Learning Commonsense Knowledge through Interactive Dialogue

Benjamin Wu, Alessandra Russo, Mark Law and Katsumi Inoue

June 2, 2018

Learning Commonsense Knowledge through Interactive Dialogue

Benjamin Wu
Imperial College London
[Department of Computing, Imperial College London, SW7 2AZ, UK]
benjamin.wu16@imperial.ac.uk

Alessandra Russo
Imperial College London
[Department of Computing, Imperial College London, SW7 2AZ, UK]
a.russo@imperial.ac.uk

Mark Law
Imperial College London
[Department of Computing, Imperial College London, SW7 2AZ, UK]
mark.law09@imperial.ac.uk

Katsumi Inoue
National Institute of Informatics
[2 Chome-1-2 Hitotsubashi, Chiyoda, Tokyo]
inoue@nii.ac.jp

Abstract

One of the most difficult problems in Artificial Intelligence is related to acquiring commonsense knowledge – to create a collection of facts and information that an ordinary person should know. In this work, we present a system that, from a limited background knowledge, is able to learn to form simple concepts through interactive dialogue with a user. We approach the problem using a syntactic parser, along with a mechanism to check for synonymy, to translate sentences into logical formulas represented in Event Calculus using Answer Set Programming (ASP). Reasoning and learning tasks are then automatically generated for the translated text, with learning being initiated through question and answering. The system is capable of learning with no contextual knowledge prior to the dialogue. The system has been evaluated on stories inspired by the Facebook's bAbI's question-answering tasks, and through appropriate question and answering is able to respond accurately to these dialogues.

2012 ACM Subject Classification Computing methodologies → Knowledge representation and reasoning

Keywords and phrases Commonsense Reasoning, Answer Set Programming, Event Calculus, Inductive Logic Programming.

1 Introduction

Learning commonsense knowledge is one of the major long-term goals in the research of Artificial Intelligence [6]. In recent years, there have been major developments in the area of Natural Language Processing, particularly in the automation of linguistic structure analysis [10], however the challenge of disambiguation and learning commonsense still remains [26]. Consider the sentence, "I pulled the pin out of the apple and there was a hole in it." Immediately we understand that the "it" in the sentence is referring to the apple and not the pin. Consider another example, "I saw a bird flying with binoculars." This could be interpreted in two different ways: either I used the binoculars to see the bird; or the bird was in possession of the binoculars while flying. These are examples of some of the current difficulties and challenges faced by computers in text comprehension and is what motivates our research.

Research into the problem of Commonsense reasoning is divided mainly into knowledge-based approaches and statistical machine learning approaches that use large corpora of data. Some of these approaches include: Word Sense Disambiguation using Topic Models [4]; Learning Common Sense through Visual Abstraction [28]; Representation Learning for Predicting Commonsense Ontologies [19]; and Commonsense Knowledge Base Completion [18] amongst others. For humans, learning commonsense through dialogue is a very natural thing to do. In fact, students who are placed in a learning environment may very well need to interact with a teacher or some learning facilitator in order to receive feedback and guidance [5]. Just as it is natural for humans to learn through dialogue, this paper presents a system that is able to automatically acquire common-sense knowledge through dialogic interaction.

There are two main aspects to the system: the knowledge representation, reasoning and inductive learning; and understanding and conversion from text to logic. For the first aspect, we use Answer Set Programming (ASP) in combination with a suitable form of Event Calculus [14] to represent the knowledge, along with ILASP [15] [16] [17] and clingo [8] as our main systems for reasoning and learning commonsense knowledge; and for the second, we use spaCy [11] as the syntactic parser along with WordNet [23] to help with checking synonymy for the translation from text to logic. The project uses stories as inspired by those from the Facebook's bAbI dataset [29] to see if the system can understand simple sentences with little ambiguity and develop the system's ability to gradually learn from and form valid hypotheses about such stories.

In Section 2 of this paper we will first discuss some background knowledge required to understand the tools used in our system. Following this in Section 3 we overview and explain the approach taken towards solving the problem at hand. Section 4 then discusses how we have evaluated the system and its capabilities and Section 5 touches briefly upon other related works that have been done on machine learning from dialogues. The paper is then concluded in Section 6 with remarks about potential areas for future work followed by the Appendix which contains examples of interactive dialogues between a user and our system.

2 Background

2.1 Answer Set Programs

Answer Set Programming (ASP) is a form of declarative programming directed at complex search problems [20]. ASP is based on stable model semantics of logic programming [9] and the search problems in ASP are reduced to computing such stable models using solvers that perform such search tasks. For the purposes of this paper, we will assume the following subset of the ASP language:

A *literal* can be either an atom p or its *default negation*, $not\, p$. A *normal rule* is of the form $h \leftarrow b_1, ..., b_n, not\, c_1, ..., not\, c_m$ where h is the *head* and $b_1, ..., b_n, not\, c_1, ..., not\, c_m$ collectively is the *body* of the rule, where b_i and c_j are *atoms*. A *constraint* is a rule with an empty head and is of the form $\leftarrow b_1, ..., b_n, not\, c_1, ..., not\, c_m$. An expression of the form $l\{h_1, ..., h_k\}u$ is called an *aggregate*, where h_i are atoms for $1 \leq i \leq k$, and l and u are integers such that $0 \leq l \leq u \leq k$. A variable V that occurs in a rule R is considered *safe* if it occurs in at least one positive body literal of R.

An Answer Set Program P is a finite set of normal rules and constraints. Given an ASP program P, the Herbrand Base of P, denoted as HB_P , is the set of all ground atoms that can be formed from the predicates and constants that appear in P. When P includes only normal rules, a set $A \subseteq HB_P$ is an Answer Set of P iff it is the minimal model of the reduct P^A. The reduct P^A is constructed from the grounding of P by removing any rule whose body contains a literal $not\, c_i$ where $c_i \in A$, and removing any negative literals in the remaining rules. Given an Answer Set Program P, we denote the set of all Answer Sets of P as $AS(P)$. A *partial interpretation* e is a pair $e = \langle e^{inc}, e^{exc} \rangle$ of sets of ground atoms, called the *inclusions* and *exclusions* respectively. An Answer Set A *extends* $e = \langle e^{inc}, e^{exc} \rangle$ if and only if $(e^{inc} \subseteq A) \wedge (e^{exc} \cap A = \emptyset)$.

