Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Addressing a Question Answering Challenge by Combining Statistical Methods with Inductive Rule Learning and Reasoning

Arindam Mitra
Arizona State University
amitra7@asu.edu

Chitta Baral
Arizona State University
chitta@asu.edu

Abstract

A group of researchers from Facebook has recently proposed a set of 20 question-answering tasks (Facebook's bAbl dataset) as a challenge for the natural language understanding ability of an intelligent agent. These tasks are designed to measure various skills of an agent, such as: fact based question-answering, simple induction, the ability to find paths, co-reference resolution and many more. Their goal is to aid in the development of systems that can learn to solve such tasks and to allow a proper evaluation of such systems. They show existing systems cannot fully solve many of those toy tasks. In this work, we present a system that excels at all the tasks except one. The proposed model of the agent uses the Answer Set Programming (ASP) language as the primary knowledge representation and reasoning language along with the standard statistical Natural Language Processing (NLP) models. Given a training dataset containing a set of narrations, questions and their answers, the agent jointly uses a translation system, an Inductive Logic Programming algorithm and Statistical NLP methods to learn the knowledge needed to answer similar questions. Our results demonstrate that the introduction of a reasoning module significantly improves the performance of an intelligent agent.

Developing intelligent agents is one of the long term goals of Artificial Intelligence. To track the progress towards this goal, several challenges have been recently proposed that employs a Question-Answering (QA) based strategy to test an agent's understanding. The *Allen Institute for AI*'s flagship project ARISTO, (Richardson, Burges, and Renshaw 2013)'s *MCTest* and the *Winograd Schema Challenge* (Levesque, Davis, and Morgenstern 2012) are all examples of this. As mentioned in the work of (Weston et al. 2015), even though these tasks are promising and provide real world challenges, successfully answering their questions require competence on many sub-tasks (deduction, use of common-sense, abduction, coreference etc.); which makes it difficult to interpret the results on these benchmarks. Often the state-of-the-art systems are highly domain specific and rely heavily on the prior knowledge. In this light, they (Weston et al. 2015) have proposed a new dataset (Facebook bAbl dataset) that put together several question-answering tasks where solving each task develops a new skill set into an agent.

In the following paragraph, we provide some examples of the tasks from (Weston et al. 2015). A detailed description of all the tasks can be found there. Each task is noiseless, provides a set of training and test data and a human can potentially achieve 100% accuracy.

Example 1. Task 8: List/Sets

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

Example 2. Task 19: Path Finding

The office is east of the hallway.
The kitchen is north of the office.
The garden is west of the bedroom.
The office is west of the garden.
The bathroom is north of the garden.
How do you go from the kitchen to the garden? A:s,e

In this work, we describe an agent architecture that simultaneously works with a formal reasoning model and a statistical inference based model to address the task of question-answering. Human beings in their lifetime learn to perform various tasks. For some tasks they may have a clear reasoning behind their actions. For example, the knowledge needed to answer the previous question "What is Mary carrying?" is clear and can be described formally. On the other hand, there are tasks such as Named Entity Recognition that we can do easily, however, we may not be able to describe it well enough for anyone else to use the description for recognition. In these cases, a statistical inference model that allows to learn by observing a distribution may be a better fit. In this research, thus, we work with a heterogeneous agent model. In our current implementation, the agent model contains three layers.

Statistical Inference Layer This layer contains statistical NLP models. In this case study, it contains only an Ab-

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 1: The basic predicates and axioms of Simple Discrete Event Calculus (SDEC)

stract Meaning Representation Parser (AMR) (Banarescu et al. 2013; Flanigan et al. 2014).

Formal Reasoning Layer This layer is responsible for formal reasoning. It uses the Answer Set Programming (ASP) (Gelfond and Lifschitz 1988) language as the knowledge representation and reasoning language. The knowledge required for reasoning is learned with a modified version of the Inductive Logic Programming algorithm XHAIL (Ray 2009). The reasoning module takes sentences represented in the logical language of Event calculus which is a temporal logic for reasoning about the events and their efforts. The ontology of the Event calculus comprises of *time points*, *fluent* (i.e. properties which have certain values in time) and *event* (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 1.

