Grounding Linguistic Analysis in Control Applications

S.R.K. Branavan

February 8, 2012

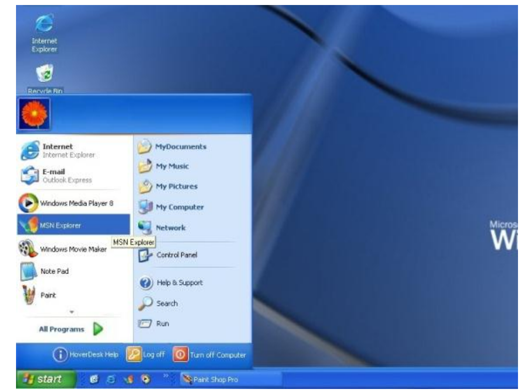
Language and Control Applications

Windows help document

Right click "*My Computer*" on the desktop, and click the *Manage* menu option.

Click *Services* after expanding "*Services and applications*"

Microsoft Windows



Game user guide

...

Cities built on or near water sources can irrigate to increase their crop yields.

...

Strategy game

Our Goal

Leverage language-control connection to:

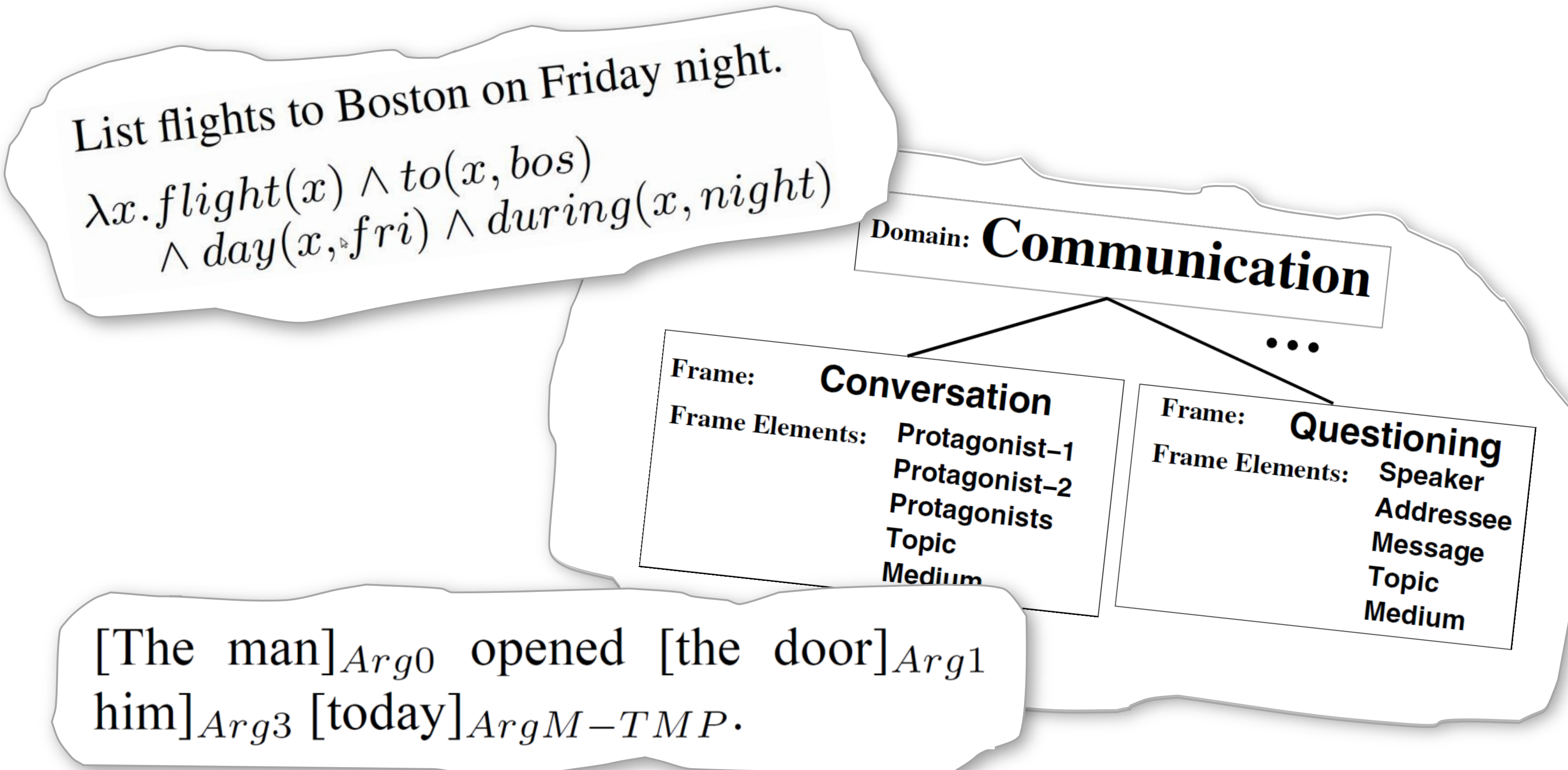
- *Learn language analysis from control feedback*

- *Improve control performance using textual information*

Text interpretation can enable novel automation

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

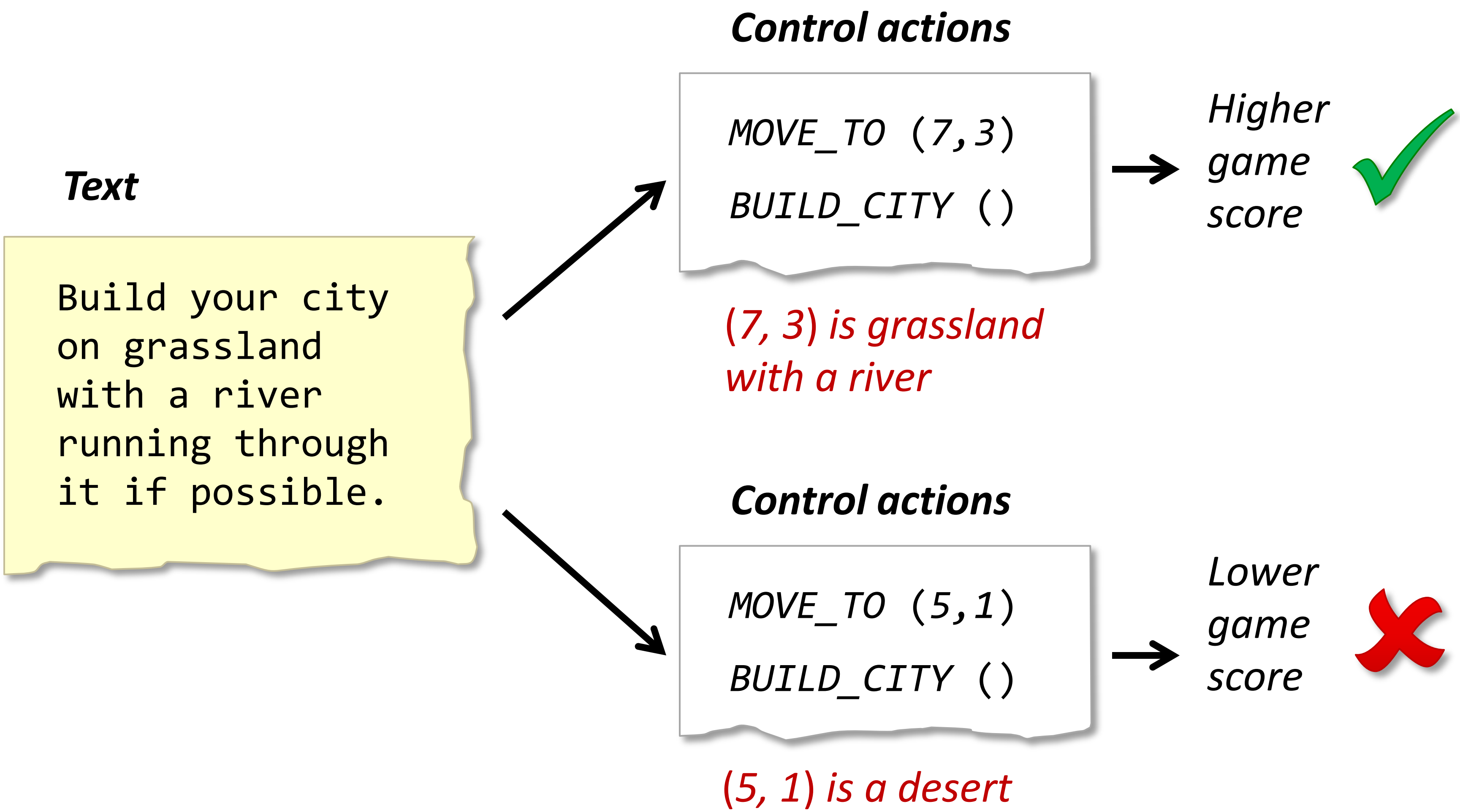
```
MOVE_TO (7,3)

BUILD_CITY ()

...
```

Enables language learning from control feedback

Learning from Control Feedback



Learn Language via Reinforcement Learning

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

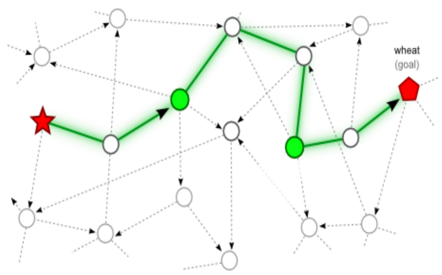
  > *"Build your city on grassland with a river running through it if possible."*
  >
  > *(what if there are no rivers nearby?)*

Contributions

- Learn language analysis from control feedback
- Improve control performance using text information

Instruction Interpretation

Complex Game-play



High-level Planning

General Setup

Input:

- *Text documents*
- *Interactive access to control application*

Prior knowledge:

- *Text provides information useful for the control application*

Goal:

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

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
   - *Abstractions*
   - *Situational relevance*
   - *Incompleteness*
   - *Learning from 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
   - *Abstractions*
   - *Situational relevance*
   - *Incompleteness*
   - *Learning from control feedback*

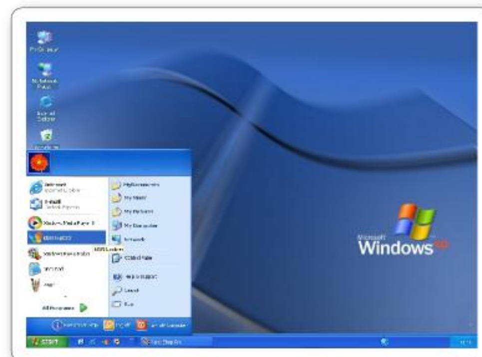
Interpreting Instructions to Actions

Input

Instructions:
step-by-step descriptions of commands

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 commands need to be executed



