Interpreting Natural Language Sources Using Transition Diagrams

Emily C. LeBlanc and Marcello Balduccini

College of Computing and Informatics
Drexel University
{leblanc, mbalduccini}@drexel.edu

Abstract. We introduce the early stages of an investigation into a novel method for interpreting natural language sources that describe sequences of actions. Beyond representing the actions themselves, the approach leverages natural language processing techniques and constraint-based reasoning to represent and reason about the state of the world before, during, and after them. We believe that deep reasoning can be carried out over the representation by Information Retrieval and Question Answering systems.

1 Introduction

Natural language understanding is central to modern Information Retrieval and Question Answering research. Tasks in both fields are concerned with gaining an understanding of the content of natural language sources. In some works, the approach to understanding is strictly syntactic or statistical while others attempt to understand and represent the meaning of the content of the sources (discussed in Section 2). In this paper, we are concerned with the latter approach and present our initial investigation into a novel method for interpreting natural language sources that describe sequences of actions. Beyond extracting and representing the actions themselves, the approach leverages natural language processing techniques and constraint-based reasoning over domain and commonsense knowledge to learn and represent information about the state of the world before, during, and after them. Inspired by techniques from reasoning about actions and change, we assume that the knowledge exists as constraints using Answer Set Programming (ASP). These constraints embody a transition diagram describing all possible world states of the domain and the actions that trigger transitions between them. Given a natural language source describing a sequence of actions occurring in some domain, our task is to map that sequence and corresponding state information, via inference, to one or more paths in the transition diagram. The resulting path (or paths) describe possible evolutions of the domain that are consistent with the content of the source. The path itself, then, is a detailed semantic representation upon which deep reasoning can be carried out by Information Retrieval and Question Answering systems.

The organization of this paper is as follows. In the next section, we describe the motivation of the work in greater detail and discuss related research. Following that, we introduce the high-level steps that comprise the current approach. Then, we provide background for our formalization of knowledge and actions using Answer Set Programming. In the section that follows, we describe the details of the present approach to mapping natural language sources to paths of the transition diagram. Following that, we present two use cases to demonstrate the current capabilities as well as immediate goals for our ongoing research. We conclude with a summary of the work and some closing discussion.

2 Motivation

This work is a step forward towards emulating the level of intuition used by humans when we read and comprehend documents describing events. For example, when we read a news article we do not only retain the chain of events that it describes. We also use our understanding of the context in which the events occurred together with our commonsense knowledge to form a picture of the evolution of the state of the world throughout the scenario and how the world may be affected by it. In this section, we discuss a number of approaches to representing the content of natural language sources and provide a demonstration of the level of understanding that motivates this investigation.

Information Retrieval is concerned with retrieving documents that are most likely to fulfill the information need represented by queries. In the classical IR approach to representing sources, the texts are broken down into lists of keywords, terms, and other content descriptors. When presented with a query, an IR system typically determines the relevance of a document to the query by measuring the overlap of terms between the query and a particular source [11]. A number of mature approaches have been developed to improve search results using techniques such as temporal ordering [6], query expansion [7], and graph based term weighting [5], however these approaches can fail to capture the deeper semantic meaning of the source as the representations are constructed using syntactical methods. Recent work in event-centered IR [8] has focused on modeling the content of sources as a graph composed of mentioned events and their temporal relationships; however, the approach does not consider the evolution of the state of the world in correspondence to these actions. The primary goal of the work presented here is to address this gap and attempt to overcome the shortcomings of syntactic-based methods.

Consider the following (heavily) simplified example of knowledge of the factors that affect a geographical area's socioeconomic status (SES). The knowledge base consists of the following four rules:

1. When new stores open, commercial property values may increase.
2. Rising commercial property values may drive out small business stores.
3. When stores close, the number of jobs in the area may decrease.

4. An area's SES may decrease when the number of jobs in the area decreases.

Assume that we have access to the following set of natural language sources:

- Ten new jobs were created after four new stores opened on Lancaster Avenue in 2000.
- Two small businesses on Lancaster Ave closed in 2005.