Translation layer The translation layer encodes the natural language sentences to the syntax of Event calculus with the help of the AMR parser from the statistical inference layer. This layer communicates with both the other layers and allows information to be passed from one layer to another. In this case study, we use a naive deterministic algorithm to model the translation layer.

Given a question-answer text such as the one shown in Example 1 (Task 8), 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 sentences (*narratives*) and the question-answer pairs (*annotation*) of Example 1 in Event calculus is shown in Table 2. The narratives in Table 2 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 state that both the fluents specifying Mary is carrying an apple and Mary is carrying a football holds at time point 3. The $not\ holdsAt$ annotation states that at time point 7 Mary is not carrying a football. Given such a set of narratives and annotations the reasoning module employs an Inductive Logic Programming algorithm to derive a Hypothesis $\mathcal{H}$, that can explain all the annotations. Given more examples it updates the existing Hypothesis $\mathcal{H}$ incrementally to remain consistent with the data.

The rest of the paper is organized as follows: in section 1, we provide a brief overview of Answer Set Programming and Inductive Logic Programming; In section 2, we describe the way the task specific ASP reasoning rules are learned. Section 3 presents training of the coreference resolution system with reasoning. In section 4, we describe the related works. In section 5, we present a detailed experimental evaluation of our system. Finally, section 6 concludes our paper. Further details are available at *http://goo.gl/JMzHbG*.

```
(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?"

Narrative
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).
Annotation
holdsAt(carry(mary,football),4).
holdsAt(carry(mary,apple),4).
holdsAt(carry(mary,apple),7).
not holdsAt(carry(mary,football),7).

Table 2: Representation of the Example 1 in Event Calculus

1 Background

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 (Baral 2003) . The left-hand side of an ASP rule is called the *head* and the right-hand side is called the *body*. The semantics of ASP is based on the stable model (answer set) semantics of logic programming (Gelfond and Lifschitz 1988).

Example

$$initiatedAt(carry(A, O), T) \leftarrow$$
$$happensAt(take(A, O), T). \quad (1)$$

The above rule represents the knowledge that the fluent $carry(A, O)$, denoting A is carrying O, gets initiated at time point T if the event $take(A, O)$ occurs at T. Following Prolog's convention, throughout this paper, predicates and ground terms in logical formulae start with a lower case letter, while variable terms start with a capital letter. A rule with no *head* is often referred to as a $constraint$. A rule with empty *body* is referred to as a $fact$. An answer set program $\mathcal{P}$ containing the above rule (Rule 1) and the axioms of Event calculus (from Table 1) along with the fact $happensAt(take(mary, football), 1)$ logically entails ($\models$) that *mary* is *carrying* a *football* at time point 2 i.e. $holdsAt(carry(mary, football), 2)$. Since it can be safely assumed that *mary* is not *carrying* a *football* at time point 1, $\mathcal{P} \models not\ holdsAt(carry(mary, football), 1)$ or equivalently $\mathcal{P} \not\models holdsAt(carry(mary, football), 1)$.

It should be noted that it is also true that $\mathcal{P} \models holdsAt(carry(mary, football), 3)$, due to the axioms in Table 1. However, if we add the following two rules in the program $\mathcal{P}$:

$$terminatedAt(carry(A, O), T) \leftarrow$$
$$happensAt(drop(A, O), T). \quad (2)$$
$$happensAt(drop(marry, football), 2). \quad (3)$$

then the new program $\mathcal{P}$ will no longer entail $holdsAt(carry(mary, football), 3)$ due the axioms of Event calculus. This is an example of non-monotonic reasoning when adding more knowledge changes one's previous beliefs and such thing is omnipresent in human reasoning. First Order Logic does not allow non-monotonic reasoning and this is one of the reasons why we have used the Answer Set Programming language as the formal reasoning language.