2.2 ILASP

ILASP [15] is an ILP algorithm targeted at learning answer set programs. In this paper, we consider a simplification of ILASP's full learning framework presented in [17], called *Context-Dependent Learning from Answer Sets*. In this framework, examples are context-dependent partial interpretations, which each consist of a partial interpretation and an ASP program called the *context* of the example. Contexts allow the expression of background concepts that only apply to specific examples. They can also be used to further structure the background knowledge in an example-specific manner, thus bringing about improvements in the performance of the learning algorithm [17].

▶ **Definition 1.** A context-dependent partial interpretation is a tuple $\langle e, C \rangle$, where $e = \langle e^{inc}, e^{exc} \rangle$ is a partial interpretation and C is an ASP program called the *context*.

The hypothesis space is defined by $M = \langle M_h, M_b \rangle$ called the *language bias* of the task which is made up of a set of *head mode declarations* (M_h) and a set of *body mode declarations* (M_b). A rule $h \leftarrow b_1, ..., b_n, not\, c_1, ..., not\, c_m$ is contained within the search space S_M if and only if it satisfies the following:

1. The head is empty; or h is an atom compatible with a mode declaration in M_h.
2. The atoms b_i and c_j are all compatible with mode declarations in M_b, $\forall i \in [1, n]$ and $\forall j \in [1, m]$.
3. All variables in the rule are safe.

Each rule R in S_M is given a unique identifier R_{id}.

▶ **Example 2.** Consider the mode declarations $M = \langle M_h, M_b \rangle$ with $M_h = \{is\_in(v, v), is\_holding(v, v)\}$ and $M_b = \{went\_to(v, v), picked\_up(v, v)\}$, and where v denotes that the arguments of the predicates are variables. Some of the possible rules that are contained in the hypothesis space S_M include:

$is\_in(V0, V1) \leftarrow went\_to(V0, V1)$
$is\_holding(V0, V1) \leftarrow picked\_up(V0, V1)$
$\leftarrow went\_to(V0, V0)$

Examples of rules that are not in S_M are:

$is\_in(V0, V1) \leftarrow is\_holding(V0, V1)$
$went\_to(V0, V1) \leftarrow picked\_up(V0, V1)$

▶ **Definition 3.** A *Context-Dependent Learning from Answer Sets* task is a tuple $T = \langle B, S_M, E^+, E^- \rangle$ where B is the background knowledge, S_M is the hypothesis space defined by a language bias M, E^+ is a set of context-dependent partial interpretations, called the *positive examples* and E^- is a set of context-dependent partial interpretations called the *negative examples*. An hypothesis H is an *inductive solution* of T if and only if:

1. $H \subseteq S_M$

2. $\forall \langle e, C\rangle \in E^{+} \quad \exists A \in AS(B \cup H \cup C)$ *such that* A *extends* e
3. $\forall \langle e, C\rangle \in E^{-} \quad \nexists A \in AS(B \cup H \cup C)$ *such that* A *extends* e
Such a solution is written as $H \in ILP_{LAS}^{context}(T)$.

▶ **Example 4.** An example of a Context-Dependent Learning from Answer Sets ($ILP_{LAS}^{context}$) task can be represented in the following manner:

```
% Background Knowledge
picked_up(john,football).

% Context dependent partial interpretations
% positive examples have the form:
% #pos(id, inclusions, exclusions, Context).
#pos(p1, { is_in(mary,garden), is_holding(john,football) },
         { is_in(john,garden) },
         { went_to(mary,garden). } ).

#pos(p2, { is_in(john,garden), is_holding(john,football) },
         { is_holding(mary,football) },
         { went_to(john,garden). } ).

% Mode declarations
#modeh(is_in(var(person),var(location))).
#modeh(is_holding(var(person),var(object))).
#modeb(1, went_to(var(person),var(location))).
#modeb(1, picked_up(var(person),var(object))).
```

In this example, the Context-Dependent Learning from Answer Sets task includes a single fact in the Background Knowledge and two positive examples with no negative examples. The mode head (#modeh) and mode body (#modeb) declarations here generate the same hypothesis space as discussed in Example 2. Here both the positive examples share the background knowledge that John picked up the football, however the contexts of each example differs in that Mary went to the garden in the first positive example and John in the second. The above learning task would produce as optimal solution the following hypothesis, H:

```
is_in(V0,V1) :- went_to(V0,V1).
is_holding(V0,V1) :- picked_up(V0,V1).
```

The Answer Set produced by $AS(B \cup H \cup C_1)$ extends the first example and $AS(B \cup H \cup C_2)$ extends the second positive example. This illustrates how Context-Dependent Learning by Answer Sets allows for consistent hypotheses to be learned even in the presence of conflicting facts from different contexts. If both contexts were added directly to the background knowledge, the learning task would have no solution for the given examples.

2.3 Event Calculus

With the comprehension of stories involving agents and their actions, there needs to be some logic-based formalism to represent actions and effects of actions. Such formalisms include *event calculus* [14] and *situation calculus* [22] among others. Even among the formalisms of Event Calculus there exists multiple variants but for the purposes of this paper, we will look at a particular variant developed for the use in Inductive Logic Programming from [13].