Output

Command sequence
executable in the environment

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

Instruction Interpretation: Challenges

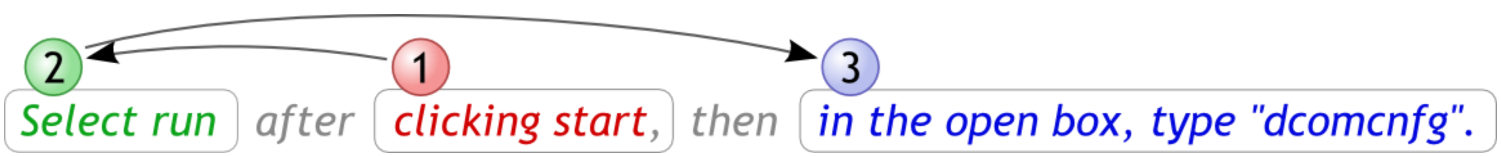
Segment text to chunks that describe individual commands

[*Select run*] *after* [*clicking start*,] *then* [*in the open box, type "dcomcnfg".*]

Learn translation of words to environment commands

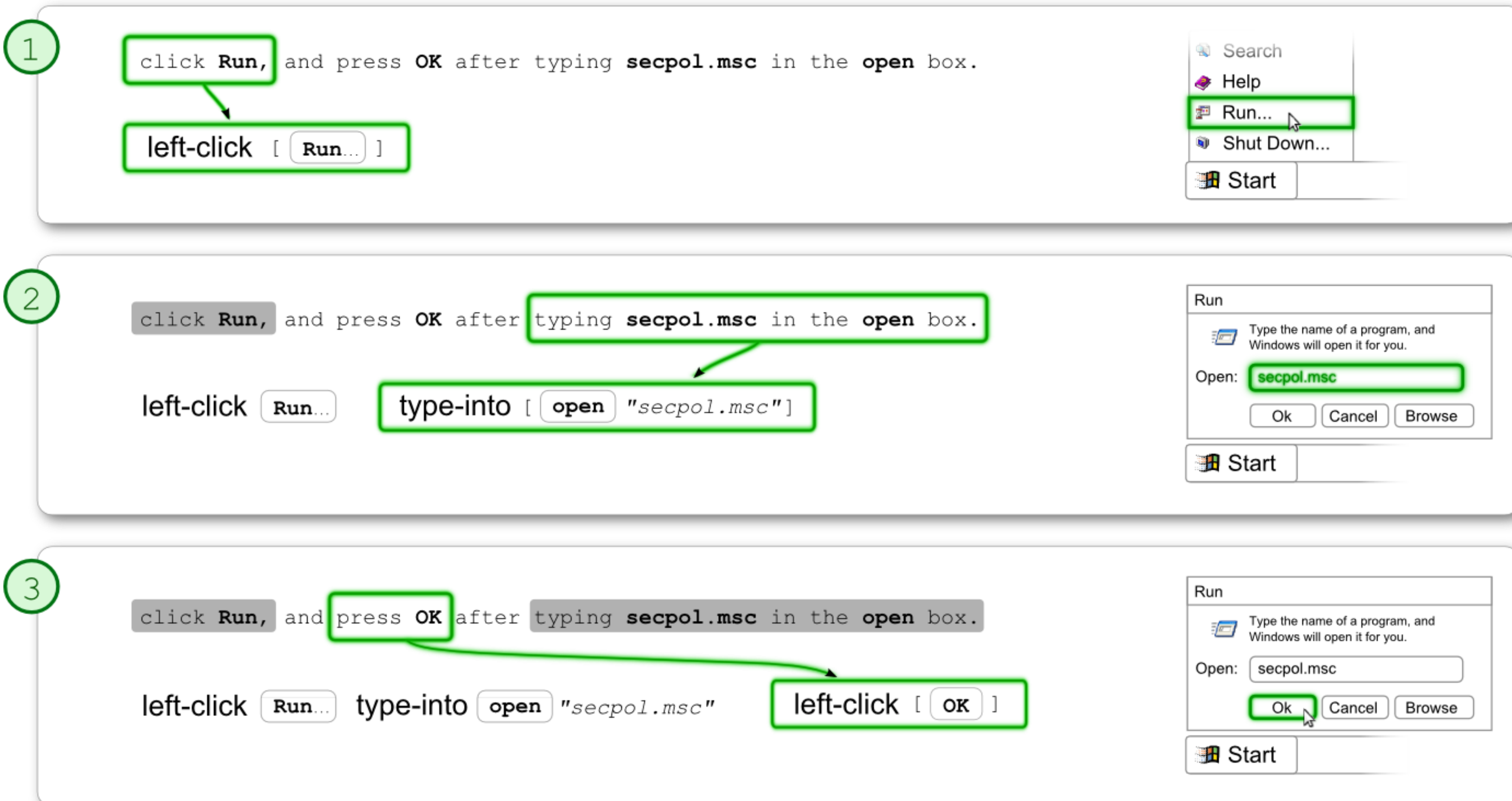
"select run" ➡ *LEFT_CLICK* [run]

Reorder environment commands



Instruction Interpretation: Representation

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



Instruction Interpretation: Representation

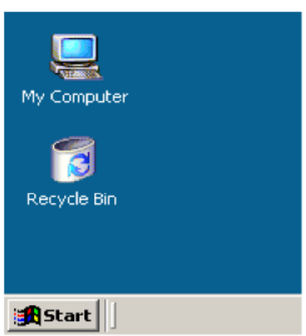
State s = Observed Text + Observed Environment

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)

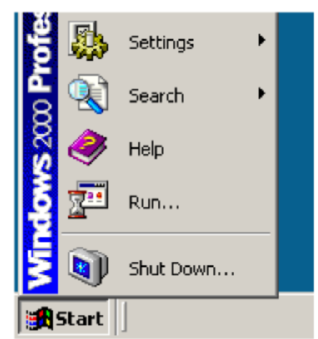
Policy function

$$p(\,a \mid s\,)$$

State 2

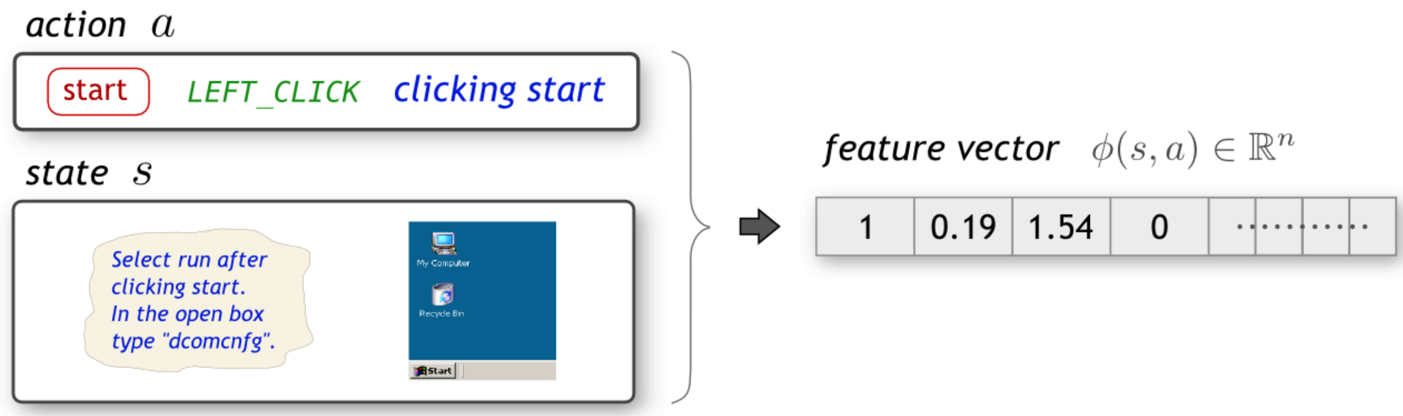
Observed text and environment

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



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*

Reward Signal

Ideal:

Test for task completion

Alternative Indication of Error:

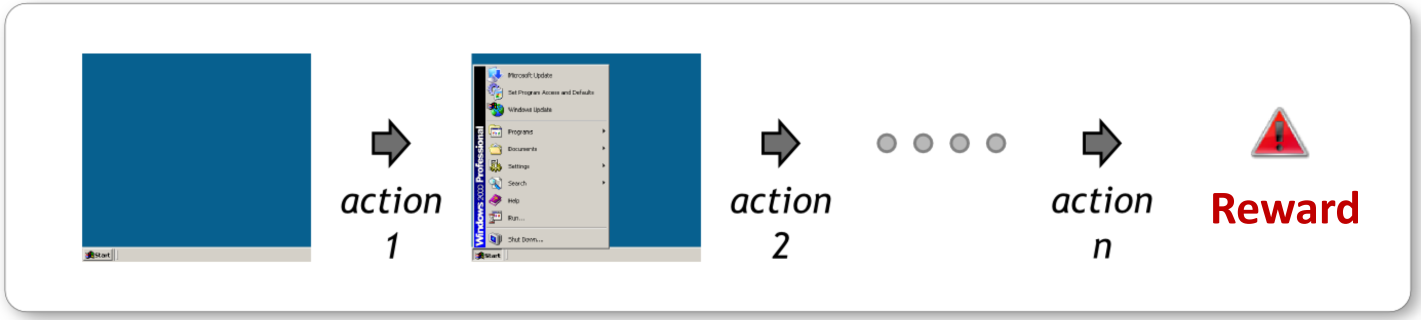
Text specifies objects not visible on the screen

Approximation:

