The Learning-Knowledge-Reasoning Paradigm For Natural Language Understanding and Question Answering

Arindam Mitra[1]
Arizona State University
[Tempe, USA]
amitra7@asu.edu
https://orcid.org/0000-0003-0089-510X

Abstract

Given a text, several questions can be asked. For some of these questions, the answer can be directly looked up from the text. However for several other questions, one might need to use additional knowledge and sophisticated reasoning to find the answer. Developing AI agents that can answer this kinds of questions and can also justify their answer is the focus of this research. Towards this goal, we use the language of Answer Set Programming as the knowledge representation and reasoning language for the agent. The question then arises, is how to obtain the additional knowledge? In this work we show that using existing Natural Language Processing parsers and a scalable Inductive Logic Programming algorithm it is possible to learn this additional knowledge (containing mostly commonsense knowledge) from question-answering datasets which then can be used for inference.

2012 ACM Subject Classification Computing methodologies → Natural language processing, Computing methodologies → Knowledge representation and reasoning

Keywords and phrases Natural Language Understanding, Question Answering, Knowledge Acquisition, Inductive Logic Programming, Knowledge Representation and Reasoning

Digital Object Identifier 10.4230/OASIcs.CVIT.2016.23

1 Introduction

Developing agents that can understand text is one of the long term goals of Artificial Intelligence. To track the progress towards this goal, several question-answering challenges have been proposed, such as, the science question answering challenge *aristo*, [1], project *euclid*'s math word problem solving [3, 4] and *facebook research*'s *bAbI* question answering challenge [9]. In all these challenges, a small text is provided describing a scenario and one or more questions based on that scenario. Table 1 shows an example from each of these three tasks.

It should be noted that answering these questions (Table 1) requires knowledge that goes beyond the text. For example, to answer the questions from the bAbI task (Table 1) one needs to know the effect of certain actions. Similarly, answering the math question requires the knowledge that the games one has won or lost is a subset of the games one has played and also that the value of a whole is equal to the sum of its parts. The later is popularly known as the *part-whole* formula. The science question on the other hand requires one to know the dynamics of predator-prey population. Some of these knowledge such as the math

[1] [funding]

Mary grabbed the football.
Mary traveled to the office.
Mary took the apple there.
What is Mary carrying? A:football,apple
Mary left the football.
Daniel went back to the bedroom.
What is Mary carrying? A:apple

(a) An example from a bAbI challenge

Sara's high school played 12 basketball games this year. The team won most of their games. They were defeated during 4 games. How many games did they win ?

(b) An example of a word arithmetic problem

In one area, a large source of prey for eagles is rabbits. If the number of rabbits suddenly decreases, what effect will it most likely have on the eagles? (A) Their numbers will increase. (B) Their numbers will decrease. (C) They will adapt new behaviors. (D) They will migrate to new locations. 296 story; example and general concept; causality interdependence food web

(c) An example of a science question

Table 1 shows an example problem from the datasets of bAbI, word math problems and Aristo.

formula or the prey-predator population dynamics, can be easily collected from books and can be provided to the agent as a background knowledge. However, some types of knowledge such as the affect of the actions or the commonsense knowledge about part whole relations between verbs might be difficult to write down manually as there exists a vast amount of such knowledge. In this research, thus we aim to learn such knowledge from question-answering dataset.

The proposed QA-architecture namely the Learning-Knowledge-Reasoning paradigm, has three components: 1) A semantic parser, T that converts the text into the required logical form, 2) An Inductive Logic Programming module, L that learns missing knowledge from the training data and 3) A reasoning engine, R which computes the answer given the query. In the training phase, given some background knowledge B and a training dataset D the Inductive Logic Programming module uses the semantic parser T and a rule learning algorithm to learn the necessary knowledge H from D. In the test phase, both B and H are used to answer a given question. We have used the language of Answer Set Programming for the purpose of knowledge representation and reasoning.

2 Background

2.1 Answer Set Programming

An answer set program is a collection of rules of the form,

$$L_0 \leftarrow L_1, ..., L_m, \textbf{not } L_{m+1}, ..., \textbf{not } L_n$$

where each of the L_i's is a literal in the sense of a classical logic. Intuitively, the above rule means that if $L_1, ..., L_m$ are true and if $L_{m+1}, ..., L_n$ can be safely assumed to be false then L_0 must be true. The left-hand side of an ASP rule is called the *head* and the right-hand side is called the *body*. Predicates and ground terms in a rule start with a lower case letter, while variable terms start with a capital letter. We will follow this convention throughout the paper. A rule with no *head* is called a *constraint*. A rule with empty *body* is referred to as a *fact*. The semantics of ASP is based on the stable model semantics of logic programming

[2]. In this work, both the background knowledge B and the learned knowledge H are a collection of such ASP rules.