For our use of Event Calculus in this paper, we introduce the following predicates.

initiatedAt(F,T)	terminatedAt(F,T)	holdsAt(F,T)	happensAt(E,T)

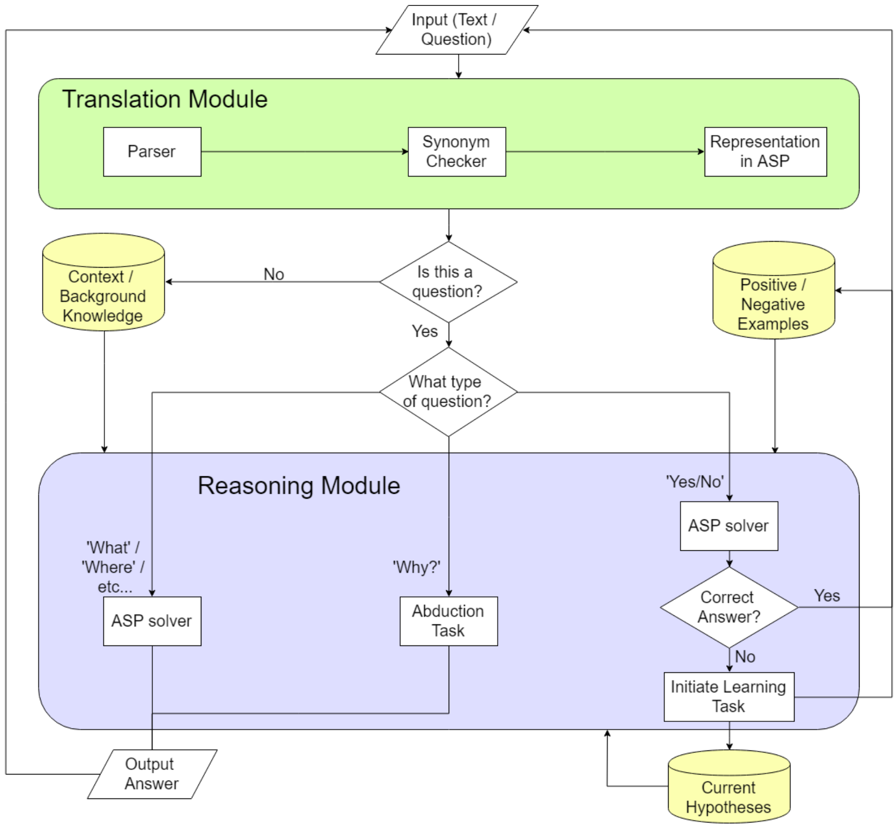

Figure 1 The general work pipeline of the system.

The variable F represents a fluent, E represents an event and T represents a time point. Along with these predicates we add the following axioms.

```
holdsAt(F,T+1) :- initiatedAt(F,T).
holdsAt(F,T+1) :- holdsAt(F,T), not terminatedAt(F,T).
```

These axioms basically mean that if a fluent is initiated at a time point, then the fluent will hold at the next time point, and if a fluent holds at a particular time point and is not terminated at that time, it will continue to hold.

3 Learning Commonsense Knowledge

In this section, we present our system for learning commonsense rules by interacting with a user through simple dialogue. The system starts with a very limited background knowledge that includes only the domain-independent axioms of Event Calculus given in the previous section. The user inputs a series of sentences and responses through the keyboard to the system in a simulated conversation. Through a mixture of story telling and question-answering, the system remembers facts about the narrative being told by the user and learns to form rules and relations that are consistent with the responses given by the user about the questions that have been asked.

An illustration of the overall structure and pipeline for the system can be seen in Figure 1. There are two main components to this system: the translation module and the reasoning module, which are described in detail in the following sections. The system was coded in

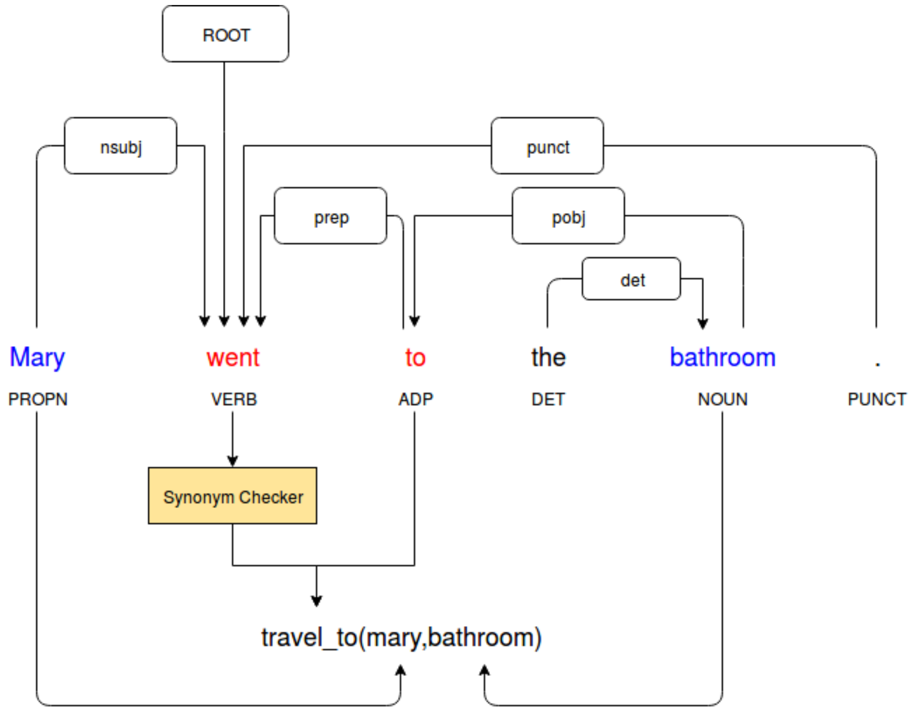

Figure 2 Illustration of how a sentence is generally translated into its logical representation.

python 2.7 using spaCy [11] as the syntactic parser, WordNet [23] for the synonym database, clingo5 [8] as the ASP solver, and ILASP [17] for the learning tasks.

3.1 Translation

Each sentence that passes through the translation module is parsed and its dependency tree is generated. Each word is tokenised and tagged with their part-of-speech (POS) tag, their dependency tag and their parent node in the tree.

After a sentence is parsed we need to generate the predicate symbol for the sentence. Firstly, the 'ROOT' of the sentence is taken and put through WordNet to find the closest general synonym. We then append any adpositions (ADP), adjectives (ADJ) or auxiliary verbs (aux / VERB) to the root verb to complete the predicate symbol. Then the sentence is scanned through for nouns (NOUN), pronouns (PRON) and proper nouns (PROPN). The lemmas of the n-number of nouns are then added to the predicate symbol to form an n-ary predicate. This is illustrated in Figure 2. For the rest of this paper we will refer to this translated predicate as the logical representation of the sentence.

