Object-oriented Neural Programming (OONP) for Document Understanding

Zhengdong Lu[1] **Xianggen Liu**[2,3,4,*] **Haotian Cui**[2,3,4,*] **Yukun Yan**[2,3,4,*] **Daqi Zheng**[1]
`luz@deeplycurious.ai`,
`{liuxg16,cht15,yanyk13}@mails.tsinghua.edu.cn`, `da@deeplycurious.ai`
[1] DeeplyCurious.ai
[2] Department of Biomedical Engineering, School of Medicine, Tsinghua University
[3] Beijing Innovation Center for Future Chip, Tsinghua University
[4] Laboratory for Brain and Intelligence, Tsinghua University

Abstract

We propose Object-oriented Neural Programming (OONP), a framework for semantically parsing documents in specific domains. Basically, OONP reads a document and parses it into a predesigned object-oriented data structure that reflects the domain-specific semantics of the document. An OONP parser models semantic parsing as a decision process: a neural net-based Reader sequentially goes through the document, and builds and updates an intermediate ontology during the process to summarize its partial understanding of the text. OONP supports a big variety of forms (both symbolic and differentiable) for representing the state and the document, and a rich family of operations to compose the representation. An OONP parser can be trained with supervision of different forms and strength, including supervised learning (SL) , reinforcement learning (RL) and hybrid of the two. Our experiments on both synthetic and real-world document parsing tasks have shown that OONP can learn to handle fairly complicated ontology with training data of modest sizes.

1 Introduction

Mapping a document into a structured "machine readable" form is a canonical and probably the most effective way for document understanding. There are quite some recent efforts on designing neural net-based learning machines for this purpose, which can be roughly categorized into two groups: 1) sequence-to-sequence model with the neural net as the black box (Liang et al., 2017), and 2) neural net as a component in a pre-designed statistical model (Zeng et al., 2014). Both categories are hindered in tackling document with complicated structures, by either the lack of effective representation of knowledge or the flexibility in fusing them in the model.

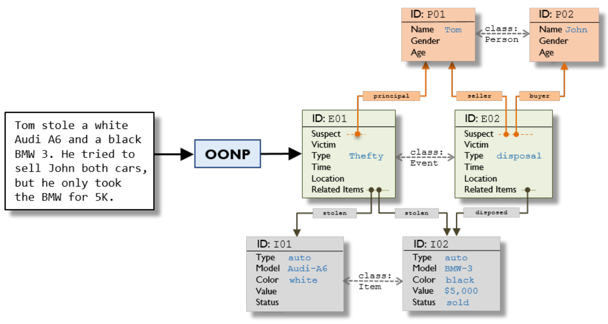


Figure 1: Illustration of OONP on a parsing task.

Towards solving this problem, we proposed Object-oriented Neural Programming (OONP), a framework for semantically parsing in-domain documents (illustrated in Figure 1). OONP maintains an object-oriented data structure, where objects from different classes are to represent entities (people, events, items etc) which are connected through links with varying types. Each object encapsulates internal properties (both symbolic and differentiable), allowing both neural and symbolic reasoning over complex structures and hence making it possible to represent rich semantics of documents. An OONP parser is neural net-based, but it has sophisticated architecture and mechanism designed for taking and yielding discrete structures, hence nicely combining symbolism (for interpretability and formal reasoning) and connectionism (for flexibility and learnability).

For parsing, OONP reads a document and parses it into this object-oriented data structure through a series of discrete actions along reading the document sequentially. OONP supports a rich fam-

* The work was done when these authors worked as interns at DeeplyCurious.ai.

ily of operations for composing the ontology, and flexible hybrid forms for knowledge representation. An OONP parser can be trained with supervised learning (SL), reinforcement learning (RL) and hybrid of the two.

OONP in a nutshell The key properties of OONP can be summarized as follows

1. OONP models parsing as a decision process: as the "reading and comprehension" agent goes through the text it gradually forms the ontology as the representation of the text through its action;
2. OONP uses a symbolic memory with graph structure as part of the state of the parsing process. This memory will be created and updated through the sequential actions of the decision process, and will be used as the semantic representation of the text at the end
3. OONP can blend supervised learning (SL) and reinforcement learning (RL) in tuning its parameters to suit the supervision signal in different forms and strength.

2 Related Works

2.1 Semantic Parsing

Semantic parsing is concerned with translating language utterances into executable logical forms and plays a key role in building conversational interfaces (Jonathan and Percy, 2014). Different from common tasks of semantic parsings, such as parsing the sentence to dependency structure (Buys and Blunsom, 2017) and executable commands (Herzig and Berant, 2017), OONP parses documents into a predesigned object-oriented data structure which is easily readable for both human and machine. It is related to semantic web (Berners-Lee et al., 2001) as well as frame semantics (Charles J, 1982) in the way semantics is represented, so in a sense, OONP can be viewed as a neural-symbolic implementation of semantic parsing with similar semantic representation.

2.2 State Tracking

OONP is inspired by Daumé III et al. (2009) on modeling parsing as a decision process, and the work on state-tracking models in dialogue system (Henderson et al., 2014) for the mixture of symbolic and probabilistic representations of dialogue state. For modeling a document with entities, Yang et al. (2017) use coreference links to recover entity clusters, though they only model entity mentions as containing a single word. However, entities whose names consist of multiple words are not considered. Entity Networks (Henaff et al., 2017) and EntityNLM (Ji et al., 2017) have addressed above problem and are the pioneers to model on tracking entities, but they have not considered the properties of the entities. In fact, explicitly modeling the entities both with their properties and contents is important to understand a document, especially a complex document. For example, if there are two persons named 'Avery', it is vital to know their genders or last names to avoid confusion. Therefore, we propose OONP to sketch objects and their relationships by building a structured graph for document parsing.