Using our knowledge base, we look at the events described in the sources and begin to infer facts about the state of the world. Considering the first source with respect to rule (1) in the knowledge base leads us to believe that commercial property values may increase as a result of the new stores opened. From rule (2), we also see that small businesses could eventually be driven out by the value increase. Considering the second source with respect to rule (3), we can intuit that the stores closing may reduce the number of jobs and from rule (4) we know that the reduction in the number of jobs can possibly result in a lower SES for the area. We propose that this level of reasoning can be employed to construct a representation of paths corresponding to the example sources. Suppose that for both of the sources, we have a corresponding representation of the ordering and the intuition described here. If we were to query an Information Retrieval system for relevant documents containing information related to a decrease in SES for Lancaster Avenue, we would expect that the system would leverage the available knowledge with its semantic understanding of the sources to return these sources.

The approach may also contribute to advances in Question Answering. Assuming that a QA system has access to the knowledge described above, consider the following question: *Why did Lancaster Avenue experience a drop in Socioeconomic Status (SES)?*

Considering our knowledge base and the two documents that we have already interpreted, we can intuitively conclude that Lancaster Ave may have experienced a drop in SES because of a chain of events set in motion when four new stores opened. The resulting increase in property values could have driven out two of the small business which in turn decreased the number of jobs in the area. Finally, the area's SES may have decreased because the number of jobs in the area decreased. The somewhat more distant goals of this work includes combining multiple models of documents related to question in order to form a deep semantic representation of the domain.

3 Approach

In this section, we present the high-level approach to interpreting natural language and building the model. Our approach to interpreting a source is composed of two high-level steps: Processing a natural language source, and reasoning about state information in order to enable the construction of a path.

3.1 Processing the Natural Language Source

The first step involves extracting mentions of actions from a natural language source. After collecting the actions, they are placed in a chronologically ordered list. Finally, we translate each action into a corresponding logical statement describing the occurrence of the event at its specific step. For example, if an action is at the first position in the list, we say that it *occurs* at time step 0. The resulting set of statements is called the *problem instance*.

3.2 Reasoning About State Information and Path Construction

Our approach assumes the existence of commonsense and domain-specific knowledge in our approach. Building knowledge repositories is the subject of much research activity (e.g., Cyc). The knowledge is expressed as ASP rules in a program called the *domain description*. Inference is applied to the *instance* and *domain description* are reasoned over together to determine how the effects of the actions are propagated. The result of the reasoning process is a collection of facts about each step in the evolution of the domain. Combining the collection of actions extracted during the source processing step and the state information from the reasoning step, enough information is available to reveal one or more possible paths of the domain's transition diagram. The edges of a path correspond to the actions mentioned in the source document and the nodes contain information about the state of the world before, during, and after them.

4 Preliminaries

We preface the description of the details of our approach with a discussion about the syntax and semantics of ASP [10] as it pertains to our work. The discussion is necessary to enable a clear description of the work of this paper.

Let Σ be a signature containing constant, function and predicate symbols. Terms and atoms are formed as in first-order logic. A *literal* is an atom a or its strong negation $\neg a$. A *rule* is a statement of the form: $h_1, \vee \ldots \vee, h_k \leftarrow l_1, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n$ where h_i 's (the head) and l_i's (the body) are literals and *not* is the so-called *default negation*. Its intuitive meaning in terms of a rational agent reasoning about its beliefs is "if you believe $\{l_1, \ldots, l_m\}$ and have no reason to believe $\{l_{m+1}, \ldots, l_n\}$, then you must believe one of h_i's.' Symbol $\leftarrow$ is omitted if the body is empty. Rules of the form $h \leftarrow l_1, \ldots, \text{not } l_n, \text{not } h$ are abbreviated $\leftarrow l_1, \ldots, \text{not } l_n$, and called *constraints*, intuitively meaning that $\{l_1, \ldots, \text{not } l_n\}$ must not be satisfied. A rule with variables denoted by an uppercase initial is interpreted as a shorthand for the set of rules obtained by replacing the variables with all possible variable-free terms. An ASP rule with an empty body is called a fact, and that in writing facts, the $\leftarrow$ connective is dropped. A *program* is a set of rules over Σ. A consistent set S of literals is closed under a rule if $\{h_1, \ldots, h_k\} \cap S \neq \emptyset$ whenever $\{l_1, \ldots, l_m\} \subseteq S$ and $\{l_{m+1}, \ldots, l_n\} \cap S = \emptyset$. Set S is an answer set of a *not*-free program Π if S

