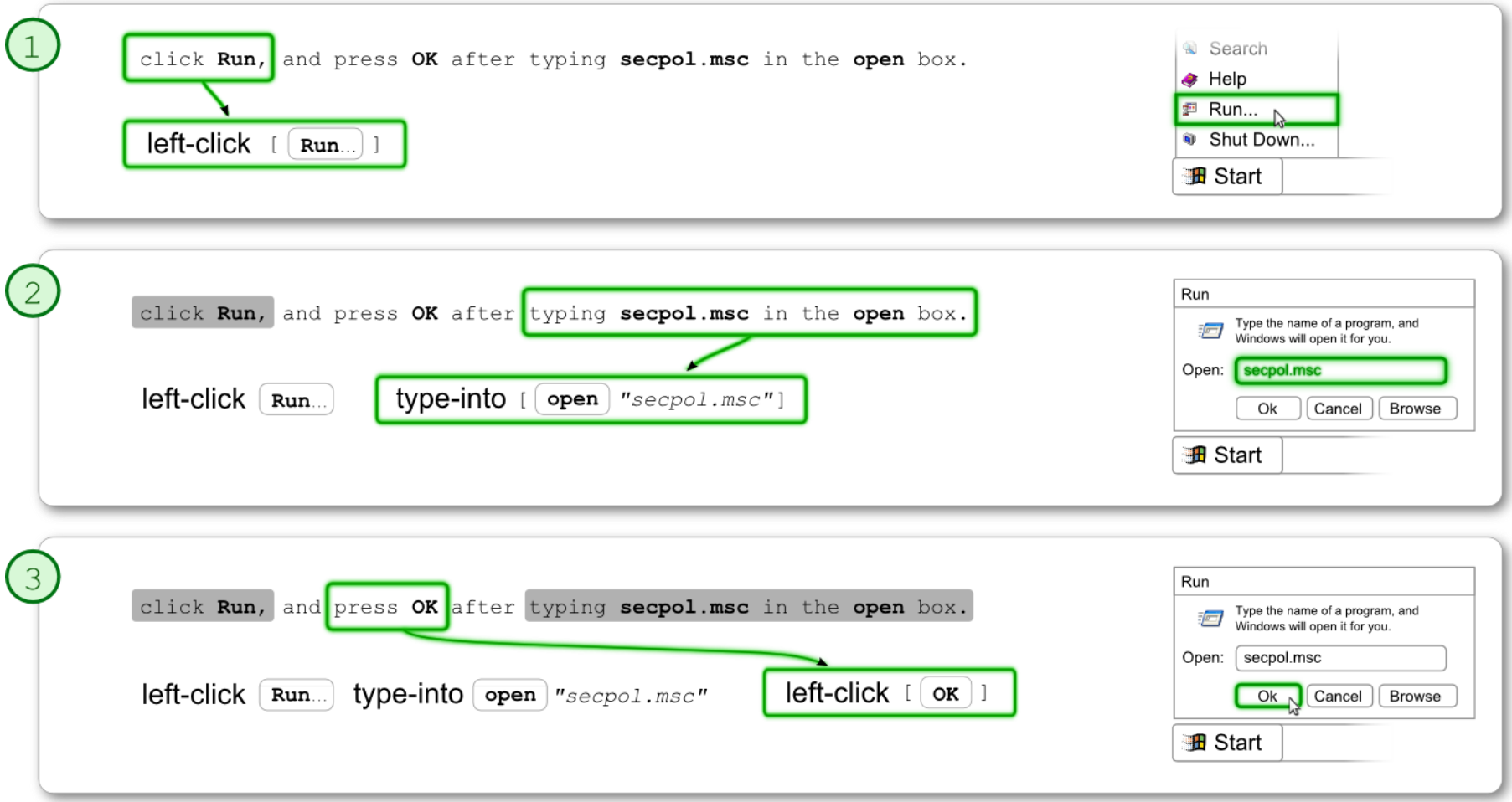
Learn translation of words to environment commands

Reorder environment commands



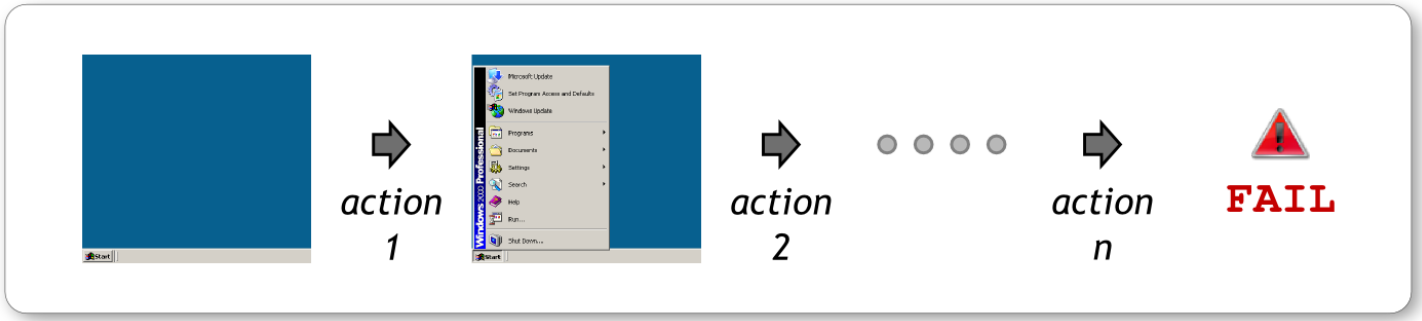
Instruction Interpretation: Representation

Markov Decision Process - select text segment, translate & execute:



Learning Using Reward Signal: Challenges

1. Reward can be delayed



⇒ *How can reward be propagated to individual actions*

2. Number of candidate action sequences is very large

⇒ *How can this space be effectively searched?*

Use Reinforcement Learning

Reinforcement Learning: Representation

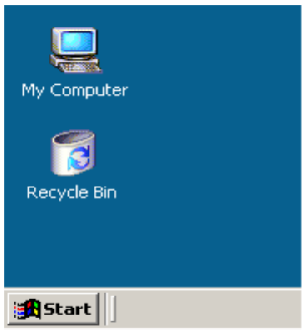
State s = Observed Text + Observed Environment

Action a = Word Selection + Environment Command

State 1

Observed text and environment

Select run after clicking start. In the open box type "dcomcnfg".



Action 1

words:
clicking start

command:
LEFT_CLICK(start)

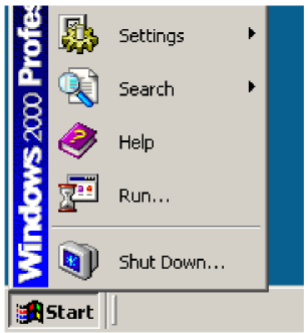
Policy function

$$p(a \mid s)$$

State 2

Observed text and environment

Select run after ~~clicking start.~~ In the open box type "dcomcnfg".



Reward Signal

Ideal:

Test for task completion

Alternative Indication of Error:

The instructions don't make sense anymore

E.g. the text specifies objects not visible on the screen

Approximation:

If a sentence matches no GUI labels, a preceding action is wrong

Critique on the Critique

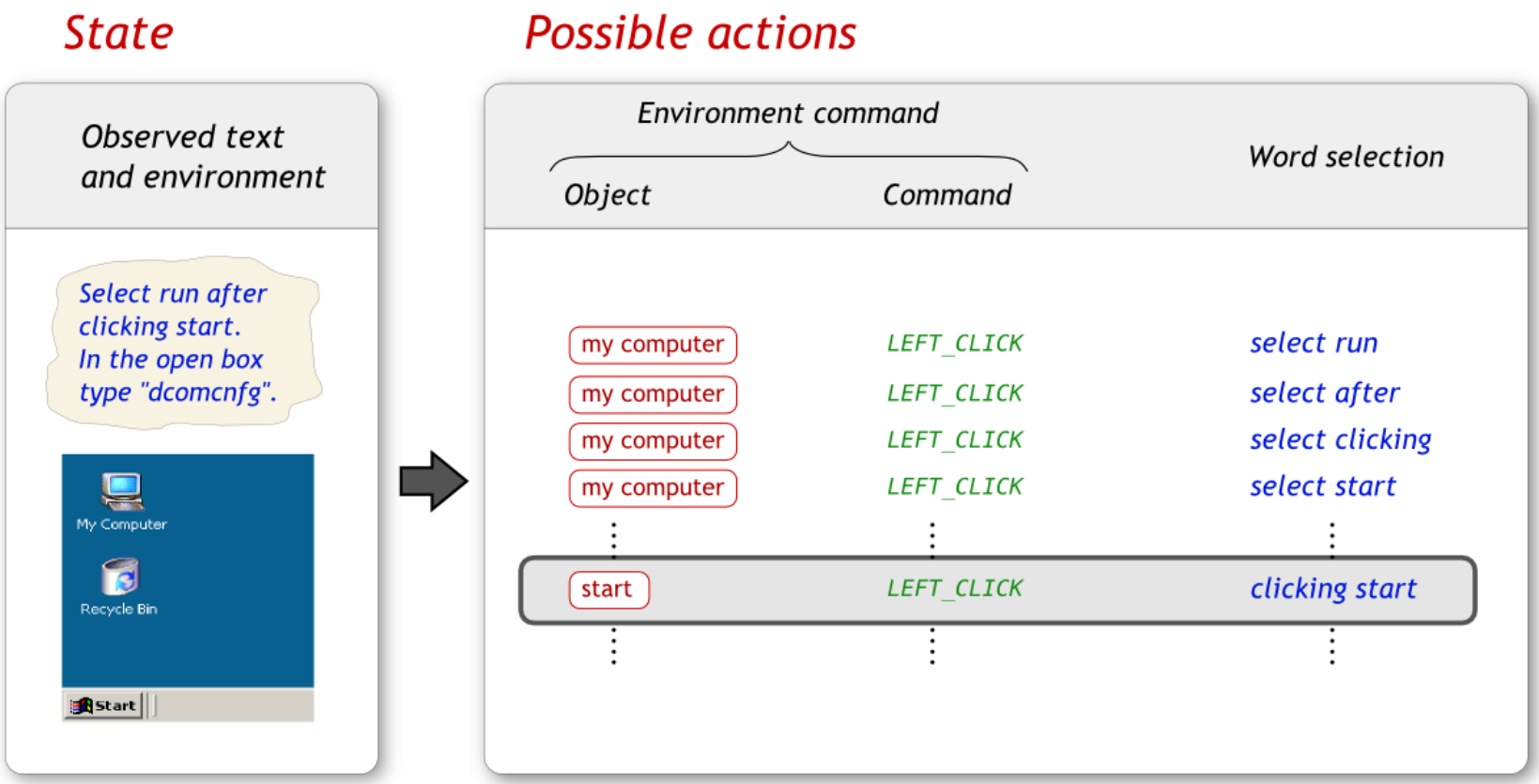
These papers both address what might roughly be called the grounding problem, or at least trying to learn something about semantics by looking at data. I really really like this direction of research, and both of these papers were really interesting. Since I really liked both, and since I think the directions are great, I'll take this opportunity to say what I felt was a bit lacking in each. In the Branavan paper, the particular choice of reward was both clever and a bit of a kludge. I can easily imagine that it wouldn't generalize to other domains: thank goodness those Microsoft UI designers happened to call the Start Button something like UI_STARTBUTTON.

No free lunch, Hal!