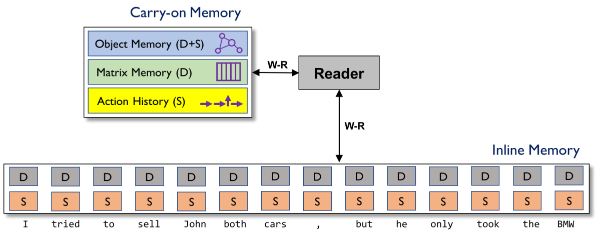


Figure 2: The overall diagram of OONP, where S stands for symbolic representation, D for distributed representation, and S+D for a hybrid of symbolic and distributed parts.

3 OONP: Overview

An OONP parser (illustrated in Figure 2) consists of a Reader equipped with read/write heads, Inline Memory that represents the document, and Carry-on Memory that summarizes the current understanding of the document at each time step. For each document to parse, OONP first preprocesses it and puts it into the Inline Memory, and then Reader controls the read-heads to sequentially go through the Inline Memory and at the same time update the Carry-on Memory. We will give a more detailed description of the major components below.

3.1 Memory

we have two types of memory, Carry-on Memory and Inline Memory. Carry-on Memory is designed to save the state in the decision process and summarize current understanding of the document based on the text that has been "read", while Inline Memory is designed to save location-specific information about the document. In a sense, the information in Inline Memory is low-level and unstructured, waiting for Reader to fuse and integrate into more structured representation.

Carry-on Memory has three compartments:

- Object Memory: denoted M_{obj}, the object-oriented data structure constructed during the parsing process;
- Matrix Memory: denoted M_{mat}, a matrix-type memory with fixed size, for differentiable read/write by the controlling neural net (Graves et al., 2014). In the simplest case, it could be just a vector as the hidden state of conventional RNN;
- Action History: symbolic memory to save the entire history of actions made during the parsing process.

Intuitively, Object Memory stores the extracted knowledge of the document with defined structure and strong evidence, while Matrix Memory keeps the knowledge that is fuzzy, uncertain or incomplete, waiting for further information to confirm, complete or clarify.

Object Memory

Object Memory stores an object-oriented representation of document, as illustrated in Figure 3. Each object is an instance of a particular class[*], which specifies the innate structure of the object, including internal properties, operations, and how this object can be connected with others. The internal properties can be of different types, for example string or category, which usually correspond to different actions in specifying them: the string-type property is usually "copied" from the original text in Inline Memory, while the category properties need to be rendered by a classifier. The links are in general directional and typed, resembling a special property viewing from the "source object". In Figure 3, there are six "linked" objects of three classes (namely, PERSON, EVENT, and ITEM) . Taking ITEM-object `I02` for example, it has five internal properties (`Type`, `Model`, `Color`, `Value`, `Status`), and is linked with two EVENT-objects through `stolen` and `disposed` link respectively.

In addition to the symbolic properties and links, each object had also its *object-embedding* as the distributed interface with Reader. For description simplicity, we will refer to the symbolic part of this hybrid representation of objects as the Ontology, with some slight abuse of this word. Object-embedding is complementary to the symbolic part of the object, recording all the relevant information associated with it but not represented in the Ontology, e.g., the contextual information when the object is created. Both Ontology and the object embeddings will be updated in time by the class-dependent operations driven by the actions issued by the Policy-net in Reader.

[*]We only consider flat structure of classes, but it is possible to have a hierarchy of classes with different levels of abstractness, and to allow an object to go from abstract class to its child during parsing with more information obtained.

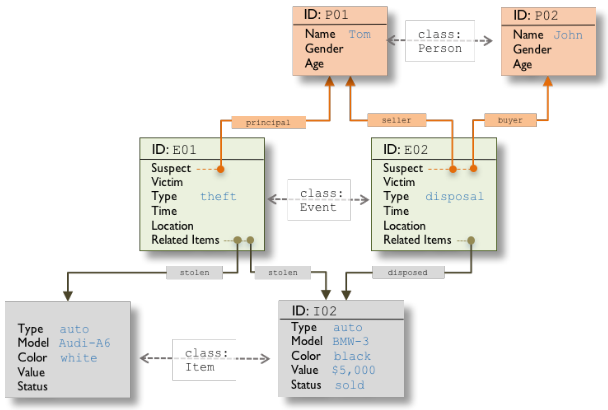


Figure 3: An example of objects of three classes.

According to the way the Ontology evolves with time, the parsing task can be roughly classified into two categories: 1) **Stationary:** there is a final ground truth that does not change with time, and 2) **Dynamical:** the truth changes with time. For stationary Ontology, see Section 5.2 and 5.3 for example, and for dynamical Ontology, please see Section 5.1.

Inline Memory

Inline Memory stores the relatively raw representation of the document with the sequential structure. Basically, Inline Memory is an array of memory cells, each corresponding to a pre-defined language unit (e.g., word) in the same order as they are in the original text. Each cell can have distributed part and symbolic part, designed to save the result of preprocessing of text, e.g., plain word embedding, hidden states of RNN, or some symbolic processing.

Inline Memory provides a way to represent locally encoded "low level" knowledge of the text, which will be read, evaluated and combined with the global semantic representation in Carry-on Memory by Reader. One particular advantage of this setting is that it allows us to incorporate the local decisions of some other models, including "higher order" ones like local relations across multiple language units, as illustrated in Figure 4.

3.2 Reader

Reader is the control center of OONP, coordinating and managing all the operations of OONP. More

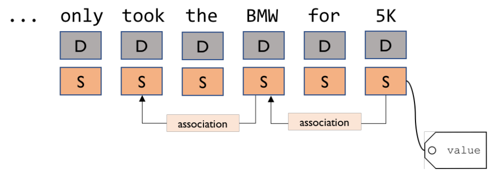


Figure 4: Inline Memory with symbolic knowledge.

