Identifying Various Kinds of Event Mentions in K-Parser Output

Arpit Sharma, Nguyen H. Vo, Somak Aditya & Chitta Baral
School of Computing, Informatics & Decision Systems Engineering
Arizona State University
Tempe, AZ 85281, USA
{asharm73, nhvo1, saditya1, chitta}@asu.edu

Abstract

In this paper we show how our semantic parser (Knowledge Parser or K-Parser) identifies various kinds of event mentions in the input text. The types include recursive (complex) and non recursive event mentions. K-Parser outputs each event mention in form of an acyclic graph with root nodes as the verbs that drive those events. The children nodes of the verbs represent the entities participating in the events, and their conceptual classes. The on-line demo of the system is available at http://kparser.org

1 Introduction

Identifying the events mentioned in a text is an essential task for any semantic parsing system. Many Natural Language Understanding applications such as Question Answering (Berant et al., 2013; Kwiatkowski et al., 2013) and semantics-based machine translation (Bazrafshan and Gildea, 2013; Jones et al., 2012) use semantic parsers to translate both questions and answer sources into a desired representation. Several semantic parsers, both application-independent (Bos, 2008b; Allen et al., 2007; Dzikovska et al., 2003) and the ones for specific application (Berant and Liang, 2014; Fader et al., 2014; Kwiatkowski et al., 2013; Yao and Van Durme, 2014) have been developed for the assistance. However, most of them do not very effectively represent the different kinds of event mentions.

In this paper we demonstrate, with the help of examples, how our semantic parser (Knowledge Parser or K-Parser) is able to identify the semantics of various event types and output them in form of an acyclic graph.

The sections below explain, in order, the basic overview of K-Parser (along with its output), a brief explanation of various kinds of event mentions and examples demonstrating how K-Parser output is able to identify event mentions in those examples.

2 The Knowledge Parser (K-Parser)

K-Parser[1] is a semantic parser which produces a graphical semantic representation of the input text. The output of the parser is a mapping between the dependency parse of input text and the ontological relations from KM component library (Clark et al., 2004). The mapping process uses Word Sense Disambiguation and a set of rules to map syntactic dependencies to appropriate semantic relations. Furthermore, the output of the parser contains commonsense information about the words in the text i.e. the conceptual classes. For example in Fig. 1 *Barack-Obama_1* has superclass *person*. To sum up, the output of the parser has following properties:

1. An acyclic graphical representation in the form of interconnected event mentions.

2. A rich ontology (KM) to represent semantic relations (Event-Event relations such as *causes, caused_by*, Event-Entity relations such as *agent*, and Entity-Entity relations such as *related_to*).

[1]The system is available online at http://kparser.org

Proceedings of the 3rd Workshop on EVENTS at the NAACL-HLT 2015, pages 82–88,
Denver, Colorado, June 4, 2015.

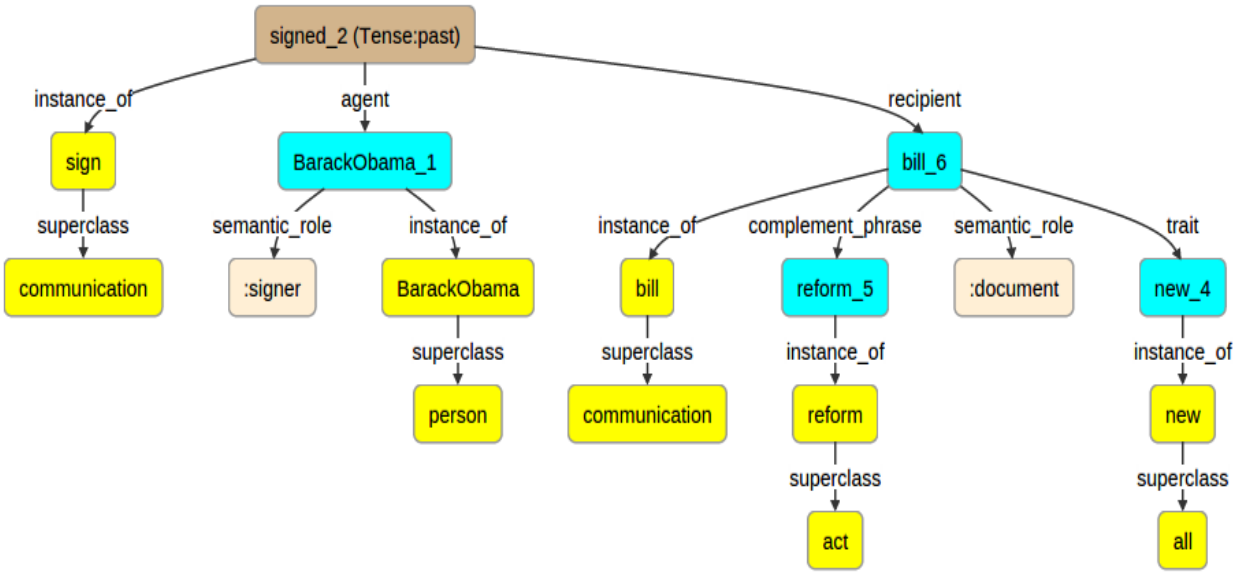

Figure 1: K-Parser output for *"Barack Obama signed the new reform bill."*

3. Special relations (*instance_of* and *prototype_of*) to represent the existential and universal quantification of entities. (For example, sentences *Every boxer walks.* and *Some boxer walks.*)

