An Architecture of Semantic Information Extraction Tool Text2ALM

1st Yuliya Lierler
University of Nebraska Omaha
Omaha,USA
ylierler@unomaha.edu

2nd Craig Olson
University of Nebraska Omaha
Omaha,USA
craig.olson19@gmail.com

***Abstract*—In this work we design a narrative understanding tool TEXT2ALM. This tool uses an action language $\mathcal{ALM}$ to perform inferences on complex interactions of events described in narratives. The methodology used to implement the TEXT2ALM system was originally outlined by Lierler, Inclezan, and Gelfond [11] via a manual process of converting a narrative to an $\mathcal{ALM}$ model. It relies on a conglomeration of resources and techniques from two distinct fields of artificial intelligence, namely, natural language processing and knowledge representation and reasoning. The effectiveness of system TEXT2ALM is measured by its ability to correctly answer questions from the bAbI tasks by Facebook Research in 2015. This tool matched or exceeded the performance of state-of-the-art machine learning methods in six of the seven tested tasks.**

I. INTRODUCTION

The field of Information Extraction (IE) is concerned with gathering snippets of meaning from text and storing the derived data in structured, machine interpretable form. Consider a sentence *BBDO South in Atlanta, which handles corporate advertising for Georgia-Pacific, will assume additional duties for brands like Angel Soft toilet tissue and Sparkle paper towels, said Ken Haldin, a spokesman for Georgia-Pacific from Atlanta.* A sample IE system that focuses on identifying organizations and their corporate locations may extract the following predicates from this sentence: $locatedIn(BBDOSouth, Atlanta)$ and $locatedIn(GeorgiaPacific, Atlanta)$. These predicates can then be stored either in a relational database or a logic program, and queried accordingly by well-known methods in computer science. Thus, IE allows us to turn unstructured data present in text into structured data easily accessible for automated querying. In this paper, we focus on an IE system that is capable of processing simple narratives with *action verbs*, in particular, verbs that express physical acts such as *go*, *give*, and *put*. Consider a sample narrative, named *JS*, discourse:

John traveled to the hallway. (1)

Sandra journeyed to the hallway. (2)

The actions *travel* and *journey* in the narrative describe changes to the narrative's environment, and can be coupled with the reader's commonsense knowledge to form and alter the reader's mental picture for the narrative. For example, after reading sentence (1), a human knows that (i) *John* is the subject of the sentence and *traveled* is an action verb describing an action performed by *John*; and (ii) *traveled* describes the act of motion, and specifically that *John*'s location changes from an arbitrary initial location to a new destination, the *hallway*. Lierler et al. [11] outline a methodology for constructing a Question Answering (QA) system by utilizing IE techniques. Their methodology focuses on performing inferences using the complex interactions of events in narratives. Their process utilizes an action language $\mathcal{ALM}$ [7] and an extension of the VERBNET lexicon [10], [16]. Language $\mathcal{ALM}$ enables a system to structure knowledge regarding complex interactions of events and implicit background knowledge in a straight-forward and modularized manner. The represented knowledge is then used to derive inferences about a given text. The proposed methodology assumes the extension of the VERBNET lexicon with interpretable semantic annotations in $\mathcal{ALM}$. The VERBNET lexicon groups English verbs into classes allowing us to infer that such verbs as *travel* and *journey* practically refer to the same class of events.

The processes described in [11] are exemplified via two sample narratives that were completed manually. The authors translated those narratives to $\mathcal{ALM}$ programs by hand and wrote the supporting $\mathcal{ALM}$ modules to capture knowledge as needed. A narrative understanding system developed within this work, TEXT2ALM, automates the method from [11]. When considering the *JS* discourse as an example, system TEXT2ALM produces a set of facts in spirit of the following:

$$move(john, hallway, 0) \quad move(sandra, hallway, 1) \qquad (3)$$

$$loc\_in(john, hallway, 1) \quad loc\_in(john, hallway, 2) \qquad (4)$$

$$loc\_in(sandra, hallway, 2) \qquad (5)$$

where $0, 1, 2$ are time points associated with occurrences of described actions in the *JS* discourse. Intuitively, time point 0 corresponds to a time prior to utterance of sentence (1). Time point 1 corresponds to a time upon the completion of the event described in (1). Facts in (3)-(5) allow us to provide grounds for answering

This is an abstract of the paper titled "Information Extraction Tool Text2Alm: From Narratives to Action Language System Descriptions" that appeared at Proceedings of the 35th International Conference on Logic Programming in 2019. The work was partially supported by the NSF 1707371 grant.