Generating Possible Actions

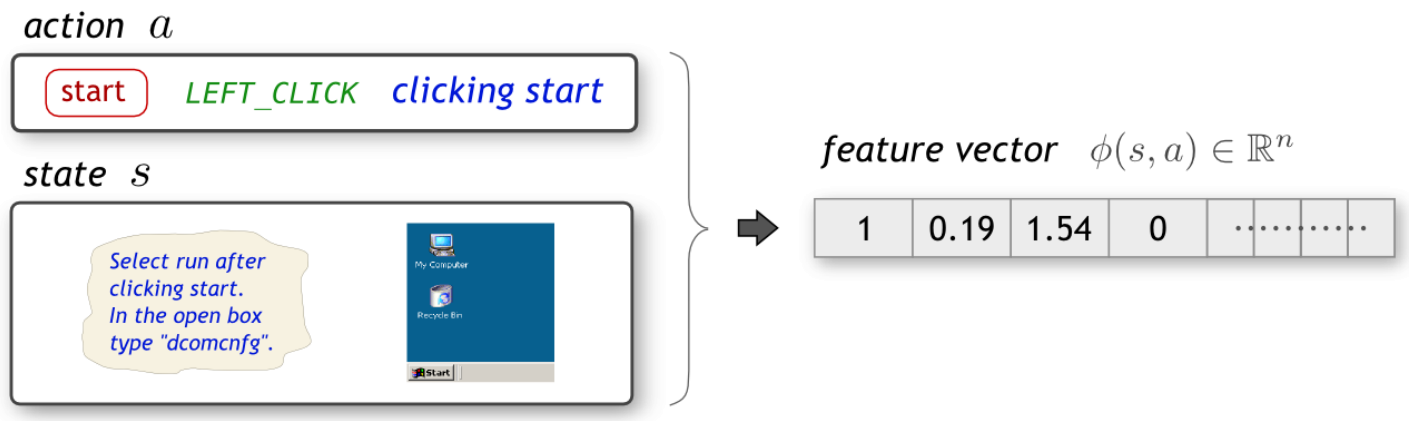
State s = Observed Text + Observed Environment

Action a = Word Selection + Environment Command



Model Parameterization

Represent each action with a feature vector:



$\phi(s,a) \in \mathbb{R}^n$ - *real valued feature function on state* s *and action* a

Define policy function as a log-linear distribution:

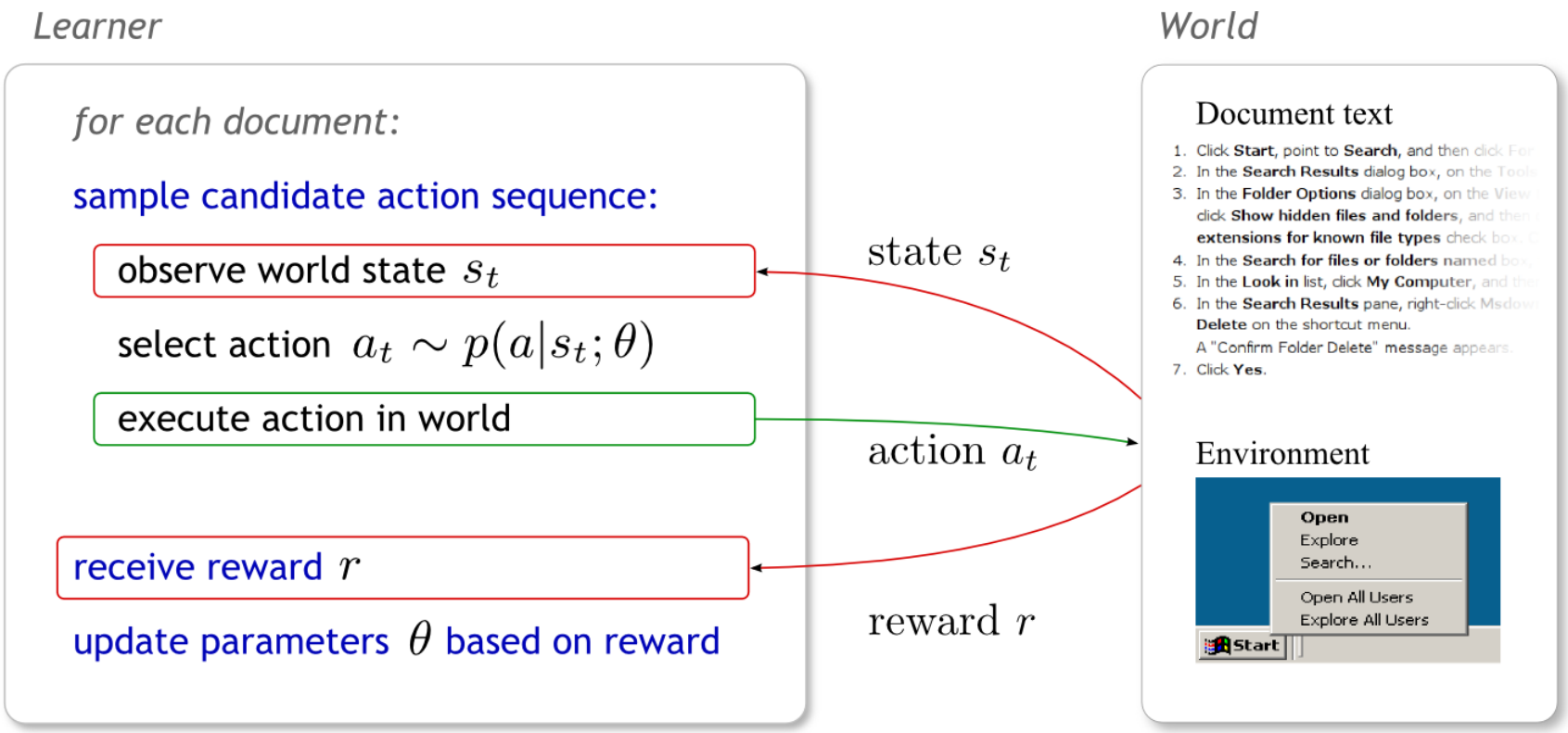
$$p(a \mid s;\theta) = \frac{e^{\theta \cdot \phi(s,a)}}{\sum_{a'} e^{\theta \cdot \phi(s,a')}}$$

θ - *parameters of model*

Learning Algorithm

Goal: *Find θ that maximizes the expected reward*

Method: *Policy gradient algorithm* *(stochastic gradient ascent on θ)*



Policy Function Factorization

$$p(a \mid s, \theta) = p(w_c \mid s, \theta_{wc}) \;\; p(w_o \mid s, w_c, \theta_{wo}) \;\; \times$$
$$p(o \mid s, w_o, \theta_o) \;\; p(c \mid s, o, w_c, \theta_c)$$

$p(w_c \mid s, \theta_{wc})$ - Select command word i.e. *clicking*

$p(w_o \mid s, w_c, \theta_{wo})$ - Select object word i.e. *start*

$p(o \mid s, w_o, \theta_o)$ - Select object i.e. the button **Start**

$p(c \mid s, o, w_c, \theta_c)$ - Select command i.e. *LEFT_CLK (left click)*

Example Features

Features on words and environment command

- Edit distance between word and object label
- Binary feature on each (*word*, *command*) pair
- Binary feature on each (*word*, *object type*) pair

Features on environment objects

- Object is visible
- Object is in foreground
- Object was previously interacted with
- Object became visible after last action

Features on words

- Word type
- Distance from last used word

Total number of features: 4438

Windows Configuration Application

Windows 2000 help documents
from *support.microsoft.com*

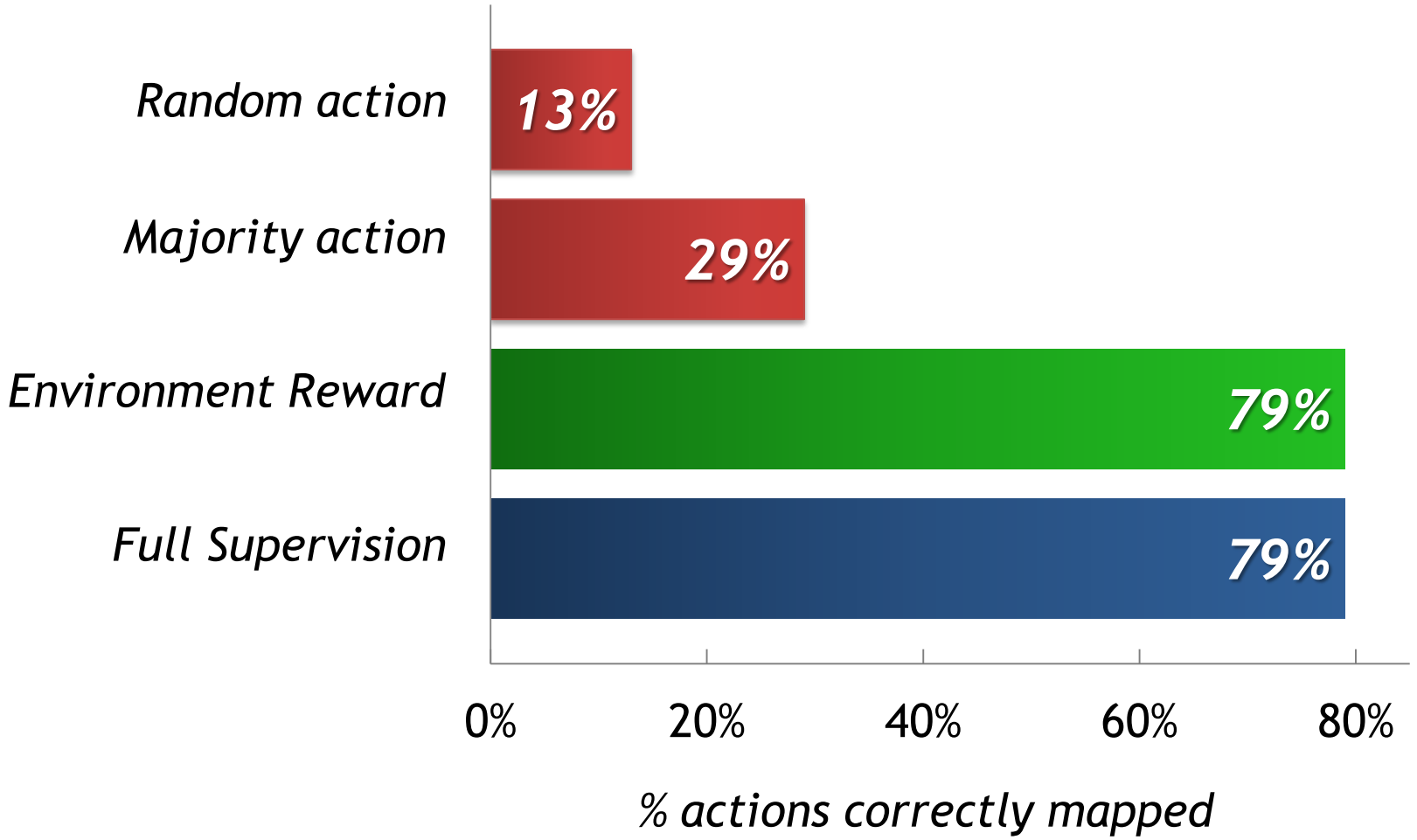
Total # of documents	128
Train/development/test	70 / 18 / 40
Total # of words	5562
Vocabulary size	610
Avg. words per sentence	9.93
Avg. sentences per document	4.38
Avg. actions per document	10.37

Human Performance

"We tested the automation capability of WikiDo on 5 computer tasks, completed by 12 computer science students. *Even with detailed instructions, the students failed to correctly complete the task in **20%** of the cases.*"

(Kushman et al. Hotnets 2009)

Results: Action Accuracy



Applications of Instruction Mapping: WikiDo

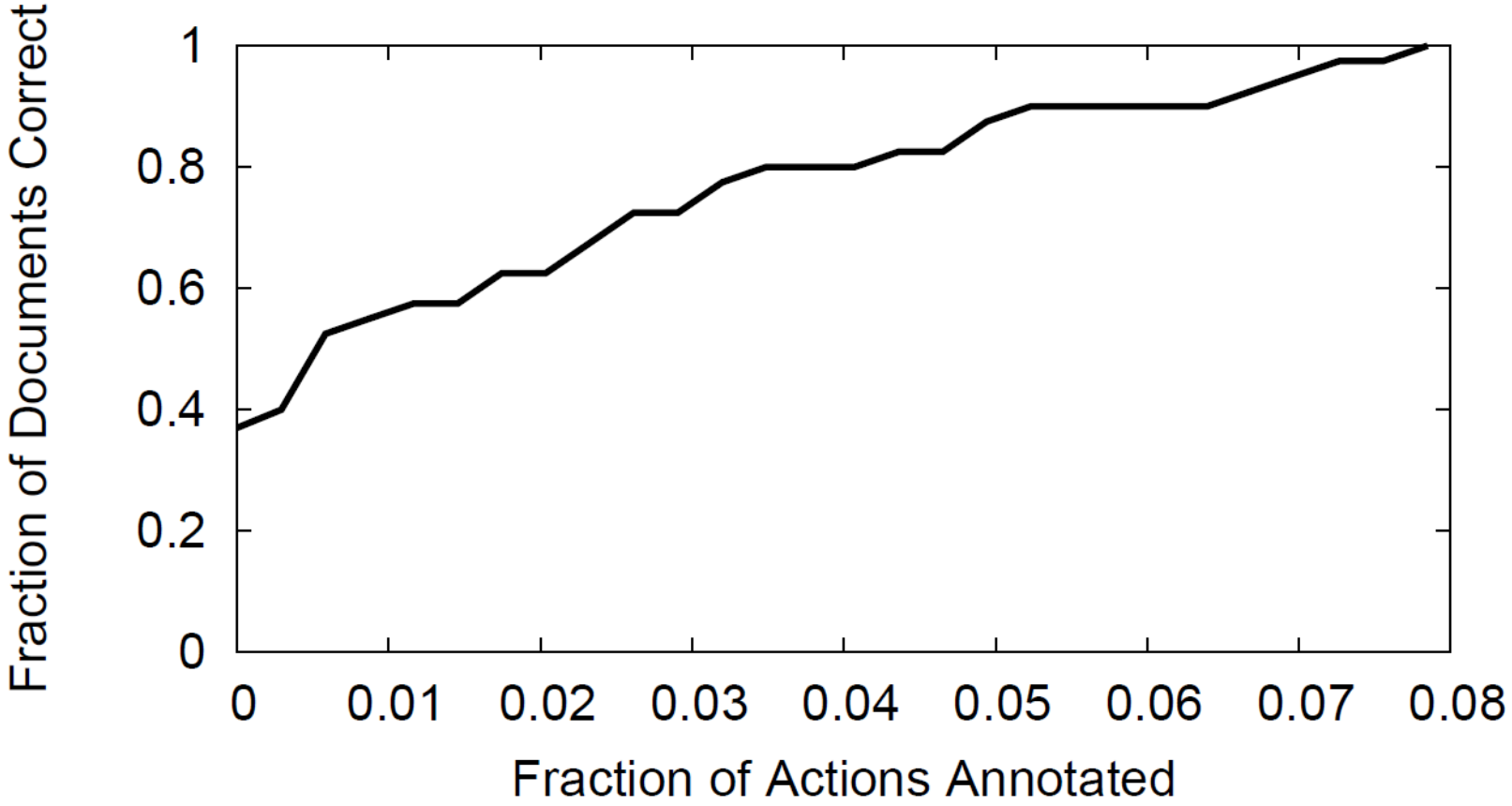
But we want
99% accuracy!