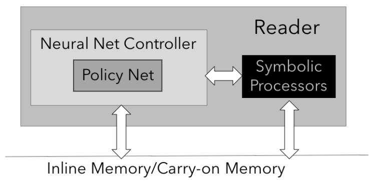


Figure 5: The overall digram of OONP

specifically, it takes the input of different forms (reading), processes it (thinking), and updates the memory (writing). As shown in Figure 5, Reader contains Neural Net Controller (NNC) and multiple symbolic processors, and NNC also has Policy-net as its sub-component. Similar to the controller in Neural Turing Machine (Graves et al., 2014), NNC is equipped with multiple read-heads and write-heads for differentiable read/write over Matrix Memory and (the distributed part of) Inline Memory, with a variety of addressing strategies (Graves et al., 2014). Policy-net however issues discrete outputs (i.e., *actions*), which gradually builds and updates the Object Memory in time. The symbolic processors are designed to handle information in symbolic form from Object Memory, Inline Memory, Action History, and Policy-net, while that from Inline Memory and Action History is eventually generated by Policy-net. In Appendix.A[†], we give a particular implementation of Reader with more details.

4 OONP: Actions

The actions issued by Policy-net can be generally categorized as the following

- `New-Assign` : determine whether to create an new object for the information at hand or assign it to a certain existed object;
- `Update.X` : determine which internal property or link of the selected object to update;
- `Update2what` : determine the content of the updating, which could be about string, category or links;

The typical order of actions is `New-Assign` → `Update.X` → `Update2what`, but it is common to have `New-Assign` action followed by nothing, when, for example, an object is mentioned but no substantial information is provided. As shown in Figure 6, we give an example of the entire episode of OONP parsing on the short text given in Figure 1, to show that a sequence of actions gradually forms the complete representation of the document.

5 An examples of actions

5.1 New-Assign

With any information at hand (denoted as $\mathcal{S}_t$) at time t, the choices of `New-Assign` include the following three categories of actions: 1) creating (`New`) an object of a certain type, 2) assigning $\mathcal{S}_t$ to an existed object, and 3) doing nothing for $\mathcal{S}_t$ and moving on. For Policy-net, the stochastic policy is to determine the following probabilities:

$$\begin{aligned} \mathsf{prob}(c, \mathsf{new}|\mathcal{S}_t), &\qquad c = 1, 2, \cdots, |\mathcal{C}| \\ \mathsf{prob}(c, k|\mathcal{S}_t), &\qquad \text{for } \mathcal{O}_t^{c,k} \in \mathsf{M}_{\mathsf{obj}}^t \\ \mathsf{prob}(\mathtt{none}|\mathcal{S}_t) & \end{aligned}$$

where $|\mathcal{C}|$ stands for the number of classes, $\mathcal{O}_t^{c,k}$ stands for the k^{th} object of class c at time t. Determining whether to new an object always relies on the following two signals

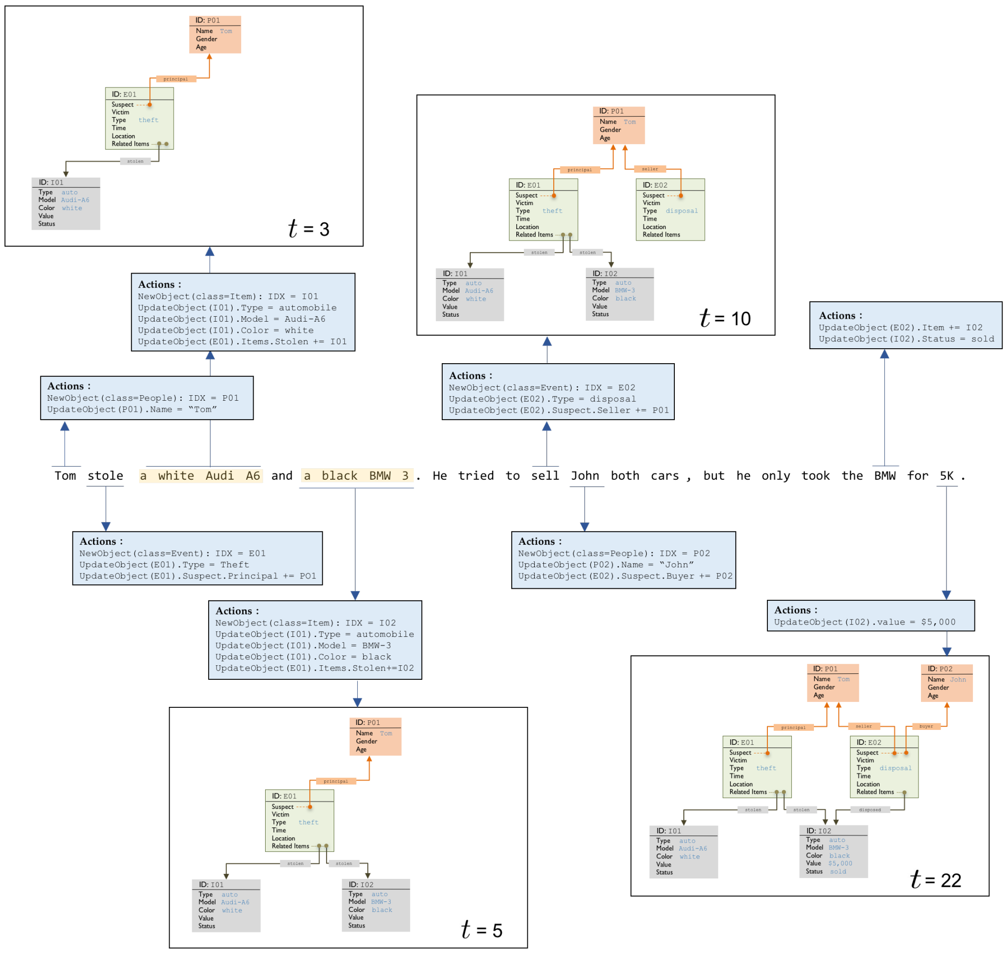
1. The information at hand cannot be contained by any existed objects;
2. Linguistic hints that suggest whether a new object is introduced.