is the minimal set closed under its rules. The reduct, Π^S, of a program Π w.r.t. S is obtained from Π by removing every rule containing an expression "not l" s.t. $l \in S$ and by removing every other occurrence of not l. Set S is an answer set of Π if it is the answer set of Π^S. For a convenient representation of choices, in this paper we also use *constraint literals*, which are expressions of the form $m\{l_1, l_2, \ldots, l_k\}n$, where m, n are arithmetic expressions and l_i's are basic literals. A constraint literal is satisfied w.r.t. S whenever $m \leq |\{l_1, \ldots, l_k\} \cap S| \leq n$. Constraint literals are especially useful to reason about available choices. For example, a rule $1\{p, q, r\}1$ intuitively states that exactly one of $\{p, q, r\}$ should occur in every answer set.

For the formalization of the evolution of the domain over time, we employ techniques from reasoning about actions and change. *Fluents* are first-order terms denoting properties of interest in a domain whose truth value may change over time. For example, *opened(s_1,o_2)* may represent the fact that store s_1 was opened by owner o_2. Fluents are further distinguished as *inertial*, whose truth value persists over time. A *fluent literal* is a fluent f or its negation $\neg f$. A set A of fluent literals is *consistent* if $\forall f, \{f, \neg f\} \not\subseteq A$ and *complete* if $\forall f, \{f, \neg f\} \cap A \neq \emptyset$. The fact that a fluent f holds at a step i is represented by the ASP atom $holds(f, i)$. Similarly, if f does not hold at step i, we write $\neg holds(f, i)$. Occurrences of actions are represented by $occurs(f, i)$. Actions are represented by first-order terms as well. The occurrence of an action a at step i is represented by an ASP atom $occurs(a, i)$.

A *domain description* consists of a collection of ASP rules determining the direct and indirect effects of the domain's actions and the actions' executability conditions – statements describing dependencies between fluents [2].

The three main types of ASP rules used in a domain description are discussed next. In the following, S is a variable ranging over all possible steps in the evolution of the domain. Given a fluent literal l, the abbreviation $\chi(l, S)$ stands for $holds(f, S)$ if $l = f$, and $\neg holds(f, S)$ if $l = \neg f$.

A *dynamic (causal) law* is a statement of the form:

$$\chi(l_0, S+1) \leftarrow \chi(l_1, S), \ldots, \chi(l_n, S), occurs(a, S). \tag{1}$$

where a is an action and l_i's are fluent literals. The statement intuitively means that if a is executed in a state in which $l_1, \ldots, l_n$ hold, it causes l_0 to hold in the resulting state.

A *state constraint* is a statement of the form:

$$\chi(l_0, S) \leftarrow \chi(l_1, S), \ldots, \chi(l_n, S). \tag{2}$$

where l_i's are fluent literals. The statement says that l_0 holds whenever $l_1, \ldots, l_n$ hold.

An *executability condition* is a statement of the form:

$$\leftarrow \chi(l_1, S), \ldots, \chi(l_n, S), occurs(a, S). \tag{3}$$

where a and l_i's are as above. The statement says that action a cannot be executed if $l_1, \ldots, l_n$ hold.