To complete the conversion of the sentence into its representation in ASP for the system, the logical representation of the sentence is wrapped with the appropriate predicate of Event Calculus. The system has an internal counter that begins at '1' for each story and increments after each sentence/question that is input by the user. This counter is used to determine the time stamp for each predicate of Event Calculus while the logical representations form the events/fluents. With the exception of sentences that contain the root verb translating to 'be' and sentences that contain negation, all facts are wrapped with the 'happensAt' predicate. These sentences are all treated as events that happen at that particular time stamp. For sentences with the root verb 'be', these are wrapped with the 'holdsAt' predicate as they are concerned with the state something is in rather than a particular event.

Some additional rules for translation have been added to deal with sentences that include negation, conjunctions, disjunctions and coreferencing. These rules have been constructed to be general in nature however they are only applicable to sentences of relatively simple structures. If a negation modifier (neg) is found linked directly to the root verb, then the resulting logical representation is wrapped in the 'terminatedAt' predicate for Event Calculus. This stops the fluent from holding from that time point onwards, regardless of if it were true or not beforehand. For conjunctions and disjunctions, the noun that is tagged as the conjunct (conj) is stored and a predicate is formed whilst disregarding the conjunct. A second predicate is then formed by substituting in the conjunct in the place of the noun it is linked to. In the case of conjunctions (sentences involving 'and') the extra predicate is added on as an additional fact. In the case of disjunctions (sentences involving 'or') the two predicates are combined into an aggregate where only one may be chosen, thus creating an exclusive or.

For all inputs that are not questions, the sentences are translated, stored into the context of the story, then the system waits for the next input. When the translation module notices that the input sentence is a question, the reasoning module will be called.

3.2 Reasoning

The reasoning module deals with a variety of types of questions, however the most important type of question for this system are 'yes/no' questions. Questions that ask 'who', 'what', 'where' and 'why' will only generate a response that the system can give using its current knowledge, whereas asking 'yes/no' questions may initiate learning tasks depending on the user's feedback.

'Yes/No' Questions

All of the learning that happens in the system is through the interactions that result from 'yes/no' questions. All our reasoning tasks are run using clingo5 with the files created dynamically during the conversations. The system begins with only the axioms of Event Calculus as described from Section 2.3 except with an added predicate for time.

```
holdsAt(F,T+1) :- initiatesAt(F,T), time(T).
holdsAt(F,T+1) :- holdsAt(F,T), not terminatedAt(F,T), time(T).
```

The type predicate for time is used here to limit the relevant grounding of the reasoning tasks. Without specifying this type predicate, our ASP solver will continue to generate 'holdsAt' predicates indefinitely.

All the sentences that are not questions pass through the translation module and are stored directly into the *context* of the story without interacting with the reasoning module. When a 'yes/no' question is asked, the background knowldge axioms, the context up to that time and the current hypothesis is written into a '.lp' file along with the translated query as the goal. The ASP solver will try to satisfy the goal using the given context with the current hypothesis. If the result of the reasoning task comes back as 'UNSATISFIABLE' then the system will respond 'No' to the question; otherwise it will respond 'Yes' or 'Maybe' in some exceptions. The user will then tell the system if its conclusion was correct or not. After the user responds, the query is stored as a positive example if it is supposed to true, a negative example if it is supposed to be false, and if the system is told that it was incorrect a learning task will be initiated.

In the case where a disjunction was present in the context, a second reasoning task is run alongside the original except with the negation of the query as the goal. If the results of running both reasoning tasks returns 'SATISFIABLE' (i.e. the query can be considered both true or false with respect to the context), then the system responds with 'Maybe'. If the response of 'Maybe' is deemed correct by the user, then the query is not stored in the examples, however if 'Maybe' is incorrect, then the query is added to the negative examples. For example, if in the story we have "John went to either the hallway or the garden." then we ask "Is John in the hallway?", two reasoning tasks will be run trying to satisfy John being in the hallway and John not being in the hallway. In this case, both cases are satisfiable and therefore the system would respond with 'Maybe'.

Learning

When a learning task is triggered, the system will construct an $ILP_{LAS}^{context}$ task that contains the background knowledge, bias constraints, the context-dependent partial interpretations and the mode declarations. We use ILASP here to run the learning task from the generated file. The background knowledge added to the task is the axioms of Event Calculus as it is for the reasoning tasks. The context-dependent partial interpretations are generated by using the positive and negative examples that were stored from the 'yes/no' questions in the dialogue, along with the context of the story thus far.

The mode declarations are automatically generated for each learning task. To generate the mode declarations, for each logical representation that is seen throughout the story, the original nouns in the arguments are replaced with variable types. With the current implementation, variable types are given as inputs by the user; when the system encounters a noun that it has not yet seen, it will prompt the user to enter the noun's variable type. For each type of predicate that occurs in the context, a corresponding mode body declaration is created. For each type of logical representation that occurs as an example, additional mode head declarations are made using both 'initiatedAt' and 'terminatedAt' wrappers, along with a mode body declaration using the 'holdsAt' wrapper if it is not already present.

Additional bias constraints are used to help decrease the size of the hypothesis search space by restricting ILASP to only searching for hypotheses about a single time point rather than multiple. As all of the hypotheses that we are currently aiming to learn consist of rules that are each triggered by events that happen at one specific time point, this constraint does not negatively impact our system's ability to learn.

The $ILP_{LAS}^{context}$ task is solved with the maximum number of variables of possible hypotheses set to three. The resulting hypothesis is stored and added to subsequent reasoning tasks as the current hypothesis. The current hypothesis is continually overwritten with each learning task. If ILASP is unable to find a suitable hypothesis, the maximum number of variables is incremented and the task is run again. The system halts if it still fails to find an inductive solution with the maximum number of variables set to five.