Based on those intuitions, we take a score-based approach to determine the above-mentioned probability. More specifically, for a given $\mathcal{S}_t$, Reader forms a "temporary" object with its own structure (denoted $\hat{\mathcal{O}}_t$) with both symbolic and distributed sections. We also have a virtual object for the New action for each class c, denoted $\mathcal{O}_t^{c,\mathsf{new}}$, which is typically a time-dependent vector formed by Reader based on information in Matrix Memory. For a given $\hat{\mathcal{O}}_t$, we can then define the following $|\mathcal{C}| + |\mathsf{M}_{\mathsf{obj}}^t| + 1$ types of score functions:

$$\begin{aligned} \mathtt{New:}\quad & \mathsf{score}_{\mathsf{new}}^{(c)}(\mathcal{O}_t^{c,\mathsf{new}}, \hat{\mathcal{O}}_t; \theta_{\mathsf{new}}^{(c)}),\ c = 1, 2, \cdots, |\mathcal{C}| \\ \mathtt{Assign:}\quad & \mathsf{score}_{\mathsf{assign}}^{(c)}(\mathcal{O}_t^{c,k}, \hat{\mathcal{O}}_t; \theta_{\mathsf{assign}}^{(c)}),\ \text{for } \mathcal{O}_t^{c,k} \in \mathsf{M}_{\mathsf{obj}}^t \\ \mathtt{Do\ nothing:}\quad & \mathsf{score}_{\mathsf{none}}(\hat{\mathcal{O}}_t; \theta_{\mathsf{none}}). \end{aligned}$$

to measure the level of matching between the information at hand and existed objects, as well as the likeliness for creating an object or doing nothing. This process is pictorially illustrated in Figure 7. We therefore can define the following probability for the stochastic policy

[†]The appendix is also available at `https://arxiv.org/abs/1709.08853`


t = 3
t = 5
t = 10
t = 22
Tom stole a white Audi A6 and a black BMW 3 . He tried to sell John both cars , but he only took the BMW for 5K .

```
Actions :
NewObject(class=People): IDX = P01
UpdateObject(P01).Name = "Tom"

Actions :
NewObject(class=Event): IDX = E01
UpdateObject(E01).Type = Theft
UpdateObject(E01).Suspect.Principal += P01

Actions :
NewObject(class=Item): IDX = I01
UpdateObject(I01).Type = automobile
UpdateObject(I01).Model = Audi-A6
UpdateObject(I01).Color = white
UpdateObject(E01).Items.Stolen += I01

Actions :
NewObject(class=Item): IDX = I02
UpdateObject(I01).Type = automobile
UpdateObject(I01).Model = BMW-3
UpdateObject(I01).Color = black
UpdateObject(E01).Items.Stolen+=I02

Actions :
NewObject(class=Event): IDX = E02
UpdateObject(E02).Type = disposal
UpdateObject(E02).Suspect.Seller += P01

Actions :
NewObject(class=People): IDX = P02
UpdateObject(P02).Name = "John"
UpdateObject(E02).Suspect.Buyer += P02

Actions :
UpdateObject(E02).Item += I02
UpdateObject(I02).Status = sold

Actions :
UpdateObject(I02).value = $5,000
```

Figure 6: A pictorial illustration of a full episode of OONP parsing, where we assume the description of cars (highlighted with shadow) are segmented in preprocessing.

$$\text{prob}(c, \text{new}|\mathcal{S}_t) = \frac{e^{\text{score}_{\text{new}}^{(c)}(\mathcal{O}_t^{c,\text{new}}, \hat{\mathcal{O}}_t; \theta_{\text{new}}^{(c)})}}{Z(t)} \quad (1)$$

$$\text{prob}(c, k|\mathcal{S}_t) = \frac{e^{\text{score}_{\text{assign}}^{(c)}(\mathcal{O}_t^{c,k}, \hat{\mathcal{O}}_t; \theta_{\text{assign}}^{(c)})}}{Z(t)} \quad (2)$$

$$\text{prob}(\text{none}|\mathcal{S}_t) = \frac{e^{\text{score}_{\text{none}}(\hat{\mathcal{O}}_t; \theta_{\text{none}})}}{Z(t)} \quad (3)$$

where $Z(t) = \sum_{c' \in \mathcal{C}} e^{\text{score}_{\text{new}}^{(c')}(\mathcal{O}_t^{c',\text{new}}, \hat{\mathcal{O}}_t; \theta_{\text{new}}^{(c')})} + \sum_{(c'',k') \in \text{idx}(\mathsf{M}_{\text{obj}}^t)} e^{\text{score}_{\text{assign}}^{(c'')}(\mathcal{O}_t^{c'',k}, \hat{\mathcal{O}}_t; \theta_{\text{assign}}^{(c'')})} + e^{\text{score}_{\text{none}}(\hat{\mathcal{O}}_t; \theta_{\text{none}})}$ is the normalizing factor.

5.2 Updating Objects

In `Update.X` step, Policy-net needs to choose the property or external link (or none) to update for the selected object determined by `New-Assign` step. If `Update.X` chooses to update an external link, Policy-net needs to further determine which object it links to. After that, `Update2what` updates the chosen property or links. In task with static Ontology, most internal properties and links will be "locked" after they are updated for the first time, with some exception on a few semi-structured

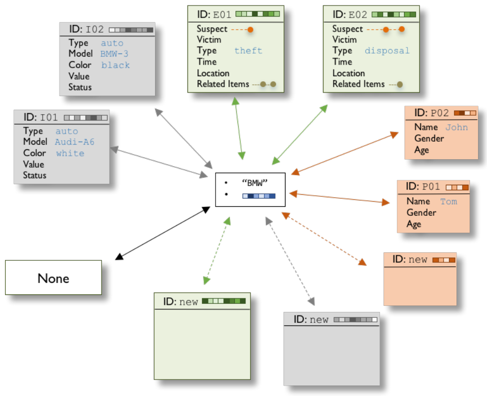


Figure 7: A pictorial illustration of what the Reader sees in determining whether to `New` an object and the relevant object when the read-head on Inline Memory reaches the last word in the text in Figure 2. The color of the arrow line stands for different matching functions for object classes, where the dashed lines are for the new object.

properties (e.g., the `Description` property in the experiment in Section 7.2). For dynamical Ontology, on the contrary, some properties and links are always subject to changes.