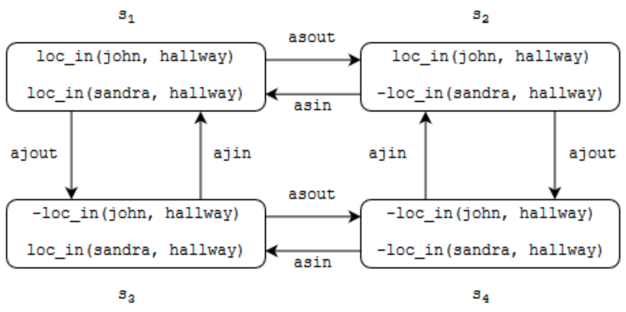

Figure 1. Sample transition diagram capturing the *JS* discourse.

questions related to the *JS* discourse such as: Is *John* inside the *hallway* at the end of the story (time 2)? Indeed, given the fact $loc\_in(john, hallway, 2)$. Who is in the *hallway* at the end of the story? John and Sandra constitute an answer given $loc\_in(john, hallway, 2)$ and $loc\_in(sandra, hallway, 2)$.

II. Background

a) NLP Resource VerbNet*:* VerbNet is a domain-independent English verb lexicon organized into a hierarchical set of verb classes [10], [16]. The verb classes aim to achieve syntactic and semantic coherence between members of a class. Each class is characterized by a set of verbs and their thematic roles. For example, the verb *run* is a member of the VerbNet class run-51.3.2. This class is characterized by (i) 96 members including verbs such as *bolt, frolic, scamper,* and *weave*, (ii) four thematic roles, namely, *theme*, *initial location*, *trajectory* and *destination*.

b) Dynamic Domains, Transition Diagrams, and Action Language $\mathcal{ALM}$: Action languages are formal KRR languages that provide convenient syntactic constructs to represent knowledge about dynamic domains. The knowledge is compiled into a transition diagram, where nodes correspond to possible states of a considered dynamic domain and edges correspond to actions/events whose occurrence signal transitions/changes in the dynamic system.

The *JS* discourse exemplifies a narrative modeling a dynamic domain with three entities *John, Sandra, hallway* and four actions, specifically:

1) *ajin* – *John* travels into the *hallway*,
2) *ajout* – *John* travels out of the *hallway*,
3) *asin* – *Sandra* travels into the *hallway*, and
4) *asout* – *Sandra* travels out of the *hallway*.

The transition diagram capturing the possible states of this domain is given in Figure 1. State $s1$ designates the state where the location of *John* and *Sandra* is the *hallway*. Likewise, state $s2$ characterizes the state where *John*'s location is the *hallway*, but *Sandra*'s location is not the *hallway*. Occurrence of action $asout$ is responsible for the transition from state $s1$ to state $s2$.

Scenarios of a dynamic domain correspond to *trajectories* in the domain's transition diagram. Trajectories are sequences of alternating states and actions. A trajectory captures the sequence of events, starting with the initial state associated with time point 0. Each edge is associated with the time point incrementing by 1. Consider a sample trajectory $\langle s4, ajin, s2, asin, s1\rangle$ for the transition diagram in Figure 1. It captures the following scenario:

- *John* and *Sandra* are not in the *hallway* at the initial time point 0,
- *John* travels into the *hallway* at time point 0, resulting in a new state of the dynamic system to be $s2$ (*John* is in the *hallway*, while *Sandra* is not) at time 1,
- *Sandra* travels into the *hallway* at time 1, resulting in a new state of the dynamic system to be $s3$ (*John* and *Sandra* are both in the *hallway*) at time 2.

It is easy to see how this sample trajectory captures the scenario of the *JS* discourse.

In this work we utilize an advanced action language $\mathcal{ALM}$ [7] to model dynamic domains of given narratives. This language provides an ability to capture the commonalities of similar actions. We illustrate the syntax and semantics of $\mathcal{ALM}$ using the *JS* discourse dynamic domain by first defining an $\mathcal{ALM}$ "system description" and then an $\mathcal{ALM}$ "history" for this discourse.