Solution: Use active learning

1. Train a classifier to identify wrong action translations
2. Rely on crowd sourcing to correct identified actions

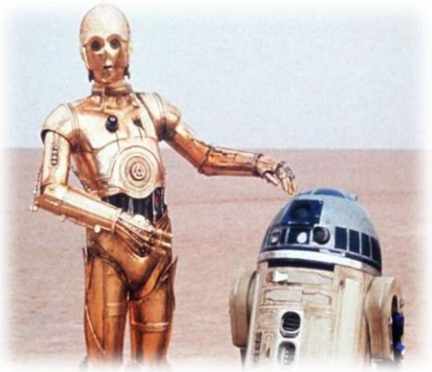
Active Learning: Performance

8% action annotation ⇒ 100% document accuracy !



Can We Do It?

1926: False Maria



1977: Star Wars

Vision: **Communicate with robots by specifying high-level goals**

- *Follow natural language instructions*
- *Select actions based on information from language and control feedback*

Outline

1. Step-by-step imperative instructions
   - *Learning from control feedback*

2. High-level strategy descriptions
   - ***Situational relevance***
   - ***Incompleteness***
   - *Learning from control feedback*

3. General descriptions of world dynamics

Solving Hard Decision Tasks

How to Load Pallets

1. Place a pallet near the boxes you are loading.
2. Carefully stack boxes in a uniform fashion onto the pallet.
3. Stretch wrap the boxes on the pallet to ensure they do not shift when you move the pallet.

Warren Buffett's Priceless Investment Advice

By John Reeves | More Articles
February 12, 2010 | Comments (0)

"It's far better to buy a wonderful company at a fair price than a fa price."

If you can grasp this simple advice from Warren Buffett, you shoul there are other investment strategies out there, but Buffett's appro demonstrably successful over more than 50 years. Why try anythir

Two words for the efficient market hypothesis: Warren Buffet

Civilization II Player's Guide

You start with two settler units. Although settlers are capable of performing a variety of useful tasks, your first task is to move the settlers to a site that is suitable for the construction of your first city. Use settlers to build the city on grassland with a river running through it

Solving Hard Decision Tasks

Objective: Maximize a utility function

Challenge: Finding optimal solution hard

- *Large decision space*
- *Expensive simulations*

Traditional solution: Manually encoded domain-knowledge

Our goal: Automatically extract required domain knowledge from text

Case Study: Adversarial Planning Problem

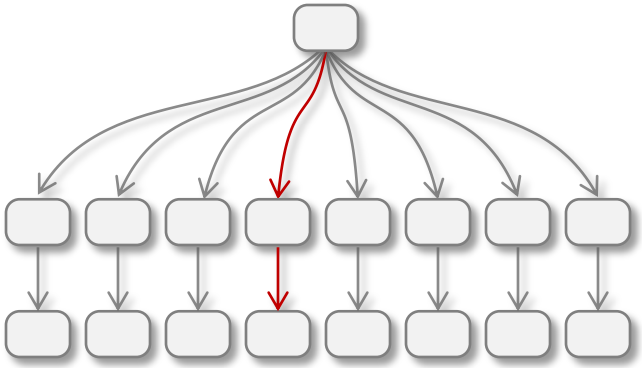
Civilization II : Complex multiplayer strategy game (branching factor $\approx 10^{20}$)

Traditional Approach: Monte-Carlo Search Framework

- *Learn action selection policy from simulations*
- *Very successful in complex games like Go and Poker*

Research Agenda

Now we need lots of simulations to identify promising candidate actions.



How can we use information automatically extracted from manuals to achieve intelligent behavior?

Cities built on or near water sources can irrigate to increase their crop yields, and cities near mineral resources can mine for raw materials.

Leveraging Textual Advice: Challenges

1. Find sentences relevant to given game state.

Game state

Strategy document

You start with two settler units. Although settlers are capable of performing a variety of useful tasks, your first task is to move the settlers to a site that is suitable for the construction of your first city. Use settlers to build the city on grassland with a river running through it if possible. You can also use settlers to irrigate land near your city. In order to survive and grow …

Leveraging Textual Advice: Challenges

1. Find sentences relevant to given game state.

Game state

Strategy document

You start with two settler units. Although settlers are capable of performing a variety of useful tasks, your first task is to move the settlers to a site that is suitable for the construction of your first city. Use settlers to build the city on grassland with a river running through it if possible. You can also use settlers to ***irrigate land near your city.*** *In order to survive and grow …*

Leveraging Textual Advice: Challenges

1. Find sentences relevant to given game state.

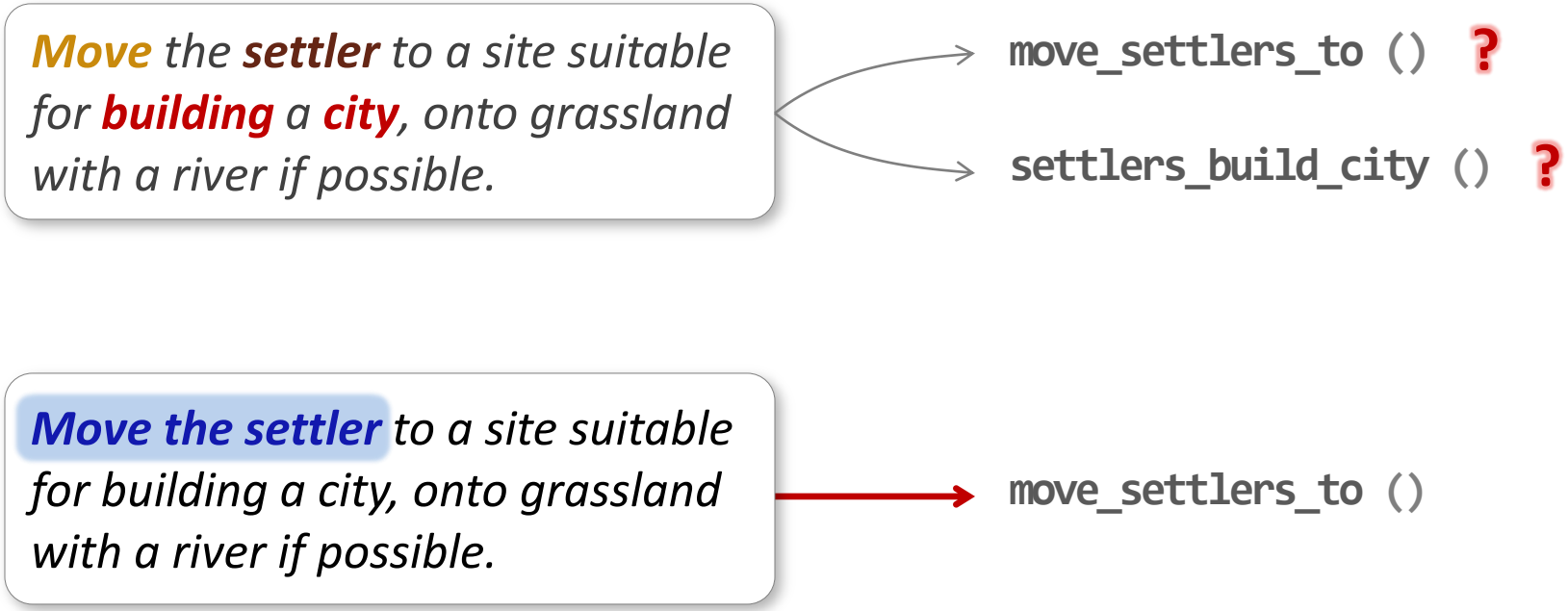
Game state

Strategy document

You start with two settler units. Although settlers are capable of performing a variety of useful tasks, your first task is to move the settlers to a site that is suitable for the construction of your first city.
Use settlers to ***build the city on grassland with a river*** *running through it if possible.*
You can also use settlers to irrigate land near your city. *In order to survive and grow …*

Leveraging Textual Advice: Challenges

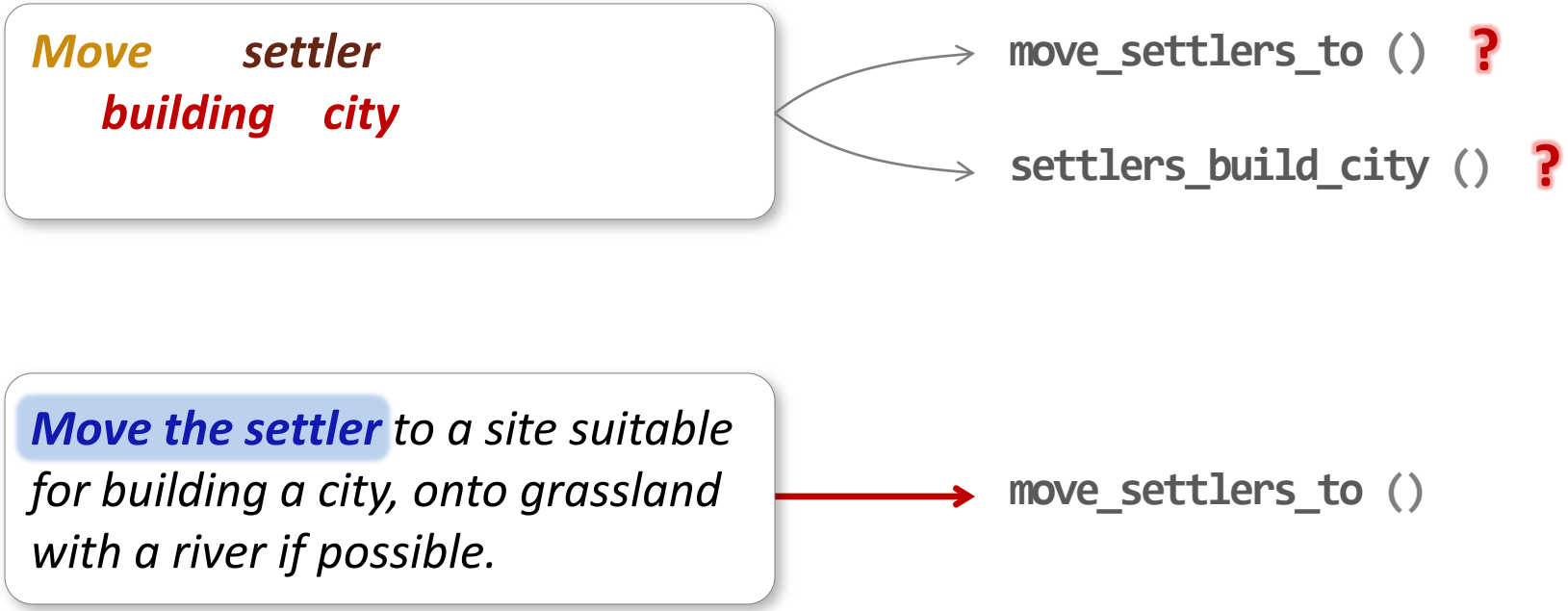
2. Label sentences with predicate structure.