Inductive Logic Programming

Inductive Logic Programming (ILP) (Muggleton 1991) 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 $\mathcal{M}$ (Muggleton 1995). A *modeh(s)* declaration denotes a literal *s* that can appear as the head of a rule (Table 3). A *modeb(s)* declaration denote a literal *s* that can appear in the body of a rule (Table 3). The argument *s* is called *schema* and comprises of two parts: 1) an identifier for the literal and 2) a list of *placemakers* for each argument of that literal. A *placemaker* is either *+type* (input), *-type* (output) or *#type* (constant), where *type* denotes the type of the argument. An answer set rule is in the hypothesis space defined by $\mathcal{L}$ (call it $\mathcal{L}(\mathcal{M})$) iff its head (resp. each of its body literals) is constructed from the schema *s* in a *modeh(s)* (resp. in a *modeb(s)*) in $\mathcal{L}(\mathcal{M})$) as follows:

- By replacing an output (-) placemaker by a new variable.
- By replacing an input (+) placemaker by a variable that appears in the head or in a previous body literal.
- By replacing a ground (#) placemaker by a ground term.

*modeh(*initiatedAt*(carrying(+arg$_1$,+arg$_3$),+time))*
*modeh(*terminatedAt*(carrying(+arg$_1$,+arg$_3$),+time))*
*modeb(*happensAt*(grab(+arg$_1$,+arg$_2$),+time))*
*modeb(*happensAt*(take(+arg$_1$,+arg$_3$),+time))*
*modeb(*happensAt*(go_back(+arg$_1$,+arg$_2$),+time))*
*modeb(*happensAt*(leave(+arg$_1$,+arg$_3$),+time))*

Table 3: Mode declarations for the problem of Task 8

Table 3 shows a set of mode declarations $\mathcal{M}$ for the example problem of Task 8. The Rule 1 of the previous section is in this $\mathcal{L}(\mathcal{M})$ and so is the fact,

$$initiated(carrying(A, O), T).$$

However the following rule is not in $\mathcal{L}(\mathcal{M})$).

$$initiated(carrying(A, O), T) \leftarrow$$
$$happensAt(take(A, O), T').$$

The set $\mathcal{E}^-$ is required to restrain $\mathcal{H}$ from being over generalized. Informally, given a ILP task, an ILP algorithm finds a hypothesis $\mathcal{H}$ that is general enough to cover all the examples in $\mathcal{E}^+$ and also specific enough so that it does not cover any example in $\mathcal{E}^-$. Without $\mathcal{E}^-$, the learned $\mathcal{H}$ will contain only facts. In this case study, negative examples are automatically generated from the positive examples by assuming the answers are complete, i.e. if a question-answer pair says that at a certain time point *mary* is carrying a *football* we assume that *mary* is not carrying anything else at that time stamp.

2 Learning Answer Set Programs for QA

In this section, we illustrate the formulation of an ILP task for a QA task and the way the answer set programs are learned. We explain our approach with the XHAIL (Ray 2009) algorithm and specify why ILED (Katzouris, Artikis, and Paliouras 2015) algorithm is needed. We continue with the example of Task 8 and conclude with path finding.

Task 8: Lists/Sets

Given an ILP task $ILP(\mathcal{B}, \mathcal{E} = \{\mathcal{E}^+ \cup \mathcal{E}^-\}, \mathcal{M})$, XHAIL derives the hypothesis in a three step process. For the example of task 8, $\mathcal{B}$ contains both the axioms of SDEC and the

narratives from Table 1. The set $\mathcal{E}$ comprises of the annotations from Table 1 which contains three positive and one negative examples. $\mathcal{M}$ is the set of mode declarations in Table 2.

Step 1 In the first step the XHAIL algorithm finds a set of ground (variable free) atoms $\triangle = \cup_{i=1}^{n} \alpha_i$ such that $\mathcal{B} \cup \triangle \models \mathcal{E}$ where each α_i is a ground instance of the *modeh(s)* declaration atoms. For the example ILP problem of task 8 there are two *modeh* declarations. Thus the set $\triangle$ can contain ground instances of only those two atoms described in two *modeh* declarations. In the following we show one possible $\triangle$ that meets the above requirements for the ILP task of Example 1.