The domain description also contains rules that capture the principle of inertia, which states that things generally stay as they are [12].

$$holds(F, S+1) \leftarrow fluent(F), holds(F, S), \text{not } \neg holds(F, S+1). \quad (4)$$

$$\neg holds(F, S+1) \leftarrow fluent(F), \neg holds(F, S), \text{not } holds(F, S+1). \quad (5)$$

The next rule is called the "awareness axiom" and ensures that a reasoner considers both possible cases of any fluent whose initial truth value is unknown[9].

$$holds(F, 0) \vee \neg holds(F, 0) \leftarrow fluent(F). \quad (6)$$

Note that, if complete knowledge about the initial state is available in the problem instance, then the awareness axiom is rendered inapplicable [3].

A set A of fluent literals is closed under a state constraint if either $l_1, \ldots, l_n \not\subseteq A$ or $l_0 \in A$. A *state* of a domain description D is a complete and consistent set of fluent literals closed under the state constraints of D. Given state σ, $h(\sigma, i)$ denotes $\{holds(f, i) \mid f \in \sigma\} \cup \{\neg holds(f, i) \mid \neg f \in \sigma\}$. Action a is executable in state σ iff $D \cup h(\sigma, 0) \cup \{occurs(a, 0)\}$ has at least one answer set. The set of the possible evolutions of D is represented by a *transition diagram*, i.e., a directed graph $\tau(D) = \langle N, E \rangle$ such that:

1. N is the collection of all states of D;
2. E is the set of all triples $\langle \sigma, a, \sigma' \rangle$ where σ, σ' are states, a is an action that is executable in σ, and $D \cup h(\sigma, 0) \cup \{occurs(a, 0)\}$ has an answer set, A, such that $h(\sigma', 1) \subseteq A$.

A sequence $\langle \sigma_0, \alpha_0, \sigma_1, \ldots, \alpha_{k-1}, \sigma_k \rangle$ is a *path of* $\tau(D)$ if every $\langle \sigma_i, \alpha_i, \sigma_{i+1} \rangle$ is a valid transition in the graph. Each path is a possible evolution of the world consistent with the sequence of events from the source. In Section 6, we will see that these paths are useful for representing the semantics of a sequence of events.

5 Mapping Natural Language to Paths of a Transition Diagram

To demonstrate how a path is constructed step by step, this section follows the processing of a sample natural language source into a sequence of actions and information about state of the world before, during, and after them. Although multiple paths may be identified by the approach, for simplicity's sake we focus on the construction of a single path.

At this early stage of the research, we have simplified a number of aspects of the implementation. We will directly address these simplifications as they arise in the discussion and address how we intend to lift them as the research continues.

We are concerned in part with the ordering of action mentions and so our initial investigation considers a specific format for natural language sources. Much research exists regarding the detection of event mentions and their temporal

relationships [14,13], however at this point we employ a simple method to contribute to the proof of concept. The current approach requires that the natural language sources contain a single sentence conforming to the following template:

$$e_1 \langle temporal\ relation \rangle e_2$$

where e_1 and e_2 are mentions of actions and the temporal relation tag is one of the markers from Allen's interval calculus[1], for example *before, after*. This marker represents the temporal relationship of events $e1$ and $e2$. Consider the following sentence: *"Store owners created jobs after they opened new stores."*

Intuitively, e_1 maps to the statement that jobs were created by store owners, e_2 corresponds to the statement that new stores were opened, and the temporal relation *after* indicates that the four new stores opened prior to the creation of the new jobs. As our work continues, we will incorporate temporal relationship extraction which will enable us to lift the constraints placed on the format of the sources.

The first task is to extract the event mentions from the sources. To do so, the system uses a syntactic translation to detect the events and translate them to an equivalent ASP form, however, work exists to translate natural language to ASP using more advanced techniques. NL2KR [4] is a sophisticated system for such a translation up to the point of being able to learn meanings of words that it has not encountered before. We intend to explore the integration of this work into our implementation to improve the natural language translation. Presently, the system parses the source text using the Stanford CoreNLP libraries[1] which yields tokens, syntactic dependency information, coreference information, and other syntactic attributes of the input text. This information is used to further interpret the natural language text.