Label words as *action*, *state* or *background*

Leveraging Textual Advice: Challenges

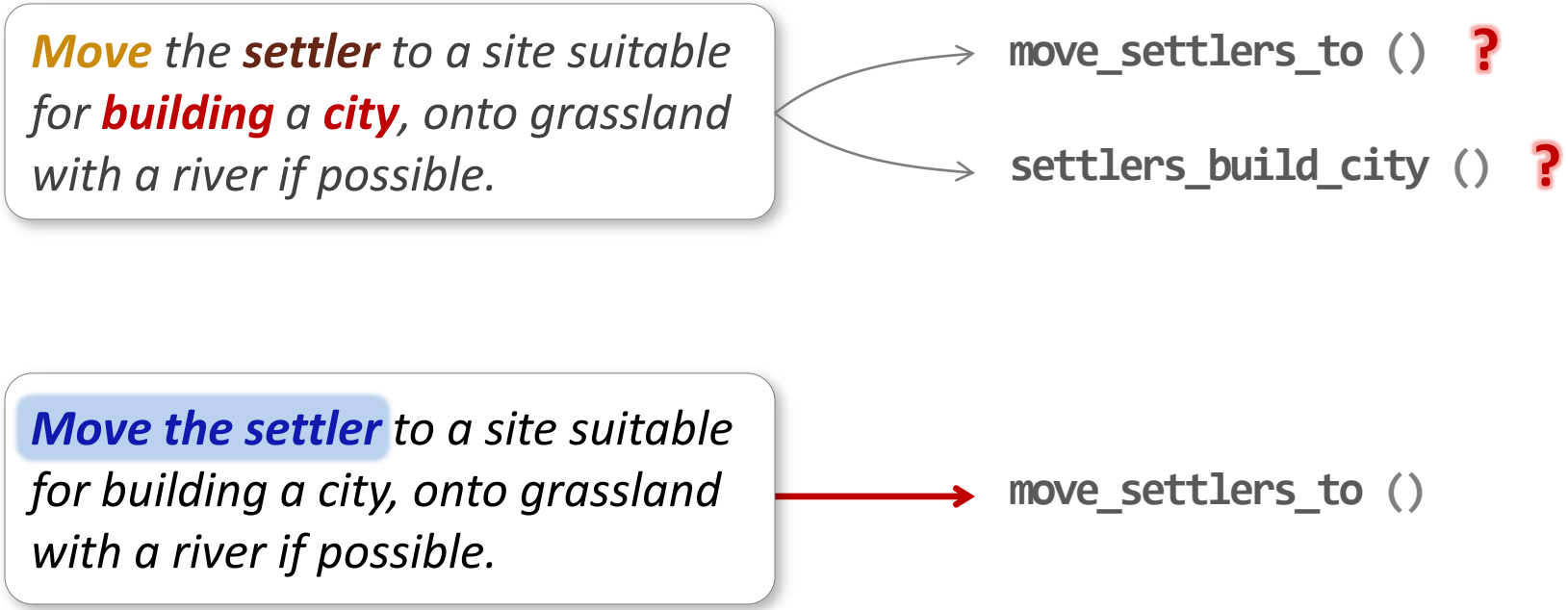
2. Label sentences with predicate structure.



Label words as *action*, *state* or *background*

Leveraging Textual Advice: Challenges

2. Label sentences with predicate structure.

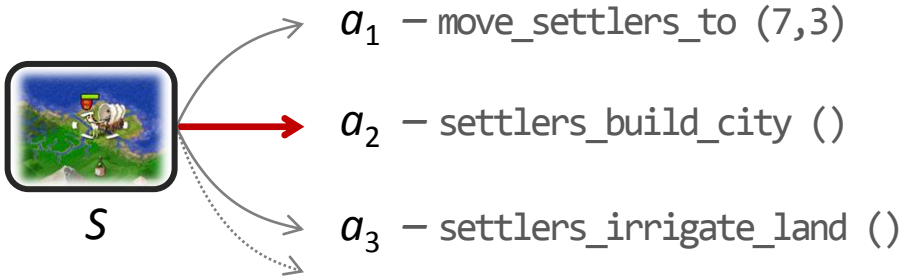


Label words as *action*, *state* or *background*

Leveraging Textual Advice: Challenges

3. Guide action selection using relevant text

Build the city on plains or grassland with a river running through it if possible.



Model Overview

Monte-Carlo Search Framework

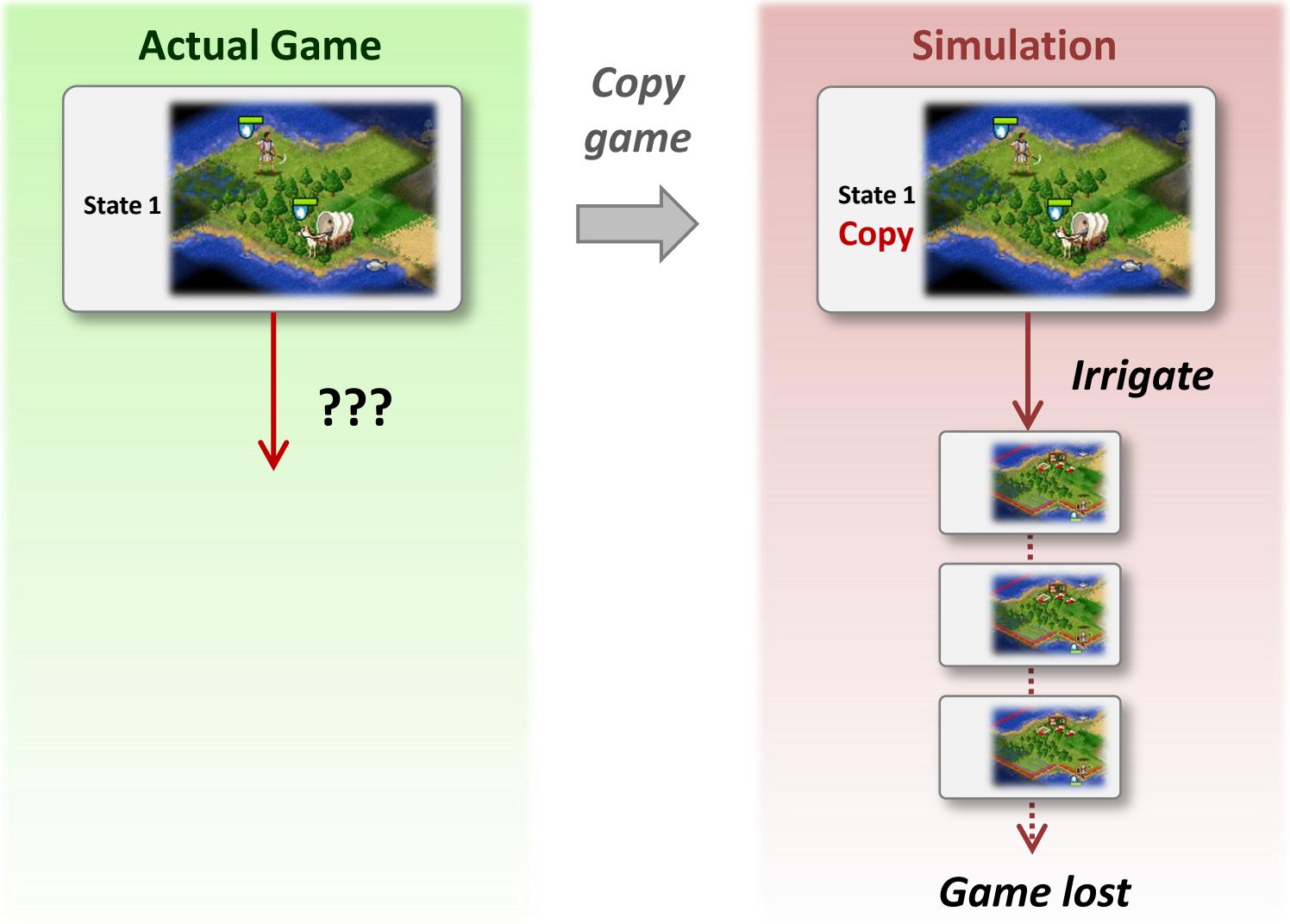
- *Learn action selection policy from simulations*

Our Algorithm

- *Learn text interpretation from simulation feedback*
- *Bias action selection policy using text*