$$\triangle = \left\{ \begin{array}{l} initiatedAt(carry(mary, football), 1) \\ initiatedAt(carry(mary, apple), 3) \\ terminatedAt(carry(mary, football), 5) \end{array} \right\}$$

Step 2 In the second step, XHAIL computes a clause $\alpha_i \leftarrow \delta_i^1 ... \delta_i^{m_i}$ for each α_i in $\triangle$, where $\mathcal{B} \cup \triangle \models \delta_i^j, \forall 1 \leq i \leq n, 1 \leq j \leq m_i$ and each clause $\alpha_i \leftarrow \delta_i^1 ... \delta_i^{m_i}$ is a ground instance of a rule in $\mathcal{L}(\mathcal{M})$. In the running example, $\triangle$ contains three atoms that each must lead to a clause $k_i, i = 1, 2, 3$. The first atom $\alpha_1 = initiatedAt(carry(mary, football), 1)$ is initialized to the head of the clause k_1. The body of k_1 is saturated by adding all possible ground instances of the literals in *modeb(s)* declarations that satisfy the constraints mentioned above. There are six ground instances (all the narratives) of the literals in the *modeb(s)* declarations; however only one of them, i.e. $happensAt(grab(mary, football), 1)$ can be added to the body due to restrictions enforced by $\mathcal{L}(\mathcal{M})$. In the following we show the set of all the ground clauses $\mathcal{K}$ constructed in this step and their variabilized version $\mathcal{K}_v$ that is obtained by replacing all input and output terms by variables.

$$K = \left\{ \begin{array}{l} initiatedAt(carry(mary, football), 1) \\ \quad \leftarrow happensAt(grab(mary, football), 1). \\ initiatedAt(carry(mary, apple), 3) \\ \quad \leftarrow happensAt(take(mary, apple), 3). \\ terminatedAt(carry(mary, football), 6) \\ \quad \leftarrow happensAt(leave(mary, apple), 6). \end{array} \right\}$$

$$K_v = \left\{ \begin{array}{l} initiatedAt(carry(X, Y), T) \\ \quad \leftarrow happensAt(grab(X, Y), T). \\ initiatedAt(carry(X, Y), T) \\ \quad \leftarrow happensAt(take(X, Y), T). \\ terminatedAt(carry(X, Y), T) \\ \quad \leftarrow happensAt(leave(X, Y), T). \end{array} \right\}$$

Step 3 In this step XHAIL tries to find a compressive theory $\mathcal{H}$ by deleting from K_v as many literals (and clauses) as possible while ensuring that $\mathcal{B} \cup \mathcal{H} \models \mathcal{E}$. In the running example, working out this problem will lead to $\mathcal{H} = \mathcal{K}_v$.

Scalability of the learning algorithm The discovery of a hypothesis $\mathcal{H}$ depends on the choice of $\triangle$. Since the value of $\triangle$ that satisfies the constraints described in Step 1 is not unique, we employ an iterative deepening strategy to select $\triangle$ of progressively increasing size until a solution is found. Furthermore, in Step 2 of XHAIL we restricted the algorithm to consider only those ground instances of *modeb* declarations that are not from the future time points. This method works when the size of the example is small. However, the dataset of Task 8 like other tasks contains 1000 examples, where each example comprises of a set of narrative and annotations (as we have shown before) and the choice of $\triangle$ will be numerous. This issue is addressed by learning rules from each example and then using the learned rules to learn new rules from yet unsolved examples. A recent incremental learning algorithm, ILED (Katzouris, Artikis, and Paliouras 2015) can be used to address the scalability issue. The first step of the ILED algorithm is the XHAIL algorithm. After finding an initial hypothesis $\mathcal{H}_1$ by XHAIL, the ILED algorithm incrementally revises the current hypothesis $\mathcal{H}_i$ when subjected to a new example E_i so that the revised hypothesis $\mathcal{H}_{i+1}$ is consistent with the current example E_i and all the previous ones $E_0, ..., E_{i-1}$. It will be interesting to see if ILED can scale up to this dataset.

Task 19 : Path Finding

In this task (Example 2), each example first describes the relative positions of several places and then asks a question about moving from one place to another. The answer to the question is then a sequence of directions. For the question "How do you go from the kitchen to the garden?" in Example 2, the answer "s,e" tells that to reach garden from kitchen, you should first head south and then head east.

