tExplain: Information Extraction with Explanations

Pedro Cabalar, Adrian Dorsey, Jorge Fandinno, Yuliya Lierler,
Brais Muñiz, and Joel Sare

University of A Coruna, University of Nebraska Omaha
{cabalar,brais.mcastro}@udc.es
{jfandinno, ylierler,joelsare}@unomaha.edu
dorsey.adrian.adrian@gmail.com

Abstract. We present a narrative understanding tool that takes as an input a narrative in natural English language that contains both sentences describing actions as well as questions and outputs answers to those questions together with natural language explanations justifying those answers. We evaluate our tool on the several tasks of Facebook's bAbI challenge where it achieves 100% accuracy.

1 Introduction

In this work, we design a narrative understanding tool named tExplain. This tool takes as an input a narrative in natural English language that contains both sentences describing actions as well as questions, regarding the entities described in the narrative. Consider the following narrative as an example:

```
1 Daniel went to the bedroom.
2 Daniel picked up the apple there.
3 Mary went to the bathroom.
4 John moved to the hallway.
5 Mary travelled to the office.
6 Daniel put down the apple there.
7 Where is the apple?
```

The output of tExplain are the answers to the questions contained in the narrative together with an explanation about the reasoning the system followed to achieve that conclusion. Figure 1 illustrates the output obtained from tExplain given the above narrative that asks where the location of the apple is. This answer states that the apple is located at the bedroom and shows two different reasons how that can be deduced. The first, by noting that Daniel moved to the bedroom and, then, picked the apple. Since Daniel did not move afterwards, the apple is still in the bedroom where it was picked. The second explanation focus on the fact that Daniel dropped the apple while he was in the bedroom.

It is due to remark that the sample narrative provided above stems from the bAbI tasks published by Facebook Research in 2015 [15]. Conglomeration of

```
Answer 1
  *
  |__apple is located at bedroom
  |  |__daniel moved to bedroom in sentence 1
  |  |__since daniel obtained the apple in sentence 2

  *
  |__apple is located at bedroom
  |  |__daniel moved to bedroom in sentence 1
  |  |__since daniel dropped the apple in sentence 6
```

Fig. 1. Output of tExplain stating that the apple is in the bedroom and the two alternative explanations that justify this conclusion.

such narratives together with valid answers and explanations in the form of the numbers of related sentences in deriving an answer forms this synthetic dataset.

In building tExplain, we leverage two different existing tools such as information extraction system text2alm [12] and answer set solver xclingo [4], which enhances clingo [8] with explanations capabilities. Both text2alm and xclingo are built on the backbone of Answer Set Programming [7] or ASP — a knowledge representation declarative programming paradigm with roots in deductive databases, logic programming and satisfiability testing. The system text2alm produces an ASP program from natural language narratives whose solutions, called answer sets, can be used to answer several queries about the narrative. xclingo [4] is an ASP based tool that constructs explanations for found answer sets based on the rules occurring in a program. It also allows users to annotate their programs with snippets of text to produce natural language explanations. System tExplain, developed in this work, combines xclingo and text2alm to provide natural language answers and explanations for questions about simple English narratives focused on action verbs.

The natural language understanding capabilities of the tool target the bAbI benchmark Tasks 2, 3, and 6. Tasks 2 and 3 involve describing where an entity is located and using two or three supporting facts to achieve those conclusions, respectively. Task 6 involves answering yes/no questions about the locations of entities throughout the narrative. As part of the evaluation efforts, system tExplain is benchmarked on these tasks where it achieved 100% accuracy.

The rest of the paper is organized as follows. We start by reviewing the building blocks of tExplain, namely, text2alm and xclingo (Section2). Then, describe the new system (Section 3) and we report on the experimental evaluation on the bAbI challenge (Section 4). Finally, Section 5 concludes the paper.

2 Background

This section reviews the two building blocks of tExplain, namely, ASP based systems text2alm and xclingo.