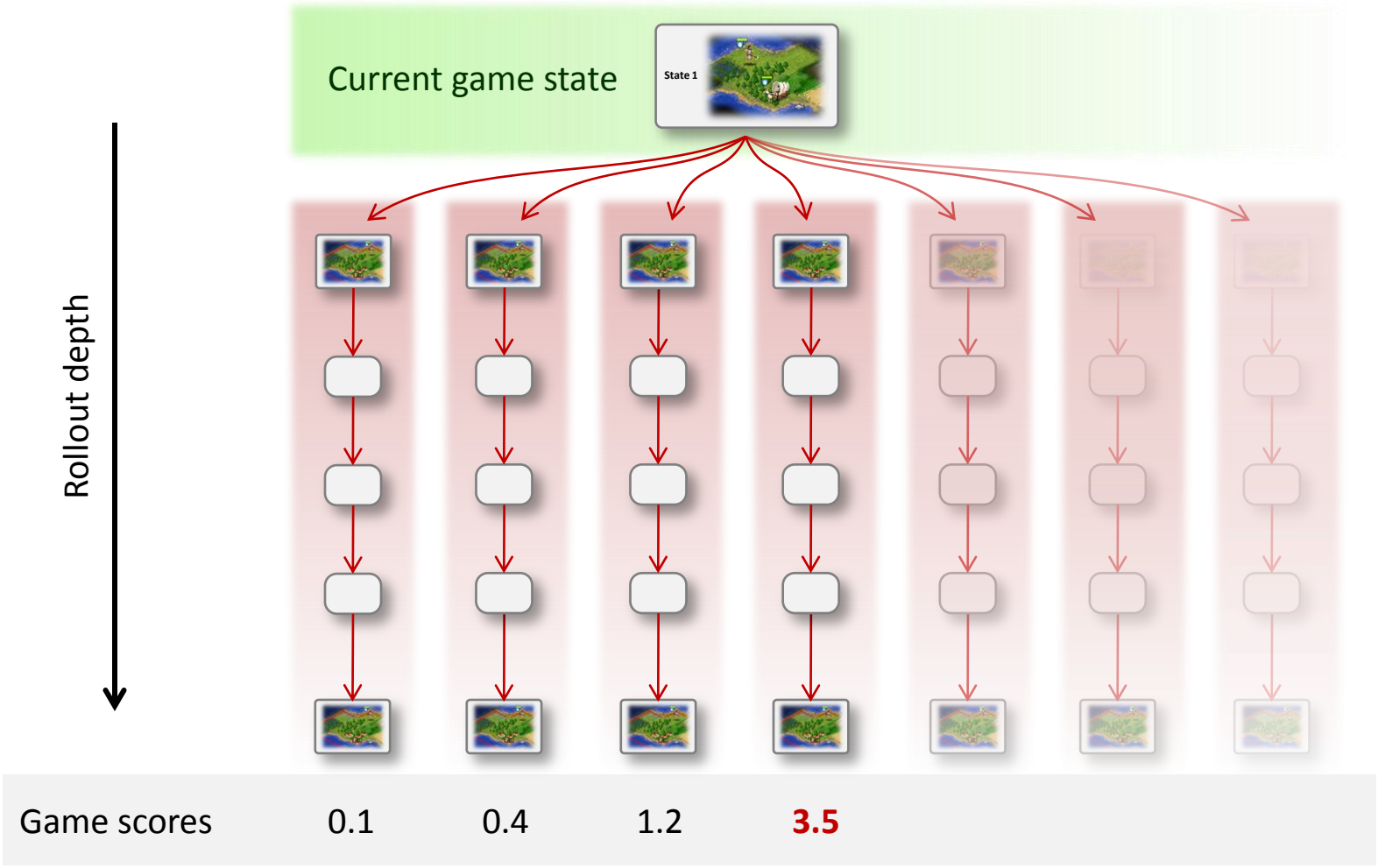
Monte-Carlo Search

Select actions via simulations, game and opponent can be stochastic



Monte-Carlo Search

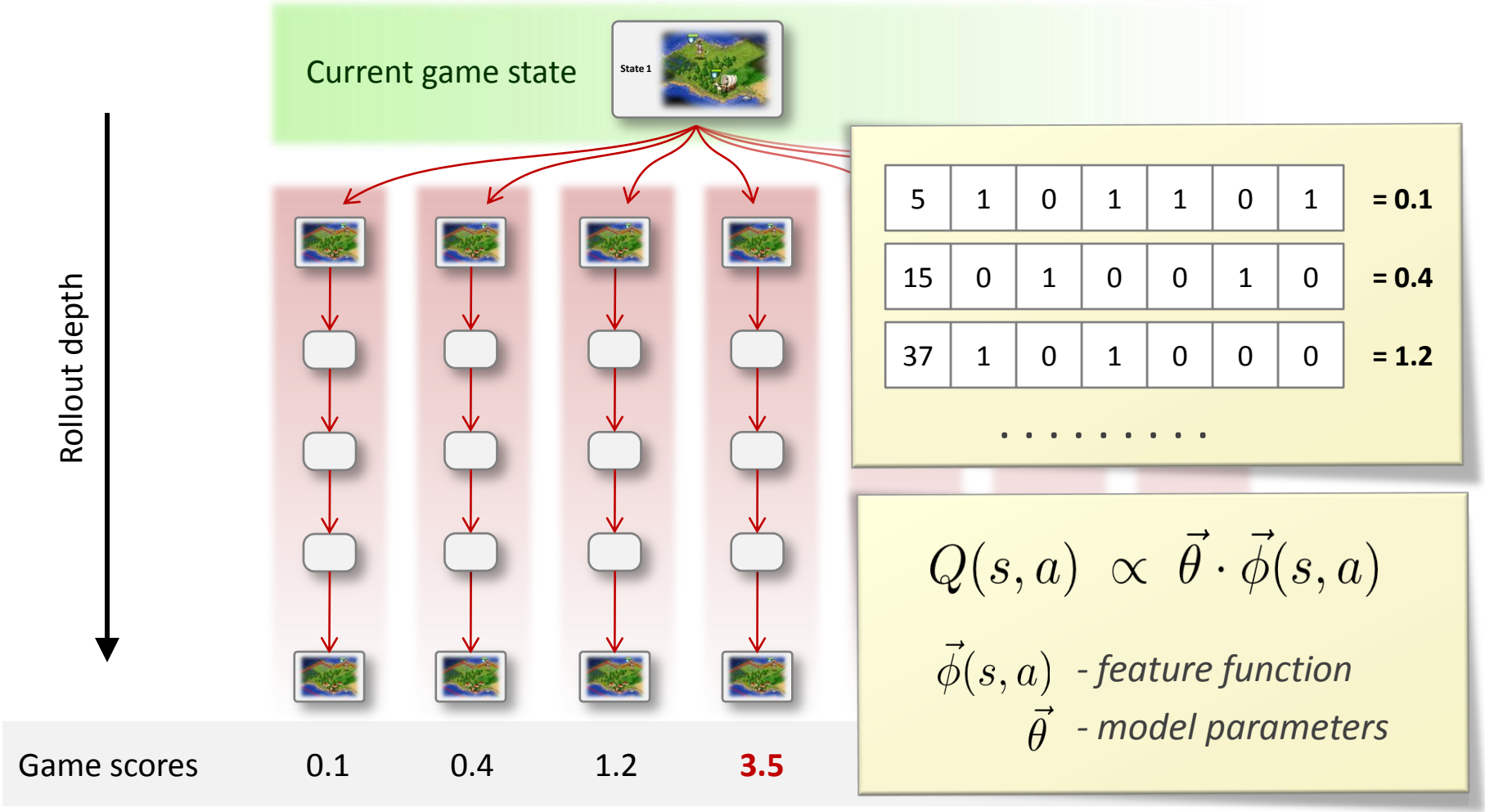
Try many candidate actions from current state & see how well they perform.



Monte-Carlo Search

Try many candidate actions from current state & see how well they perform.

Learn feature weights from simulation outcomes



Model Overview

Monte-Carlo Search Framework

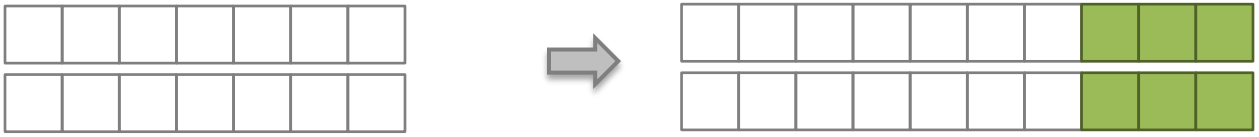
- *Learn action selection policy from simulations*

Our Algorithm

- *Bias action selection policy using text*
- *Learn text interpretation from simulation feedback*

Getting Advice from Text

Traditionally: Policy captures relation between state and action



$$Q(s,a) \propto \vec{\theta} \cdot \vec{\phi}(s,a) \qquad Q(s,a,\vec{w}) \propto \vec{\theta} \cdot \vec{\phi}(s,a,\vec{w})$$

Our approach: *Bias action selection policy using text*

➡ Enrich policy with text features

Modeling Requirements

- *Identify sentence relevant to game state*


Build cities near rivers or ocean.

- *Label sentence with predicate structure*

Build cities near rivers or ocean.

Build cities *near* ***rivers or ocean.***

- *Estimate value of candidate actions*

Build cities *near* ***rivers or ocean.***

Irrigate :	*-10*
Fortify :	*-5*
. . . .	
Build city :	*25*

Sentence Relevance

Identify sentence relevant to game state and action

State s*, candidate action* a*, document* d

$$p(y = y_i | s, a, d) \propto e^{\vec{u} \cdot \vec{\phi}(y_i, s, a, d)}$$

Sentence y_i *is selected as relevant*

Log-linear model:

- $\vec{u}$ - *weight vector*
- $\vec{\phi}(y_i, s, a, d)$ - *feature function*

Predicate Structure

Select word labels based on sentence + dependency info

E.g., "***Build cities*** *near* ***rivers or ocean.***"

Word index j*, sentence* y*, dependency info* q

$$p(e_j \mid j, y, q) \propto e^{\vec{v} \cdot \vec{\psi}(e_j, j, y, q)}$$

Predicate label e_j = { ***action***, ***state***, ***background*** }

Log-linear model:

- $\vec{v}$ - *weight vector*
- $\vec{\psi}(e_j, j, y, q)$ - *feature function*

Final Q function approximation

Predict expected value of candidate action

State s, candidate action a

$$Q(s, a, d, y_i, \vec{e}_i) = \vec{w} \cdot \vec{f}(s, a, d, y_i, \vec{e}_i)$$

Document d, relevant sentence y_i, predicate labeling $\vec{e}_i$

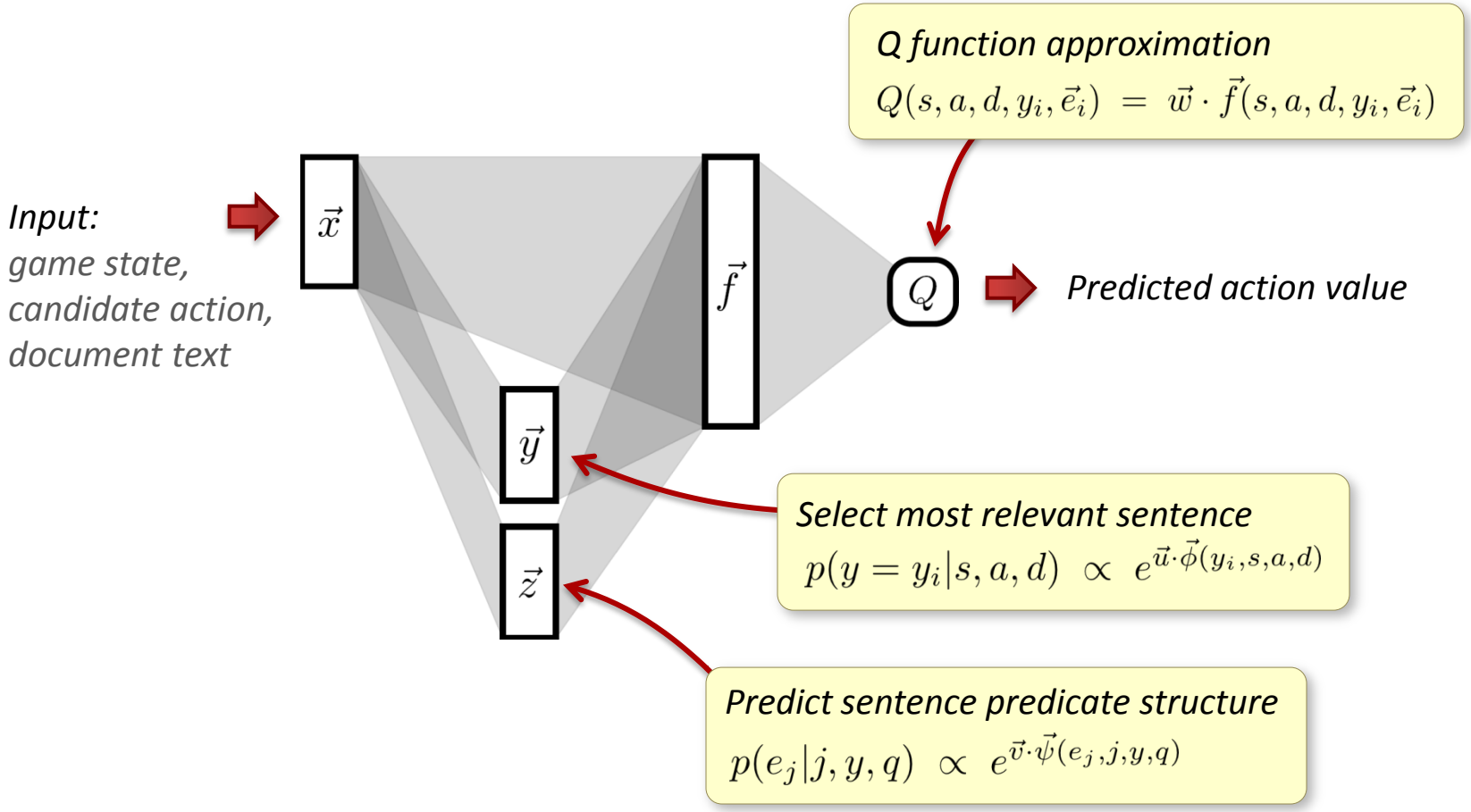
Linear model:

$\vec{w}$ - *weight vector*

$\vec{f}(s, a, d, y_i, \vec{e}_i)$ - *feature function*

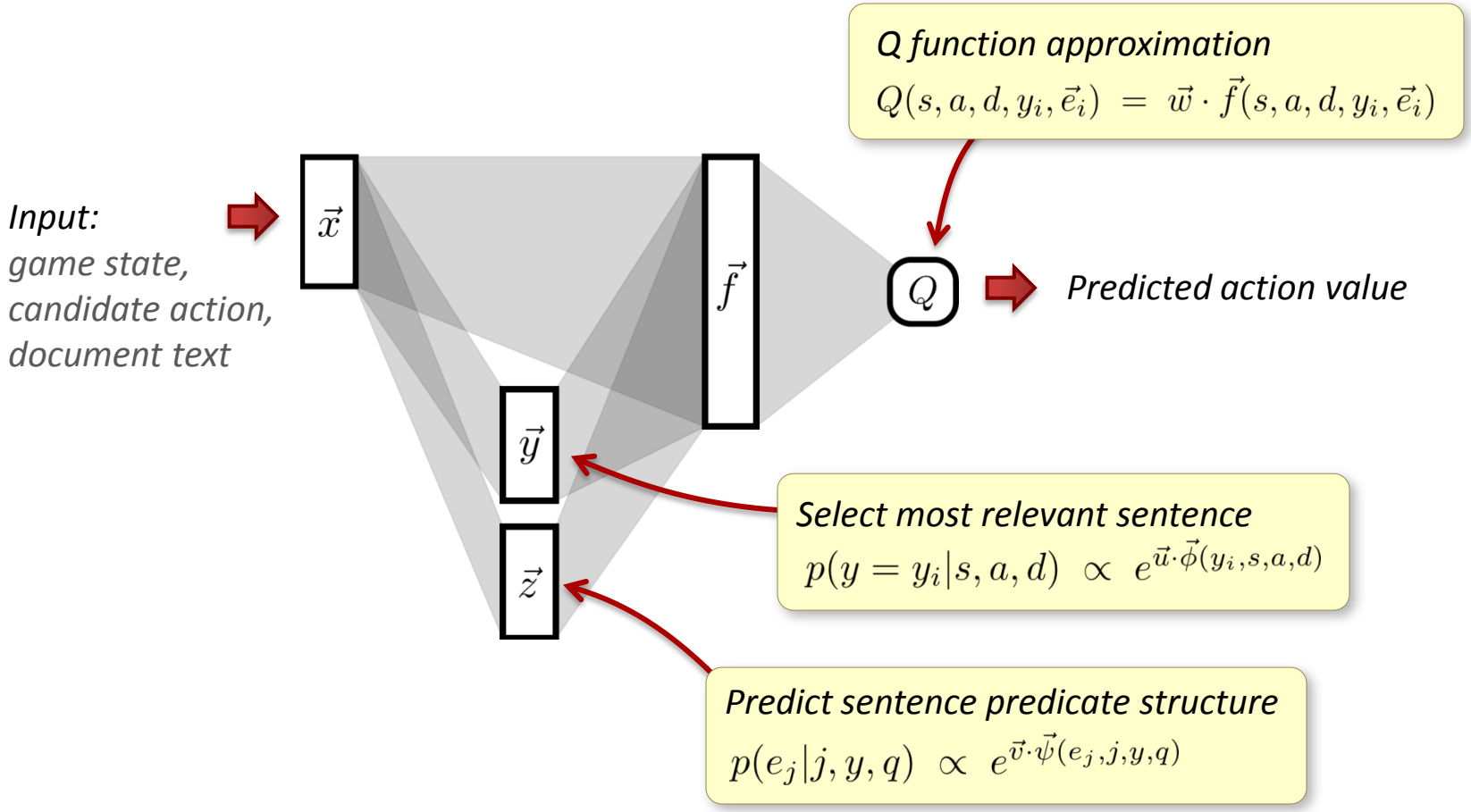
Model Representation

Multi-layer neural network: *Each layer represents a different stage of analysis*



Model Representation

Multi-layer neural network: *Each layer represents a different stage of analysis*