Given such an example, an agent learns how moving towards a direction changes its current location with respect to the particular orientation of the places. Let us say, $mt(X, Y)$ denotes the event of X moving towards the direction Y. Similar to the earlier problem, the natural language text is first translated to the syntax of ASP (Table 4). However, in this task the background knowledge $\mathcal{B}$ also contains the rules learned from the task 4. In the following we show an example of such rules:

$$holdsAt(relative\_position(X, Y, east), T) \leftarrow holdsAt(relative\_position(Y, X, west), T).$$

The above rule says that if Y is to the west of X at time point T then X is to the east of Y at T. Similar rules were learned for each direction pair from the Task 4 which were used in the process of hypothesis generation for the task of path finding. Table 4 shows the corresponding ILP task for the example of path finding and the hypothesis generated by the XHAIL algorithm. This example illustrates how the task of path finding can be easily learned when a formal representation is used. While the state-of-the-art neural network based systems have achieved 36% accuracy on this task with an average of 93% on all tasks, our system is able to achieve 100% with the two compact rules shown in Table 4.

Input

Narrative

holdsAt(relative_position(office,hallway,east),1).
holdsAt(relative_position(kitchen,office,north),2).
holdsAt(relative_position(garden,bedroom,west),3).
holdsAt(relative_position(office,west,garden),4).
holdsAt(relative_position(bathroom,garden,north),5).
holdsAt(location(you,kitchen),6).
happensAt(mt(you,south),6).
happensAt(mt(you,east),7).

Annotation

$not\ holdsAt(location(you, garden), 6).$
$holdsAt(location(you, garden), 8).$
$not\ holdsAt(location(you, kitchen), 8).$

Mode declarations

*modeh(*initiatedAt*(location(+arg$_1$,+arg$_2$),+time))*
*modeh(*terminatedAt*(carrying(+arg$_1$,+arg$_2$),+time))*
*modeb(*happensAt*(mt(+arg$_1$,+direction),+time))*
*modeb(*holdsAt*(location(+arg$_1$,+arg$_2$),+time))*
*modeb(*holdsAt*(relative_position(+arg$_2$,+arg$_2$, +direction),+time))*

Background Knowledge

Axioms of SDEC (Table 1)

Output

$initiatedAt(location(X, Y), T) \leftarrow$
$happensAt(move\_towards(X, D), T),$
$holdsAt(relative\_position(Y, Z, D), T),$
$holdsAt(location(X, Z), T).$
$terminatedAt(location(X, Y), T) \leftarrow$
$happensAt(mt(X, D), T).$

Table 4: Hypothesis Generation For Path Finding

3 Learning Coreference Resolution with Reasoning

The dataset contains contains two tasks related to coreference resolution : 1) task of basic coreference resolution and 2) task of compound coreference resolution. Examples of the tasks are shown below :

Task 11: Basic Coreference

Mary went back to the bathroom.
After that she went to the bedroom.
Daniel moved to the office.
Where is Mary? bedroom

Task 13: Compound Coreference

Daniel and Sandra journeyed to the office.
Then they went to the garden.
Sandra and John travelled to the kitchen.
The office is west of the garden.
After that they moved to the hallway.
Where is Daniel? A:garden

We formulate both the coreference resolution tasks as ILP problems and surprisingly it learns answer set rules that can fully explain the test data. For the task of basic coreference, it learns a total of five rules one for each of the five different events *go, travel, go back, move, journey* that appeared in the training data. The rule corresponding to the event *go* (Table 5) states that if a narrative at time point $T+1$ contains a pronoun, then the pronoun is referring to the arg_1 (agent) of the event *go* that happened at time point T. Similar rules were learned for the other four events. Here, $corefId(X, T)$ denotes that the pronoun with id X has appeared in a narrative at time point at $T + 1$.

$initiatedAt(coref(X, Y), T) \leftarrow corefId(X, T),$
$happensAt(go(Y, Z), T).$

Table 5: One rule for coreference resolution