2.1 Information extraction tool text2alm

System text2alm provides the basis for this work. It is an information extraction tool capable of narrative understanding with a focus on action verbs [12]. This tool uses an action language ALM [10] to perform inferences on complex interactions of events described in narratives. An ALM model is then translated into a logic program under answer set semantics [7] and a tool called answer set solver, specifically, clingo [8] is used to obtain solutions to this program. The text2alm system relied on a conglomeration of resources and techniques from two distinct fields of artificial intelligence (i) knowledge representation and reasoning and (ii) natural language processing. In a nutshell, text2alm takes a narrative in English and converts the narrative into a logic program under answer set semantics. The process of conversion involves "invocation" of the relevant background knowledge that is triggered by the action verbs utilized within the considered narrative. The knowledge is obtained from the knowledge base called CoreAlmLib [9] and is also converted into the form of a logic program. The CoreAlmLib is an ALM library of generic commonsense knowledge for modeling dynamic domains developed by Inclezan [9]. The library's foundation is the Component Library or CLib [6], which is a collection of general, reusable, and interrelated components of knowledge[1]. A logic program representing a given narrative together with related background knowledge is then processed with the state-of-the-art answer set solver clingo, which finds an answer set for the program that represents entities and events occurring within the narrative as well as encodes the state of affairs at various points of the narrative timeline.

We now provide a brief overview of text2alm by considering a sample input and resulting output from the system. The same sample input will be used in the sequel presentation of tExplain. As mentioned, text2alm takes narratives in natural language as input. Here is a shortened narrative from Section 1 which is referred to as *narrative 1*:

```
Mary went to the bathroom.
John moved to the hallway.
Mary travelled to the office.
```

Given this narrative, text2alm produces facts about this text in a structured form that machines can interpret. Figure 2 shows a subset of the text2alm output that is grouped together for readability by related predicates. The first group tells us basic information about the three events that happened within the narrative, namely, `e1`, `e2`, and `e3`. For example, the three events in this narrative are instances of action `move`. The text2alm tool relies on the verbnet [11, 13] ontology to identify different verbs occurring in given texts with specific actions. Note how different verbs occurring in our sample narrative, i.e., *went*, *moved*, *travel*, are mapped by text2alm into the same action `move`. The `event_agent` and `event_destination` facts of Group 2 provide us with the details of the participants of events `e1`, `e2`, and `e3`. For example, `event_agent(e1, mary)`

[1] CLib was populated with knowledge stemming from linguistic and ontological resources, such as verbnet [2], wordnet [3], framenet [1], and English dictionaries.

```
Group 1:
    instance(e1, move),
    instance(e2, move),
    instance(e3, move)
Group 2:
    event_agent(e1, mary),
    event_agent(e2, john),
    event_agent(e3, mary)
    event_destination(e1, bathroom),
    event_destination(e2, hallway),
    event_destination(e3, office)
Group 3:
    occurs(e1, 0),
    occurs(e2, 1),
    occurs(e3, 2)
Group 4:
    location(mary, bathroom, 1),
    location(mary, bathroom, 2),
    location(mary, office, 3),
    location(john, hallway, 2),
    location(john, hallway, 3)
```

Fig. 2. A subset of the facts produced by TEXT2ALM.

and `event_destination(e1, bathroom)` tell us that `mary` is the acting entity of the `move` action `e1` and bathroom is the destination of `mary` in the scope of `e1`. Facts, for example `occurs(e1,0)`, contain information pertaining to the time point associated with event `e1`. Within the TEXT2ALM framework, each sentence is associated with a time point value, where `0` denotes the time point for the first sentence within the narrative. For every sentence following the first, the time point is incremented by one. The facts within Group 4 tell us the location of various entities within the narrative for time points `1`, `2`, and `3`. Let us look at the first fact within Group 4: `location(mary, bathroom, 1)`. This fact contains information that Mary is located in the `bathroom` at time point `1`. This information is associated with the completion of the first sentence within the narrative. There are more conclusions than what is shown in these four groups as TEXT2ALM encodes more information for the full answer set given this three-sentence narrative. For example, looking at Group 1, one can assume that the events `e1`, `e2`, and `e3` relate to the idea of "movement".