In language $\mathcal{ALM}$, a dynamic domain is described via a *system description* that captures a transition diagram specifying the behavior of a given domain. An $\mathcal{ALM}$ system description consists of a theory and a structure. A *theory* is comprised of a hierarchy of modules, where a module represents a unit of general knowledge. A module contains declarations of sorts, attributes, and properties of the domain, together with axioms describing the behavior of actions and properties. There are four types axioms, namely, (i) dynamic causal laws, (ii) executability conditions, (iii) state constraints, and (iv) function definitions. The properties that can be changed by actions are called fluents and modeled by functions in $\mathcal{ALM}$. The *structure* declares instances of entities and actions of the domain. Figure 2 illustrates these concepts with the $\mathcal{ALM}$ formalization of the *JS* discourse domain. The resulting formalization depicts the transition diagram that we can obtain from the diagram in Figure 1 by erasing the edges annotated with $ajout$ and $asout$.

The *JS* discourse theory is composed of a single module containing the necessary knowledge associated with the domain. The module starts with the declarations of sorts (*agents*, *points*, *move*) and fluents (`loc_in`). Sorts *universe* and *actions* are predefined in $\mathcal{ALM}$ so that any entity of a domain is considered of *universe* sort, whereas any declared action/event is considered of *actions* sort. While declaring actions, the $\mathcal{ALM}$ user specifies its attributes, which are roles that entities participating in the

```
system description JS_discourse
  theory JS_discourse_theory
   module JS_discourse_module
   sort declarations
    points, agents :: universe
    move :: actions
     attributes
      actor : agents -> booleans
      origin : points -> booleans
     destination : points -> booleans
   function declarations
    fluents
    loc_in : agents * points -> booleans
   axioms
   dynamic causal laws
     occurs(X) causes loc_in(A,D)
      if instance(X,move), actor(X,A),
           destination(X,D).
   executability conditions
    impossible occurs(X) if
      instance(X,move), actor(X,A),
      loc_in(A,P), origin(X,O), P!=O.
    impossible occurs(X) if
      instance(X,move), actor(X,A),
      loc_in(A,P), destination(X,D), P=D.
   structure john_and_sandra
   instances
     john, sandra in agents
     hallway in points
       ajin in move
        actor(john) = true
        destination(hallway) = true
        asin in move
        actor(sandra) = true
        destination(hallway) = true
```

Figure 2. An $\mathcal{ALM}$ system description formalizing the *JS* discourse dynamic domain

action take. For instance, the attributes of *move* include *actor*, *origin*, and *destination*. Here we would like a reader to draw a parallel between the notions of an attribute and a VERBNET thematic role.

There are two types of axioms in the *JS* discourse theory: dynamic causal laws and executability conditions. The only dynamic causal law states that if a *move* action occurs with a given *actor* and *destination*, then the *actor*'s location becomes that of the *destination*. The executability conditions restrict an action from occurring if the action is an instance of *move*, where the *actor* and *actor*'s location are defined, but either (i) the *actor*'s location is not equal to the *origin* of the *move* event or (ii) the *actor*'s location is already the *destination*.

An $\mathcal{ALM}$ structure in Figure 2 defines the entities of sorts *agents*, *points*, and *actions* that occurred in the *JS* discourse. For example, it states that *john* and *sandra* are *agents*. Then, the structure declares an action $ajin$ as an instance of *move* where *john* is the *actor* and *hallway* is the *destination*. Likewise, $asin$ is declared as an instance of *move*, where *sandra* is the *actor* and *hallway* is the *destination*.

In $\mathcal{ALM}$, a *history* is a particular scenario described by observations about the values of fluents and events that occur. In the case of narratives, a history describes the sequence of events by stating occurrences of specific actions at given time points. For instance, the *JS* discourse history contains the events

- *John moves* to the *hallway* at the beginning of the story (an action $ajin$ occurs at time 0) and
- *Sandra moves* to the *hallway* at the next point of the story (an action $asin$ occurs at time 1).