Other Questions

With questions starting with 'What', 'Where' or 'Who', a slightly different reasoning task to the one generated for 'yes/no' questions is created. We create a file for the reasoning task with the background knowledge axioms, the context of the story and the current hypothesis, however we do not add in a goal to this ASP program. Rather than add the translation of the query as the goal, we generate a pattern from the translation and see what positive atoms match this pattern in the resulting answer set. From those atoms we extract the

Input	Description
end	ends the current session and exits the program
new story	starts a new story with empty context
save hypothesis	stores the current hypothesis into the Background Knowledge
check hypothesis	prints the stored and current hypothesis

Table 1 List of special inputs and their functions.

variables that have been matched and output them to the screen. For questions that start with 'How many', the same task as what has just been described is generated but rather than return the strings that result from pattern matching, it returns the number of items matched.

For questions that ask 'Why', a different type of reasoning task is generated. The background knowledge and learned hypothesis are written into the '.lp' file along with the translated query as the goal, and the facts from the context of the story are written into an aggregate. For each fact in the aggregate, a weight is applied so that when the ASP program is then run, only the minimum number of facts from the story will be chosen to make the goal 'SATISFIABLE'. This in essence is an abductive task where we look for the causal relationship between what events have happened and how the goal has been reached. This differs to the type of 'why' questions asked in Facebook bAbI's task 20 where it asks for the motivation of the agent in question, which is outside the scope of the story.

3.3 Special inputs

This system runs in a constant loop where it will keep waiting for the user's next input. Whenever a new sentence is expected by the system, there are a few specific inputs that are recognised by the system as different function calls. These special inputs are described in Table 1.

One thing to note is that when 'new story' is called, the context and examples up until that point all get stored as a context-dependent partial interpretation which is still used for subsequent learning tasks so that what has been learned previously is not forgotten, however the reasoning tasks will not be affected by facts from previous stories. By checking and saving the hypothesis, the user can also choose to keep rules that they consider to be desirable. The 'save hypothesis' function also clears all mode head declarations up to that point; this aids the systems scalability and helps when learning more difficult concepts.

4 Evaluation

The system was tested with various stories that draw inspiration from the themes of those seen in Facebook bAbI's question-answering data set [29]. Many of these stories are to do with a number of people moving to different locations and then asking about their whereabouts or about objects that they are or were carrying. By the nature of our system, our results are more qualitative than quantitative, and so it is easier for us to demonstrate the capabilities of the system via an example.

▶ **Example 5.** For this example, consider Listings 1, 2 and 3 (the Listings are found in the Appendix). This story is inspired by tasks 1, 2, 6, 7, 8, 11 and 12 of Facebook bAbI's QA dataset.

In Listing 1 it can be seen how initially when the system is asked about the location of ‘Mary’ from the story, it cannot answer correctly. It then forms a concept to initiate the state of Mary being in the location due to travelling there. However with what it had first learned, it did not understand that if Mary moved to a different location, she would no longer be in the previous location. This is then corrected after making its second mistake, which results in learning the correct hypothesis, which is displayed after the input ’check hypothesis’.

Following the story from Listing 1 we look at the dialogue from Listing 2. Here we can see that initially when asked about the items that Mary is carrying, the system responds incorrectly as it has not learned anything about the concept of ‘carrying’ yet. So then the concept of carrying is then taught to the system as a result of picking up or dropping objects through additional questioning and the system is then able to answer the questions correctly. By further interactions with the system through dialogue, the system is able to learn more interesting and complex concepts, such as what is displayed in Listing 3. Here the system is able to answer questions that require two supporting facts, this being equivalent to questions from task 2 of the bAbI dataset. More specifically, in Listing 3, it has learned that objects will be in the locations they are picked up in or will move to new locations with the person who is carrying them, and that they will no longer be in previous locations if the person carrying them has moved.

Using Facebook bAbI’s QA dataset as a means for comparison, the presented system is able to learn concepts that are able to deal with dialogues that are equivalent to eleven of the bAbI tasks. The tasks that can be solved are shown in Table 2. However due to some limitations in our system with translation, some of these tasks need to be slightly adapted for our system. For questions such as “Where is the apple?”, our system needs the question to be phrased using the same language as what was used to teach it. Since the concept determining the location of an object or person would have been translated as something ‘being in’ a location, we would need to change the question to “Where is the apple in?” for the system to be able to correctly understand and answer the question. Another translation limitation that has been found during various tests is that some names are not always recognised as proper nouns by the spaCy parser. Names such as Emily and Will are sometimes tagged as adverbs or verbs by the parser. To deal with this problem, it is sufficient to replace these names in the tasks with other less ambiguous names such as Emma and Brian. With these minor changes to the eleven bAbI tasks, after appropriate questioning and answering, this system is able to learn to solve them completely.

The number of questions required to learn each concept correctly is also difficult to quantify as it can vary greatly depending on the complexity of the concept, the context of the story, the types of questions you ask and the order in which you ask them. Some of these concepts can be learned in as few as two or three questions as can be seen from Listings 1 and 2, however the same concepts can also take much longer to learn if not questioned appropriately. To minimise the number of questions required to learn a concept, questions should be directed towards the gaps of any incomplete learned hypothesis. Since learning is only initiated when a ‘yes/no’ question is answered incorrectly, actively trying to increase the number of mistakes the system makes will allow the learning to progress much faster.

We have yet to implement a way for dialogue with the system to be automated and allow for quantitative analyses to be generated. With the current interactive approach, to do a quantitative analysis over a large dataset is impractical. Due to the size of these data sets, it is likely that many examples will be covered by the same hypotheses. Since ILASP2i [16] iteratively computes a subset of the examples which are relevant to the search, the size of

Task	Description	Examples of sentences that feature in the task
1	single supporting fact	Where is Mary? (see Listing 1)
2	two supporting facts	Where is the apple? (see Listing 3)
6	yes/no questions	Is Mary in the kitchen?
7	counting	How many objects is Mary carrying?
8	lists / sets	What is Mary carrying?
9	simple negation	Sandra is no longer in the bedroom.
10	indefinite knowledge	John is either in the office or the bathroom.
11	basic coreference	Then he moved to the hallway.
12	conjunctions	Mary and Sandra journeyed to the garden.
13	compound coreference	After that they went back to the kitchen.
15	basic deduction	What is gertrude afraid of? (see Listing 4)

Table 2 The Facebook bAbI question-answering tasks that are able to be solved by the system.

which generally being much smaller than the entire set of examples, we may be able to take advantage of this capability when automating the process to scale on entire bAbI datasets. This could potentially scale better compared to other batch learning systems, however this has yet to be tested.