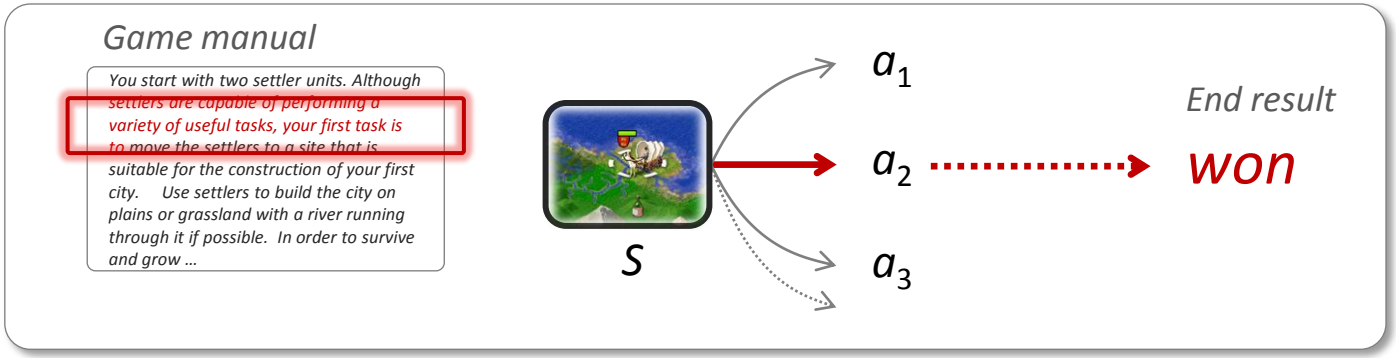


Learning from Game Feedback

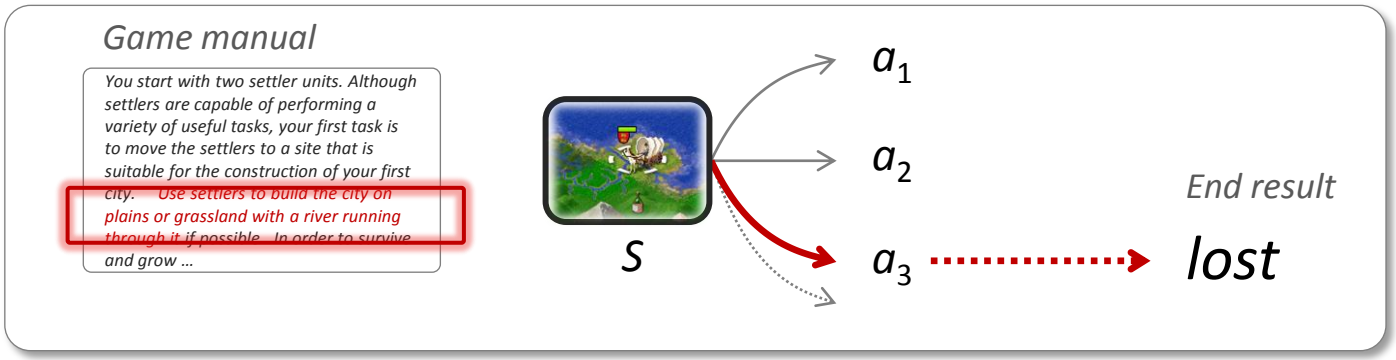
Goal: *Learn from game feedback as only source of supervision.*

Key idea: *Better parameter settings will lead to more victories.*

Model params: θ_1

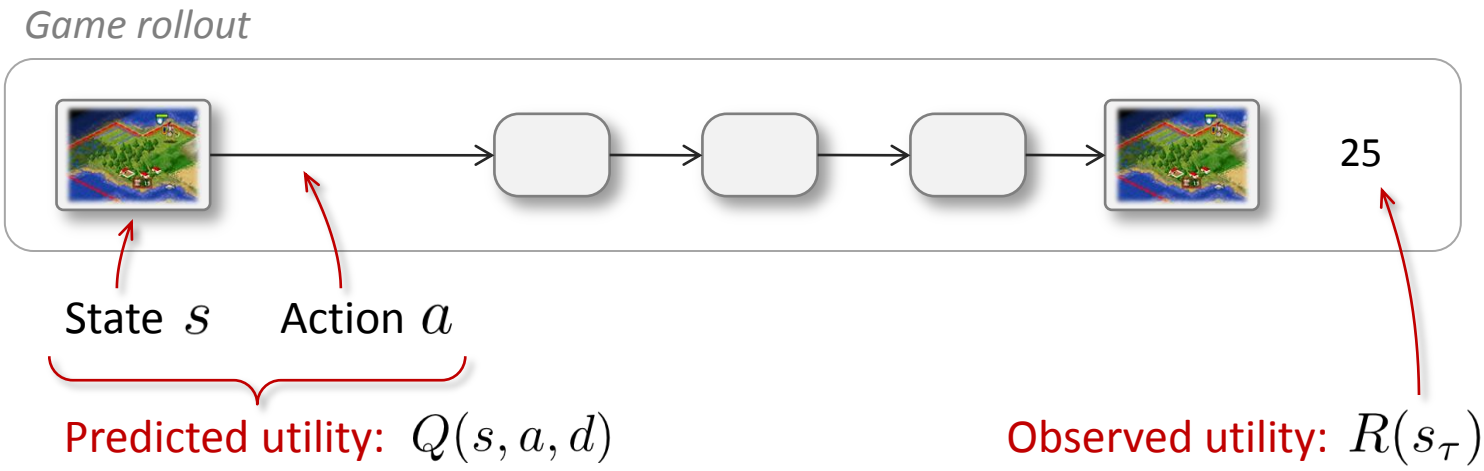


Model params: θ_2



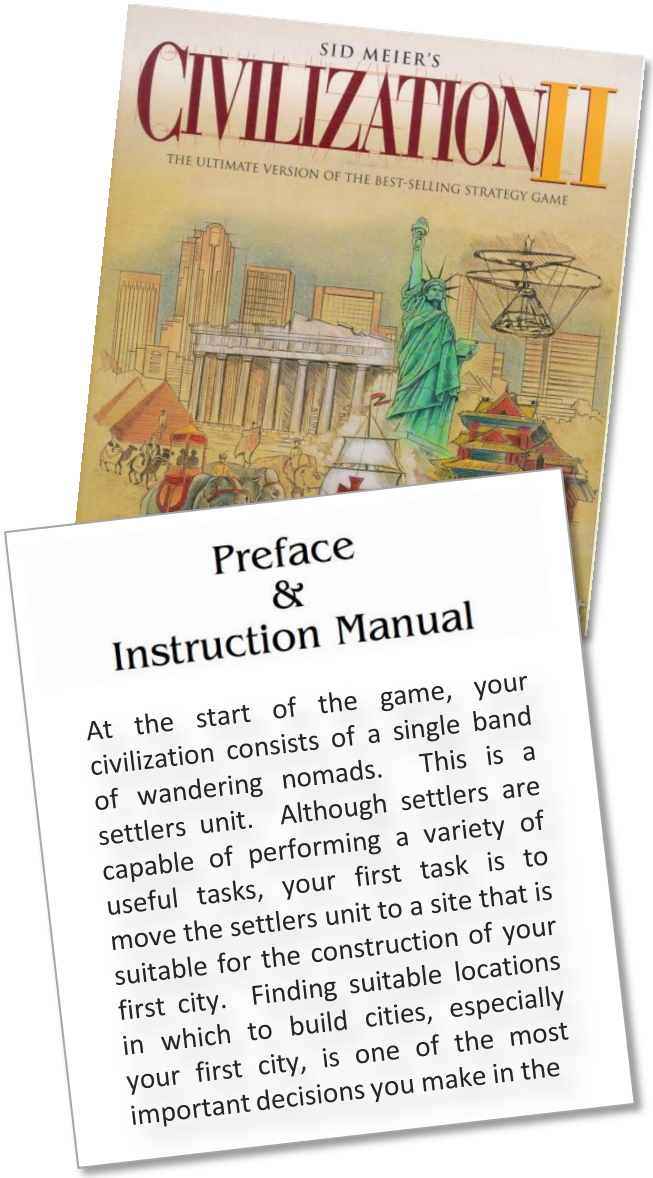
Parameter Estimation

Objective: Minimize *mean square error* between predicted utility $Q(s, a, d)$ and observed utility $R(s_\tau)$



Method: Gradient descent – i.e., Backpropagation.

Experimental Domain



Game:

- *Complex, stochastic turn-based strategy game Civilization II.*
- *Branching factor: 10^{20}*

Document:

- *Official game manual of Civilization II*

Text Statistics:

Sentences:	***2083***
Avg. sentence words:	***16.7***
Vocabulary:	***3638***

Experimental Setup

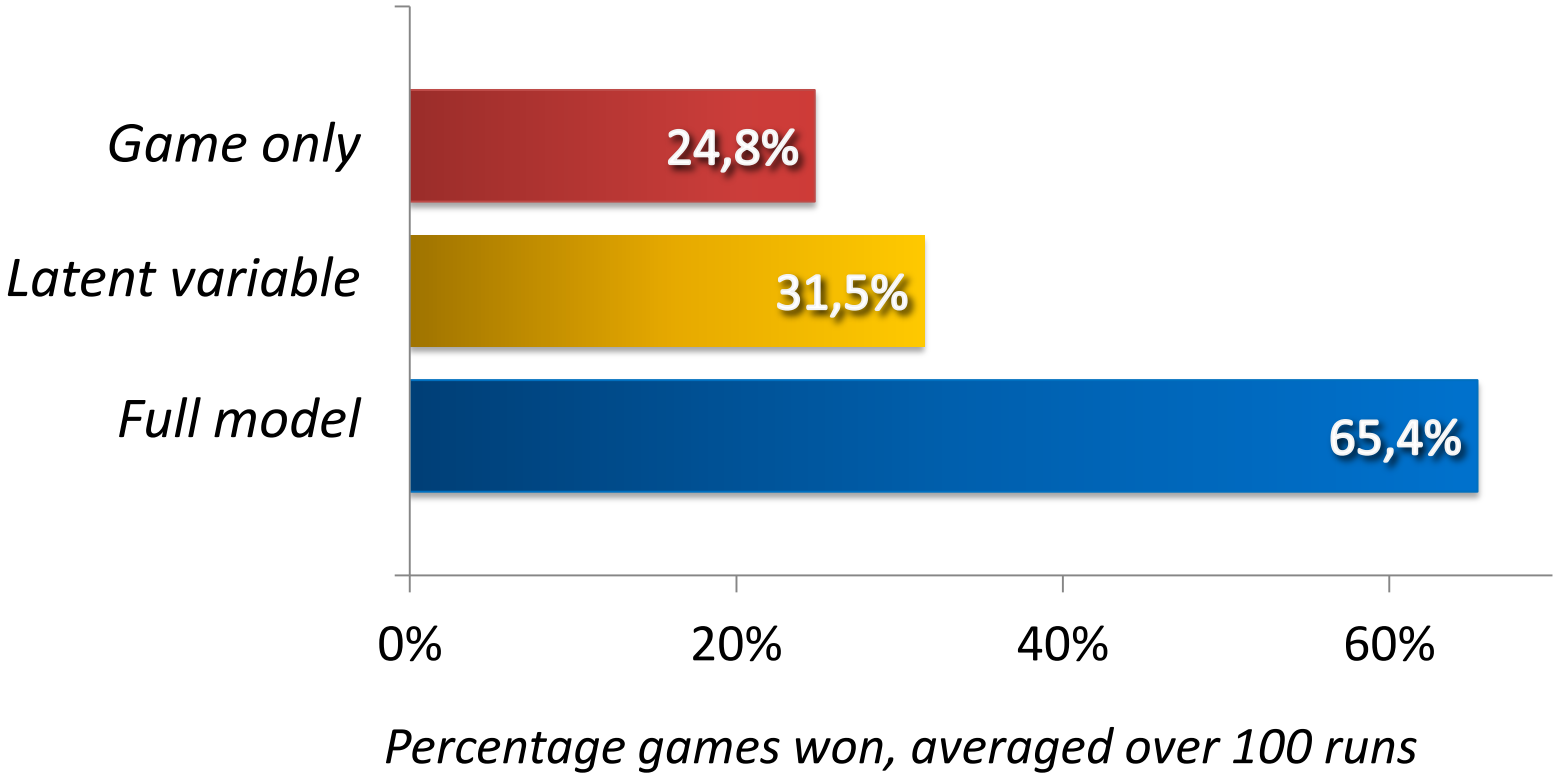
Game opponent:

- *Built-in AI of Game.*
- *Domain knowledge rich AI, built to challenge humans.*

Evaluation:

- *Full games won.*
- *Averaged over 100 independent experiments.*
- *Avg. experiment runtime: 4 hours*