There is a noticable distinction between the facts in Groups 1-3 and Group 4 pertaining to their nature. The facts in Groups 1-3 stem directly from text. TEXT2ALM uses various natural language processing tools to retrieve information and generate these facts directly. However, for Group 4, TEXT2ALM uses knowledge contained in the CoreALM Library. This library contains axioms for multiple actions which give TEXT2ALM commonsense knowledge about behavior. For our input narrative, axioms for the action `move` are retrieved and used. Then, TEXT2ALM uses facts within Groups 1-3, common sense knowledge, and the law of inertia to obtain information that is implicitly expressed within a narrative to generate the facts occurring in Group 4. This information allows us to use TEXT2ALM to answer questions like: *"where was John at the end of the story"*? The answer to this is the *hallway* because John explicitly moved to the hallway in sentence two and there's no information to show that he moved somewhere else after sentence three. We can obtain this answer from TEXT2ALM by looking at fact `location(john, hallway, 3)` in Group 4 of its output.

Figure 3 shows the building blocks for TEXT2ALM. The Text2LP block is responsible for parsing natural language text, translating information carried

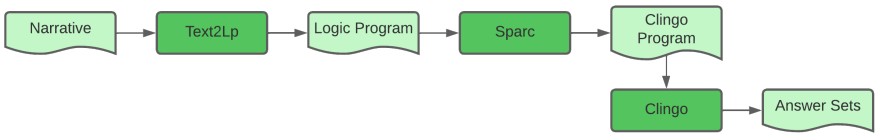

Fig. 3. System text2alm

by the text into logic program form and expanding it with background knowledge also in the form of a logic program. The Text2LP block translates a given narrative into a logic program in the language of answer set solver `Sparc` [5]. The answer set solver Sparc translates this program into the format of answer set solver clingo [8] and invokes clingo to compute answer sets of a given program. The groups of facts presented within our running example stem from computed answer sets.

2.2 "Explainable" answer set solver xclingo

System xclingo is a tool in answer set programming that allows the user to create annotations for logic programs so the solved program is shown together along with explanations in natural language. Without these annotations, xclingo solves and outputs just like the answer set solver clingo [8]. We illustrate xclingo annotations with an example. We consider an ASP program that is inspired by the first line within *narrative 1*: `Mary went to the bathroom.`

```
instance(e1,move).      event_agent(e1,mary).
occurs(e1,0).           event_destination(e1, bathroom).

location(O,D,T) :- occurs(E,T-1), instance(E,move),
                   event_agent(E,O), event_destination(E,D).
```

The answer set of this program follows:

```
instance(e1,move)              event_agent(e1,mary)
occurs(e1,0)                   event_destination(e1,bathroom)
location(mary,bathroom,1)
```

To transform this output into natural language statements, we may add xclingo annotations. For this example, we focus on the predicate `location/3`:

```
%!trace {"% is located at % at time %",O,L,T} location (O,L,T).
%!show_trace location(O,L,T).
```

The `%!trace` annotation creates a label for the atom `location/3`. Whenever an atom that has a label is shown in the answer set of a program, an explanation is triggered. In our running example, a label has the form of the tuple

```
{"% is located at % at time %",O,L,T},
```

the first element within this tuple contains a string that serves as the output phrase. The various `%` symbols occurring within this string serve as placeholders for the remaining values in the tuple. For example, the first `%` symbol will get replaced with whatever value variable `O` has. Atoms of the form `location(O,D,T)` are associated with this `%!trace` annotation within our example.

Only atoms with a `%!show_trace` annotation are displayed as output. Below is the output by running XCLINGO on our ASP program with the above `%!trace` and `%!show_trace` annotations:

```
Answer: 1
>> location(mary,bathroom,1)
  *
  |__"mary is located at bathroom at time 1"
```

Another example was given in Figure 1.

3 System tExplain and what goes into annotations

In this work, we propose to utilize answer set solver XCLINGO to augment system TEXT2ALM capabilities with explanations. Figure 4 presents an enhanced version of TEXT2ALM titled tExplain that

- introduces annotations for logic programs produced within TEXT2ALM, namely the XCLINGO Trace File which then extend a logic program produced by original TEXT2ALM components,
- turns a question associated with the end point of the narrative into a query supported by XCLINGO annotations, and
- utilizes XCLINGO in place of CLINGO

so that the explanations on relations of interest are provided to the user.