From the syntactic dependencies, the system may extract two kinds of action tuples by linking dependencies across the sentence. The first is of the form ⟨*action, subject/object*⟩ and represents actions related to a subject or object, but not both (e.g. "stores opened" or "jobs were created"). The second is of form ⟨*action, subject, object*⟩ and represents actions performed by a subject with respect to some object (e.g. *"Store owners created jobs."* becomes ⟨*created, owners, jobs*⟩).

Once the system has extracted the action tuples, it uses the coreference information obtained from the parsing step for anaphora resolution if needed. Finally all verbs are lemmatized to obtain the base forms. In the case of our running example, the system returns the predicates ⟨*create,owners,jobs*⟩ and ⟨*opened, owners, stores*⟩.

The goal of the next step is to store the action predicates in a chronologically ordered list. The current approach extracts the temporal tag from the natural language source. In our example, the source contains the word "after" which is interpreted to signify that ⟨*opened, owners, stores*⟩, or e_1 from our specification of source format, should appear in the list after ⟨*create,owners,jobs*⟩, or e_2.

[1] http://stanfordnlp.github.io/CoreNLP/

Step	Action	*is_open(stores)*	*exists_new(jobs)*
0	*open(owners,stores)*	*false*	*false*
1	*create(owners,jobs)*	*true*	*false*
2	–	*true*	*true*

Table 1. Values of domain fluents before, during, and after example events.

The system then translates the predicates to the ASP problem instance as follows. It first converts the predicate tuples into corresponding ASP form *action(subject)* or *action(actor,target)*. In our example, we have *open(owners,stores)* and *create(owners,jobs)*. Finally, we assign occurrences of the actions to steps based on the ordering, and we encode the information by means of $occurs(a, i)$ statements. For the example above, the corresponding facts are:

occurs(open(owners,stores),0).
occurs(create(owners,jobs),1).

The domain description contains information about opening and creating things. Knowledge of the act of opening a store is represented using the following two rules:

fluent(is_open(Y)) $\leftarrow$ *occurs(open(X,Y),S).*
holds(is_open(Y),S+1) $\leftarrow$ *occurs(open(X,Y),S).*

The first rule defines a fluent of the form $is\_open(Y, X)$ for every occurrence of an action $open(X, Y)$[2]. The second rule states that the effect of X opening Y is that Y is open. Knowledge of X creating Y is similarly encoded by rules stating that, if X creates Y, then there exists new Y. The task of finding the corresponding paths in the transition diagram can now be reduced to that of invoking an ASP solver, such as `clingo`[3], to find the answer sets of the problem instance and the domain description.

Finally, the system extracts from the answer set(s) the information about each state of the domain. The results are quite intuitive - once the stores have been opened, they remain open in all subsequent states thanks to the inertia axiom (and unless otherwise specified by other portions of the source). Similarly, once the jobs are created, they remain in existence in subsequent states. The system additionally displays the path(s) in a graphical way, as shown in Figure 1.

[2] This is a simple way of defining the fluents relevant to the source, but more sophisticated techniques could be applied.
[3] http://potassco.sourceforge.net/

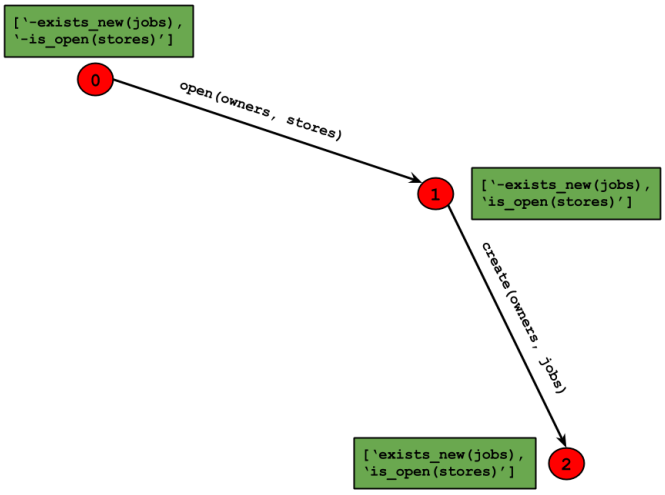

Fig. 1. Path for the sentence *"Store owners created jobs after they opened new stores."*

6 Use Cases

In this section, we present two use cases to illustrate the depth of understanding that is achieved with our approach and some immediate goals of the research.