6 Learning

The parameters of OONP models (denoted Θ) include that for all operations and that for composing the distributed sections in Inline Memory. They can be trained with supervised learning (SL) , reinforcement learning (RL), and a hybrid of the two in different ways. With pure SL, the oracle gives the ground truth about the "right action" at each time step during the entire decision process, with which the parameter can be tuned to maximize the likelihood of the truth, with the following objective function

$$\mathcal{J}_{\mathsf{SL}}(\Theta) = -\frac{1}{N}\sum_{i}^{N}\sum_{t=1}^{T_i}\log(\pi_t^{(i)}[a_t^\star]) \quad (4)$$

where N stands for the number of instances, T_i stands for the number of steps in decision process for the i^{th} instance, $\pi_t^{(i)}[\cdot]$ stands for the probabilities of the actions at t from the stochastic policy, and $a_t^\star$ stands for the ground truth action in step t.

With RL, the supervision is given as rewards during the decision process, for which an extreme case is to give the final reward at the end of the decision process by comparing the generated Ontology and the ground truth, e.g.,

$$r_t^{(i)} = \begin{cases} 0, & \text{if } t \neq T_i \\ \mathsf{match}(\mathsf{M}_{\text{obj}}^{T_i}, \mathcal{G}_i), & \text{if } t = T_i \end{cases} \quad (5)$$

where the $\mathsf{match}(\mathsf{M}_{\text{obj}}^{T_i}, \mathcal{G}_i)$ measures the consistency between the Ontology of in the Object Memory $\mathsf{M}_{\text{obj}}^{T_i}$ and the ground truth $\mathcal{G}^\star$. We can use policy search algorithm to maximize the expected total reward, e.g. the commonly used REINFORCE (Williams, 1992) for training, with the gradient

$$\nabla_\Theta \mathcal{J}_{\mathsf{RL}}(\Theta) = -\mathbb{E}_{\pi_\theta}\left[\nabla_\Theta \log \pi_\Theta\left(a_t^i|s_t^i\right) r_{t:T}^{(i)}\right] \quad (6)$$

$$\approx -\frac{1}{NT_i}\sum_{i}^{N}\sum_{t=1}^{T}\nabla_\Theta \log \pi_\Theta\left(a_t^i|s_t^i\right) r_{t:T_i}^{(i)}. \quad (7)$$

When OONP is applied to real-world tasks, there is often quite natural supervision signals for both SL and RL. More specifically, for static Ontology one can infer some actions from the final ontology based on some basic assumption, e.g.,

- the system should `New` an object the first time it is mentioned;
- the system should put an extracted string (say, that for `Name`) into the right property of right object at the end of the string.

For those that can not be fully inferred, say the categorical properties of an object (e.g., `Type` for event objects), we have to resort to RL to determine the *time* of decision, while we also need SL to train Policy-net on the *content* of the decision. Fortunately it is quite straightforward to combine the two learning paradigms in optimization. More specifically, we maximize this combined objective

$$\mathcal{J}(\Theta) = \mathcal{J}_{\mathsf{SL}}(\Theta) + \lambda\mathcal{J}_{\mathsf{RL}}(\Theta), \quad (8)$$

where $\mathcal{J}_{\mathsf{SL}}$ and $\mathcal{J}_{RL}$ are over the parameters within their own supervision mode and λ coordinates the weight of the two learning mode on the parameters they share. Equation (8) actually indicates a deep coupling of supervised learning and reinforcement learning, since for any episode the samples of actions related to RL might affect the inputs to the models under supervised learning.

For dynamical Ontology (see Section 7.1 for example), it is impossible to derive most of the decisions from the final Ontology since they may change over time. For those we have to rely mostly on the supervision at the time step to train the action (supervised mode) or count on OONP to learn the dynamics of the ontology evolution by fitting the final ground truth. Both scenarios are discussed in Section 7.1 on a synthetic task.

7 Experiments

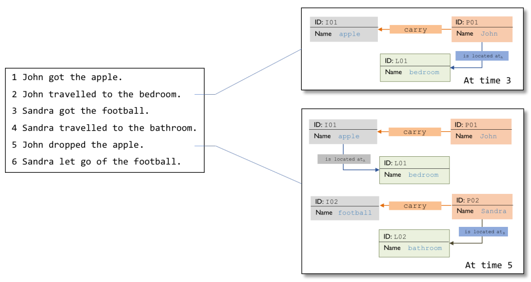
We applied OONP on three document parsing tasks, to verify its efficacy on parsing documents with different characteristics and investigate different components of OONP.

7.1 Task-I: bAbI Task

Data and Task

We implemented OONP on enriched version of bAbI tasks (Johnson, 2017) with intermediate representation for history of arbitrary length. In this experiment, we considered only the original bAbi task-2 (Weston et al., 2015), with an instance shown in the left panel Figure 8. The ontology has three types of objects: PERSON-object, ITEM-object, and LOCATION-object, and three types of links specifying relations between them (see Figure 8 for an illustration). All three types of objects have `Name` as the only internal property.

The task for OONP is to read an episode of story and recover the trajectory of the evolving ontology. We choose bAbI for its dynamical ontology that evolves with time and ground truth given for each snapshot. Comparing with the real-world tasks we will present later, bAbi has almost trivial internal properties but relatively rich opportunities for links, considering that any two objects of different types could potentially have a link.



```
1 John got the apple.
2 John travelled to the bedroom.
3 Sandra got the football.
4 Sandra travelled to the bathroom.
5 John dropped the apple.
6 Sandra let go of the football.
```

Figure 8: One instance of bAbI (6-sentence episode) and the ontology of two snapshots.

Action	Description
`NewObject`(c)	New an object of class-c.
`AssignObject`(c, k)	Assign the current information to existed object (c, k)
`Update`(c, k)`.AddLink`(c', k', ℓ)	Add an link of type-ℓ from object-(c, k) to object-(c', k')
`Update`(c, k)`.DelLink`(c', k', ℓ)	Delete the link of type-ℓ from object-(c, k) to object-(c', k')

Table 1: Actions for bAbI.