Logic programs produced by the `Text2LP`/`Sparc` block of the TEXT2ALM system are complex and therefore some considerations are required in order to create annotations. In this section, we show XCLINGO annotations and some additional rules used in this to accomplish our goal. Both trace rules and new predicates are introduced within the XCLINGO Trace File. These predicates play

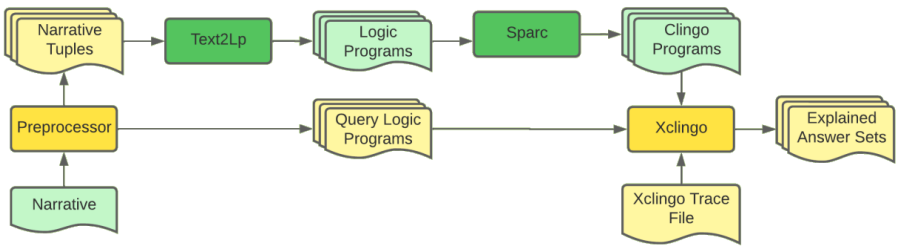

Fig. 4. tExplain in a Nutshell

two roles. First, some of these predicates may streamline the form of explanations that we obtain. Second, some of these predicates serve a role of "query-predicates" so that questions about a considered narrative can be formulated by means of the specific instances of these predicates and the help of `%!show_trace` directive of the xclingo.

Consider the following rule and annotation in the xclingo Trace File:

```
locationT2(Oname,Lname,T) :- location(Oref,L,T),
                             is_a(Oref,Oname),
                             is_a(Lref Lname).

%!trace {"% is located at % in sentence %", O, L, T}
         locationT2(O, L, T).
```

The provided trace rule encodes an explanation for the location of an entity at a specific sentence. Logic programs produced by the Text2LP block of text2alm use location argument referents such as `r1` or `r2` instead of names such as "ball" or "mary". These referents and names are linked together through the relation `is_a(_,_)` elsewhere in the program. The first argument in relation `is_a(_,_)` is for the referent and the second is for the English name. Therefore, we use the auxiliary `is_a(Oref, Oname)` together to instruct the trace to display "ball" and "mary" instead of `r1` and `r2`. Similarly, `is_a(Lref, Lname)` is used for locations.

Note how the trace rule is instructed to produce explanations by tracing newly defined predicate `locationT2` in place of `location`. Besides replacing the identifier by its corresponding name, this is essential because predicate `locationT2` is never used within the program produced by text2LP and hence there may not be "nested" explanations due to this predicate. We use `locationT2` and not `location` for explanations because various logic rules that encode background knowledge rely on `location/3` in many ways. Because of this, `location/3`-atoms get used as explanations according to other `location/3`-atoms. This means the xclingo system would perform a chain of explanations recursively for every `location/3`-atom. This level of detail is hard to follow and would not be considered natural to human reasoning. Introducing `locationT2/3` allows us to avoid these undesired explanations. It is worth recalling that the rules for `location/3` are automatically produced as output of `Sparc`, something we cannot change. This shows that xclingo can be used to add explanations to existing answer set programs with a great deal of elaboration tolerance: only new rules and xclingo directives are added and no modification to the code produced by `Sparc` was necessary.

Here we show an example of the use `%!show_trace`, which selects atoms with predicate symbol `locationT2` as the ones to be be explained:

```
%!show_trace locationT2(O,D,T).
```

In addition to `%!show_trace` statements, the xclingo Trace File includes `%!trace` statements to produce human-readable subexplanations. For instance, consider the following group of rules from this xclingo Trace File:

```
top_concept(move).   top_concept(obtain).  top_concept(relinquish).
```

```
link_r(X,X') :- link(X,X').
link_r(X,X') :- link_r(X,X1), link(X1,X').
event_top_concept(E,C) :- link_r(E,C), top_concept(C).

%!trace {"Since % %ed the % in sentence %",X0', A, B', X2 + 1}
    happened(X1,X2) :
    is_a(X1, X1'), event_recipient(X1, X0), event_object(X1, B),
    is_a(X0, X0'), event_top_concept(X1', A), is_a(B, B').
```