Results: Full Games



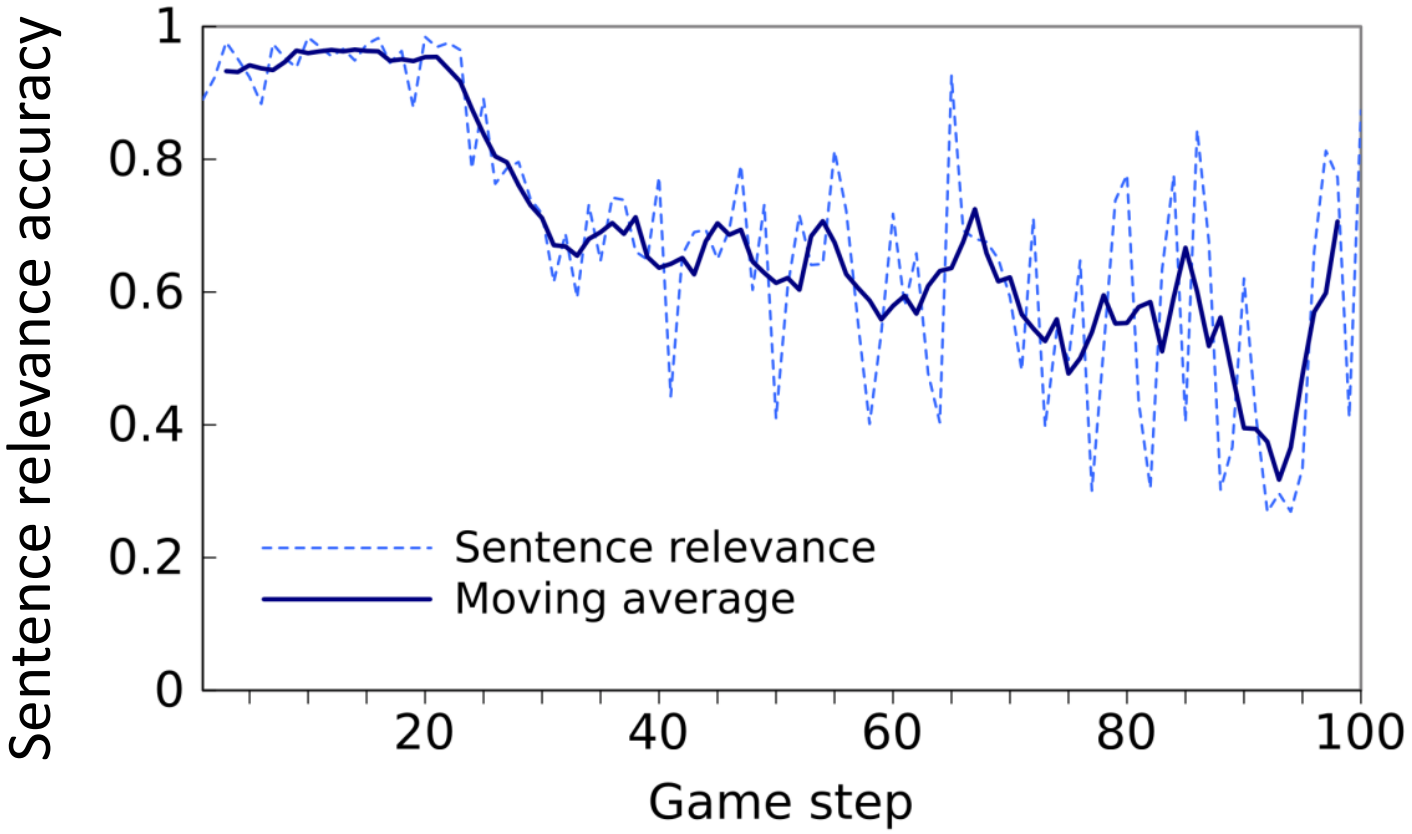
Results: Sentence Relevance

Problem: *Sentence relevance depends on game state.*
States are game specific, and not known a priori!

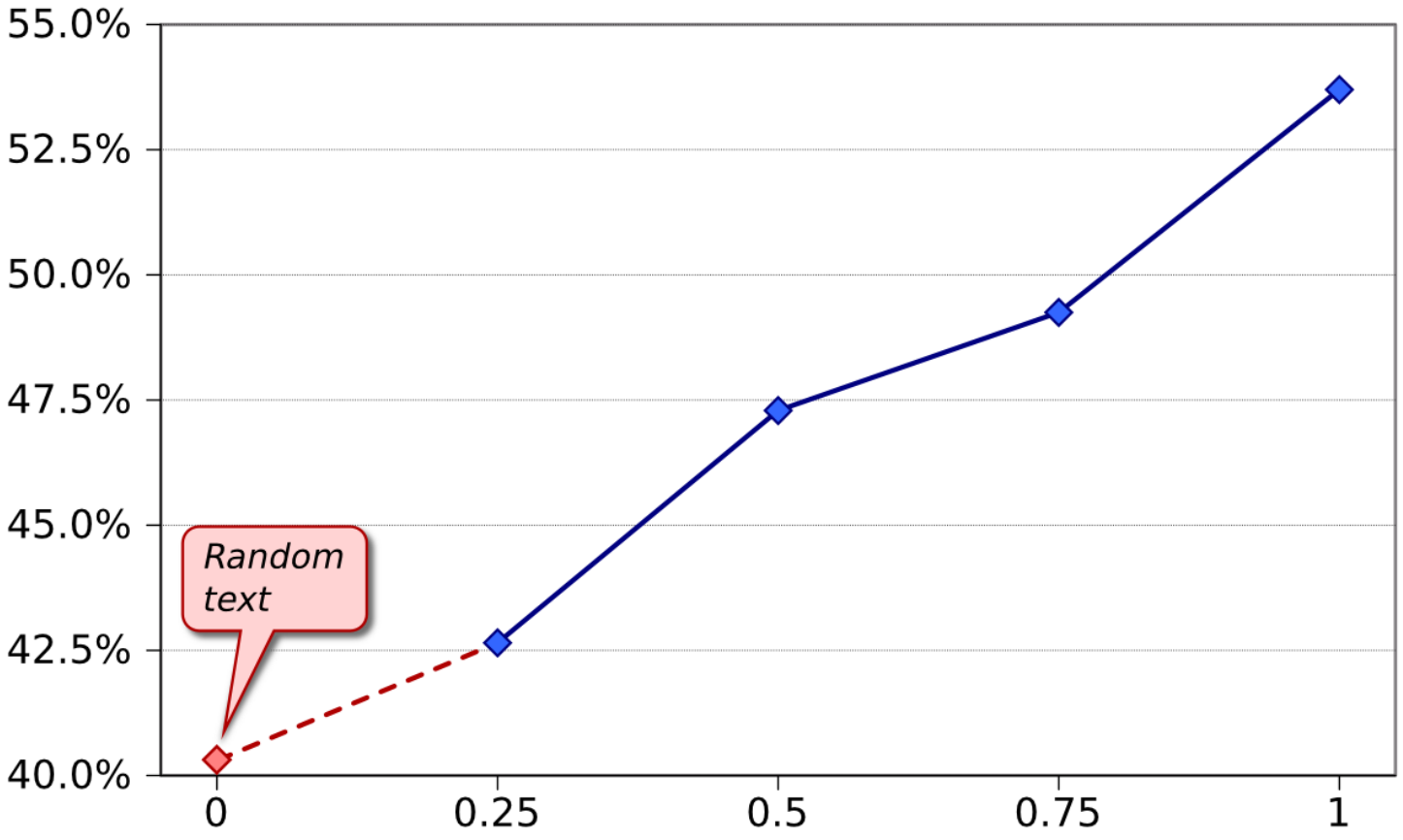
Solution: *Add known non-relevant sentences to text.*
E.g., sentences from the Wall Street Journal corpus.

Results: ***71.8%*** *sentence relevance accuracy…*
Surprisingly poor accuracy given game win rate!

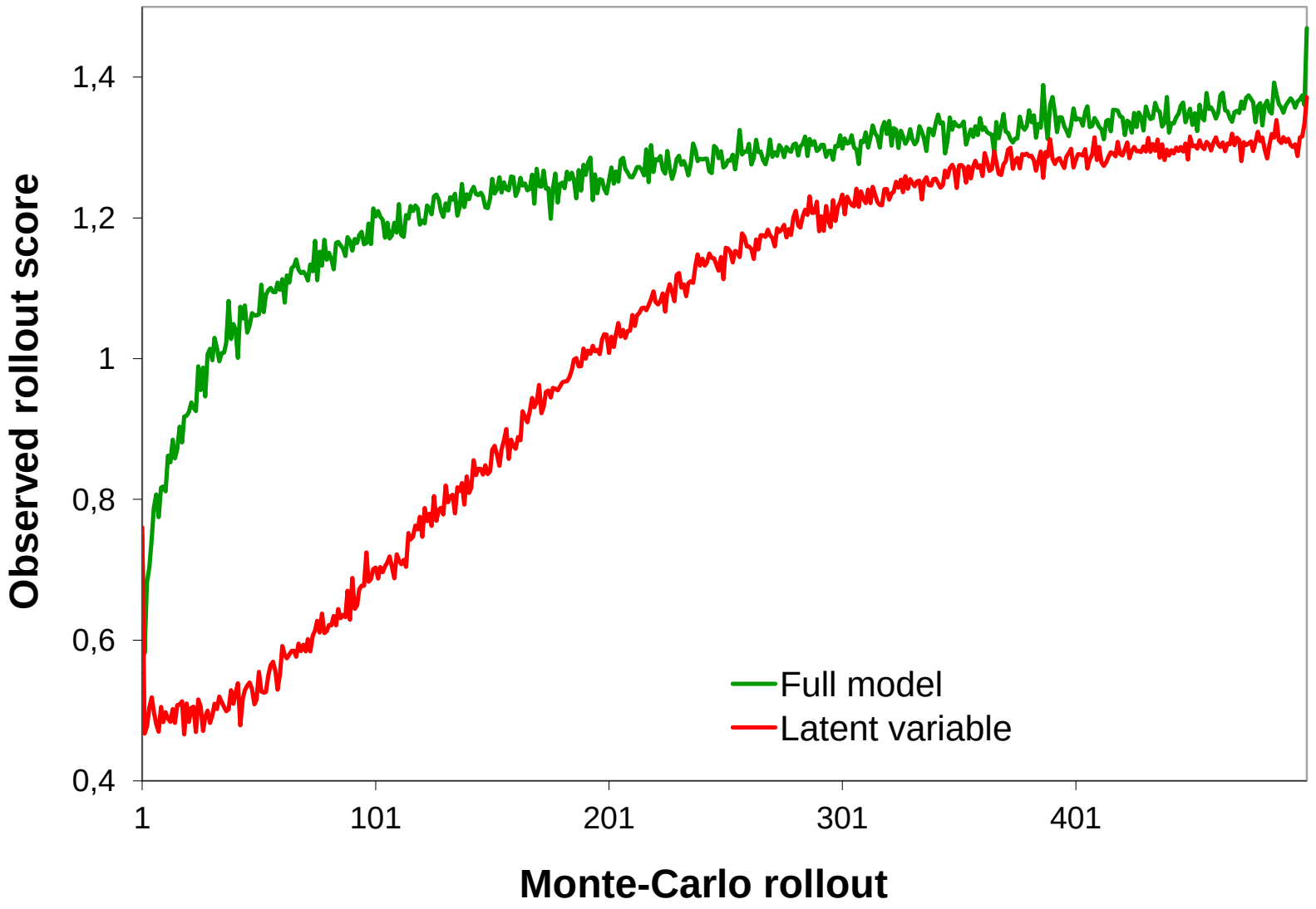
Results: Sentence Relevance



Good Advice Helps!



Performance as a function of ratio of relevant sentences


Observed rollout score
1,4
1,2
1
0,8
0,6
0,4
1
101
201
301
401
Monte-Carlo rollout
Full model
Latent variable

Outline

1. Step-by-step imperative instructions
2. High-level strategy descriptions
3. General descriptions of world dynamics
   - ***Abstractions***
   - *Situational relevance*
   - *Incompleteness*
   - *Learning from control feedback*

Solving Hard Planning Tasks

Objective: Compute plan to achieve given goal

Challenge: Exponential search space

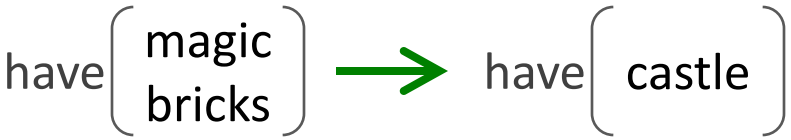
Traditional solution: Analyze domain structure to induce sub-goals

Our goal: Use precondition information from text to guide sub-goal induction

Precondition/Effects Relationships

Castles are built with magic bricks

Classical Planning:



NLP: *Discourse Relation*

Goal: Show that planning can be improved by utilizing precondition information in text

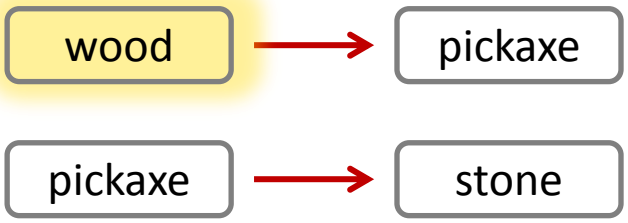
How Text Can Help Planning

Minecraft : Virtual world allowing tool creation and complex construction.