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

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

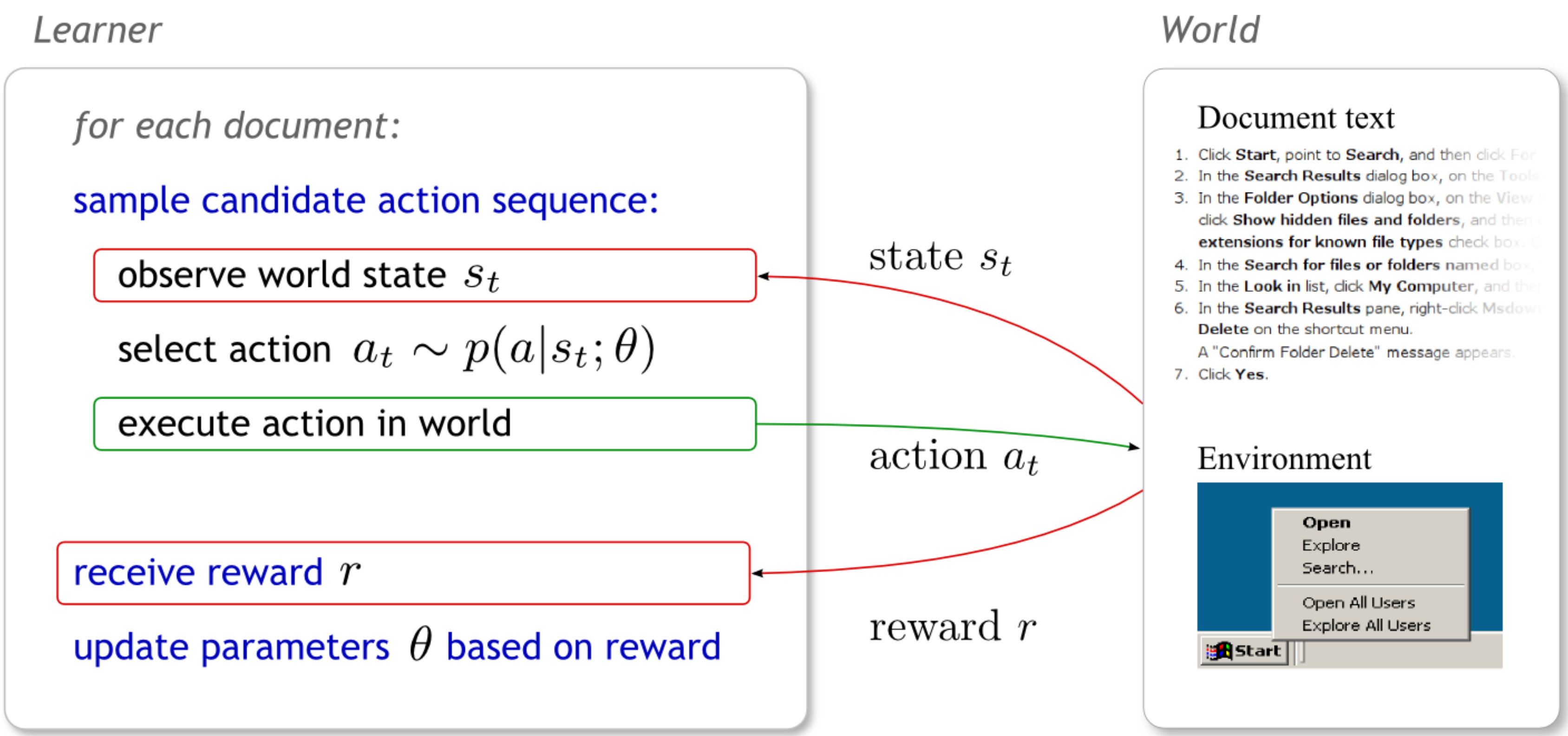
⇒ How can this space be effectively searched?

Use Reinforcement Learning

Learning Algorithm

Goal: ***Find θ that maximizes the expected reward***

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

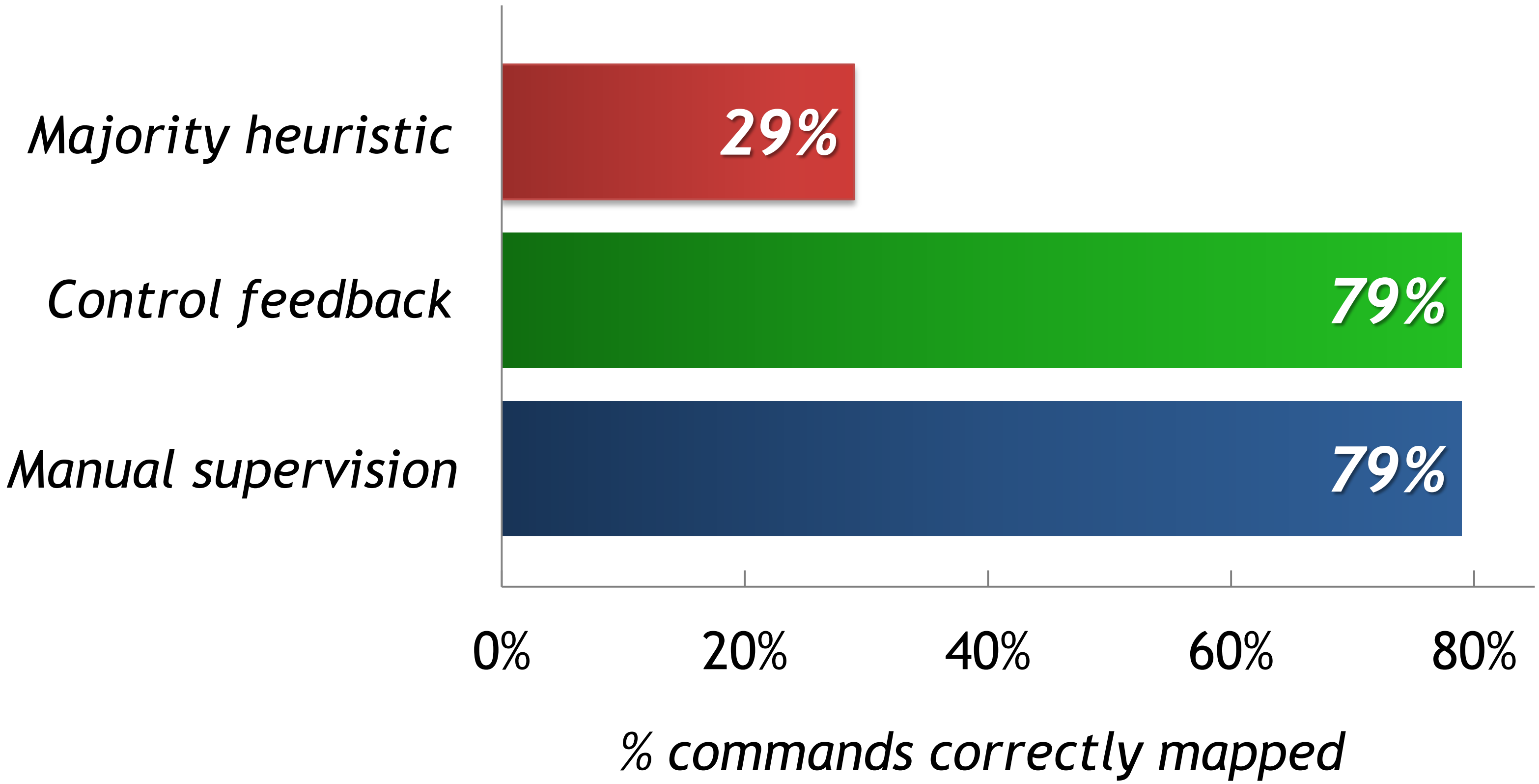


Windows Configuration Application

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*

Results: Command Accuracy



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
   - *Abstractions*
   - *Situational relevance*
   - *Incompleteness*
   - *Learning from control feedback*

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

Adversarial Planning Problem

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

Traditional Approach: Monte-Carlo Search Framework

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

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.

*Move the **settler** to a site suitable for **building** a **city**, onto grassland with a river if possible.*

→ `move_settlers_to ()` ?

→ `settlers_build_city ()` ?

Move the settler *to a site suitable for building a city, onto grassland with a river if possible.*

→ `move_settlers_to ()`

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

Leveraging Textual Advice: Challenges

2. Label sentences with predicate structure.

Move *settler*
building *city*

→ `move_settlers_to ()` **?**

→ `settlers_build_city ()` **?**

Move the settler *to a site suitable for building a city, onto grassland with a river if possible.*

→ `move_settlers_to ()`

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

Leveraging Textual Advice: Challenges

2. Label sentences with predicate structure.

Move *the* ***settler*** *to a site suitable for* ***building*** *a* ***city****, onto grassland with a river if possible.*

→ move_settlers_to () ?

→ settlers_build_city () ?

Move the settler *to a site suitable for building a city, onto grassland with a river if possible.*

→ move_settlers_to ()

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.

S

a_1 – move_settlers_to (7,3)

a_2 – settlers_build_city ()

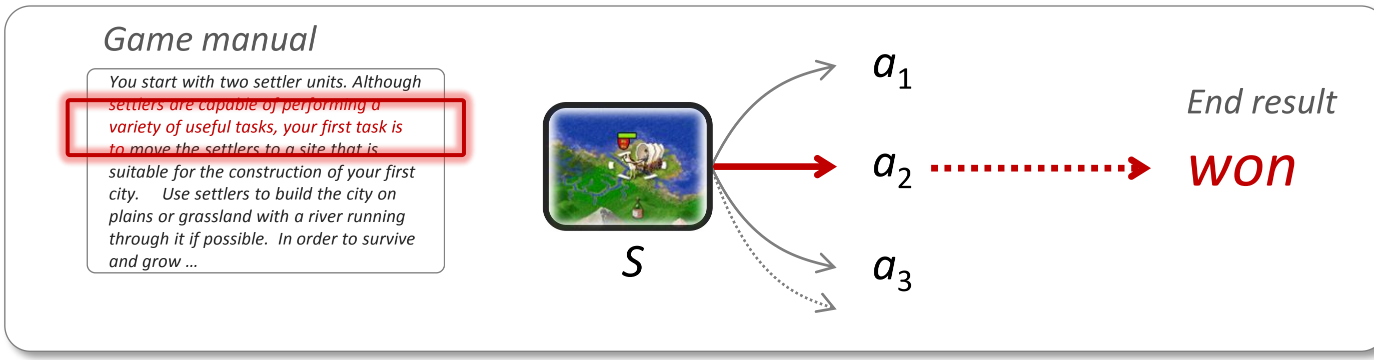
a_3 – settlers_irrigate_land ()

Learning from Game Feedback

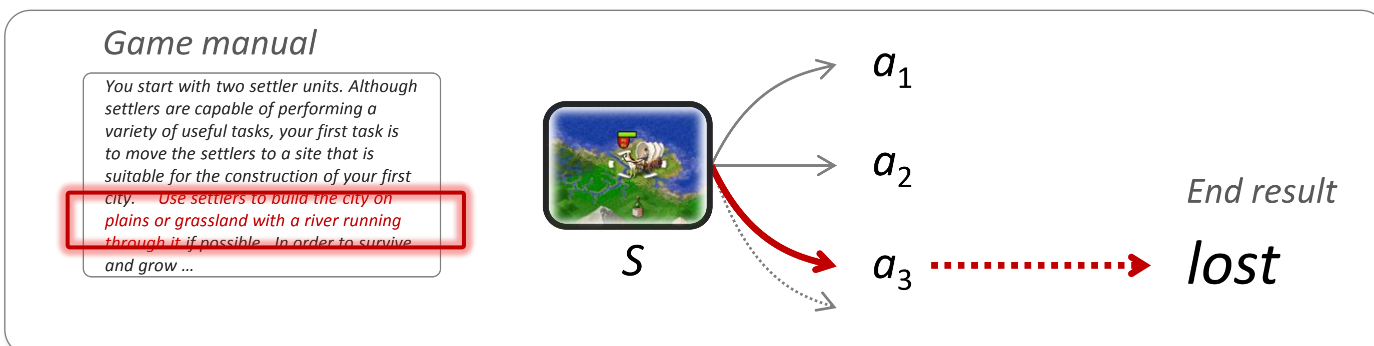
Goal: *Learn from game feedback as only source of supervision.*