The following history is appended to the end of the system description in Figure 2 to form an $\mathcal{ALM}$ program for the *JS* disocurse. We note that $happened$ is a keyword that captures the occurrence of actions.

```
happened(ajin, 0).
happened(asin, 1).
```

c) An $\mathcal{ALM}$ Solver CALM*:* System CALM is an $\mathcal{ALM}$ solver developed at Texas Tech University by Wertz, Chandrasekan, and Zhang [18]. It uses an $\mathcal{ALM}$ program to produce a "model" for an encoded dynamic domain. Behind the scene system CALM (i) constructs a logic program under stable model/answer set semantics [5], whose answer sets/solutions are in one-to-one correspondence with the models of the $\mathcal{ALM}$ program, and (ii) uses an answer set solver SPARC [1] for finding these models. The $\mathcal{ALM}$ program in Figure 2 follows the CALM syntax. However, system CALM requires two additional components for this program to be executable. The user must specify (i) the computational task and (ii) the max time point considered.

System CALM can solve temporal projection and planning computational tasks. Our work utilizes *temporal projection*, which is the process of determining the effects of a given sequence of actions executed from a given initial situation (which may be not fully determined). In the case of a narrative, the initial situation is often unknown, whereas the sequence of actions are provided by the discourse. Inferring the effects of actions allows us to properly answer questions about the domain. To perform temporal projection, we insert the line following statement in the $\mathcal{ALM}$ program prior to the history:

```
  temporal projection
```

Additionally, CALM requires the max number of time points/steps to be stated. Intuitively, we see this number as an upper bound on the "length" of considered trajectories. In temporal projection problems, this information denotes the final state's time point. To define the max step for the *JS* discourse $\mathcal{ALM}$ program, we insert the following line in the $\mathcal{ALM}$ program:

```
   max steps 3
```

For the case of the temporal projection task, a model of an $\mathcal{ALM}$ program is a trajectory in the transition system captured by the $\mathcal{ALM}$ program that is "compatible" with the provided history. For example, a trajectory $\langle s4, ajin, s2, asin, s1\rangle$ is the only model for the *JS* discourse $\mathcal{ALM}$ program. For the *JS* discourse $\mathcal{ALM}$ program, the CALM computes a model that includes the following expressions:

```
  happened(ajin, 0),  happened(asin, 1),
  loc_in(john, hallway, 1),
  loc_in(sandra, hallway, 2),
  loc_in(john, hallway, 2)
```

d) $\mathcal{ALM}$ Knowledge Base COREALMLIB*:* The COREALMLIB is an $\mathcal{ALM}$ library of generic commonsense knowledge for modeling dynamic domains developed by Inclezan [6]. The library's foundation is the Component Library or CLib [2], which is a collection of general, reusable, and interrelated components of knowledge. CLib was populated with knowledge stemming from linguistic and ontological resources, such as VERBNET, WORDNET, FRAMENET, a thesaurus, and an English dictionary. The COREALMLIB was formed by translating CLib into $\mathcal{ALM}$ to obtain descriptions of 123 action classes grouped into 43 reusable modules. The modules are organized into a hierarchical structure and contain action classes.

III. SYSTEM TEXT2ALM ARCHITECTURE

Lierler, Inclezan, and Gelfond [11] outline a methodology for designing IE/QA systems to make inferences based on complex interactions of events in narratives. This methodology is exemplified with two sample narratives that were completed manually by the authors. System TEXT2ALM automates the process outlined in [11]. Figure 3 pretenses the architecture of the system. It implements four main tasks/processes:

1) TEXT2DRS Processing – Entity, Event, and Relation Extraction
2) DRS2ALM Processing – Creation of $\mathcal{ALM}$ Program
3) CALM Processing – $\mathcal{ALM}$ Model Generation and Interpretation
4) QA Processing

In Figure 3, each process is denoted by its own column. Ovals identify inputs and outputs. Systems or resources are represented with white, grey, and black rectangles. White rectangles denote existing, unmodified resources. Grey rectangles are used for existing, but modified resources. Black rectangles signify newly developed subsystems. The first three processes form the core of TEXT2ALM, seen as an IE system. The QA Processing component is specific to the bAbI QA benchmark that we use for illustrating the validity of the approach advocated by TEXT2ALM. The system is available at `https://github.com/cdolson19/Text2ALM`.