2.2 Event Calculus

Event calculus is a temporal logic for reasoning about the events and their effects. The ontology of the Event calculus comprises of *time points*, *fluents* (i.e. properties which have certain values at a time point) and *events* (i.e. occurrences in time that may affect fluents and alter their value). The formalism also contains two domain-independent axioms to incorporate the commonsense *law of inertia*, according to which fluents persist over time unless they are affected by an event. The building blocks of Event calculus and its domain independent axioms are presented in Table 2.

Predicate	**Meaning**
happensAt(F, T)	Event E occurs at time T
initiatedAt(F, T)	At time T a period of time for which fluent F holds is initiated
terminatedAt(F, T)	At time T a period of time for which fluent F holds is terminated
holdsAt(F, T)	Fluent F holds at time T
Axioms	
$holdsAt(F, T+1) \leftarrow initiatedAt(F, T).$	$holdsAt(F, T+1) \leftarrow holdsAt(F, T), not\ terminatedAt(F, T).$

Table 2 The basic predicates and axioms of Simple Discrete Event Calculus (SDEC)

3 Inductive Logic Programming for Mutually Distinct Examples

Inductive Logic Programming (ILP) [7] is a subfield of Machine learning that is focused on learning logic programs. Given a set of positive examples $\mathcal{E}^+$, negative examples $\mathcal{E}^-$ and some background knowledge $\mathcal{B}$, an ILP algorithm finds an Hypothesis $\mathcal{H}$ (answer set program) such that $\mathcal{B} \cup \mathcal{H} \models \mathcal{E}^+$ and $\mathcal{B} \cup \mathcal{H} \not\models \mathcal{E}^-$. The possible hypothesis space is often restricted with a language bias that is specified by a series of mode declarations $\mathcal{M}$ [8].

This definition however does not consider the fact that a statistical machine learning dataset contains several *context dependent* examples. We recently proposed a variation of the standard ILP task namely, Inductive Logic Programming for "mutually Distinct Examples" [6] which is more suitable for working with this machine learning datasets. An ILP task for "mutually Distinct Examples" [6] (denoted as ILP^{DE}) is defined as follows:

▶ **Definition 1 (Inductive Logic Programming for Mutually Distinct Examples).** An ILP task for *Distinct Examples* (denoted as ILP^{DE}) is a tuple $\langle B, M, D\rangle$, where B is an Answer Set Program, called the background knowledge, M defines the set of rules allowed in hypotheses (the hypothesis space) and D is the dataset containing a series of mutually distinct examples $\langle E_1, E_2, ..., E_n\rangle$. Here each E_i is a tuple $\langle O_i, E_i^+, E_i^-\rangle$ where, O_i is a logic program, called *observation*, E^+ is a set of positive ground literals and E^- is a set of negative ground literals. A hypothesis H is an inductive solution of T (written as $H \in ILP^{DE}(B, M, D)$) *iff*,

$$H \cup B \cup O_i \vdash E_i^+,\ \forall i = 1...n$$

$$H \cup B \cup O_i \nvdash E_i^-,\ \forall i = 1...n$$

An iterative and incremental algorithm, has also been developed [6] to compute the solution of an ILP^{DE} task.

4 Learning Knowledge from dataset

To learn the missing knowledge H from the training dataset D, first an instance of the ILP^{DE} task is created. The iterative and incremental algorithm for ILP^{DE} in [6] is then used which outputs the desired H. In this section we describe this procedure with the example of the bAbI question answering challenge.

Background Knowledge B

The background knowledge contains the two commonsense *law of inertia* from Event calculus, according to which fluents persist over time unless they are affected by an event.

Mapping an bABI Example to an ILP^{DE} Example

The bAbI challenge contains 20 different question answering tasks. One of such task is about reasoning with sets. An example of that which is shown in table 1. The training dataset for each tasks contains 1000 of such examples. Each of such example is translated into an ILP^{DE} example $E_i =< O_i, E_i^+, E_i^- >$ in the following manner.

