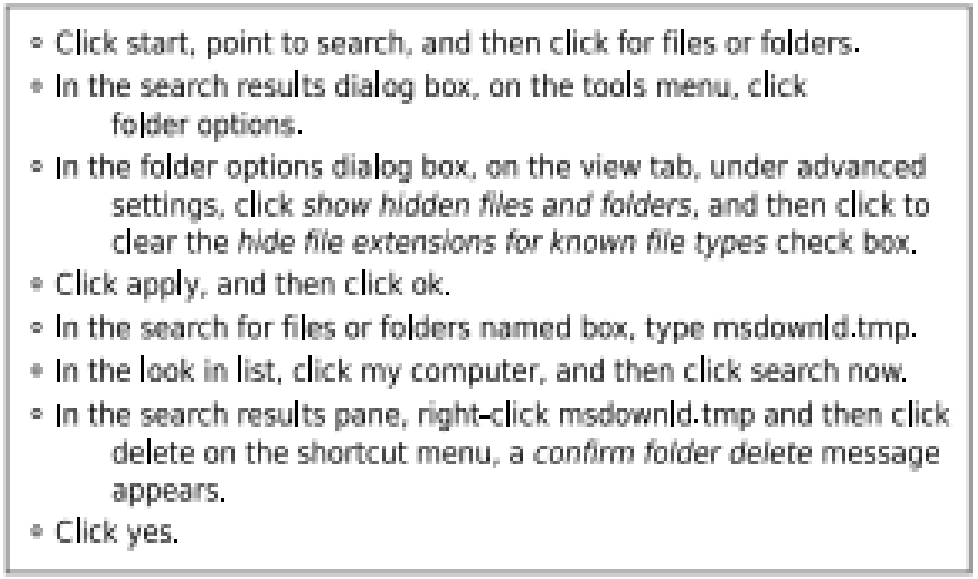
Reinforcement Learning for Mapping Instructions to Actions

S.R.K. Branavan, Harr Chen, Luke S. Zettlemoyer,
Regina Barzilay
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology

Introduction

- In this paper, they presented a reinforcement learning approach for inducing a mapping between instructions and actions. This approach is able to use environment-based rewards, such as task completion, to learn to analyze text. They showed that having access to a suitable reward function can significantly reduce the need for annotations.
- During training, the learner repeatedly constructs action sequences for a set of documents, executes those actions, and observes the resulting reward.
- Their policy is modeled in a log-linear fashion, allowing them to incorporate features of both the instruction text and the environment. They have employed a policy gradient algorithm to estimate the parameters of this model and to learn efficiently while exploring the small subsets of the states.

Example



- Click start, point to search, and then click for files or folders.
- In the search results dialog box, on the tools menu, click folder options.
- In the folder options dialog box, on the view tab, under advanced settings, click *show hidden files and folders*, and then click to clear the *hide file extensions for known file types* check box.
- Click apply, and then click ok.
- In the search for files or folders named box, type msdownld.tmp.
- In the look in list, click my computer, and then click search now.
- In the search results pane, right-click msdownld.tmp and then click delete on the shortcut menu, a *confirm folder delete* message appears.
- Click yes.

Figure 1: A Windows troubleshooting article describing how to remove the "msdownld.tmp" temporary folder.

The aim is to map this text to the corresponding low-level commands and parameters. For example, properly interpreting the third instruction requires clicking on a tab, finding the appropriate option in a tree control, and clearing its associated checkbox.

The key idea of their approach is to leverage the validation process as the main source of supervision to guide learning.

This form of supervision allows to learn interpretations of natural language instructions when standard supervised techniques are not applicable, due to the lack of human-created annotations.