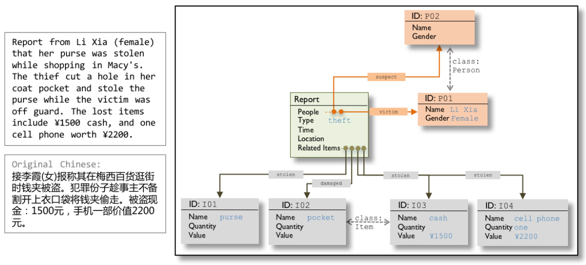
Implementation Details

For preprocessing, we have a trivial NER to find the names of people, items and locations (saved in the symbolic part of Inline Memory) and word-level bi-directional GRU for the distributed representations of Inline Memory. In the parsing process, Reader goes through the inline word-by-word in the temporal order of the original text, makes `New-Assign` action at every word, leaving `Update.X` and `Update2what` actions to the time steps when the read-head on Inline Memory reaches a punctuation (see more details of actions in Table 1). For this simple task, we use an almost fully neural Reader (with MLPs for Policy-net) and a vector for Matrix Memory, with however a Symbolic Reasoner to maintain the logical consistency after updating the relations with the actions (see Appendx.B for more details).

Results and Analysis

For training, we use 1,000 episodes with length evenly distributed from one to six. We use just REINFORCE with only the final reward defined as the overlap between the generated ontology and the ground truth, while step-by-step supervision on actions yields almost perfect result (result omitted). For evaluation, we use the F1 (Rijsbergen, 1979) between the generated links and the ground truth averaged over all snapshots of all test instances, since the links are sparse compared with all the possible pairwise relations between objects, with which we get F1= 94.80% without Symbolic Reasoner and F1= 95.30% with it.

Clearly OONP can learn fairly well on recovering the evolving ontology with such a small training set and weak supervision (RL with the final reward), showing that the credit assignment over



```
Report from Li Xia (female)
that her purse was stolen
while shopping in Macy's.
The thief cut a hole in her
coat pocket and stole the
purse while the victim was
off guard. The lost items
include ¥1500 cash, and one
cell phone worth ¥2200.
```

Original Chinese:
接李霞(女)报称其在梅西百货逛街时钱夹被盗。犯罪份子趁事主不备割开上衣口袋将钱夹偷走。被盗现金：1500元，手机一部价值2200元。

Figure 9: Example of police report & its ontology.

to earlier snapshots does not cause much difficulty in the learning of OONP even with a generic policy search algorithm. It is not so surprising to observe that Symbolic Reasoner helps to improve the results on discovering the links, while it does not improve the performance on identifying the objects although it is taken within the learning.

7.2 Task-II: Parsing Police Report

Data & Task

We implement OONP for parsing Chinese police report (brief description of criminal cases written by policeman), as illustrated in the left panel of Figure 9. We consider a corpus of 5,500 cases with a variety of crime categories, including theft, robbery, drug dealing and others. Although the language is reasonably formal, the corpus covers a big variety of topics and language styles, and has a high proportion of typos. The ontology we designed for this task mainly consists of a number of PERSON-objects and ITEM-objects connected through an EVENT-object with several types of relations, as illustrated in the right panel of Figure 9. A PERSON-object has three internal properties: `Name` (string), `Gender` (categorical) and `Age` (number), and two types of external links (`suspect` and `victim`) to an EVENT-object. An ITEM-object has three internal properties: `Name` (string), `Quantity` (string) and `Value` (string), and six types of external links (`stolen`, `drug`, `robbed`, `swindled`, `damaged`, and `other`) to an EVENT-object. On average, a sample has 95.24 Chinese words and the ontology has 3.35 objects, 3.47 mentions and 5.02 relationships. Compared with bAbI in Section 7.1, the police report ontology has less pairwise links but much richer internal properties for objects of all three objects.

Implementation Details

The OONP model is to generate the ontology as illustrated in Figure 9 through a decision process with actions in Table 2. As pre-processing, we performed third party NER algorithm to find people names, locations, item etc. For the distributed part of Inline Memory, we used dilated CNN with different choices of depth and kernel size (Yu and Koltun, 2016), all of which will be jointly learned during training. In updating objects with its string-type properties (e.g., `Name` for a PERSON-object), we use Copy-Paste strategy for extracted string (whose NER tag already specifies which property in an object it goes to) as Reader sees it. For undetermined category properties in existed objects, Policy-net will determine the object to update (a `New-Assign` action without `New` option), its property to update (an `Update.X` action), and the updating operation (an `Update2what` action) at milestones of the decision process , e.g., when reaching an punctuation. For this task, since all the relations are between the single by-default EVENT-object and other objects, the relations can be reduced to category-type properties of the corresponding objects in practice. For category-type properties, we cannot recover `New-Assign` and `Update.X` actions from the label (the final ontology), so we resort RL for learning to determine that part, which is mixed with the supervised learning for `Update2what` and other actions for string-type properties.

Action	Description
`NewObject` (c)	New an object of class-c.
`AssignObject` (c, k)	Assign the current information to existed object (c, k)
`UpdateObject` (c, k) `.Name`	Set the name of object-(c, k) with the extracted string.
`UpdateObject` (PERSON, k) `.Gender`	Set the name of a PERSON-object indexed k with the extracted string.
`UpdateObject` (ITEM, k) `.Quantity`	Set the quantity of an ITEM-object indexed k with the extracted string.
`UpdateObject` (ITEM, k) `.Value`	Set the value of an ITEM-object indexed k with the extracted string.
`UpdateObject` (EVENT, 1) `.Items.x`	Set the link between the EVENT-object and an ITEM-object, where `x` $\in$ {`stolen`, `drug`, `robbed`, `swindled`, `damaged`, `other`}
`UpdateObject` (EVENT, 1) `.Persons.x`	Set the link between the EVENT-object and an PERSON-object, and `x` $\in$ {`victim`, `suspect`}

Table 2: Actions for parsing police report.

Results & Discussion

We use 4,250 cases for training, 750 for validation an held-out 750 for test. We consider the following four metrics in comparing the performance of different models:

Assignment Accuracy	the accuracy on `New-Assign` actions made by the model
Category Accuracy	the accuracy of predicting the category properties of all the objects
Ontology Accuracy	the proportion of instances for which the generated Objects is exactly the same as the ground truth
Ontology Accuracy-95	the proportion of instances for which the generated Objects achieves 95% consistency with the ground truth

which measures the accuracy of the model in making discrete decisions as well as generating the final ontology.