This `%!trace` statements produces a subexplanation for location whenever the atom `happended/2` was used to derive the atom being explained. We define three `top_concept`s: 'move', 'obtain', and 'relinquish'. Relation `link_r/2` and `event_top_concept/2` are also defined, with the latter itself being defined in terms of `top_concept` and `link`. The rationale behind these helper rules is that in the answer set program produced by TEXT2ALM before processed by XCLINGO, the immediate events of, for example, an entity picking up an object, are defined using VERBNET terms. In the case of an entity picking up an object, the VERBNET term would be 'get_13_5_1_1'. However, the VERBNET term for grabbing eventually links to the overarching verb 'obtain.' The Text2ALM answer set program contains the notion of `link/2` which we leverage in our `link_r/2` rule in order to propagate the VERBNET term for grabbing up to our `top_concept/1` of 'obtain.' The same applies to the event of an entity moving to a location. The VERBNET term associated with moving gets propagated up to our `top_concept/1` of 'move'. Verbs associated with entities dropping objects are propagated up to our `top_concept/1` of 'relinquish.' This propagation from VERBNET terms to `top_concept/1` allows us to accomplish our goal of generating output in natural language.

The relation `happened/2` encodes an event with a time point. The trace rule includes several `is_a/2` relations so that the explanation can display English words instead of abstractions such as `r1`. Additionally, it utilizes the relations `event_object/2` and `even_recipient/2` in order to associate the entity that participated in the event `happened/2` with the recipient of the action of the event in `happened/2`. The `event_top_concept/2` relation is defined above the discussed trace rule and is part of XCLINGO Trace File. In total, there are three trace rules in the XCLINGO trace file that help justify the location of entities.

The `%!mute` directive of XCLINGO marks atoms of some form as untraceable. This means the generated explanations will not include/follow them. In the above example, all atoms of the form `dom_location(X,Y,Z)` will not appear in XCLINGO output. In our XCLINGO trace file, there are 86 `%!mute` trace rules.

4 Evaluation and enhancements of TEXPLAIN

We evaluate TEXPLAIN on Facebook's bAbI dataset. In particular, we chose bAbI Tasks 1 (single supporting fact), 2 (two supporting facts), 3 (three supporting facts), and 6 (Yes/No questions). Each task splits its narratives into training and testing sets so that we can follow the best practice of the machine learning

community and focus on the extensions of tExplain utilizing examples from a training set of bAbI, while the evaluation data that we present stems from testing portion of the dataset. The difference though was that it was a knowledge engineer who inspected the bAbI training dataset tasks to identify whether and how tExplain can be equipped to tackle it. This is achieved by adding annotations that instruct a system what kind of information and explanations a user is interested in as described in Section 3. For each of these tasks, we

1. annotate the relations of interest to produce English sentence that read as a valid answer to a bAbI question that accompanies a narrative
2. annotate the relations of interest to produce explanations for derived answers
3. turn a question posed by bAbI into a statement within annotation language of xclingo so that tExplain is capable of taking as an input a narrative-question pair and produce an English sentence as an answer accompanied with the explanations.

4.1 bAbI dataset

The bAbI dataset [15] by Facebook AI Research was designed to be a baseline benchmark for automated text understanding and reasoning. The bAbI dataset consists of several tasks for systems to experiment on, ranging from yes/no questions, basic deduction, and path finding. In these tasks, there are entities of various types such as *locations, objects, people*, etc. and various actions can operate upon these entities. These entities also have internal states such as location, whether they have objects on top of or in them, the mental state of actions, and properties such as size, color, and entity. Figure 5 provides a snippet from one of the tasks of bAbI, namely, Task 2 called "two supporting facts". Note how bAbI narratives are simple stories, where various entities perform actions. These narratives also contain questions prompted at various time points. The bAbI dataset is of especial interest in this project as many of its tasks specifically target reasoning of action verbs that original text2alm system was geared for.