Reinforcement Learning

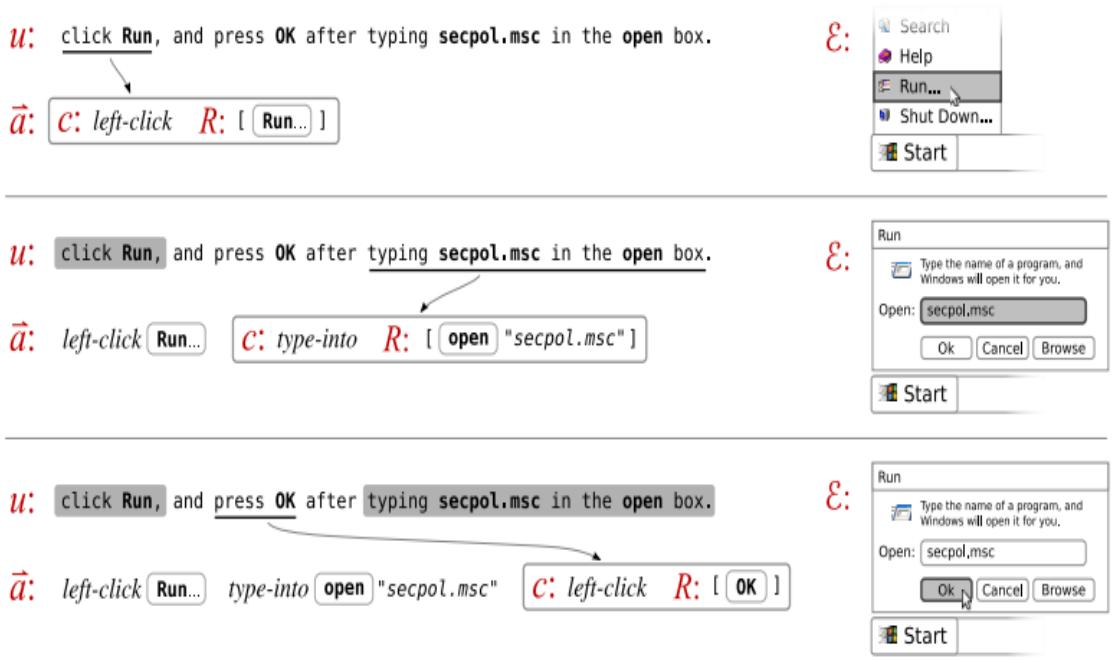
• It is concerned with how one ought to take actions in an environment so as to maximize some notion of cumulative reward.

•The basic reinforcement learning model consists of:

- a set of environment states ;
- a set of actions ;
- rules of transitioning between states;
- rules that determine the scalar immediate reward of a transition; and
- rules that describe what the agent observes.

•A reinforcement learning agent interacts with its environment in discrete time steps. At each time , the agent receives an observation , which typically includes the reward . It then chooses an action from the set of actions available, which is subsequently sent to the environment. The environment moves to a new state and the reward associated with the *transition* is determined. The goal of a reinforcement learning agent is to collect as much reward as possible.

•The main problem lies in the fact to decide a policy for selecting appropriate action so that some cumulative function of reward is maximized.



$d = (u_1, \ldots, u_\ell)$	Document
u	Sentence
$\vec{a} = (a_0, \ldots, a_{n-1})$	Sequence of actions
$a = (c, R, W')$	Action
c	Command
R	Command parameters
W'	Words mapped to action
W	Words mapped to previous actions
$\mathcal{E}$	Environment state

Here,

- Document(d) -> sequence of sentences ($u_1, u_2, u_3, \ldots, u_l$)
- *Our goal is to map d* to a sequence of actions (a) =(c, R, W') where,

c-> command ; R-> command's parameter ; W'-> words specifying c and R

- The environment state E specifies the set of objects available for interaction, and their properties. The environment state E changes in response to the execution of command c with parameters R according to a transition distribution : p(E'|E, c, R)

- To track the state of the document-to-actions mapping over time a mapping state tuple s is defined as (E , d ,j , W), where E refers to the current environment state; j is the index of the sentence currently being interpreted in document d; and W contains words that were mapped by previous actions for the same sentence.

- The initial mapping state s_0 for document d is (E_d, d, 0, Φ). E_d is the unique starting environment state for d. Performing action a in state s = (E , d , j , W) leads to a new state s' according to distribution p(s'|s , a), defined as follows: E transitions according to p(E'|E ,c,R), W is updated with a's selected words, and j is incremented if all words of the sentence have been mapped.