Limitations

For the majority of the other tasks in the bAbI dataset, the reason why the tasks are unable to be completed is because of the limitations of the translation module. Some of these troubles are to do with inconsistent parsing of sentences with similar structure and some are to do with the challenges in representing sentences with multiple arguments. For instance, take the sentences "Fred gave John the football.", "Who gave the football to John?" and "Who did Fred give the football to?". By using the current method of translation, the logical representation of the first sentence would be 'give(Fred, John, football)', and the two questions would ideally translate to 'give(?, John, football)' and 'give(Fred, ?, football)'. However, rules that determine the order in which the multiple arguments are put into the predicate, and also the position of the missing argument that needs to be found, are very hard to generalise without programming it for a specific sentence structure. Currently the system is unable to translate these types of sentences well enough.

Another challenge this system faces is a problem with scalability. As stories get more involved and new types of predicates get introduced, the hypothesis space gets exponentially larger and can expand to the point where learning takes too long to be considered reasonable for dialogue. Although this problem arises from our choice of automating the generation of mode declarations, we believe that this does not outweigh the benefits we gain from being able to generalise with no prior knowledge of the contexts. An example of encountering scalability problems can be seen when trying to solve task 2 (two supporting facts) of the bAbI dataset. This task is currently solvable by using the 'save hypothesis' functionality that can be used to clear the old mode head declarations. By being a bit more strategic with the order in which you ask questions and help the system learn, concepts that otherwise would be too difficult for it to learn in one go can be broken down into manageable steps for incremental learning.

5 Related Work

To our knowledge, the problem of learning commonsense knowledge through a dialogic interaction is a novel task. However there has been significant research done on solving Facebook bAbI's question-answering tasks.

Mitra and Baral [24] developed a system to solve the toy tasks from Facebook's bAbI dataset. In their work, they describe an agent architecture that works with a formal reasoning model together with a statistical inference based model in tandem, to face the task of question answering. There are three layers to their implementation: the Statistical Inference Layer, the Formal Reasoning Layer and the Translation Layer. The Statistical Inference Layer contains the statistical NLP models which uses an Abstract Meaning Representation (AMR) Parser [1] [7]. The Formal Reasoning Layer uses a modified version of the ILP algorithm XHAIL [25] to learn the knowledge for reasoning in ASP. The Translation Layer encodes the sentences from the text into the syntax of Event Calculus with the help of the AMR Parser. This layer enables the communication between the two aforementioned layers and allows information to be passed from one to the other. Their system achieved a mean accuracy of 99.68% over the entire bAbI dataset and shows that with the addition of a formal reasoning layer, the reasoning capability of an agent increases significantly. With this approach, mode declarations had to be manually defined for each task, and some tasks had hypotheses learned from previous tasks added to the background knowledge of other more complex tasks. This differs to our approach as mode declarations are automatically defined during the dialogue and we do not augment our background knowledge.

An approach to the problem of machine comprehension of text has recently been developed by Chabierski [3]. The approach here utilises Combinatory Categorial Grammar [27] and Montague-style semantics [12] to perform a semantic analysis of text to derive Answer Set Program representations expressed in the form of λ-ASP calculus [2]. These representations are used to automatically derive the mode declarations for the generation of ILP tasks to be computed by the ILASP algorithm. To evaluate the performance of this approach, the system was tested using a subset of the bAbI question-answering tasks. Using only 25 training examples for each task, the system is able to fully solve six of the twenty QA tasks from the bAbI dataset, namely tasks 1, 6, 8, 9, 12 and 15. This approach also automatically derives its mode declarations from the task context and background knowledge, however the background knowledge is manually added for each task to increase the capability of the learner.

6 Conclusion and future work

What distinguishes our presented work from others is the interactive nature of our system and its ability to be able to learn from dialogues without having any prior knowledge about the contexts of their stories. More specifically, our system has the ability to automatically define the mode declarations for each task without the need to manually augment the system's background knowledge. This allows our system to be more flexible and able to understand contexts outside the scope of stories in the bAbI tasks. Although interactions with the system are limited to relatively simple sentences, these sentences are not confined to a preset type of contexts. Another strength of our system is that it is able to learn concepts with relatively few questions; it does not require large datasets for it to be trained on.

To expand the scope of what the system is able to understand and learn from, the translation module needs to be improved. It is very hard to define general rules that would allow for accurate translations of all the complexities in the English language, therefore it would probably be better to use statistical methods of machine learning to train the translation module instead. Many measures have already been taken to improve the scalability of the system but further improvements should be looked into as well. Allowing the user to define the variable types is one such measure taken to improve the scalability. This is not a very elegant implementation and ideally the system would be able to accurately define the variable types itself in a way which is not too specific to the point where the system is unable to generalise, yet not too general so that the scalability is affected. It may be possible to introduce some way of using ontologies or by using other relational databases such as ConceptNet [21] to map these variables to similar types.

A Demonstrations

The following listings here in the Appendix show examples of dialogue between a user and the system during a session. For clarity, the dialogues have been annotated with 'U:' for what the user has input and 'S:' to signify the system's responses.

References

1 Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract Meaning Representation for Sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL: `http://www.aclweb.org/anthology/W13-2322`.

2 Chita Baral, Juraj Dzifcak, and Tran Cao Son. Using answer set programming and lambda calculus to characterize natural language sentences with normatives and exceptions. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI'08, pages 818–823. AAAI Press, 2008. URL: `http://dl.acm.org/citation.cfm?id=1620163.1620199`.

3 Piotr Chabierski, Alessandra Russo, Mark Law, and Krysia Broda. Machine comprehension of text using combinatory categorial grammar and answer set programs. In *COMMONSENSE*, 2017.