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

Model params: θ_1



Model params: θ_2



Model Overview

Monte-Carlo Search Framework

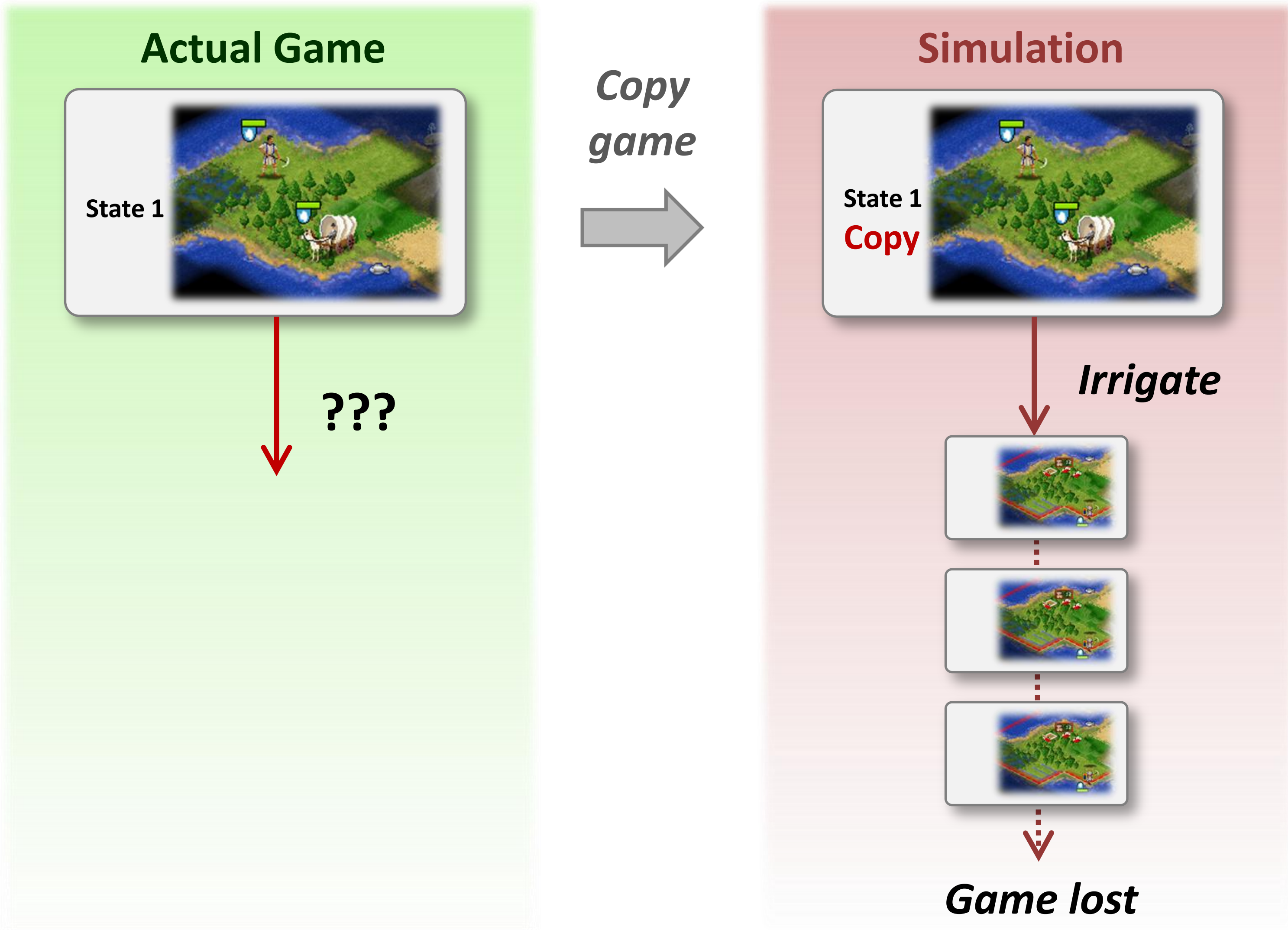
- *Learn action selection policy from simulations*
- *Very successful in complex games like Go and Poker.*

Our Algorithm

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

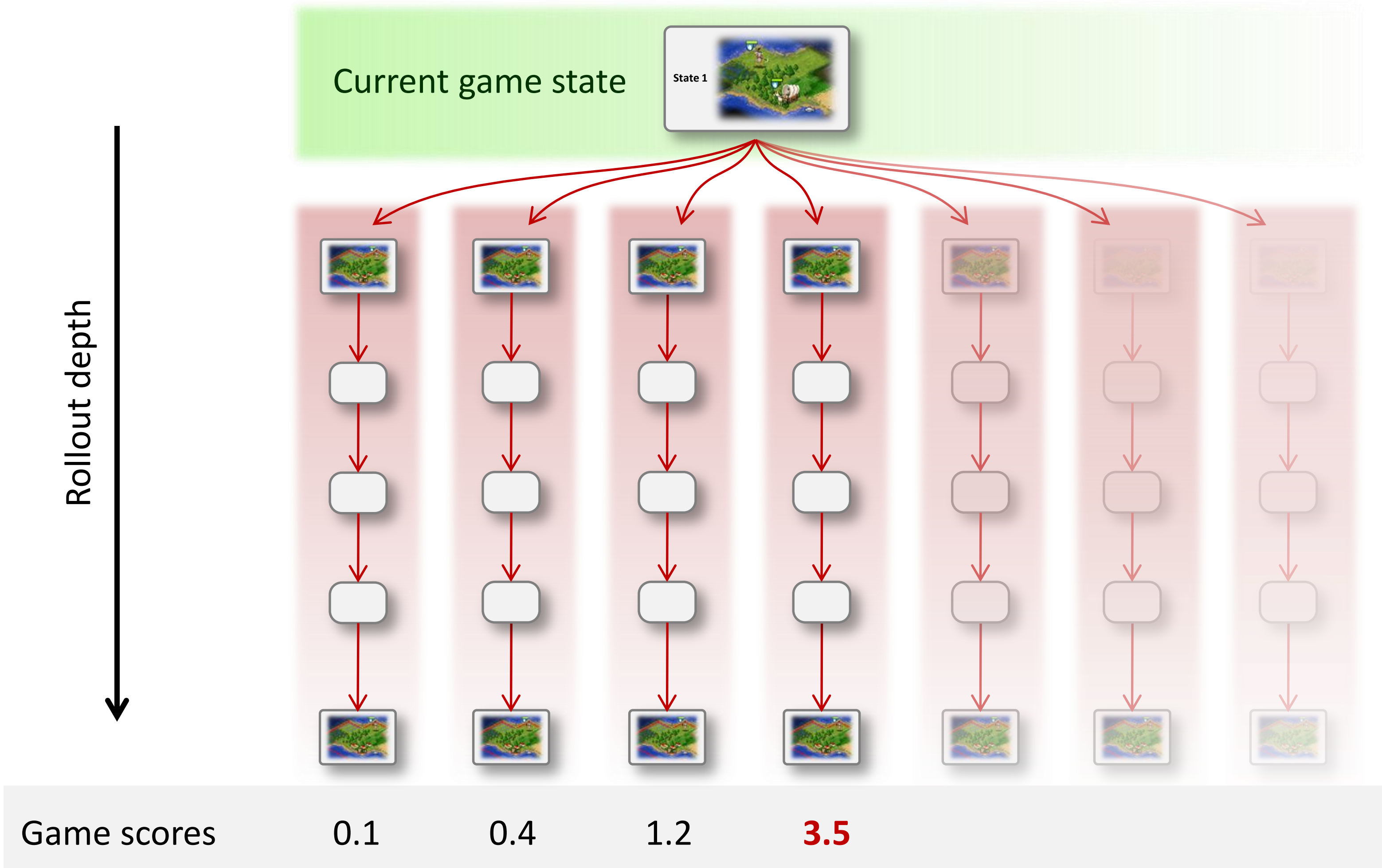
Monte-Carlo Search

Select actions via simulations, game and opponent can be stochastic



Monte-Carlo Search

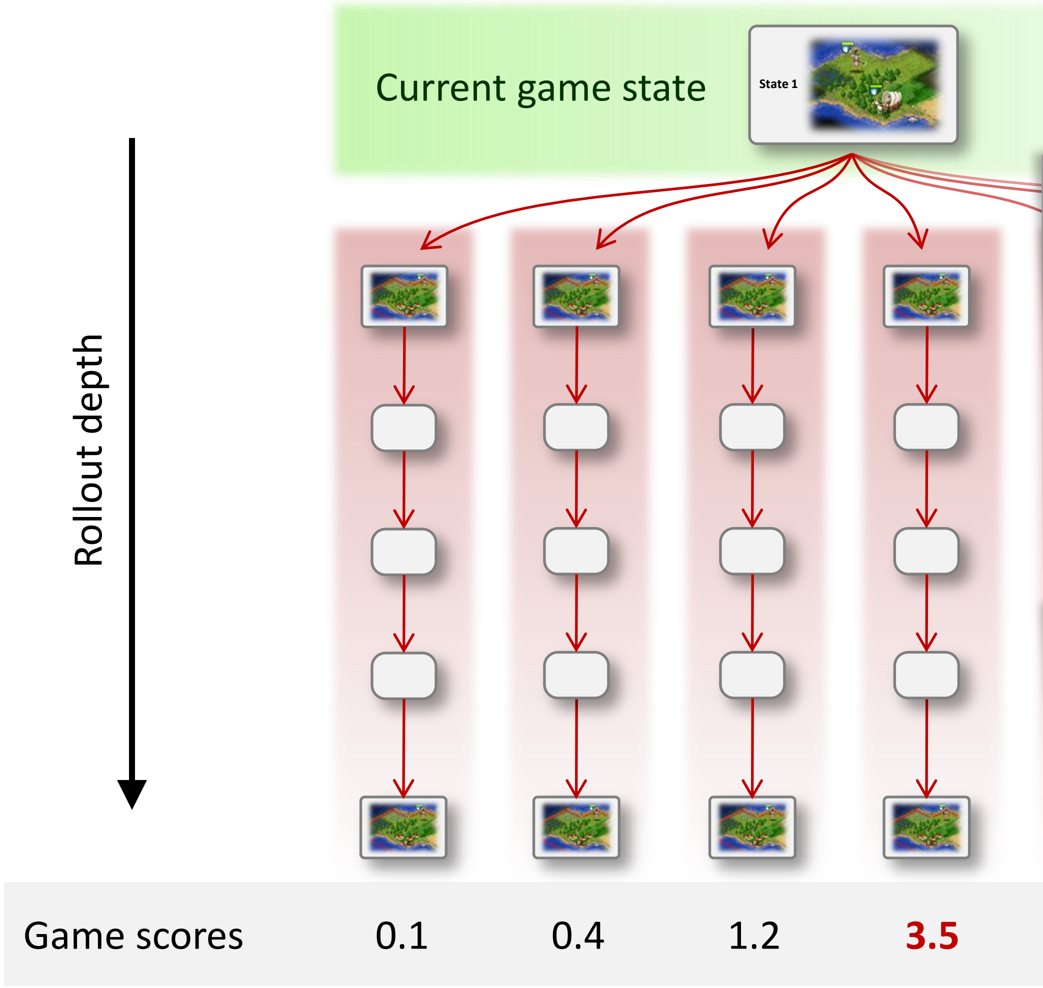
Try many candidate actions from current state & see how well they perform.