Model	Assign Acc. (%)	Type Acc. (%)	Ont. Acc. (%)	Ont. Acc-95 (%)
Bi-LSTM (baseline)	73.2 ± 0.58	-	36.4 ± 1.56	59.8 ± 0.83
ENTITYNLM (baseline)	87.6 ± 0.50	84.3 ± 0.80	59.6 ± 0.85	72.3 ± 1.37
OONP (neural)	88.5 ± 0.44	84.3 ± 0.58	61.4 ± 1.26	75.2 ± 1.35
OONP (structured)	91.2 ± 0.62	87.0 ± 0.40	65.4 ± 1.42	79.9 ± 1.28
OONP (RL)	**91.4** ± 0.38	**87.8** ± 0.75	**66.7** ± 0.95	**80.7** ± 0.82

Table 3: OONP on parsing police reports.

We empirically investigated two competing models, Bi-LSTM and EntityNLM , as baselines. Both models can be viewed as simplified versions of OONP. Bi-LSTM consists of a bi-directional LSTM as Inline Memory encoder and a two-layer MLP on top of that as Policy-net. Bi-LSTM does not support categorical prediction for objects due to the lack of explicit object representation, which will only be trained to perform `New-Assign` actions and evaluated on them (with the relevant metrics modified for it). EntityNLM, on the other hand, has some modest capability for modeling entities with the original purpose of predicting entity mentions (Ji et al., 2017) which has been adapted and re-implemented for this scenario. For OONP , we consider three variants:

- OONP (neural): simple version of OONP with only distributed representation in Reader;
- OONP (structured): OONP that considers the matching between two structured objects in `New-Assign` actions;
- OONP (RL): another version of OONP (structured) that uses RL[‡] to determine the time for predicting the category properties, while OONP (neural) and OONP (structured) use a rule-based approach to determine the time.

The experimental results are given in Table 3. As shown in Table 3, Bi-LSTM struggles to achieve around 73% Assignment Accuracy on test set, while OONP (neural) can boost the performance to 88.5%. Arguably, this difference in performance is due to the fact that Bi-LSTM lacks Object Memory, so all relevant information has to be stored in the Bi-LSTM hidden states along the reading process. When we start putting symbolic representation and operation into Reader, as shown in the result of OONP (structure), the performance is again significantly improved on all four metrics.

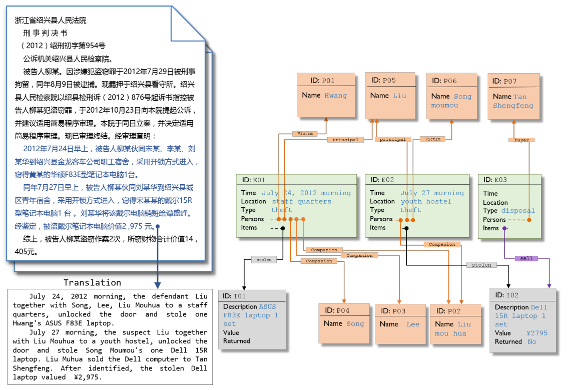
From the result of OONP (RL), RL improves not only the prediction of categorical property (and hence the overall ontology accuracy) but also tasks trained with purely SL (i.e., learning the `New-Assign` actions). This indicates there might be some deep entanglement between SL and RL through the obvious interaction between features in parsing and/or sharing of parameters.

7.3 Task-III: Parsing court judgment docs

Data and Task

Comparing with Task-II, court judgements are typically much longer, containing multiple events

[‡] A more detailed exposition of this idea can be found in (Liu et al., 2018), where RL is used for training a multi-label classifier of text


浙江省绍兴县人民法院

刑 事 判 决 书

（2012）绍刑初字第954号

公诉机关绍兴县人民检察院。

被告人柳某。因涉嫌犯盗窃罪于2012年7月29日被刑事拘留，同年8月9日被逮捕。现羁押于绍兴县看守所。绍兴县人民检察院以绍县检刑诉（2012）876号起诉书指控被告人柳某犯盗窃罪，于2012年10月23日向本院提起公诉，并建议适用简易程序审理。本院于同日立案，并决定适用简易程序审理。现已审理终结。经审理查明：

2012年7月24日早上，被告人柳某伙同宋某、李某、刘某华到绍兴县金龙客车公司职工宿舍，采用开锁方式进入，窃得黄某的华硕F83E型笔记本电脑1台。

同年7月27日早上，被告人柳某伙同刘某华到绍兴县城区青年宿舍，采用开锁方式进入，窃得宋某某的戴尔15R型笔记本电脑1 台 。刘某华将该戴尔电脑销赃给谭盛峰。经鉴定，被盗戴尔笔记本电脑价值2 ,975 元。

综上，被告人柳某盗窃作案2次，所窃财物合计价值14，405元。

Translation

July 24, 2012 morning, the defendant Liu together with Song, Lee, Liu Mouhua to a staff quarters, unlocked the door and stole one Hwang's ASUS F83E laptop.

July 27 morning, the suspect Liu together with Liu Mouhua to a youth hostel, unlocked the door and stole Song Moumou's one Dell 15R laptop. Liu Muhua sold the Dell computer to Tan Shengfeng. After identified, the stolen Dell laptop valued ¥2,975.

Figure 10: Left: the judgement document with highlighted part being the description the facts of crime; right: the corresponding ontology

of different types and large amount of irrelevant text. The dataset contains 4056 Chinese judgement documents, divided into training/dev/testing set 3256/400/400 respectively. The ontology for this task mainly consists of a number of PERSON-objects and ITEM-objects connected through a number EVENT-object with several types of links. An EVENT-object has three internal properties: `Time` (string), `Location` (string), and `Type` (category, ∈{`theft`, `restitution`, `disposal`}), four types of external links to PERSON-objects (namely, `principal`, `companion`, `buyer`, `victim`) and four types of external links to ITEM-objects (`stolen`, `damaged`, `restituted`, `disposed`). In addition to the external links to EVENT-objects , a PERSON-object has only the `Name` (string) as the internal property. An ITEM-object has three internal properties: `Description` (array of strings), `Value` (string) and `Returned`(binary) in addition to its external links to EVENT-objects , where `Description` consists of the words describing the corresponding item, which could come from multiple segments across the document. An object could be linked to more than one EVENT-object, for example a person could be the principal suspect in event A and also a companion in event B. An illustration of the judgement document and the corresponding ontology can be found in Figure 10.