Text

A **pickaxe**, which is used to harvest **stone**, can be made from **wood**.

Preconditions



Plan

- Move to location: <3,3>
- Harvest: wood
- Retrieve: harvested wood

wood

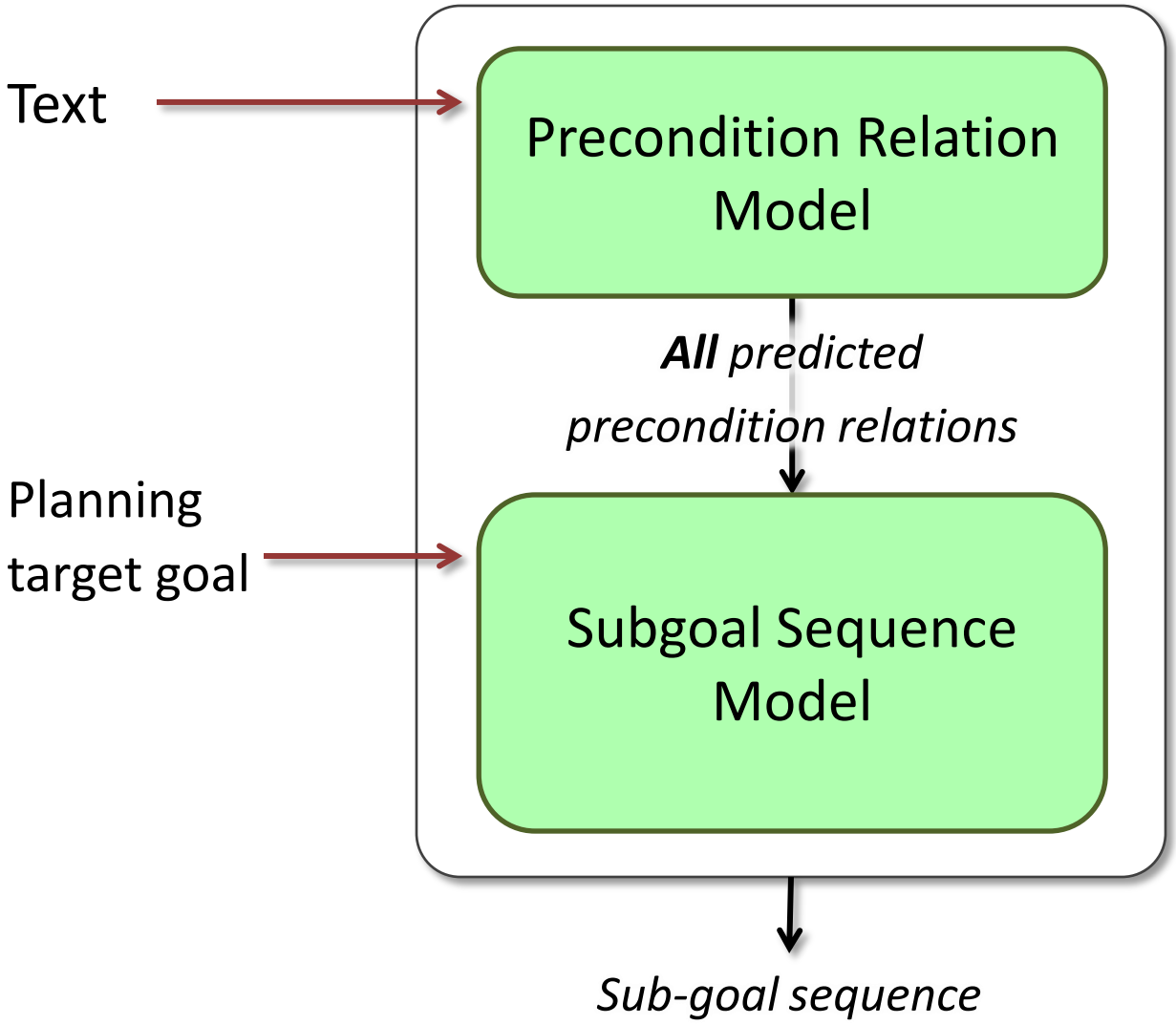
- Setup crafting table
- Place on crafting table: wood
- Craft: pickaxe
- Retrieve: pickaxe

pickaxe

- Move to location: <1,2>
- Pickup tool: pickaxe
- Harvest: stone with: pickaxe
- Retrieve: stone

stone

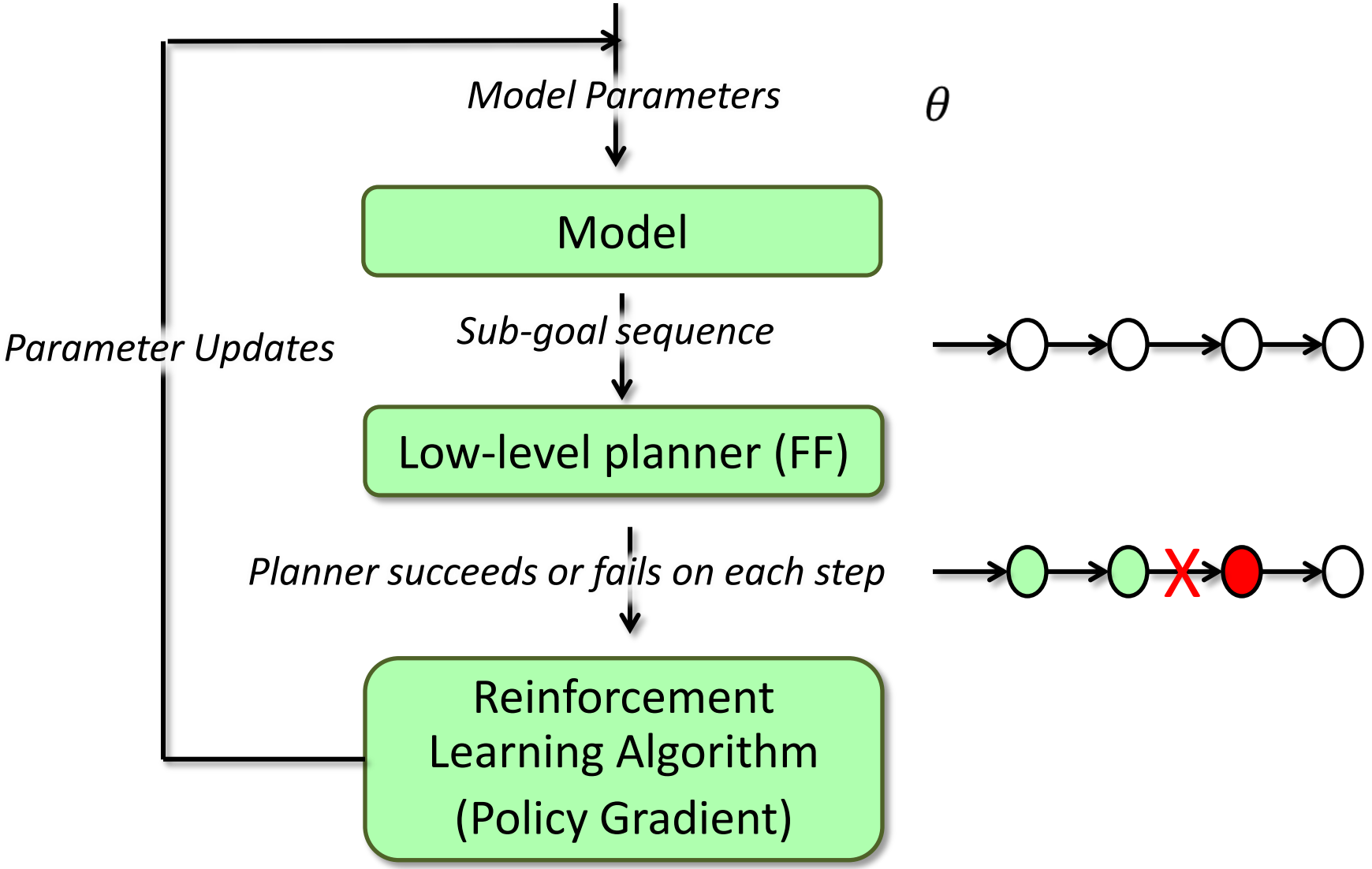
Solution: Model both the relations in the text as well as the relations in the world itself



Given text predicts all domain preconditions (independent of the goal)

Starting with the goal state work backwards, predicting the previous subgoal given the precondition relations and the current subgoal

Learn Parameters Using Feedback from the World



Experimental Domain

World:

Minecraft virtual world

Documents:

User authored wiki articles

Text Statistics:

Sentences:	*242*
Vocabulary:	*979*

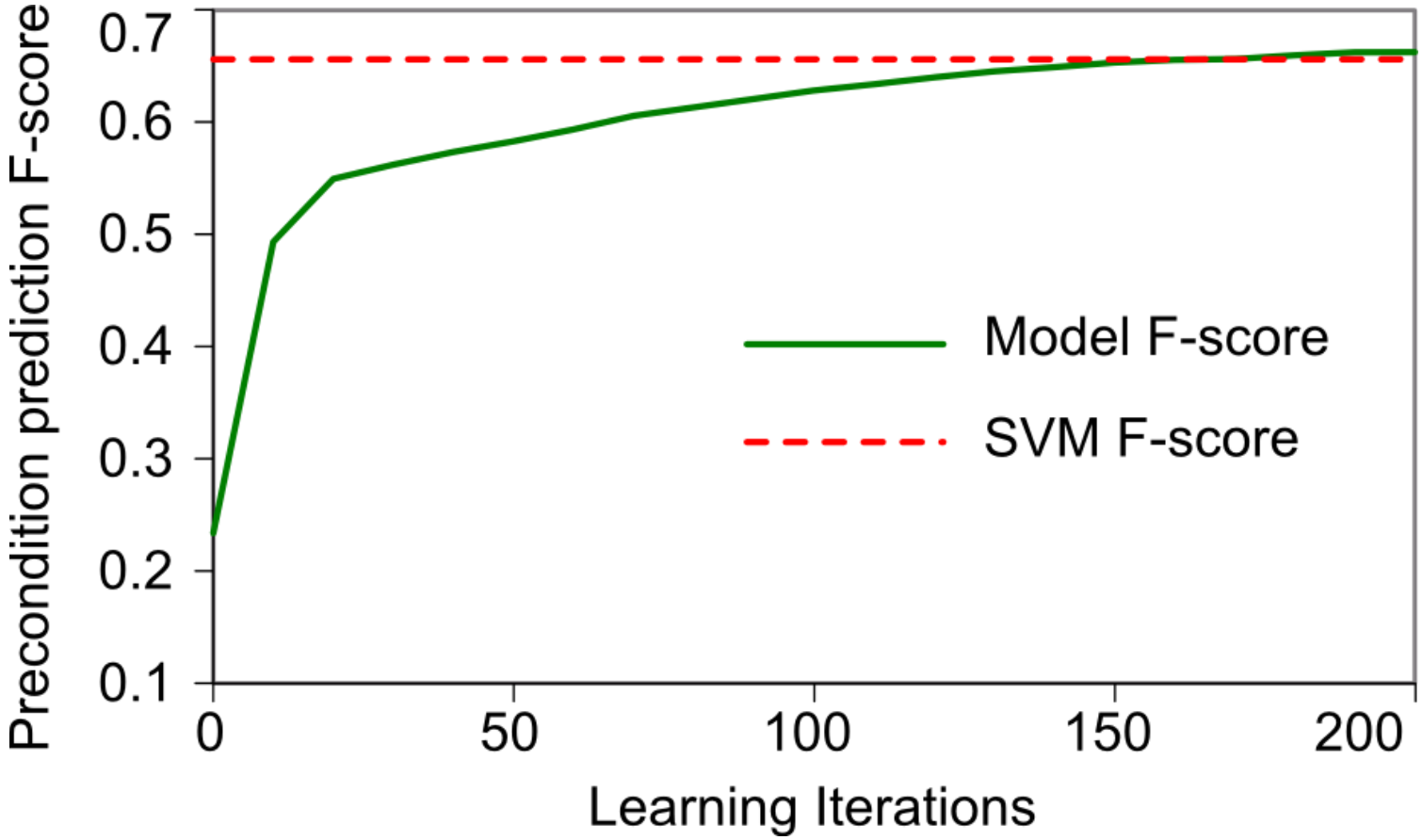
Planning task Statistics:

Tasks:	*98*
Avg. plan length:	*35*

Results

Method	% plans solved
Low-level planner (FF)	40.8
No text	69.4
Full model	**80.2**
Manual text connections	84.7

Results: Text Analysis



Conclusions

- Human knowledge encoded in natural language can be automatically leveraged to improve control applications.
- Environment feedback is a powerful supervision signal for language analysis.
- Method is applicable to control applications that have an inherent success signal, and can be simulated.

Code, data & experimental framework available at:
http://groups.csail.mit.edu/rbg/code/civ

- Full model with relevant sentences removed (sentences identified as relevant less than 5 times): **20%** (after 30/200 runs)