Monte-Carlo Search

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

Learn feature weights from simulation outcomes



5	1	0	1	1	0	1	**= 0.1**
15	0	1	0	0	1	0	**= 0.4**
37	1	0	1	0	0	0	**= 1.2**

. . . .

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

$\vec{\phi}(s,a)$ - *feature function*

$\vec{\theta}$ - *model parameters*

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*

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

1

2

3

Select word labels based on sentence + dependency info

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

Word index j, sentence y, dependency info q

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

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

1
2
3

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 $\vec{e}_i$

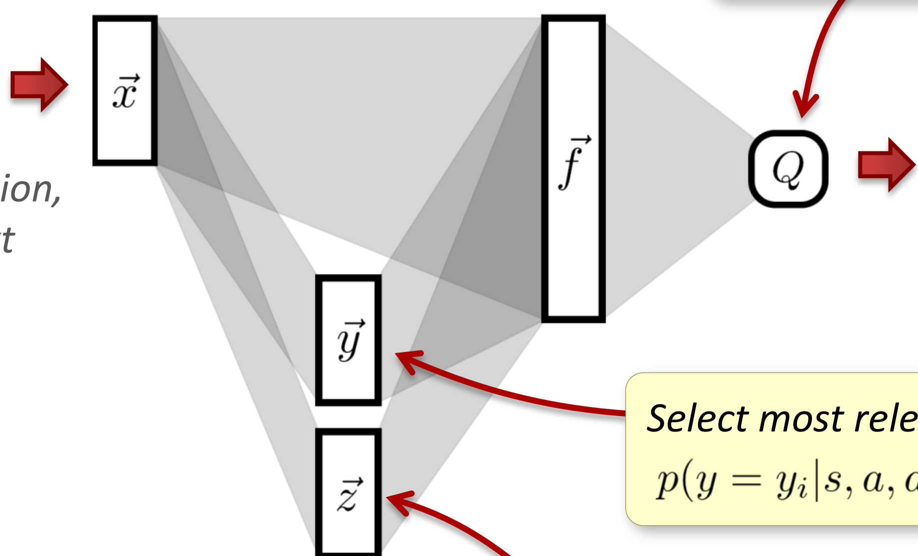
Linear model:

- $\vec{w}$ - *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*

Input:
game state,
candidate action,
document text



Predicted action value

Q function approximation

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

Select most relevant sentence

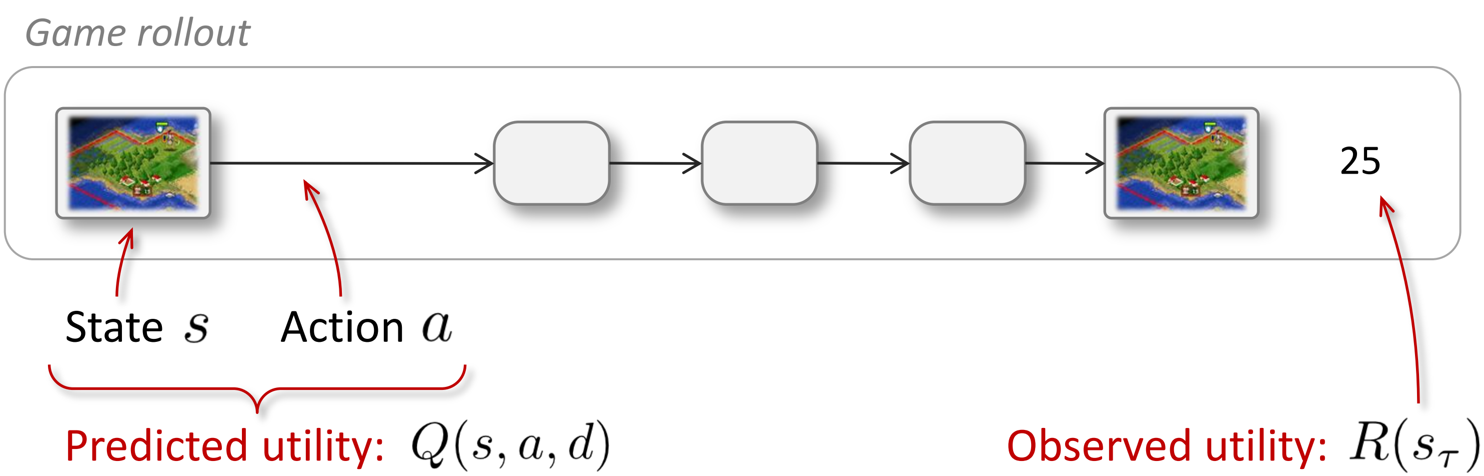
$p(y = y_i | s, a, d) \propto e^{\vec{u} \cdot \vec{\phi}(y_i, s, a, d)}$

Predict sentence predicate structure

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

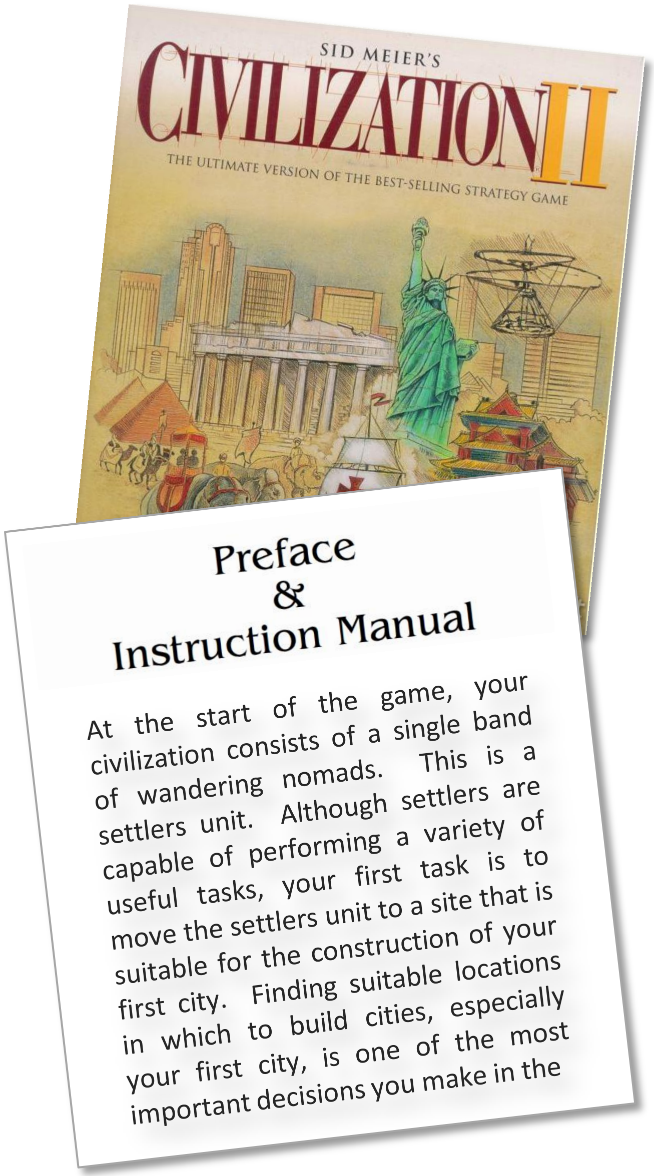
Parameter Estimation

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



Preface
&
Instruction Manual

At the start of the game, your civilization consists of a single band of wandering nomads. This is a settlers unit. Although settlers are capable of performing a variety of useful tasks, your first task is to move the settlers unit to a site that is suitable for the construction of your first city. Finding suitable locations in which to build cities, especially your first city, is one of the most important decisions you make in the

