Learning to Win by Reading Manuals in a Monte-Carlo Framework

S.R.K. Branavan, David Silver, Regina Barzilay

MIT
Semantic Interpretation

Traditional view:

*Map text into an abstract representation*

Alternative view:

*Map text into a representation which helps performance in a control application*
Semantic Interpretation for Control Applications

Complex strategy game

action 1

action 2

action 3

End result

lost

won

lost

Traditional approach:

Learn action-selection policy from game feedback.

Our contribution:

Use textual advice to guide action-selection policy.
Leveraging Textual Advice: Challenges

1. Find sentences relevant to given game state.

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.

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

  - move_settlers_to()
  - settlers_build_city()

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: $\theta_1$

Game manual:
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 plains or grassland with a river running through it if possible. In order to survive and grow ...

Model params: $\theta_2$

Game manual:
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 plains or grassland with a river running through it if possible. In order to survive and grow ...

End result: won

End result: lost
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

Actual Game

Simulation

Copy game

State 1

Copy

Irrigate

Game lost

State 1
Monte-Carlo Search

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

Game scores

| Rollout depth | 0.1 | 0.4 | 1.2 | 3.5 |
Monte-Carlo Search

Try many candidate actions from current state & see how well they perform.
Learn feature weights from simulation outcomes

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

\[ \vec{\phi}(s, a) \quad \text{- feature function} \]

\[ \vec{\theta} \quad \text{- model parameters} \]
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.

• Estimate value of candidate actions
  
  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:
$$\begin{cases} 
\vec{u} & \text{weight vector} \\
\vec{\phi}(y_i, s, a, d) & \text{feature function}
\end{cases}$$
Predicate Structure

Select word labels based on sentence + dependency info

E.g., “Build cities near rivers or ocean.”

Log-linear model:

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

Word index \( j \), sentence \( y \), dependency info \( q \)

Predicate label \( e_j = \{ \text{action, state, background} \} \)

- \( \vec{v} \) - weight vector
- \( \vec{\psi}(e_j, j, y, q) \) - feature function
Final Q function approximation

Predict expected value of candidate action

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

State \( s \), candidate action \( a \)

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

Linear model:

\[
\begin{align*}
\mathbf{w} & \quad \text{- weight vector} \\
\mathbf{f}(s, a, d, y_i, \vec{e}_i) & \quad \text{- feature function}
\end{align*}
\]
Multi-layer neural network: *Each layer represents a different stage of analysis*

**Model Representation**

**Input:**
- game state,
- candidate action,
- document text

**Q function approximation**

\[ Q(s, a, d, y_i, e_i) = \mathbf{w} \cdot \mathbf{f}(s, a, d, y_i, e_i) \]

**Predicted action value**

**Select most relevant sentence**

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

**Predict sentence predicate structure**

\[ p(e_j | j, y, q) \propto e^{\mathbf{u} \cdot \psi(e_j, j, y, q)} \]
Parameter Estimation

**Objective:** Minimize *mean square error* between predicted utility $Q(s, a, d)$ and observed utility $R(s_τ)$.
Parameter Estimation

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

**Parameter updates:**

\[ \tilde{u}_i \leftarrow \tilde{u}_i + \alpha_u [Q - R(s_{\tau})] Q \bar{x} [1 - p(y_i | \cdot)] \]

\[ \tilde{v}_i \leftarrow \tilde{v}_i + \alpha_v [Q - R(s_{\tau})] Q \bar{x} [1 - p(e_i | \cdot)] \]

\[ \tilde{w} \leftarrow \tilde{w} + \alpha_w [Q - R(s_{\tau})] \tilde{f}(s, a, d, y_i, z_j) \]
Features

State features:
- Amount of gold in treasury
- Government type
- Terrain surrounding current unit

Action features:
- Unit type (settler, worker, archer, etc)
- Unit action type

Text features:
- Word
- Parent word in dependency tree
- Word matches text label of unit
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.*

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 50 independent experiments.*
- *Avg. experiment runtime: 4 hours*
Results

Built-in AI

Full model

% games won in 100 turns, averaged over 200 runs.

0%
53.7%
Does Text Help?

- **Built-in AI**: 0%
- **Game only**: 17.3%
- **Full model**: 53.7%

% games won in 100 turns, averaged over 200 runs.

Linear Q fn. approximation, No text
Text vs. Representational Capacity

% games won in 100 turns, averaged over 200 runs.

- Built-in AI: 0%
- Game only: 17.3%
- Latent variable: 26.1%
- Full model: 53.7%

Non-Linear Q fn. approximation, No text
Linguistic Complexity vs. Performance Gain

- **Built-in AI**: 0%
- **Game only**: 17.3%
- **Latent variable**: 26.1%
- **Sentence relevance**: 46.7%
- **Full model**: 53.7%

% games won in 100 turns, averaged over 200 runs.
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

Sentence relevance accuracy

Game step

Sentence relevance
Moving average

Text feature importance

Game features

Text features
Results: Full Games

- **Game only**: 45.7%
- **Latent variable**: 62.2%
- **Full model**: 78.8%

*Percentage games won, averaged over 50 runs*
Related Work

**Grounded Language Acquisition: Instruction Interpretation**

*Branavan et al. 2009, 2010, Vogel & Jurafsky 2010*

- Imperative descriptions of action sequences
- Assume relevance of text to current world state

**Language Analysis in Games**

*Eisenstein et al. 2009*

- Extract high-level semantic representation from text
- Learn game rules from labeled traces + extracted formulae

*Gorniak & Roy 2005*

- Interpret spoken commands to control game character
- Learn from labeled parallel corpus
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
Monte-Carlo Search: Summary

Game states and actions

Monte-Carlo Rollouts (simulations)

Use observed rollout scores to select game action
Model Complexity, Time and Performance

- Full model
- Non-linear, no text
- Linear, no text

Win rate vs. Computation time per game step (s)
Dependency Information