4 Devendra Singh Chaplot and Ruslan Salakhutdinov. Knowledge-based word sense disambiguation using topic models. *CoRR*, abs/1801.01900, 2018. URL: `http://dblp.uni-trier.de/db/journals/corr/corr1801.html#abs-1801-01900`.

5 John Cook. Dialogue in Learning: Implications for the Design of Computer-based Educational Systems. In *ICCE*, pages 987–991, 2002. URL: `http://dblp.uni-trier.de/db/conf/icce/icce2002-2.html#Cook02`.

6 Ernest Davis and Gary Marcus. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Communications of the ACM*, 58(9):92–103, 9 2015. `doi:10.1145/2701413`.

7 Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. A Discriminative Graph-Based Parser for the Abstract Meaning Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL: `http://www.aclweb.org/anthology/P14-1134`.

8 M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub. *Clingo* = ASP + control: Preliminary report. In M. Leuschel and T. Schrijvers, editors, *Technical Communications of the Thirtieth International Conference on Logic Programming (ICLP'14)*, volume arXiv:1405.3694v1, 2014. Theory and Practice of Logic Programming, Online Supplement.

9 M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In *ICLP '88*, pages 1070–1080, Cambridge, Massachusetts, 1988. MIT Press.

10 Julia Hirschberg and Christopher D. Manning. Advances in natural language processing. *Science*, 349(6245):261–266, 2015. URL: `http://science.sciencemag.org/content/349/6245/261`, `arXiv:http://science.sciencemag.org/content/349/6245/261.full.pdf`, `doi:10.1126/science.aaa8685`.

11 Matthew Honnibal and Mark Johnson. An Improved Non-monotonic Transition System for Dependency Parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL: `https://aclweb.org/anthology/D/D15/D15-1162`.

12 Theo M. V. Janssen. Montague semantics. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2017 edition, 2017.

13 Nikos Katzouris, Alexander Artikis, and Georgios Paliouras. Incremental learning of event definitions with Inductive Logic Programming. *Machine Learning*, 100(2-3):555–585, 2015. URL: `http://dblp.uni-trier.de/db/journals/ml/ml100.html#KatzourisAP15`.

14 Robert Kowalski and Marek Sergot. A Logic-Based Calculus of Events. *New Generation Computing*, 4:67–95, 1986.

15 Mark Law, Alessandra Russo, and Krysia Broda. Inductive Learning of Answer Set Programs. In Eduardo Fermé and João Leite, editors, *JELIA*, volume 8761 of *Lecture Notes in Computer Science*, pages 311–325. Springer, 2014. URL: `http://dblp.uni-trier.de/db/conf/jelia/jelia2014.html#LawRB14`.

16 Mark Law, Alessandra Russo, and Krysia Broda. Learning weak constraints in answer set programming. *TPLP*, 15(4-5):511–525, 2015. URL: `http://dblp.uni-trier.de/db/journals/tplp/tplp15.html#LawRB15`.

17 Mark Law, Alessandra Russo, and Krysia Broda. Iterative Learning of Answer Set Programs from Context Dependent Examples. *TPLP*, 16(5-6):834–848, 2016. URL: `http://dblp.uni-trier.de/db/journals/tplp/tplp16.html#LawRB16`.

18 Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. Commonsense knowledge base completion. In *ACL (1)*. The Association for Computer Linguistics, 2016. URL: `http://dblp.uni-trier.de/db/conf/acl/acl2016-1.html#LiTTG16`.

19 Xiang Li, Luke Vilnis, and Andrew McCallum. Improved representation learning for predicting commonsense ontologies. *CoRR*, abs/1708.00549, 2017. URL: `http://dblp.uni-trier.de/db/journals/corr/corr1708.html#abs-1708-00549`.

20 Vladimir Lifschitz. What is answer set programming? In *23rd AAAI Press Conference on Artificial Intelligence (AAAI 2008)*, pages 1594–1597, Menlo Park, CA, USA, 2008. AAAI Press. URL: `http://www.cs.utexas.edu/~{}vl/papers/wiasp.pdf`.

21 H. Liu and P. Singh. Conceptnet — a practical commonsense reasoning tool-kit. *BT Technology Journal*, 22(4):211–226, October 2004. URL: `http://dx.doi.org/10.1023/B:BTTJ.0000047600.45421.6d`, `doi:10.1023/B:BTTJ.0000047600.45421.6d`.

22 J. McCarthy and P. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence*, 4:463–502, 1969.

23 George Miller. WordNet A Lexical Database for English. *Communications of ACM*, 38(11):39–41, 1995.

24 Arindam Mitra and Chitta Baral. Addressing a Question Answering Challenge by Combining Statistical Methods with Inductive Rule Learning and Reasoning. In Dale Schuurmans and Michael P. Wellman, editors, *AAAI*, pages 2779–2785. AAAI Press, 2016. URL: `http://dblp.uni-trier.de/db/conf/aaai/aaai2016.html#MitraB16`.

25 Oliver Ray. Nonmonotonic abductive inductive learning. *Journal of Applied Logic*, 7(3), 9 2009. Other identifier: 2001069. `doi:10.1016/j.jal.2008.10.007`.

26 D. Singh Chaplot and R. Salakhutdinov. Knowledge-based Word Sense Disambiguation using Topic Models. *ArXiv e-prints*, January 2018. `arXiv:1801.01900`.

27 Mark Steedman and Joakim Nivre. The syntactic process. 01 2000.

28 Ramakrishna Vedantam, Xiao Lin, Tanmay Batra, C. Lawrence Zitnick, and Devi Parikh. Learning common sense through visual abstraction. In *ICCV*, pages 2542–2550. IEEE Computer Society, 2015. URL: `http://dblp.uni-trier.de/db/conf/iccv/iccv2015.html#VedantamLBZP15`.