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.*

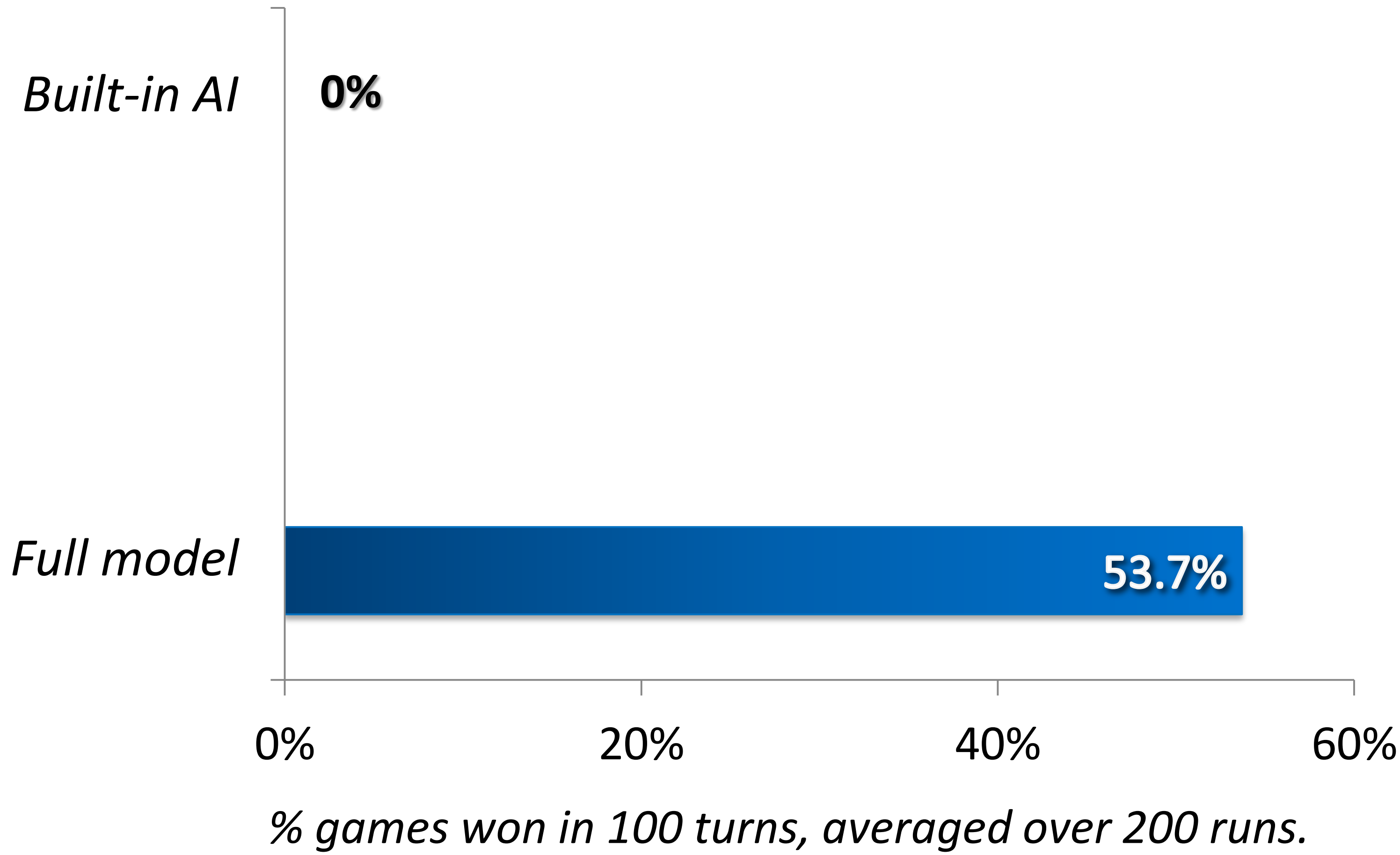
Primary evaluation:

- *Games won within first 100 game steps.*
- *Averaged over 200 independent experiments.*
- *Avg. experiment runtime: 1.5 hours*

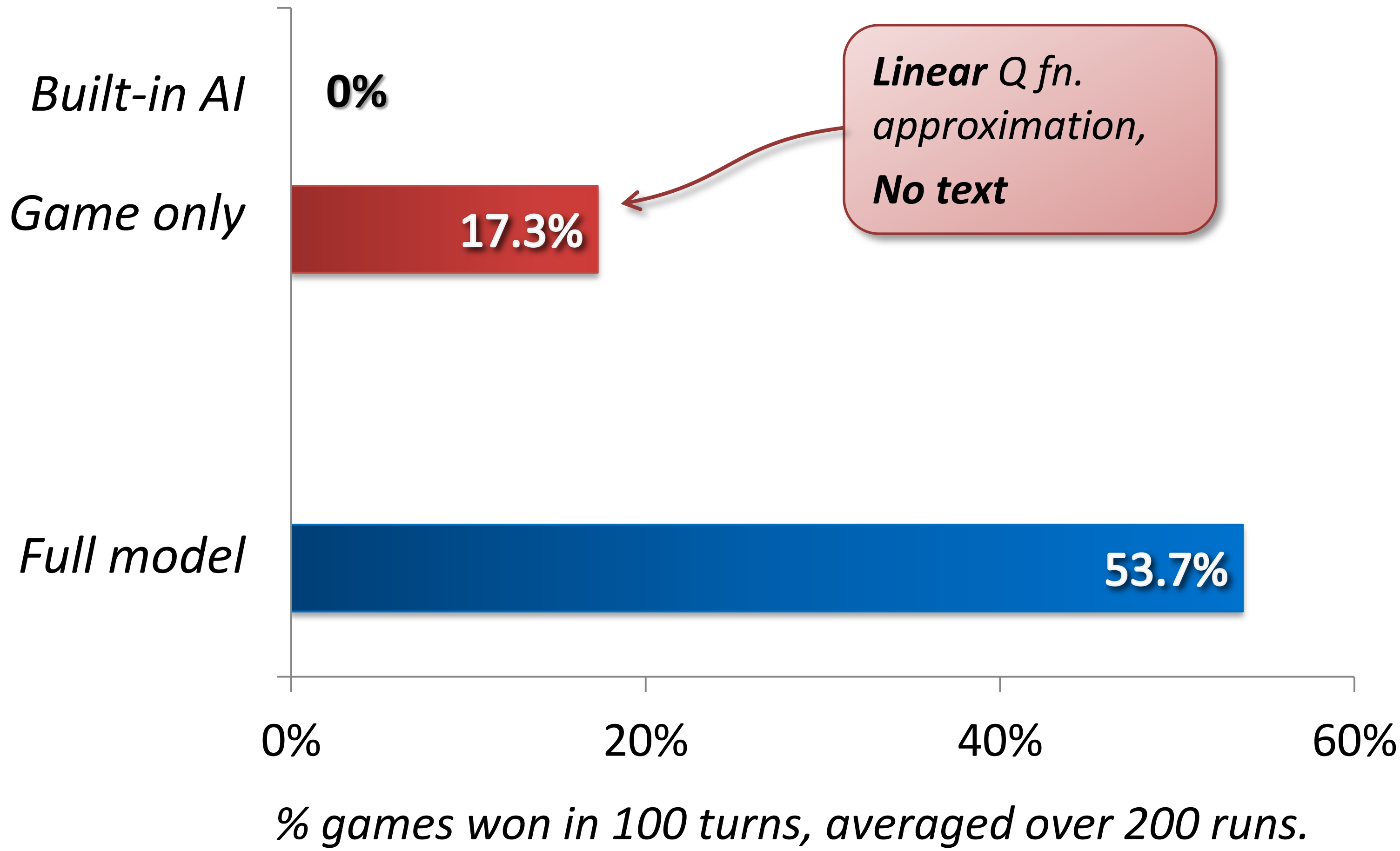
Secondary evaluation:

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

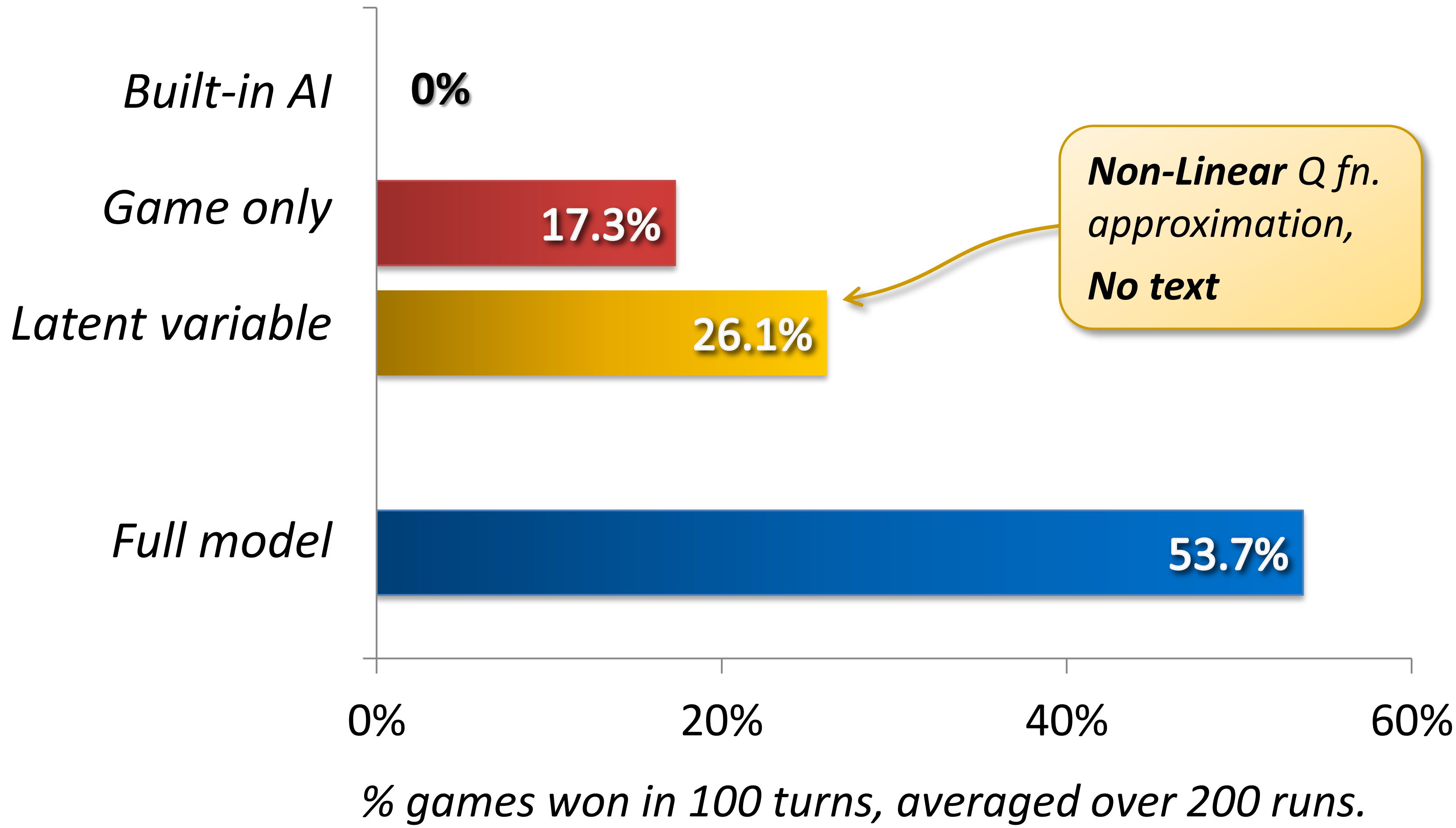
Results


Built-in AI
0%
Full model
53.7%
0%
20%
40%
60%
% games won in 100 turns, averaged over 200 runs.