Given a question-answer text such as the one shown in Table 1(a), the translation module first converts the natural language sentences to the syntax of Event calculus. While doing so, it first obtains the Abstract Meaning Representation (AMR) of the sentence from the AMR parser in the statistical NLP layer and then applies a rule-based procedure to convert the AMR graph to the syntax of Event calculus. Figure 1 & 2 show two AMR representations for the sentence “Mary grabbed the football." and the question “What is Marry carrying?". The representation of the question-answer text in $< O_i, E_i^+, E_i^- >$ form is shown in Table 3. The narratives in O_i (Table 3) describe that the event of grabbing a football by Mary has happened at time point 1, then another event named *travel* has happened at time point 2 and so on. The first two annotations in E_i^+ state that both the fluents specifying Mary is carrying an apple and Mary is carrying a football holds at time point 4. The *not holdsAt* annotation in E_i^- states that at time point 7 Mary is not carrying a football.

```
(g / grab
    :ARG0 (p / person
        :name (m / name :op1 Mary ))
    :ARG1 (f / football ))
```

Figure 1 AMR representation of “Mary grabbed the football.”

```
(c / carry
    :ARG0 (m / Marry  )
    :ARG1 (a / amr-unknown)))
```

Figure 2 AMR representation of “What is Marry carrying?”

Computing the Inductive Solution

The algorithm [6] that computes the solution roughly works as follows: Given an instance of the ILP^{DE} task, it first finds a solution H_1 of E_1. Then it expands H_1 minimally to solve only E_2 and obtains H_2 . In the next iteration it again expands H_2 minimally to solve E_1

O_i	$happensAt(grab(mary, football), 1).$ $happensAt(travel(mary, office), 2).$ $happensAt(take(mary, apple), 3).$ $happensAt(leave(mary, footbal;), 5).$ $happensAt(go\_back(daniel, bedroom), 6).$
E_i^+	holdsAt(carry(mary,football),4). holdsAt(carry(mary,apple),4). holdsAt(carry(mary,apple),7).
E_i^-	not holdsAt(carry(mary,football),7).

Table 3 Representation of the Example in Table 1(a) in ILP^{DE} format.

and it continues expanding until it finds a hypothesis that solves both E_1 and E_2. Next it starts with a solution of $\langle E_1, E_2 \rangle$ and tries to expand it iteratively until it solves all of E_1, E_2 and E_3. The process continues until a hypothesis is found that explains all the examples. The algorithm is shown to be sound and complete when $H \cup B \cup O_i$ is *stratified* for all $i = 1, ..., n$, [6]. Table 4 shows the 8 rules that are learned for this task. Our system following this learning-knowledge-reasoning method outperforms all the deep learning systems for the bAbI challenge. [5].

$initiatedAt(carry(P, O), T) \leftarrow happensAt(get(P, O), T).$
$initiatedAt(carry(P, O), T) \leftarrow happensAt(take(P, O), T).$
$terminatedAt(carry(P, O), T) \leftarrow happensAt(drop(P, O), T).$
$initiatedAt(carry(P, O), T) \leftarrow happensAt(pick\_up(P, O), T).$
$initiatedAt(carry(P, O), T) \leftarrow happensAt(grab(P, O), T).$
$terminatedAt(carry(P, O), T) \leftarrow happensAt(discard(P, O), T).$
$terminatedAt(carry(P, O), T) \leftarrow happensAt(put\_down(P, O), T).$
$terminatedAt(carry(P, O), T) \leftarrow happensAt(leave(P, O), T).$

Table 4 Rules learned from the task 8 of bABI dataset

5 Current State of Research

Currently we are trying to apply this framework of learning-knowledge-reasoning to the task of word arithmetic problem solving, where the goal is to learn human readable knowledge which can help the question answering agent to decide which arithmetic formulas to apply for a particular problem and in which order.

6 Conclusion

Earlier days of Artificial Intelligence have seen many handwritten rule based systems. Later those were replaced by better performing machine learning based systems. With the advancements of knowledge representation and reasoning languages, a natural question arises, "if machines can learn logic programs, can it achieve better accuracy than existing statistical machine learning methods such neural networks?" It should be noted that the system of [5] achieved better results than the existing deep learning models on the bAbI dataset. To further explore this possibility we need to focus on the task of learning of logic programs and need to develop systems that can learn from large datasets. In this research, we have made an attempt towards that.

References

1 Peter Clark. Elementary school science and math tests as a driver for ai: take the aristo challenge! 2015.

2 Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *ICLP/SLP*, volume 88, pages 1070–1080, 1988.

3 Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, 2014.

4 Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597, 2015.

5 Arindam Mitra and Chitta Baral. Addressing a question answering challenge by combining statistical methods with inductive rule learning and reasoning. In *AAAI*, pages 2779–2785, 2016.

6 Arindam Mitra and Chitta Baral. Incremental and iterative learning of answer set programs from mutually distinct examples. *CoRR*, abs/1802.07966, 2018. URL: `http://arxiv.org/abs/1802.07966`, `arXiv:1802.07966`.

7 Stephen Muggleton. Inductive logic programming. *New generation computing*, 8(4):295–318, 1991.

8 Stephen Muggleton. Inverse entailment and progol. *New generation computing*, 13(3-4):245–286, 1995.

9 Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. Towards ai-complete question answering: a set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.