Now let us examine bAbI questions and the numeric annotations that come with them. Consider Line 7 in Figure 5. In this line, bAbI suggests that the answer to *where the apple is located* is in the *bedroom*. The two numbers appearing after word “bedroom” are the two sentences that support this answer. Consequently, we can read this line as saying that

> Upon the completion of the narrative composed of sentences in lines 1 through 6, *apple* is located in *bedroom* because we have been told that *Daniel went to the bedroom* in sentence one and then *he put down the apple* in sentence six.

In the design of tExplain we were targeting to create a system that is capable of the output in style presented above. It is easy to see how sample entry in the bAbI task presented in Figure 5 lends itself into what we will call the bAbINQAE tuple. Indeed, we have (1) a narrative given to us composed of 6 sentences; (2) a question *Where is the apple?*; (3) a word *bedroom* that encodes an answer; and

```
 1 Daniel went to the bedroom.       4 Mary left the milk.
 2 Daniel picked up the apple there. 5 John journeyed to the office.
 3 Mary grabbed the milk there.      6 Daniel put down the apple there.
                                     7 Where is the apple? bedroom 6 1
```

Fig. 5. Example entry from the bAbI task called two supporting facts

```
1 Daniel went to the bedroom.              1 person1 <move_to> loc1.
2 Daniel picked up the apple there.        2 person1 <grab>  obj1.
3 Mary grabbed the milk there.             3 person2 <grab>  obj2.
4 Mary left the milk.                      4 person2 <drop> obj2.
5 John journeyed to the office.            5 person3 <move_to> loc2.
6 Daniel put down the apple there.         6 person1 <drop> object1.
7 Where is the apple? bedroom 6 1          7 Where is obj1? loc1 6 1
8 John picked up the milk there.           8 person3 <grab>  obj2
9 Sandra got the football there.           9 person4 <grab>  obj3.
10 Where is the apple? bedroom 6 1         10 Where is obj1? loc1 6 1
11 Daniel journeyed to the hallway.        11 person1 <move_to> loc2.
12 John left the milk.                     12 person3 <drop> obj2.
13 Where is the milk? office 12 5          13 Where is obj2? loc2 12 5
14 Sandra travelled to the office.         14 person4 <move_to> loc2.
15 Sandra put down the football there.     15 person4 <drop> obj3.
16 Where is the football? office 15 14     16 Where is obj3? loc2 15 14
17 Sandra grabbed the milk there.          17 person4 <grab>  obj2.
18 John grabbed the football there.        18 person3 <grab>  obj3.
19 Sandra moved to the bathroom.           19 person4 <move_to> loc3.
20 John went to the bedroom.               20 person3 <move_to> loc1.
21 Where is the football? bedroom 18 20    21 Where is obj3? loc1 18 20.
```

Fig. 6. Translating bAbI entry within two supporting facts into pattern format

(4) two identifiers of the sentences that can be seen as explanations for derived answer.

4.2 Evaluation observations on Task 2

We now proceed to the kind of analysis we performed on Tasks 2, 3, and 6 of the bAbI dataset. This paper is documenting the analysis of Task 2. Other tasks are similar and are left out due to space reasons. We refer to Joel's Sare MS thesis [14] for a detailed analysis.

bAbI Patterns. In the 20 narratives listed in the training datases there were a total of 100 narrative-question-answer-explanation (NQAE) tuples. Clear patterns emerge within the bAbINQAE tuples. This is due to the nature of the bAbI dataset as it has been created shadowing the simulation environment and transcribing its history [15]. We illustrate the concept of a pattern on an example of Task 2 consisting of 16 sentences and 5 questions. Figure 6 contains this sample

bAbI narrative in the left column. We can identify the content of this column with *five* NQAE tuples. For instance, the narrative of the first tuple consists of sentences in lines 1-6; the narrative of the second tuple consists of sentences in lines 1-6,8, and 9. Lines 7 and 10 enumerate question, answer and explanation of the first and second tuples, respectively. In the sequel, we frequently identify each *question* with its corresponding NQAE tuple.