6.1 Use Case 1: Google Acquires Motorola

To demonstrate the importance of awareness and inertia axioms in our approach, we will begin by processing a natural language sources using both. Then we will observe the change in quality of the representation when we remove awareness and then inertia from the knowledge base. In Use Case 1, we would like to find a path that corresponds to the following sentence[8]: *"Google bought Motorola after the government approved the acquisition."*

The domain description contains information about the effects of purchasing and approval. Knowledge of the act of purchasing is represented using the following two rules:

fluent(is_owned_by(Y,X)) $\leftarrow$ *occurs(buy(X,Y),S).*
holds(is_owned_by(Y,X),S+1) $\leftarrow$ *occurs(buy(X,Y),S).*

The first rule states defines a fluent of the form $is\_owned\_by(Y, X)$ for every occurrence of an action $buy(X, Y)$. The second rule states that the effect of X buying Y is that Y is owned by X. Knowledge of X approving Y is similarly encoded by rules stating that, if X approves Y, then Y is allowed by X.

Step	Action	*allows(acquisition, gov't)*	*owned_by(Motorola, Google)*
0	*approves(gov't,acquisition)*	*false*	*false*
1	*buy(Google,Motorola)*	*true*	*false*
2	–	*true*	*true*

Table 2. Values of fluents in each state with awareness and inertia in the knowledge base.

Step	Action	*allows(acquisition, gov't)*	*owned_by(Motorola, Google)*
0	*approves(gov't,acquisition)*	–	–
1	*buy(Google,Motorola)*	*true*	–
2	–	*true*	*true*

Table 3. Values of fluents in each state when the awareness axiom is removed from the knowledge base.

Step	Action	*allows(acquisition, gov't)*	*owned_by(Motorola, Google)*
0	*approves(gov't,acquisition)*	–	–
1	*buy(Google,Motorola)*	*true*	–
2	–	–	*true*

Table 4. Values of fluents in each state when awareness and inertia are removed from the knowledge base.

Given the above sentence and the domain description, our prototype maps the sentence to ASP statements as described earlier, invokes the solver, and produces the state information shown in Table 2 describing actions and their effects at specific time steps along the path.

As shown in the table, at steps 1 and 2 $owned\_by(Motorola, Google)$ is false, meaning that Google does not own Motorola. However, when the action $buy(Google, Motorola)$ occurs at state 1, a transition occurs to State 2 in which Google does own Motorola, i.e. the value of $owned\_by(Motorola, Google)$ is true.

To highlight the importance of the awareness axiom, let us consider the effects of removing it from the knowledge base. Table 3 shows that there are no truth values for either of the domain fluents until after the occurrence the actions defining them. That is to say that the value of a fluent is not true or false unless explicitly expressed in the knowledge base or as a result of an action. The awareness axiom allows the system to make hypotheses about the values of each possible fluent in every state[3].

Additionally, if the inertia axiom is removed, the information that can be inferred about the states of the domain is further reduced. Specifically, Google owns Motorola in the state just after the purchasing action, but the system has

no information about the ownership in any subsequent states as can be seen in Table 4. It is easy to see that both awareness and inertia are crucial to having accurate state information that reflects not only domain knowledge but human-like intuition.

6.2 Use Case 2: John Sells his Possessions

In this example, we illustrate how the representation proposed in this approach may be used in conjunction with QA techniques to enhance a QA system's ability to reason for answers that are not explicitly stated in the sources or the knowledge base. Finding such an answer may involve a deeper understanding of both the domain and the scenario in question. Note that the technical approach to the QA task itself is not the subject of this example. Rather, the purpose of this example is to illustrate that our approach can provide a QA system with useful information that is not otherwise available.