One drawback of the learned rules is, they are event dependent, i.e. if a coreference resolution text contains a pronoun which is referring to an argument of an previously unseen event, these rules will not be able to resolve the coreference. In spite of that, these rules reflect one of the basic intuitions behind coreference resolution and all of them are learned from data.

4 Related Works

In this section, we briefly describe the two other attempts on this challenge. The attempt using Memory Network (MemNN) (Weston, Chopra, and Bordes 2014) formulates the QA task as a search procedure over the set of narratives. This model takes as input the Question-Answering samples and the set of facts required to answer each question. It then learns to find 1) the supporting facts for a given question and 2) the word or set of words from the supporting facts which are given as answer. Even though this model performs well on average, the performance on the tasks of positional reasoning (65%) and path finding (36%) are far below from the average (93%).

The attempt using Dynamic Memory Network (DMN) (Kumar et al. 2015) also models the the QA task as a search procedure over the set of narratives. The major difference being the way supporting facts are retrieved. In the case of the Memory Networks, given a question, the search algorithm scans the narratives in the reverse order of time and finds the most relevant hypothesis. It then tries to find the next most relevant narrative and the process continues until a special marker narrative is chosen to be the most relevant one in which case the procedure terminates. In the case of Dynamic Memory Networks the algorithm first identifies a set of useful narratives conditioning on the question and updates the agent's current state. The process then iterates and in each iterations it finds more useful facts that were thought to be irrelevant in the previous iterations. After several passes the module finally summarizes its knowledge and provides the answer. Both the models rely only on the given narratives to answer a question. However, for many QA tasks (such as task of Path finding) it requires additional knowledge that is not present in the text (for path finding, knowledge from Task 4), to successfully answer a question.

TASK	MemNN	DMN	Our Method
1: Single Supporting Fact	100	100	100
2: Two Supporting Facts	100	98.2	100
3: Three Supporting facts	100	95.2	100
4: Two Argument Relations	100	100	100
5: Three Argument Relations	98	99.3	100
6: Yes/No Questions	100	100	100
7: Counting	85	96.9	100
8: Lists/Sets	91	96.5	100
9: Simple Negation	100	100	100
10: Indefinite Knowledge	98	97.5	100
11: Basic Coreference	100	99.9	100
12: Conjunction	100	100	100
13: Compound Coreference	100	99.8	100
14: Time Reasoning	99	100	100
15: Basic Deduction	100	100	100
16: Basic Induction	100	99.4	93.6
17: Positional Reasoning*	65	59.6	100
18: Size Reasoning	95	95.3	100
19: Path Finding	36	34.5	100
20: Agent's Motivations*	100	100	100
Mean Accuracy(%)	93.3	93.6	99.68

Table 6: Performance on the set of 20 tasks

Both MemNN and DMN models suffer in this case whereas our method can swiftly combine knowledge learned from various tasks to handle more complex QA tasks.

5 Experiments

Table 6 shows the performance of our method on the set of 20 tasks. For each task, there are 1000 questions for training and 1000 for testing. Our method was able to answer all the question correctly except the ones testing basic induction. In the following we provide a detail error analysis for the task of Induction. For each task the *modeh* and the *modeb* declarations were manually defined and can be found at the project website. The test set of Task 5 (Three argument relations) contains 2 questions that have incorrect answers. The result is reported on the corrected version of that test set. The details on the error can be found on the project website. Training of the tasks that are marked with (*) used the annotation of supporting facts present in the training dataset.

Error Analysis for Basic Induction This task tests basic induction via potential inheritance of properties. The dataset contains a series of examples like the one described below:

> Lily is a frog.
> Julius is a swan.
> Julius is green.
> Lily is grey.
> Greg is a swan.
> What color is Greg? green