The right column translates this narrative and questions into a generic pattern form. Note how, for instance, distinct verbs *picked up, grabbed, got* occurring in the left column are captured by the same concept `grab` in the right column; also specific names of people, objects, and locations are identified with ids of the form `person_p`, `obj_y`, and `loc_z`, respectively.

	bAbI example	**bAbI pattern**
P1	**A** Daniel went to the bedroom.	**A** person$_P$ <move_to> loc$_X$.
	B Daniel picked up the apple there.	**B** person$_P$ <grab> obj$_Y$.
	C Daniel put down the apple there.	**C** person$_P$ <drop> obj$_Y$.
	Q Where is the apple? bedroom **A C**	**Q** Where is obj$_Y$? loc$_X$ **A C**
P1′	**A** Sandra got the football there.	**A** person$_P$ <grab> obj$_Y$.
	B Sandra travelled to the office.	**B** person$_P$ <move_to> loc$_X$.
	C Sandra put down the football there.	**C** person$_P$ <drop> obj$_Y$.
	Q Where is the football? office **B C**	**Q** Where is obj$_Y$? loc$_X$ **B C**
P2	**A** John grabbed the football there.	**A** person$_P$ <grab> obj$_Y$.
	B John went to the bedroom.	**B** person$_P$ <move_to> loc$_Z$.
	Q Where is the football? bedroom **A B**	**Q** Where is obj$_Y$? loc$_Z$ **A B**

Table 1. Task 2 NQAE Patterns

Table 1 captures two patterns of NQAE tuples. Consider pattern **P1**. Let us examine the *bAbI example* column of Table 1. The three declarative sentences **A-C** stem from the narrative consisting of lines 1-6 in Figure 6 (left column). The question **Q** coincides with the one listed in line 7 in Figure 6 (left column). These three sentences (out of the total six sentences of the considered narrative) appear in **P1** as each of them refers to a pair of entities related to inferences required to find the answer to given question **Q**. In other words to establish the location of an entity *apple* we have to analyze its relation to entities *Daniel* and *bedroom*. All other sentences preceding the question in line 7 in Figure 6 (left column) do not refer to any of the three entities among *apple*, *Daniel*, and *bedroom*. The *Pattern format* column of Table 1 uncovers the generic form of the NQAE tuple of the considered example. Now note that the first three questions (NQAE tuples) in Figure 6 (left column) fall into this pattern **P1**. Question in line 21 in Figure 6 (left column) falls into pattern **P2** presented in Table 1. Question in line 16 in Figure 6 (left column) falls into a variant of pattern **P1** that we denote as **P1′**. Pattern **P1′** is formed from **P1** by switching the order of sentences of the form **A** and **B** presented in Table 1.

As mentioned earlier, we focused on processing the narratives listed in the training dataset. Patterns **P1**, **P1′**, **P2** are the only ones occurring within these narratives.

These patterns resurface in the same manner when narratives are being processed by tExplain. Figure 2 presents the behavior of system tExplain on patterns **P1**, **P1′**, and **P2** from bAbI: these bAbI patterns are repeated once more in the left hand side column. For all patterns, the explanations given by tExplain coincide with the ones suggested by bAbI. For patterns **P1** and **P1′** additional explanations are produced by the tExplain system. For patterns **P1** and **P1′**, these additional explanations are logical in their conclusions and can both be considered correct. However, bAbI only shows one "correct" explanation for every NQAE tuple.

	bAbI pattern	tExplain **pattern**
P1	**A** person_P <move_to> loc_X. **B** person_P <grab> obj_Y. **C** person_P <drop> obj_Y. **Q** Where is obj_Y? loc_X **A C**	**A** person_P <move_to> loc_X. **B** person_P <grab> obj_Y. **C** person_P <drop> obj_Y. **Q** Where is obj_Y? loc_X **A C** Additional explanation by tExplain: **A B**
P1′	**A** person_P <grab> obj_Y. **B** person_P <move_to> loc_X. **C** person_P <drop> obj_Y. **Q** Where is obj_Y? loc_X **B C**	**A** person_P <grab> obj_Y. **B** person_P <move_to> loc_X. **C** person_P <drop> obj_Y. **Q** Where is obj_Y? loc_X **B C** Additional explanation by tExplain: **A B**
P2	**A** person_P <grab> obj_Y. **B** person_P <move_to> loc_Z. **Q** Where is obj_Y? loc_Z **A B**	**A** person_P <grab> obj_Y. **B** person_P <move_to> loc_Z. **Q** Where is obj_Y? loc_Z **A B**