TRAINING

- For training use a predefined set D of documents and a reward function r(h) where h is the history of state-action pair, taking into account both the immediate and delayed award.
- The goal of training is to estimate parameters **θ** of the action selection distribution p(a|s , θ), called the policy. Since the reward correlates with action sequence correctness, the θ that maximizes expected reward will yield the best actions.

POLICY: Log-Linear Model

- Log-Linear models is used to analyze the diverse range of features and variables. Therefore for predicting the action given the state the following function is used:

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

Where φ(s,a') is a n-dimensional feature representation and θ is the optimizing parameter. s is the state and a is the action.

Policy Gradient Method

- Policy gradient algorithm is used to estimate the value of θ, defined above. This is used as we have a large set of states and therefore it is difficult to find θ, which maximizes the value function :

$$V_\theta(s) = E_{p(h|\theta)}\left[r(h)\right] = \sum_h r(h)p(h|\theta)$$

Here, r(h) is the reward function and the distribution p(h|θ) returns the probability of seeing history h when starting from state s and acting according to a policy with parameters θ.

$$p(h|\theta) = \prod_{t=0}^{n-1} p(a_t|s_t;\theta)p(s_{t+1}|s_t,a_t)$$

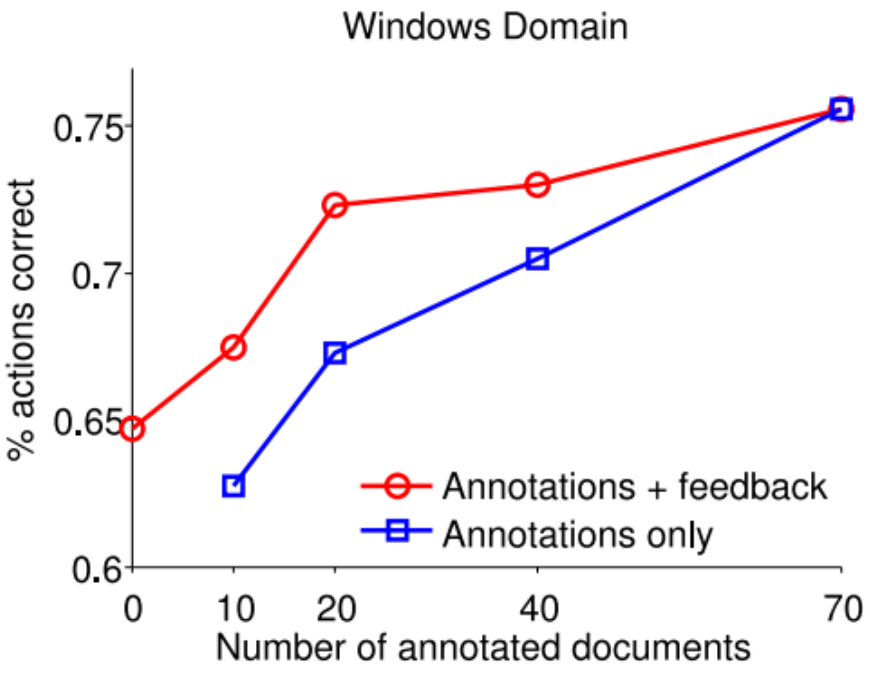
- Policy Gradient Algorithm computes local maximum but is a good measure when one have many states to deal with.
- Policy Gradient algorithms employ stochastic gradient ascent by computing a noisy estimate of the expectation using just a subset of the histories. It draws samples from $p(h|\theta)$ by acting in the target environment, and use these samples to approximate the expectation.