The learning algorithm could not find a hypothesis that can characterize the entire training data with the given set of mode declarations. So, we took the hypothesis that partially explained the data. This was obtained by ignoring the examples in the training data which resulted in a failure. The resulted hypothesis then contained the following single rule:

$$holdsAt(color(X,C),T) \leftarrow holdsAt(domain(Z,D),T), holdsAt(color(Z,C),T), holdsAt(domain(X,D),T).$$

The above rule says that X has color C at time T if there exists a Z which is of type D and has color C at time point T, where X is also of type D. This rule was able to achieve 93.6% accuracy on the test set. However it failed for the examples of following kind where there are two different entity of type D having two different colors:

Error Case 1	Error Case 2
Lily is a frog.	Lily is a rhino.
Brian is a frog.	Lily is yellow.
Greg is frog.	Bernhard is a frog.
Lily is yellow.	Bernhard is white.
Julius is a frog.	Brian is a rhino.
Brian is grey.	Greg is a rhino.
Julius is grey.	Greg is yellow.
Greg is grey.	Julius is a rhino.
Bernhard is a frog.	Julius is green.
What color is Bernhard? A:grey	What color is Brian? A:green

Table 7: Failure cases for Induction

For the error case 1, the learned rule will produce two answers stating that Bernhard has the color grey and yellow. Since, the more number of frogs are grey it may seem like the correct rule should produce the color that has appeared maximum number of times for that type (here, frog). However, error case 2 describes a complete opposite hypothesis. There are two yellow rhino and one grey rhino and the color of Brian which is a rhino is grey. The actual rule as it appears is the one that determines the color on the basis of the latest evidence. Since, Memory Networks scans the facts in the deceasing order of time, it always concludes from the recent most narratives and thus has achieved a 100% accuracy.

6 Conclusion

This paper presents a learning approach for the task of Question-Answering that benefits from the field of knowledge representation and reasoning, inductive logic programming and statistical natural language processing. We have shown that the addition of a formal reasoning layer significantly increases the reasoning capability of an agent. We anticipate our attempt to be a starting point for more sophisticated architectures of agents that combine statistical NLP methods with formal reasoning methods.

7 Acknowledgement

We thank NSF for the DataNet Federation Consortium grant OCI-0940841 and ONR for their grant N00014-13-1-0334 for partially supporting this research.

References

Banarescu, L.; Bonial, C.; Cai, S.; Georgescu, M.; Griffitt, K.; Hermjakob, U.; Knight, K.; Koehn, P.; Palmer, M.; and Schneider, N. 2013. Abstract meaning representation for sembanking.

Baral, C. 2003. *Knowledge representation, reasoning and declarative problem solving*. Cambridge university press.

Flanigan, J.; Thomson, S.; Carbonell, J.; Dyer, C.; and Smith, N. A. 2014. A discriminative graph-based parser for the abstract meaning representation.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *ICLP/SLP*, volume 88, 1070–1080.

Katzouris, N.; Artikis, A.; and Paliouras, G. 2015. Incremental learning of event definitions with inductive logic programming. *Machine Learning* 100(2-3):555–585.

Kumar, A.; Irsoy, O.; Su, J.; Bradbury, J.; English, R.; Pierce, B.; Ondruska, P.; Gulrajani, I.; and Socher, R. 2015. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.

Levesque, H. J.; Davis, E.; and Morgenstern, L. 2012. The winograd schema challenge. In *KR*.

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

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

Ray, O. 2009. Nonmonotonic abductive inductive learning. *Journal of Applied Logic* 7(3):329–340.

Richardson, M.; Burges, C. J.; and Renshaw, E. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, volume 1, 2.

Weston, J.; Bordes, A.; Chopra, S.; and Mikolov, T. 2015. Towards ai-complete question answering: a set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Weston, J.; Chopra, S.; and Bordes, A. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.