29 Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. *CoRR*, abs/1502.05698, 2015. URL: `http://dblp.uni-trier.de/db/journals/corr/corr1502.html#WestonBCM15`.

```
System: Tell me a story:
User: Mary went to the bathroom.
System: What is "mary" a type of? User: person
S: What is "bathroom" a type of? U: location

U: Is Mary in the bathroom?
S: No.
S: Am I right? U: No.
S: Alright, please continue.

U: Mary moved to the kitchen.
S: What is "kitchen" a type of? U: location

U: John journeyed to the kitchen.
S: What is "john" a type of? U: person

U: Is Mary in the kitchen?
S: Yes.
S: Am I right? U: Yes.
S: Alright, please continue.

U: Is Mary in the bathroom?
S: Yes.
S: Am I right? U: No.
S: Alright, please continue.

U: check hypothesis

S: initiatedAt(be_in(V0,V1),V2) :- happensAt(travel_to(V0,V1),V2).
S: terminatedAt(be_in(V0,V1),V2) :- holdsAt(be_in(V0,V1),V2),
        happensAt(travel_to(V0,V3),V2).

S: Please continue with the story:
U: Mary and John travelled to the garden.
S: What is "garden" a type of? U: location

U: Where is John in?
S: garden

U: Who is in the garden?
S: john
S: mary
```

1 An example of a dialogue between the system and a user showing how the concept of people being or not being in locations as a result of travelling is formed.

```
User: Mary picked up the apple and the milk.
System: What is "apple" a type of? U: object
S: What is "milk" a type of? U: object

U: How many objects is Mary carrying?
S: 0

U: Is Mary carrying the apple?
S: No.
S: Am I right? U: No.
S: Alright, please continue.

U: How many objects is Mary carrying?
S: 2

U: What is Mary carrying?
S: milk
S: apple

U: Mary dropped the milk.

U: Is Mary carrying the milk?
S: Yes.
S: Am I right? U: No.
S: Alright, please continue.

U: What is Mary carrying?
S: apple

U: check hypothesis

S: initiatedAt(be_in(V0,V1),V2) :- happensAt(travel_to(V0,V1),V2).
S: terminatedAt(be_in(V0,V1),V2) :- holdsAt(be_in(V0,V1),V2),
        happensAt(travel_to(V0,V3),V2).

S: initiatedAt(be_transport(V0,V1),V2) :- happensAt(pick(V0,V1),V2).
S: terminatedAt(be_transport(V0,V1),V2) :- happensAt(drop(V0,V1),V2).

S: Please continue with the story:
```

2 A continuation of the dialogue from Listing 1, this time showing the concept of carrying objects being learned.

```
S: initiatedAt(be_in(V0,V1),V2) :- happensAt(travel_to(V0,V1),V2).
S: terminatedAt(be_in(V0,V1),V2) :- holdsAt(be_in(V0,V1),V2),
     happensAt(travel_to(V0,V3),V2).

S: initiatedAt(be_transport(V0,V1),V2) :- happensAt(pick(V0,V1),V2).
S: terminatedAt(be_transport(V0,V1),V2) :- happensAt(drop(V0,V1),V2).

S: initiatedAt(be_in(V1,V3),V2) :- holdsAt(be_transport(V0,V1),V2),
     happensAt(travel_to(V0,V3),V2).
S: initiatedAt(be_in(V1,V3),V2) :- happensAt(pick(V0,V1),V2),
     holdsAt(be_in(V0,V3),V2).
S: terminatedAt(be_in(V1,V3),V2) :- holdsAt(be_transport(V0,V1),V2),
     holdsAt(be_in(V0,V3),V2), happensAt(travel_to(V0,V4),V2).

S: Please continue with the story:
U: new story
S: Tell me a new story:
U: Mary went to the kitchen.

U: She picked up the apple and the milk.

U: John moved to the hallway.

U: He picked up the football.

U: What is in the kitchen?
S: milk
S: mary
S: apple

U: Mary dropped the apple.

U: Then she travelled to the garden.

U: Where is the apple in?
S: kitchen

U: Where is the milk in?
S: garden

U: Where is the football in?
S: hallway
```

3 A demonstration of further concepts that can be learned through more questioning and answering (equivalent to task 2 - two-supporting facts from the bAbI dataset).

```
System: Tell me a story:
User: Tom is a cat.
S: What is "tom" a type of? U: name
S: What is "cat" a type of? U: animal

U: Jerry is a mouse.
S: What is "jerry" a type of? U: name
S: What is "mouse" a type of? U: animal

U: Mice are afraid of cats.

U: Is Jerry afraid of cats?
S: No.
S: Am I right? U: No.
S: Alright, please continue.

U: Is Jerry afraid of Tom?
S: No.
S: Am I right? U: No.
S: Alright, please continue.

U: Is Tom afraid of mice?
S: No.
S: Am I right? U: Yes.
S: Alright, please continue.

U: Is Tom afraid of Jerry?
S: No.
S: Am I right? U: Yes.
S: Alright, please continue.

U: check hypothesis

S: initiatedAt(be_afraid_of(V3,V0),V2) :- holdsAt(be(V0,V1),V2),
        holdsAt(be_afraid_of(V3,V1),V2).
S: initiatedAt(be_afraid_of(V0,V3),V2) :- holdsAt(be(V0,V1),V2),
        holdsAt(be_afraid_of(V1,V3),V2).
```

4 Running example of a dialogue inspired by bAbI task 15 that learns an extended concept.

```
System: Please continue with the story:
User: Gertrude is a wolf.
S: What is "gertrude" a type of? U: name
S: What is "wolf" a type of? U: animal

U: Cats are afraid of wolves.

U: Is Tom afraid of wolves?
S: Yes.
S: Am I right? U: Yes.
S: Alright, please continue.

U: Is Tom afraid of Gertrude?
S: Yes.
S: Am I right? U: Yes.
S: Alright, please continue.

U: What is Gertrude afraid of?
S: Nothing

U: Mary is a sheep.
S: What is "mary" a type of? U: name
S: What is "sheep" a type of? U: animal

U: Sheep are afraid of mice.

U: Sheep are afraid of wolves.

U: What is Mary afraid of?
S: wolf
S: mouse
S: jerry
S: gertrude
```

5 Continuation of the dialogue seen from Listing 4 that tests the learned hypothesis.