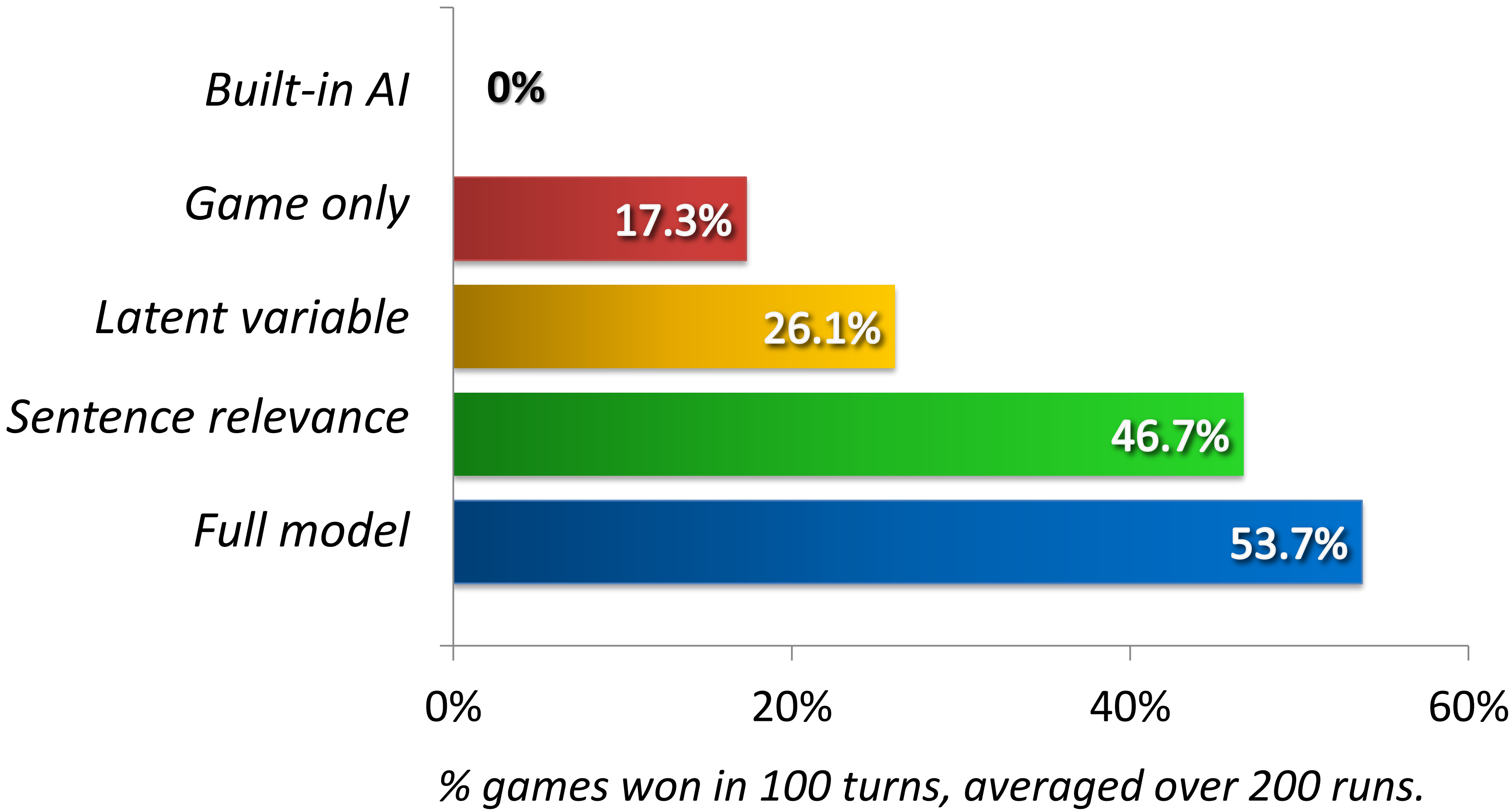
Does Text Help ?



Text vs. Representational Capacity



Linguistic Complexity vs. Performance Gain



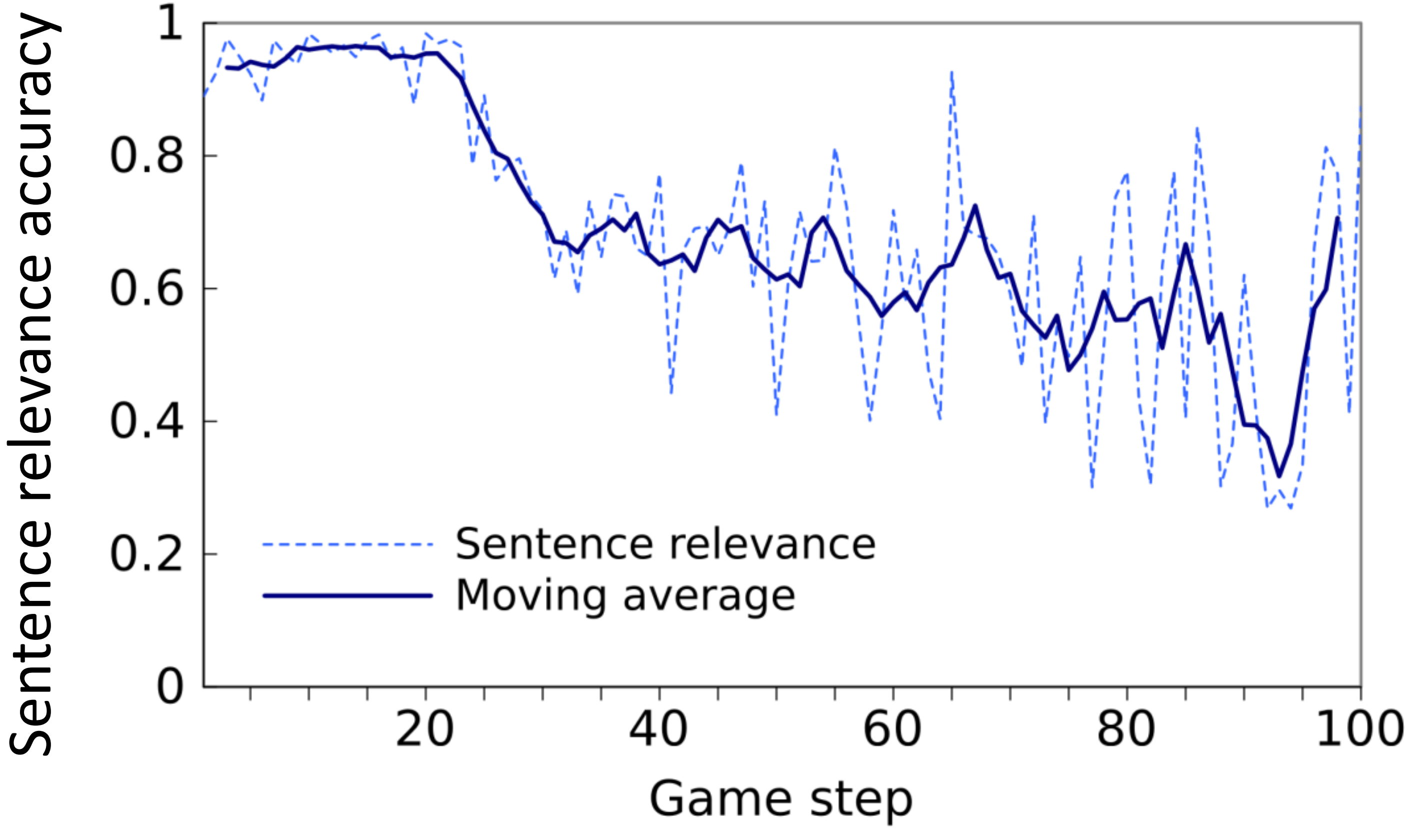
Results: Sentence Relevance

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

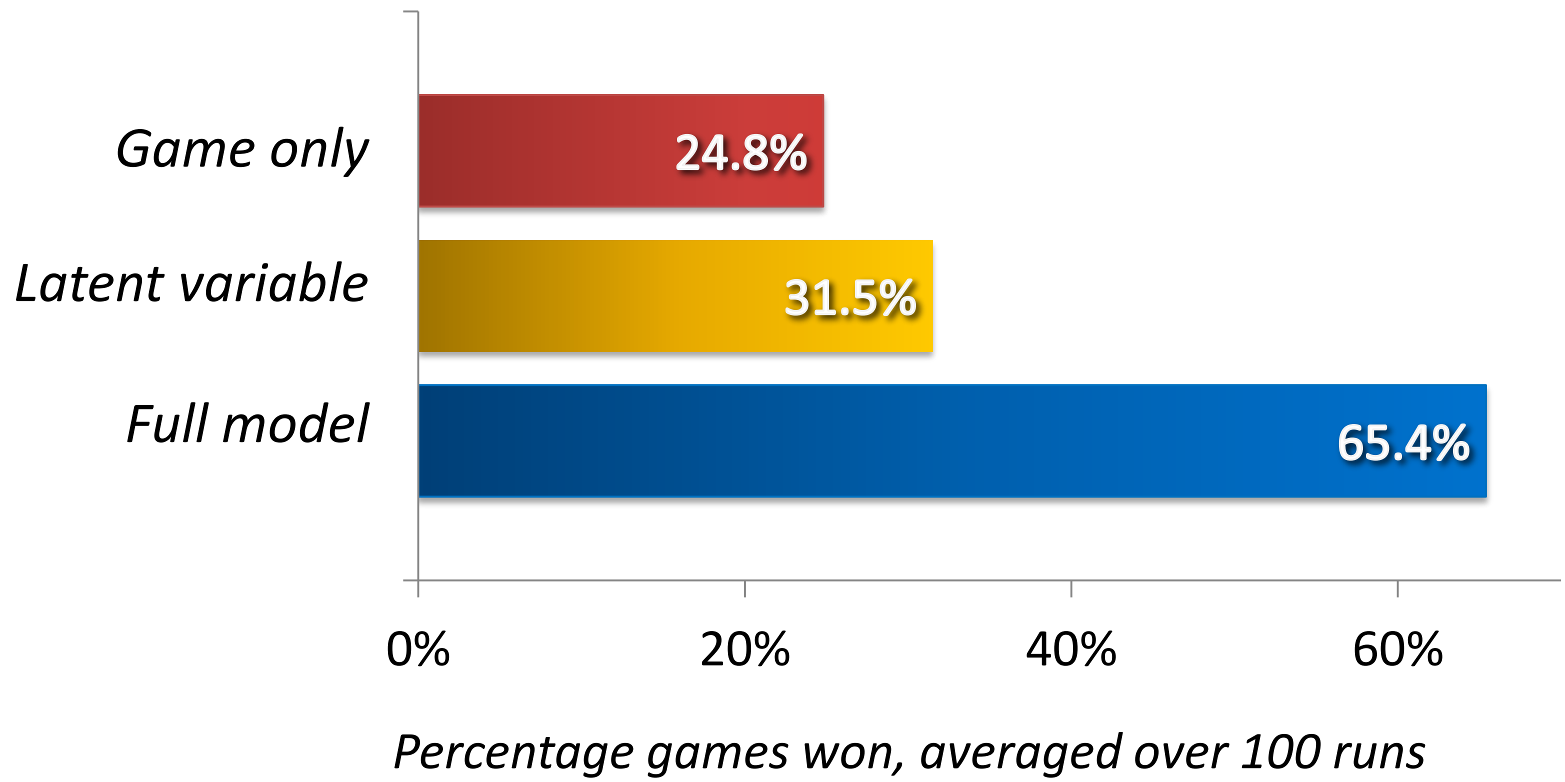
Solution: *Add known non-relevant sentences to text.*
E.g., sentences from the Wall Street Journal corpus.

