Reinforcement Learning for Mapping Instructions to Actions

S.R.K. Branavan

Joint work with: Harr Chen, Luke Zettlemoyer, Regina Barzilay

MIT
Mapping Instructions to Actions

**Instructions:**
step-by-step descriptions of actions

**Target environment:**
where actions need to be executed

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

**Instructions:** step-by-step descriptions of actions

**Target environment:** where actions need to be executed

**Action sequence** executable in the environment

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. ...

```
LEFT_CLICK(Start)
LEFT_CLICK(Search)
...
TYPE_INTO(Search for: , "msdownld.tmp")
...
```
The Conventional Approach: Supervised Learning

1. **Annotate training documents.**

```
Text: click Start, point to Settings, and then click Control Panel
```

Annotations: `LEFT_CLICK(Start)` `LEFT_CLICK(Settings)` `LEFT_CLICK(Control Panel)`

2. **Use CRF to learning the mapping.**
Our Approach

Learn through trial and error

1. Map instructions to candidate actions
2. Execute candidate actions in the environment
3. Check how well we do (reward signal)
4. Update model parameters based on reward

Key hypothesis: Reward signal is sufficient supervision
Example Application 1: An Online Puzzle

**Target environment**
online flash puzzle

**Instructions**
puzzle solution

Clear the left six from the bottom row, the bottom six from the far-right column, the right six from the top row, the top six from the left column, the second row, the second-to-bottom row, the second from the left column and the second from the right column, the top six from columns three and four, the top two rows of six, the two columns of six, then the two rows.

**Reward signal**

Check if we won the puzzle!
Example Application 2:  Windows Help Instructions

Target environment  
Windows 2000  
graphical user interface

Instructions  
Microsoft help document

Reward signal  
Check if we hit a dead-end  
(check for overlap between sentence words & GUI labels)
Learning Using Reward Signal: Challenges

1. Reward can be delayed

\[ \Rightarrow \text{How can reward be propagated to individual actions} \]

2. Number of candidate action sequences is very large

\[ \Rightarrow \text{How can this space be effectively searched?} \]

Use Reinforcement Learning
Reinforcement Learning: A Sketch

Repeat:
- Observe current state of text + environment
- Select action based on a probabilistic model
- Execute action
- Receive reward and update parameters

FAIL?
Reinforcement Learning: Representation

\[ \text{State} \ s = \ \text{Observed Text} + \ \text{Observed Environment} \]

\[ \text{Action} \ a = \ \text{Word Selection} + \ \text{Environment Command} \]

**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] )

**State 2**
- Observed text and environment
- Select run after clicking start. In the open box type "dcomcnfg".

Policy function
\[ p( a \mid s ) \]
Constructing Mappings

Action 1
words: clicking start
command:
LEFT_CLICK( start )

Action 2
words: select run
command:
LEFT_CLICK( run )

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

Mapping process allows us to:

- Segment text to chunks that describe individual commands
- Learn translation of words to environment commands
- Reorder environment commands

Select run after clicking start, then in the open box, type "dcomcnfg".
Constructing Mappings

*Mapping process allows us to:*

- Segment text to chunks that describe individual commands
- Learn translation of words to environment commands
- Reorder environment commands

"select run"  ➔  \textcolor{green}{LEFT\_CLICK} \quad \textcolor{red}{run}
Constructing Mappings

**Mapping process allows us to:**

- Segment text to chunks that describe individual commands
- Learn translation of words to environment commands
- Reorder environment commands

2. Select run after **clicking start**, then

3. In the open box, type "dcomcnfg".
Generating Possible Actions

\[ State \ s = \text{Observed Text} + \text{Observed Environment} \]

\[ Action \ a = \text{Word Selection} + \text{Environment Command} \]
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:

\[
p(a | s; \theta) = \frac{e^{\theta \cdot \phi(s, a)}}{\sum_{a'} e^{\theta \cdot \phi(s, a')}}
\]

\( \theta \) - parameters of model
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
Learning Algorithm

Goal: Find $\theta$ that maximizes the expected reward

Method: Policy gradient algorithm (stochastic gradient ascent on $\theta$)

Learner

for each document:

sample candidate action sequence:

- observe world state $s_t$
- select action $a_t \sim p(a|s_t; \theta)$
- execute action in world
- receive reward $r$
- update parameters $\theta$ based on reward

World

Environment

Document text

1. Click Start, point to Search, and then click File.
2. In the Search Results dialog box, on the Folder Options dialog box, on the View tab, click Show hidden files and folders, and then click extensions for known file types check box.
3. In the Search for files or folders named box, click My Computer, and then click Start.
4. In the Search Results pane, right-click Modern, and then click Delete on the shortcut menu.
5. A “Confirm Folder Delete” message appears.
6. Click Yes.