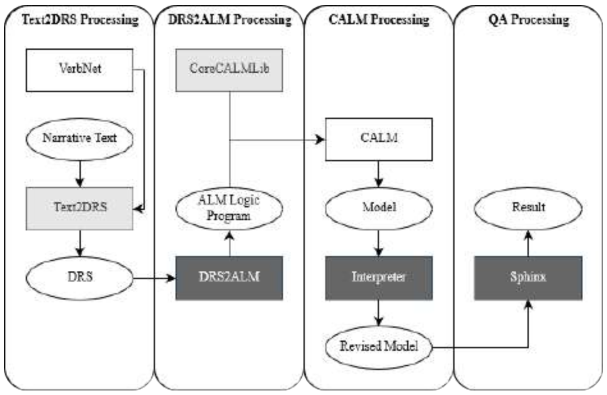

Figure 3. System TEXT2ALM Architecture

a) TEXT2DRS *Processing:* To produce $\mathcal{ALM}$ system descriptions for considered narratives, the method by Lierler et al. [11] utilizes NLP resources, such as semantic role labeler LTH [8], parsing and coreference resolution tools of CORENLP [13], and lexical resources PROPBANK [17] and SEMLINK [3]. System TEXT2DRS [12] was developed with these resources to deliver a tool that extracts entities, events, and their relations from given narratives. The TEXT2DRS tool became a starting point in the development of TEXT2ALM. The output of the TEXT2DRS system is called a discourse representation structure, or DRS [9]. A DRS captures key information present in discourse in a structured form. For example, for the *JS* discourse its DRS will include information that there are three entities and two events that take part in the *JS* narrative. It will also identify both of the events of the discourse with the VERBNET class RUN-51.3.2-1.

b) DRS2ALM *Processing:* The DRS2ALM subsystem is concerned with combining commonsense knowledge related to events in a discourse with the information from the DRS generated by TEXT2DRS. The goal of this process is to produce an $\mathcal{ALM}$ program consisting of a system description and a history pertaining the scenario described by the narrative. The system description is composed of a theory containing relevant commonsense knowledge, and a structure that is unique for a given narrative. One of the key components of the DRS2ALM Processing is the COREALMLIB knowledge base, which was modified to form CORECALMLIB to suit the needs of the TEXT2ALM system. In particular CORECALMLIB adds a layer to COREALMLIB that maps the entires in VERBNET ontology with the entries in COREALMLIB ontology. This layer allows us to properly convert the DRS of a given narrative into an $\mathcal{ALM}$ system description.

c) CALM *and QA Processing:* In the CALM Processing performed by TEXT2ALM, the CALM system is invoked on a given $\mathcal{ALM}$ program stemming from a narrative in question. The CALM system computes a model. We then perform post-processing on this model to make its content more readable for a human.

A model derived by the CALM system contains facts about the entities and events from the narrative supplemented with basic commonsense knowledge associated with the events. We use the bAbI QA tasks to test the TEXT2ALM system's IE effectiveness and implement QA capabilities within the SPHINX subsystem (see Figure 3). The SPHINX component utilizes regular expressions to identify a kind of question that is being asked in the bAbI tasks and then query the model for relevant information to derive an answer. The SPHINX system is not a general purpose question answering component.

Additional information on the components of system TEXT2ALM are given in [15].

IV. TEXT2ALM EVALUATION

a) Related Work:: Many modern QA systems predominately rely on machine learning techniques. However, there has recently been more work related to the design of QA systems combining advances of NLP and KRR. The TEXT2ALM system is a representative of the latter approach. Other approaches include the work by Clark, Dalvi, and Tandon [4] and Mitra and Baral [14]. Mitra and Baral [14] use a provided training dataset of narratives, questions, and answers to learn the knowledge needed to answer similar questions. Their approach posted nearly perfect test results on the bAbI tasks. However, this approach doesn't scale to narratives that utilize other action verbs, which are not present in the training set, including synonymous verbs. For example, if their system is trained on bAbI training data that contains verb *travel* it will process the *JS* discourse correctly. Yet, if we alter the *JS* discourse by exchanging *travel* with a synonymous word *stroll*, their system will fail to perform inferences on this altered narrative (note that *stroll* does not occur in the bAbI training set). The TEXT2ALM system does not rely upon the training narratives for the commonsense knowledge. If the verbs occurring in narratives belong to VERBNET classes whose semantics have been captured within CORECALMLIB then TEXT2ALM is normally able to process them properly.