Consider the following sentence: *"John sold his most valuable possessions to Mary before he sold his least valuable possessions to Frank."*

Assume that we have a knowledge base in which the following information exists about John, and additional commonsense rules exist regarding purchasing and ownership. First, there are definitions of John's valuable and invaluable possessions. For example, John's house at 123 Lancaster Avenue is listed among his valuable property. Suppose that we would like to know the answer to the question *"Who currently owns the house at 123 Lancaster Avenue?"*. Referring to the knowledge base alone, the system would incorrectly answer that John owns the house because it is listed among his valuable property. However, because we have processed the sentence about John selling his possessions, we now have an understanding of the effects of his sales on the state of the world. An end-to-end QA system would then be able to infer that Mary now owns all of John's valuable possessions, and because there is no additional information stating that Mary has sold the house, the system could correctly answer that she is the current owner.

7 Conclusion

In this paper, we introduced a novel method to interpreting natural language documents mentioning the occurrence of actions. The approach extends the state of the art the event-based document representation of [8] by proposing a rich semantic encoding of actions of the consequences of their occurrence on the state of the world. We described the motivation for our work, the high-level approach, our formalization of knowledge and actions, and worked through a detailed example of processing a natural language source from its original form to a path of a domain's transition diagram. Finally, we presented two use cases – the first demonstrated the role of commonsense knowledge in reasoning about sources that mention occurrences of actions and the second illustrated that given a single document and an adequate knowledge base, our approach may provide

enough information to carry out rather sophisticated QA. It is our belief that this approach is a fundamental component of future contributions to the fields of Information Retrieval and Question Answering, enabling these systems to elegantly process unstructured natural language sources at a human-like level. As this work continues, we intend to lift the simplifying assumptions to enable our implementation to perform better temporal tagging and translation. Moreover, the research will extend further to investigate the combination of actions across multiple natural language sources, the expansion, combination, and revisions of existing paths in the face of new or conflicting information, and methods by which the representations can be utilized in the Information Retrieval and Question Answering processes.

References

1. Allen, J.F.: "maintaining knowledge about temporal intervals.". Communications of the ACM 26.11, 832–843 (1983)
2. Balduccini, M., Gelfond, M.: Diagnostic reasoning with a-prolog. Theory and Practice of Logic Programming 3(4+ 5), 425–461 (2003)
3. Baral, C., Gelfond, M., Rushton, N.: International Conference on Logic Programming and Nonmonotonic Reasoning. Springer, Berlin (2004)
4. Baral, C., Dzifcak, J., Kumbhare, K., Vo, N.H.: The nl2kr system. Language Processing and Automated Reasoning (NLPAR) (2013)
5. Blanco, R., Lioma, C.: Graph-based term weighting for information retrieval. Information retrieval 15.1, 54–92 (2012)
6. Campos, R.: Survey of temporal information retrieval and related applications.ACM Computing Surveys (CSUR) 47, 2 (2015)
7. Carpineto, C., Ramano, G.: A survey of automatic query expansion in information retrieval. ACM Computing Surveys (CSUR) 44, 1 (2012)
8. G. Glavas, J.S.: Event-centered information retrieval using kernels on event graphs. TextGraphs-at Empirical Methods in Natural Language Processing EMNLP'13 8 (2013)
9. Gelfond, M., Kahl, Y.: Knowledge Representation, Reasoning, and the Design of Intelligent Agents. The Answer-Set Programming Approach. Cambridge University Press (2014)
10. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing 9, 365–385 (1991)
11. Manning, C., Christopher, D., Raghavan, P., SchŸtze, H.: Introduction to information retrieval., vol. 1. Cambridge University Press, Cambridge (2008)
12. McCarthy, J., Hayes, P.J.: Some philosophical problems from the standpoint of artificial intelligence. Readings in artificial intelligence pp. 431–450 (1969)
13. l. Minard, A.: In: Semeval-2015 task 4: Timeline: Cross-document event ordering. Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015) (2015)
14. UzZaman, N.: Tempeval-3: Evaluating events, time expressions, and temporal relations. preprint, arXiv:1206.5333 (2012)