Open
Explore
Search...
Open All Users
Explore All Users
Learning Algorithm

Parameter update:

$$\theta \leftarrow \theta + r \sum_t \left[ \phi(s_t, a_t) - \sum_a \phi(s_t, a) \ p(a \mid s_t; \theta) \right]$$

**Gradient of log-linear model**

- $r$ – reward
- $s_t$ – state at time $t$
- $a_t$ – action taken at time $t$
Incorporating Annotation in Reinforcement Learning

Reward can be based on annotations if available

\[ r(h) = \begin{cases} 
+1 & \text{if actions match annotations} \\
0 & \text{if actions don't match annotations} 
\end{cases} \]

Reinforcement learning allows a mix of annotation and environment based reward signals

**Annotated**

*Use annotation reward*

**Unannotated**

*Use environment reward*
Incorporating Annotation in Reinforcement Learning

Reward can be based on annotations if available

\[
\text{Reward } r(h) = \begin{cases} 
+1 & \text{if actions match annotations} \\
0 & \text{if actions don't match annotations}
\end{cases}
\]

If all documents are annotated: Equivalent to stochastic gradient ascent with a maximum-likelihood objective
Windows Configuration Application

Windows 2000 help documents from support.microsoft.com

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total # of documents</td>
<td>128</td>
</tr>
<tr>
<td>Train/development/test</td>
<td>70  / 18 / 40</td>
</tr>
<tr>
<td>Total # of words</td>
<td>5562</td>
</tr>
<tr>
<td>Vocabulary size</td>
<td>610</td>
</tr>
<tr>
<td>Avg. words per sentence</td>
<td>9.93</td>
</tr>
<tr>
<td>Avg. sentences per document</td>
<td>4.38</td>
</tr>
<tr>
<td>Avg. actions per document</td>
<td>10.37</td>
</tr>
</tbody>
</table>

Complex environment: 13088 observed states
Results: Baselines

Random action: 13% Randomly LEFT_CLICK, RIGHT_CLICK, DOUBLE_CLICK or TYPE on heuristically selected GUI object

Majority action: 29% Always LEFT_CLICK on heuristically selected GUI object

% actions correctly mapped
Results: Supervised

- **Random action**: 13%
- **Majority action**: 29%
- **Full supervision**: 76%

% actions correctly mapped
Results

- Random action: 13%
- Majority action: 29%
- Environment reward: 65%
- Partial supervision (30% annotated): 72%
- Full supervision: 76%

% actions correctly mapped
Trade off between Environment Reward and Manual Annotations

% of annotated documents

% actions correct

- RL annotation reward
- RL environment reward
Trade off between Environment Reward and Manual Annotations
Puzzle Application

Walk-through documents from the *Crossblock* flash puzzle

Clear the left six from the bottom row, the bottom six from the far-right column, the right six from the top row, the top six from the left column, the second row, the second-to-bottom row, the second from the left column and the second from the right column, the top six from columns three and four, the top two rows of six, the two columns of six, then the two rows.

Target environment

http://hexaditidom.deviantart.com/art/Crossblock-108669149
Results: Puzzle Game Application

![Graph showing results for different types of rewards in a Puzzle Game Application. The graph plots the percentage of actions correct against the percentage of annotated documents. The legend indicates three types of rewards: RL environment + annotation reward, RL annotation reward, and RL environment reward. The graph shows a significant improvement in performance with the addition of annotation reward.]
Results: Puzzle Game Application

Our method can leverage knowledge encoded in natural language
Related Work

Reinforcement Learning for Dialogue Management:

Scheffler and Young (2002), Roy et al. (2000),
Litman et al. (2000), Singh et al. (1999)

Fundamentally different problems

Grounded Language Acquisition:

Chen and Mooney (2008), Roy and Pentland (2002),

Assume parallel corpus of text and semantic representations (e.g. database entries)
Conclusions

- Environment feedback is an effective source of supervision
  - Reduces need for manual annotations
- Our method can leverage knowledge encoded in natural language

Code and data available at:

`groups.csail.mit.edu/rgb/code/rl`
## Results

<table>
<thead>
<tr>
<th></th>
<th>Windows</th>
<th>Puzzle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random baseline</td>
<td>0.128</td>
<td>0.101</td>
</tr>
<tr>
<td>Majority baseline</td>
<td>0.287</td>
<td>0.197</td>
</tr>
<tr>
<td>Environment reward</td>
<td>* 0.647</td>
<td>* 0.590</td>
</tr>
<tr>
<td>Partial supervision</td>
<td>◦ 0.723</td>
<td>* 0.702</td>
</tr>
<tr>
<td>Full supervision</td>
<td>◦ 0.756</td>
<td>0.714</td>
</tr>
</tbody>
</table>