Another relevant QA approach is the work by Clark, Dalvi, and Tandon [4]. This approach uses VERBNET to build a knowledge base containing rules of preconditions and effects of actions utilizing the semantic annotations that VERBNET provides for its classes. In our work, we can view $\mathcal{ALM}$ modules associated with VERBNET classes as machine interpretable alternatives to these annotations. Clark et al. [4] use the first and most basic action language STRIPS for inference that is more limited than $\mathcal{ALM}$.

b) Evaluation: We use Facebook AI Research's bAbI dataset [19] to evaluate system TEXT2ALM. These tasks were proposed by Facebook Research in 2015 as a benchmark for evaluating basic capabilities of QA systems in twenty categories. Each of the twenty bAbI QA tasks is composed of narratives and questions, where 1000 questions are given in training set and 1000 questions are given in a testing set. The goal of the tasks are to process testing sets properly while also minimizing the number of questions used from the training set to develop a solution. We evaluate the TEXT2ALM system with all 1000 questions in the testing sets for tasks 1, 2, 3, 5, 6, 7, and 8. These tasks are selected because they contain action-based narratives that are of focus in this work. Figure 4 provides an example of a narrative and a question from the training set of bAbI task 2-Two Supporting Facts. For this task, a QA system must combine information from two sentences in the given narrative. The narrative in Figure 4 consists of six sentences. A question is given in line 7, followed by the answer and identifiers for the two sentences that provide information to answer the question.

```
1 Mary moved to the bathroom.
2 Sandra journeyed to the bedroom.
3 Mary got the football there.
4 John went to the kitchen.
5 Mary went back to the kitchen.
6 Mary went back to the garden.
7 Where is the football? garden 3 6
```

Figure 4. Example entry from bAbI task 2 training set

The bAbI dataset enables us to compare TEXT2ALM's IE/QA ability with other modern approaches designed for this task. The left hand side of Figure 5 compares the accuracy of the TEXT2ALM system with the machine learning approach AM+NG+NL MemNN described by Weston et al. [19]. In that work, the authors compared results from 8 machine learning approaches on bAbI tasks and the AM+NG+NL MemNN (Memory Network) method performed best almost across the board. There were two exceptions among the seven tasks that we consider. For the Task 7-Counting the AM+N-GRAMS MemNN algorithm was reported to obtain a higher accuracy of 86%. Similarly, for the Task 8-Lists/Sets the AM+NONLINEAR MemNN algorithm was reported to obtain accuracy of 94%. Figure 5 also presents the details on the Inductive Rule Learning and Reasoning (IRLR) approach by [14]. We cannot compare TEXT2ALM performance with the methodology by [4] because their system is not available and it has not been evaluated using the bAbI tasks.

System TEXT2ALM matches the Memory Network approach by Weston et al. [19] at 100% accuracy in tasks 1, 2, 3, and 6 and performs better on tasks 7 and 8. When compared to the methodology by Mitra and Baral [14], the Text2ALM system matches the results for tasks 1, 2, 3, 6, and 8, but is outperformed in tasks 5 and 7.

	Accuracy		
bAbI Task	AM+NG+NL MemNN	IRLR	TEXT2ALM
1-Single SF	100	100	100
2-Two SF	100	100	100
3-Three SF	100	100	100
5-Three AR.	98	100	22
6-Yes/No	100	100	100
7-Counting	85	100	96.1
8-Lists/Sets	91	100	100
	Training Size		
1-Single SF	250	1000	100
2-Two SF	500	1000	100
3-Three SF	500	1000	100
5-Three AR.	1000	1000	100
6-Yes/No	500	1000	100
7-Counting	1000	1000	100
8-Lists/Sets	1000	1000	100

Figure 5. System Evaluation and Training Set Sizes

The right hand side of Figure 5 presents the number of questions in training sets used by each of the reported approaches in considered tasks. The TEXT2ALM training set comprised of 100 questions per QA bAbI task, for a total of 700 questions. These training questions and their associated narratives were used to develop the CORECALMLIB knowledge base (recall *Library* CORECALMLIB adopts the earlier COREALMLIB knowledge base). As a result of this process, the CORECALMLIB covers 20 first-level VERBNET classes out of its 274.

V. CONCLUSION

System TEXT2ALM matched or outperformed the results of modern machine learning methods in all of these tasks except task 5. It matched the results of another KRR approach [14] in tasks 1, 2, 3, 6, and 8. However, our approach adjusts well to narratives with a more diverse lexicon due to its architecture. Additionally, the ability of the CORECALMLIB to represent the interactions of events in the bAbI narratives serves as a proof of usefulness of the original COREALMLIB endeavor.