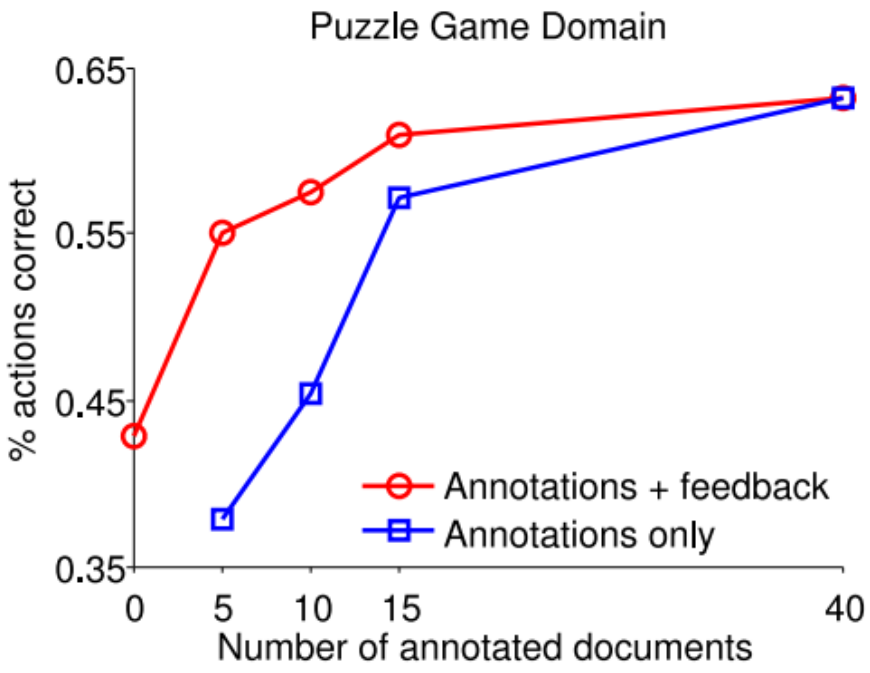
REWARD FUNCTION

- Reward functions is designed based on the availability of annotated data and environment feedback.
- Here annotated means correct sequence of action is provided with each sentence.
- Reward function can be boolean such as 1 if correct else 0 or fractional ie extent to which result is correct.
- In most reinforcement learning problems, the reward function is defined over state-action pairs, as r(s, a) but in this case,
    - $r(h) = \sum_t r(a_t, s_t)$

Results

After applying there theory on “Microsoft Help and Support” and “Crossblock:Puzzle Game” following results are obtained:





	Windows				Puzzle		
	Action	Sent.	Doc.	Word	Action	Doc.	Word
Random baseline	0.128	0.101	0.000	—	0.081	0.111	—
Majority baseline	0.287	0.197	0.100	—	—	—	—
Environment reward	∗ 0.647	∗ 0.590	∗ 0.375	0.819	∗ 0.428	∗ 0.453	0.686
Partial supervision	◊ 0.723	∗ 0.702	0.475	0.989	0.575	∗ 0.523	0.850
Full supervision	◊ 0.756	0.714	0.525	0.991	0.632	0.630	0.869

CONCLUSION

- There approach is able to use environment-based rewards, such as task completion, to learn to analyze text.
- They showed that having access to a suitable reward function can significantly reduce the need for annotations.
- The result obtained along with feedbacks are better then just using annotations.

Q/A

- Some examples of features used for window troubleshoot:
  - the similarity of a word in the sentence to the UI labels of objects in the environment.
  - Environment-specific features, such as whether an object is currently in focus, are useful when selecting the object to manipulate
- For reward in windows troubleshoot they have used following reward function:
  - Task completion
  - noisy method of checking whether execution can proceed from one sentence to the next.

Input: A document set D,
Feature representation ϕ,
Reward function $r(h)$,
Number of iterations T

Initialization: Set θ to small random values.

for $i = 1 \ldots T$ **do**
  foreach $d \in D$ **do**
    Sample history $h \sim p(h|\theta)$ where
      $h = (s_0, a_0, \ldots, a_{n-1}, s_n)$ as follows:
    for $t = 0 \ldots n-1$ **do**
      Sample action $a_t \sim p(a|s_t; \theta)$
      Execute a_t on state s_t: $s_{t+1} \sim p(s|s_t, a_t)$
    end
    $\Delta \leftarrow \sum_t \left(\phi(s_t, a_t) - \sum_{a'} \phi(s_t, a')p(a'|s_t; \theta)\right)$
    $\theta \leftarrow \theta + r(h)\Delta$
  end
end

Output: Estimate of parameters θ

Algorithm 1: A policy gradient algorithm.