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

Results: Sentence Relevance



Results: Full Games



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

   - ***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

Solving Hard Planning Tasks

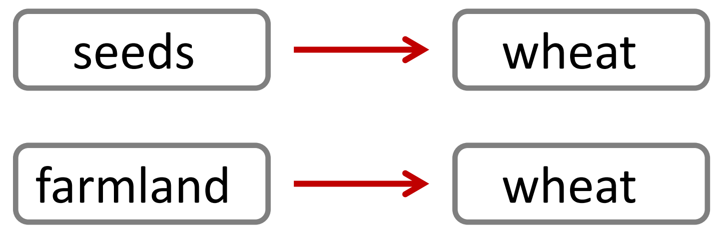
Minecraft : Virtual world allowing tool creation and complex construction.

Example Precondition Relations

Text

Seeds planted in **farmland** will grow to become **wheat** which can be harvested.

Preconditions



Challenge

Difference in level of abstraction

Text

Seeds planted in **farmland** will grow to become **wheat** which can be harvested.

Preconditions

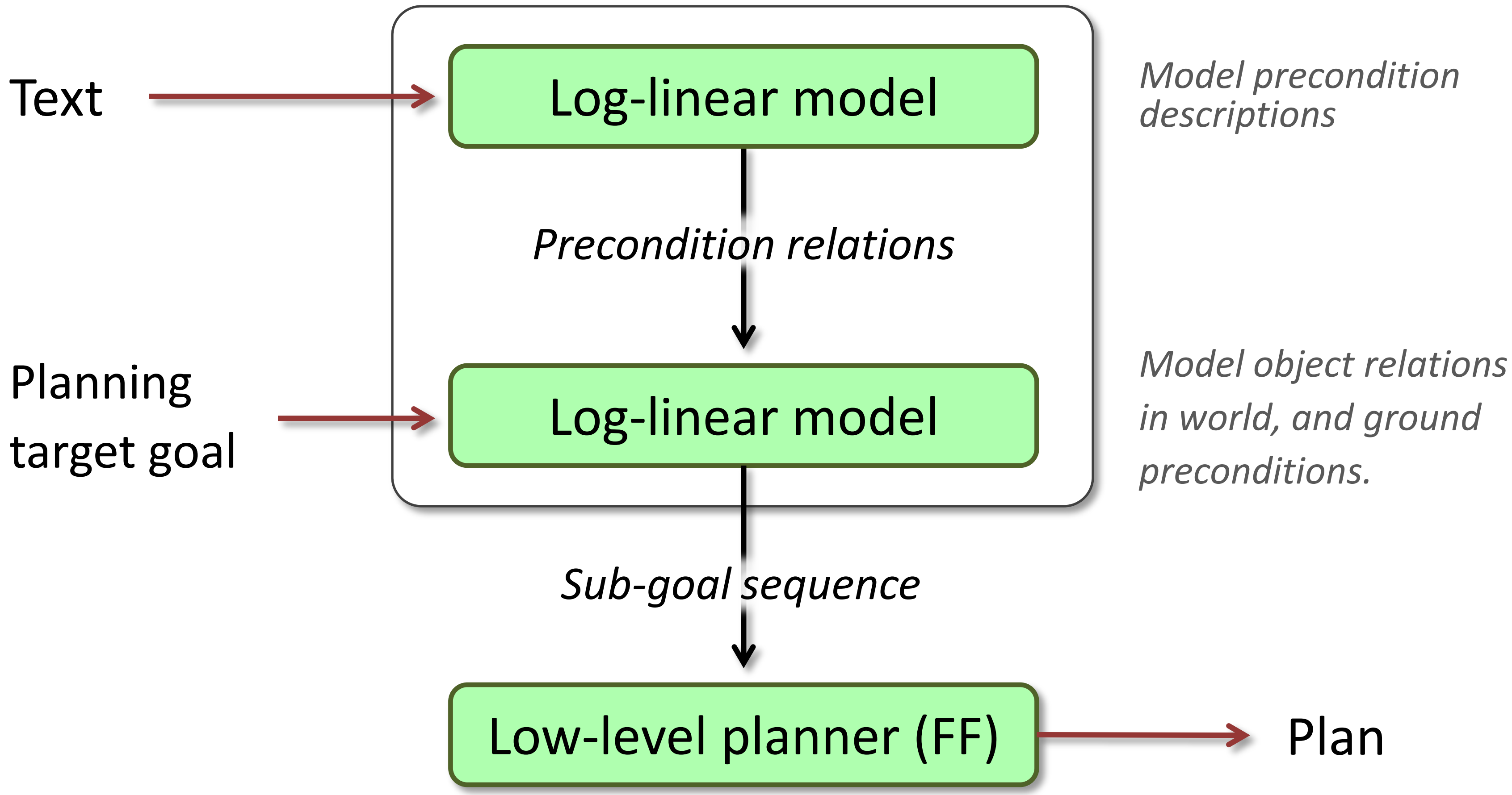
seeds → wheat

farmland → wheat

Plan

1. pickup tool: ***shears***
2. collect ***seeds*** from ***tallgrass*** using ***shears***
3. pickup tool: ***hoe***
4. plow ***land*** with ***hoe*** at (2,0) into ***farmland***
5. plant ***seeds*** at coordinates (2,0)
6. fertilize ***seeds*** at (2,0) with ***bonemeal***
7. wait for ***wheat*** to grow
8. pickup tool: ***shears***
9. harvest ***wheat*** with ***shears*** at (2,0)

Model



Learn model parameters from planning feedback

Experimental Domain

World:

Minecraft virtual world

Documents:

User authored wiki articles

Text Statistics:

Sentences:	*242*
Vocabulary:	*979*

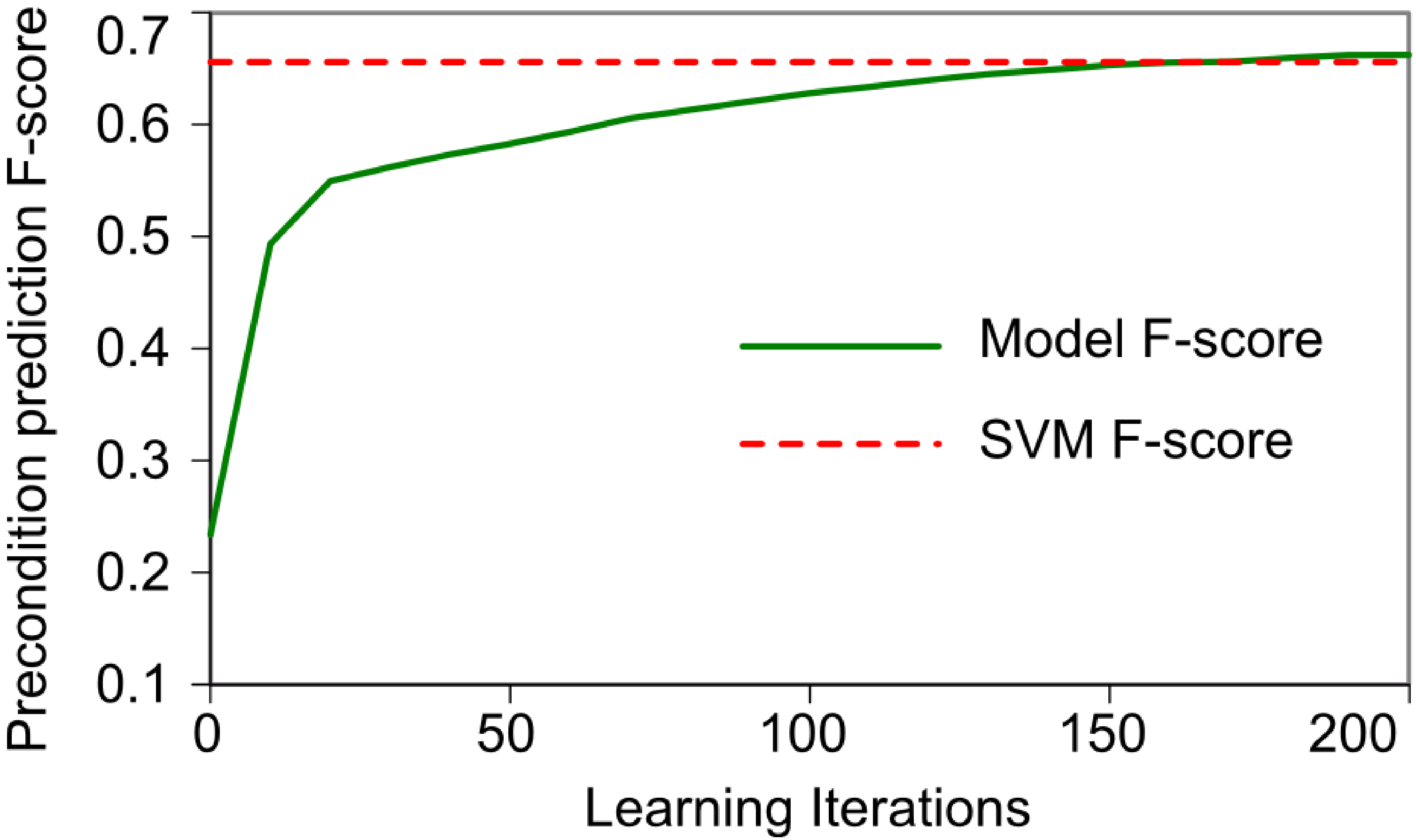
Planning task Statistics:

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



Related Work

- Instruction interpretation

  Learned from manual supervision

- Game playing

  Action selection based only on game state

- High-level planning

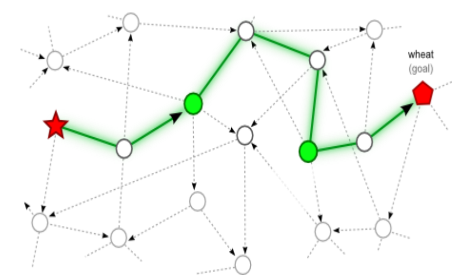
  Based on analysis of world dynamics

Contributions

- Learn language analysis from control feedback
- Improve control performance using text information

Instruction Interpretation

Complex Game-play



High-level Planning