REFERENCES

[1] E. Balai, M. Gelfond, and Y. Zhang, "Towards answer set programming with sorts," in *Logic Programming and Nonmonotonic Reasoning*, P. Cabalar and T. C. Son, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 135–147.

[2] K. Barker, B. Porter, and P. Clark, "A Library of Generic Concepts for Composing Knowledge Bases," *Proceedings of the 1st International Conference on Knowledge Capture - K-CAP*, pp. 14–21, 2001.

[3] C. Bonial, K. Stowe, and M. Palmer, "SemLink," `https://verbs.colorado.edu/semlink/`, 2013.

[4] P. Clark, B. Dalvi, and N. Tandon, "What happened? Leveraging VerbNet to predict the effects of actions in procedural text," *CoRR*, vol. abs/1804.05435, 2018. [Online]. Available: http://arxiv.org/abs/1804.05435

[5] M. Gelfond and V. Lifschitz, "The stable model semantics for logic programming," in *Proceedings of International Logic Programming Conference and Symposium*, R. Kowalski and K. Bowen, Eds. MIT Press, 1988, pp. 1070–1080.

[6] D. Inclezan, "CoreALMlib: An ALM library translated from the Component Library," *Theory and Practice of Logic Programming*, vol. 16, no. 5-6, pp. 800–816, 2016.

[7] D. Inclezan and M. Gelfond, "Modular action language ALM," *TPLP*, vol. 16, no. 2, pp. 189–235, 2016. [Online]. Available: http://dx.doi.org/10.1017/S1471068415000095

[8] R. Johansson and P. Nugues, "LTH: Semantic structure extraction using nonprojective dependency trees," in *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*.

[9] H. Kamp and U. Reyle, *From discourse to logic*. Kluwer, 1993, vol. 1,2.

[10] K. Kipper-Schuler, "VerbNet: A broad-coverage, comprehensive verb lexicon," Ph.D. dissertation, University of Pennsylvania, 2005.

[11] Y. Lierler, D. Inclezan, and M. Gelfond, "Action languages and question answering," in *IWCS 2017 - 12th International Conference on Computational Semantics - Short papers*, 2017.

[12] G. Ling, "From Narrative Text to VerbNet-Based DRSes: System Text2DRS," 2018. [Online]. Available: https://www.unomaha.edu/college-of-information-science-and-technology/natural-language-processing-and-knowledge-representation-lab/_files/papers/Text2Drses_system_description.pdf

[13] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The Stanford CoreNLP Natural Language Processing Toolkit," *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55–60, 2014.

[14] A. Mitra and C. Baral, "Addressing a Question Answering Challenge by Combining Statistical Methods with Inductive Rule Learning and Reasoning," pp. 2779–2785, 2016.

[15] C. Olson, "Processing Narratives by Means of Action Languages," Master's thesis, University of Nebraska Omaha, 5 2019.

[16] M. Palmer, "VerbNet," Boulder, 2018. [Online]. Available: https://verbs.colorado.edu/verb-index/vn3.3/

[17] M. Palmer, D. Gildea, and P. Kingsbury, "The proposition bank: An annotated corpus of semantic roles," *Computational Linguistics*, vol. 31, no. 1, pp. 71–106, Mar. 2005.

[18] E. Wertz, A. Chandrasekan, and Y. Zhang, "CALM: a Compiler for Modular Action Language ALM," unpublished draft, 2018.

[19] J. Weston, A. Bordes, S. Chopra, and T. Mikolov, "Towards AI-complete question answering: A set of prerequisite toy tasks," *CoRR*, vol. abs/1502.05698, 2015.

Архитектура приложения Text2ALM для семантической обработки языка

Юлия Лирлер и Крэйг Олсон

Приложения Text2ALM ориентируется на семантическую обработку текста с глаголами действия. Эта система использует язык программирования действий под названием ALM для выполнения выводов о сложных взаимодействиях событий, описанных в тексте. Система опирается на ресурсы и методы из двух различных областей искусственного интеллекта, а именно: обработка естественного языка и представление знаний. Эффективность приложения Text2ALM измеряется по ее способности правильно отвечать на вопросы из задач babi (Facebook Research, 2015). Text2ALM соответствовал или превышал производительность современных методов машинного обучения в шести из семи протестированных заданий.

Received 04.01.2020