Table 2. Comparison between bAbI and tExplain with Task 2 patterns

5 Conclusion

This paper describes the development of information extraction system tExplain equipped with explanations. The system takes English narratives focused on action verbs as an input together with a question and produces a logic program together with specialized annotations for invoking explanations as an output. Answer set solver xclingo is then used to provide a user with the conclusion and explanation that the system derives on the given narrative-question pair. We used the bAbI dataset to both guide the development of the system and evaluate its outcomes. The evaluation illustrates that the system achieves satisfactory performance by not only providing answers to questions but also sensible explanations. It is remarkable that the tExplain system is built from a conglomeration of the off-the-shelf tools and datasets stemming from the natural language processing and knowledge and representation communities whereas

the major genuine part of this project were the design of systematic mappings between the resources. On bAbI, tExplain achieved 100% accuracy on Task 2 along with Tasks 3 (Three Supporting Facts) and 6 (Yes/ No Questions).

References

1. Framenet. https://framenet.icsi.berkeley.edu/fndrupal/
2. Verbnet. http://verbs.colorado.edu/ mpalmer/projects/verbnet.htm
3. Wordnet. http://wordnet.princeton.edu/
4. Aguado, F., Cabalar, P., Fandinno, J., Muñiz, B., Pérez, G., Suárez, F.: A rule-based system for explainable donor-patient matching in liver transplantation. In: Bogaerts, B., Erdem, E., Fodor, P., Formisano, A., Ianni, G., Inclezan, D., Vidal, G., Villanueva, A., Vos, M.D., Yang, F. (eds.) Proceedings 35th International Conference on Logic Programming (Technical Communications), ICLP 2019 Technical Communications, Las Cruces, NM, USA, September 20-25, 2019. EPTCS, vol. 306, pp. 266–272 (2019). https://doi.org/10.4204/EPTCS.306.31, https://doi.org/10.4204/EPTCS.306.31
5. Balai, E., Gelfond, M., Zhang, Y.: Towards answer set programming with sorts. In: Cabalar, P., Son, T.C. (eds.) Logic Programming and Nonmonotonic Reasoning. pp. 135–147. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
6. Barker, K., Porter, B., Clark, P.: A Library of Generic Concepts for Composing Knowledge Bases. Proceedings of the 1st International Conference on Knowledge Capture - K-CAP pp. 14–21 (2001). https://doi.org/10.1145/500742.500744, http://portal.acm.org/citation.cfm?doid=500737.500744 http://www.cs.utexas.edu/users/mfkb/papers/kcap01.pdf
7. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Commun. ACM **54**(12), 92–103 (2011). https://doi.org/10.1145/2043174.2043195
8. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Multi-shot asp solving with clingo. Theory and Practice of Logic Programming **19**(1), 27–82 (2019). https://doi.org/10.1017/S1471068418000054
9. Inclezan, D.: CoreALMlib: An ALM library translated from the Component Library. Theory and Practice of Logic Programming **16**(5-6), 800–816 (2016). https://doi.org/10.1017/S1471068416000363
10. Inclezan, D., Gelfond, M.: Modular action language ALM. TPLP **16**(2), 189–235 (2016). https://doi.org/10.1017/S1471068415000095, http://dx.doi.org/10.1017/S1471068415000095
11. Levin, B.: English verb classes and alternations : a preliminary investigation. University Of Chicago Press (1993)
12. Lierler, Y., Ling, G., Olson, C.: Information extraction tool text2alm: From narratives to action language system descriptions and query answering. AI communications (to appear)
13. Palmer, M.: VerbNet. https://verbs.colorado.edu/verb-index/vn3.3/ (2018)
14. Sare, J.: Extending Text2Alm with Xclingo (2023)
15. Weston, J., Bordes, A., Chopra, S., Mikolov, T.: Towards AI-complete question answering: A set of prerequisite toy tasks. CoRR **abs/1502.05698** (2015), http://arxiv.org/abs/1502.05698