Implementation Details

We use a model configuration similar to that in Section 7.2, with event-based segmentation of text given by third-party extraction algorithm (Yan et al., 2017) in Inline Memory, which enables OONP to trivially `New` EVENT-objectwith rules. OONP reads the Inline Memory, fills the EVENT-objects, creates and fills PERSON-objects and ITEM-objects, and specifies the links between them, with the actions summarized in Table 4. When an object is created during a certain event, it will be given an extra feature (not an internal property) indicating this connection, which will be used in deciding links between this object and event object, as well as in determining the future `New-Assign` actions.

Action for 2nd-round	Description
`NewObject(`c`)`	New an object of class-c.
`AssignObject(`c,k`)`	Assign the current information to existed object (c,k)
`UpdateObject(PERSON,`k`).Name`	Set the name of the k^{th} PERSON-object with the extracted string.
`UpdateObject(ITEM,`k`).Description`	Add to the description of an k^{th} ITEM-object with the extracted string.
`UpdateObject(ITEM,`k`).Value`	Set the value of an k^{th} ITEM-object with the extracted string.
`UpdateObject(EVENT,`k`).Time`	Set the time of an k^{th} EVENT-object with the extracted string.
`UpdateObject(EVENT,`k`).Location`	Set the location of an k^{th} EVENT-object with the extracted string.
`UpdateObject(EVENT,`k`).Type`	Set the type of the k^{th} EVENT-object among {`theft`, `disposal`, `restitution`}
`UpdateObject(EVENT,`k`).Items.x`	Set the link between the k^{th} EVENT-object and an ITEM-object, where `x` ∈ {`stolen`, `damaged`, `restituted`, `disposed` }
`UpdateObject(EVENT,`k`).Persons.x`	Set the link between the k^{th} EVENT-object and an PERSON-object, and `x` ∈ {`principal`, `companion`, `buyer`, `victim`}

Table 4: Actions for parsing court judgements.

Results and Analysis

We use the same metric as in Section 7.2, and compare two OONP variants, OONP (neural) and OONP (structured), with two baselines EntityNLM and Bi-LSTM. The two baselines will be tested only on the second-round reading, while both OONP variants are tested on a two-round reading. The results are shown in Table 5. OONP parsers attain accuracy significantly higher than Bi-LSTM. Among, OONP (structure) achieves over 71% accuracy on getting the entire ontology right and over 77% accuracy on getting 95% consistency with the ground truth. We omitted the RL results since the model RL model chooses to predict the type properties same as the simple rules.

Model	Assign Acc. (%)	Type Acc. (%)	Ont. Acc. (%)	Ont. Acc-95 (%)
Bi-LSTM (baseline)	84.66 ± 0.20	-	18.20 ± 0.74	36.88 ± 1.01
ENTITYNLM (baseline)	90.50 ± 0.21	96.33 ± 0.39	39.85 ± 0.20	48.29 ± 1.96
OONP (neural)	94.50 ± 0.24	97.73 ± 0.12	53.29 ± 0.26	72.22 ± 1.01
OONP (structured)	**96.90** ± 0.22	**98.80** ± 0.08	**71.11** ± 0.54	**77.27** ± 1.05

Table 5: OONP on judgement documents.

8 Conclusion

We proposed Object-oriented Neural Programming (OONP), a framework for semantically parsing in-domain documents. OONP is neural net-based, but equipped with sophisticated architecture and mechanism for document understanding, therefore nicely combining interpretability and learnability. Experiments on both synthetic and real-world datasets have shown that OONP outperforms several strong baselines by a large margin on parsing fairly complicated ontology.

Acknowledgments

We thank Fandong Meng and Hao Xiong for their insightful discussion. We also thank Classic Law Institute for providing the raw data.

References

Tim Berners-Lee, James Hendler, and Ora Lassila. 2001. The semantic web. *Scientific American* 284(5):34–43.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics(ACL)*. pages 1215–1226.

Fillmore Charles J. 1982. Frame semantics. *In Linguistics in the Morning Calm* pages 111–137.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning Journal (MLJ)* .

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *CoRR* abs/1410.5401.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2017. Tracking the world state with recurrent entity networks. In *ICLR*.

Henderson, Matthew, Blaise Thomson, , and Steve Young. 2014. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*. pages 292–299.

Jonathan Herzig and Jonathan Berant. 2017. Neural semantic parsing over multiple knowledge-bases. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics(ACL)*. pages 623–628.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A. Smith. 2017. Dynamic entity representations in neural language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing(EMNLP)*. Association for Computational Linguistics, pages 1830–1839.

Daniel D. Johnson. 2017. Learning graphical state transitions. In *the International Conference on Learning Representations(ICLR)*.

Berant Jonathan and Liang Percy. 2014. Semantic parsing via paraphrasing. In *Association for Computational Linguistics (ACL)*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Association for Computational Linguistics(ACL)*.

Xianggen Liu, Lili Mou, Haotian Cui, Zhengdong Lu, and Sen Song. 2018. Jumper: Learning when to make classification decisions in reading. In *IJCAI*.

C. J. Van Rijsbergen. 1979. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR* abs/1502.05698.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8:229–256.

Yukun Yan, Daqi Zheng, Zhengdong Lu, and Sen Song. 2017. Event identification as a decision process with non-linear representation of text. *CoRR* abs/1710.00969.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-aware language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing(EMNLP)*. Association for Computational Linguistics, pages 1850–1859.

Fisher Yu and Vladlen Koltun. 2016. Multi-scale context aggregation by dilated convolutions. In *ICLR*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. *In Proceedings of COLING* .