4. Conceptual class information about words in the text.

5. Semantic roles of entities (For example in sentence *John loves Mia.*).

6. Tenses of the events in the input text.

7. Other features such as an optional Co-reference resolution.

The basic algorithm of K-Parser contains five modules. The first module is used to extract the syntactic dependency graph from the input text. We used Stanford Dependency Parser (De Marneffe et al., 2006) for the purpose. The second module is used to map the syntactic dependency relations to KM relations (Barker et al., 2001; Clark et al., 2004) and a few newly created relations (such as *related_to*). There are three techniques used for semantic mapping. First, we used the rules to map syntactic dependencies into semantic relations. For example the *nominal subject* dependency is mapped to *agent* relation. Second, we developed a multi-class multilayer perceptron classifier to disambiguate different senses of prepositions and assign the semantic relations appropriately. The training data for classification is taken from "The Preposition Project" (Litkowski, 2013) and the sense ids for prepositions are manually mapped to the KM relations. The third method uses the discourse connectives in the text to label the event-event relations. Different connectives correspond to different labels. For example, the coordinate connectives such as *but*, *and*, *comma (,)* and *stop(.)* are labeled as *next_event*. Other connectives are also labeled based on their effect, such as *because* and *so* are labeled *caused_by* and *causes* respectively. The third module in K-Parser algorithm adds two level of classes for each node in the output of Semantic Mapping module. Word Sense Disambiguation (Basile et al., 2007) along with the lexical senses from WordNet (Miller, 1995) are used for this task. The fourth module corrects the mappings done by the mapping function by using class information extracted by the third module. For example, if there is a relation *is_possessed_by* between two nodes with their superclass as *person*, then the relation is corrected to *related_to* (because a person can not possess another person). Lastly, the fifth module implements other features such as semantic roles of the entities by using Propbank Framesets (Bonial et al., 2012; Palmer et al., 2005). An option for co-

reference resolution is also provided in the system which uses state of the art, Stanford Co-reference resolver (Raghunathan et al., 2010). Furthermore, many other tools are also used at various steps in the above mentioned modules, such as Named Entity Tagging, WordNet database and Weka statistical classifier library (Witten et al., 1999).

We used KM library for labeling the relationship edges between nodes in the output graph. There are 118 total relations available in KM. Out of 118, there are 24 (12 bi-directional[2]) relations that define the relationship between events. These relations are used in K-Parser to capture event-event relations. We also defined four new relations to represent some of the edge labels that were not captured in KM. These relations are *instance_of*, *superclass*, *participant* and *related_to*. The first two are used to represent two levels of conceptual class information associated with nodes in the graph. The other two relations represent special relations between an event node and an entity node.

As mentioned before, apart from recognizing event mentions, K-Parser also have other features such as conceptual classes, semantic roles and an optional co-reference resolution.

3 Event Mentions

We believe that the event mentions in the text are driven by the verbs in it. For example in Fig. 2 the left side shows the output for the phrase *Jerry and Tom*. There are no verbs in this phrase, hence no events. The right side of the figure shows the output for *Jerry and Tom were lying in the bed*. There is a verb (*lying_5*) in this sentence, hence the output shows an event graph with root as *lying_5*. In our system, we identify event mentions based on the actions or verbs found in the text. The environment of the events i.e. the subgraph with its root as a verb, is defined using the entities and attributes found in the input text. For example, the graph in Fig. 1 represents an event mention driven by the action *signed_2*.

3.1 Types of Event Mentions

There are four aspectual types of events (namely achievements, accomplishments, process or activity and states). Pustejovsky (1991) demonstrates how same verbs can be used in different types of events (see example sentences 1(a) and 2(a) in Table 1). The difference between these types is determined by the arguments of the verb. For example in 1(a), the event is an unbounded *process* whereas in 2(a) it is an *accomplishment* because of the bounding (by the phrase *to the store*). Our parser captures these arguments and hence is useful in differentiating between the types of events. Table 1 shows example sentences for these types.

Another criteria for categorizing events is based on the complexity. An event is defined recursive or complex if there exist events with other events as their arguments. For example, the sentence *The knife was used for killing the dog* has a complex event consisting of two events *used* and *killing*. The *killing* event is an argument to the *used* event. The K-Parser output for the sentence is shown in Fig. 3. The relationship between the two events is shown with an argumentative Event-Event relation i.e *objective* (see Fig. 3).

On the other hand, there is no argumentative relationship between events in the non-recursive or simple event mentions. Example sentence 6(b) in Table 1 contains two events *killed* and *ran*. These events are not arguments of each other but they are related via an ordering edge that specifies that *ran* is the event happened after *killed* event. Temporal ordering is another criteria for categorization of events. This is used to specify the order of occurrence of atomic events in a chain of events. K-Parser parser such events by using special event-event relations such as *next_event* and *previous_event*.

Example sentences of all the types are provided in Table 1. K-Parser outputs for only a few are demonstrated in this paper because of space constraints. We encourage the reader to try out all other examples in the table in the on-line demo of K-Parser, which is available at *www.kparser.org*

4 Evaluations

K-Parser is developed based on the training sentences collected from many sources such as the example sentences from stanford dependency manual (De Marneffe and Manning, 2008) and dictionary examples for sentences with conjunctions. We eval-

[2] *causes, defeats, enables, inhibits, by-means-of, first-subevent, objective, next-event, prevents, resulting-state, subevent, supports*

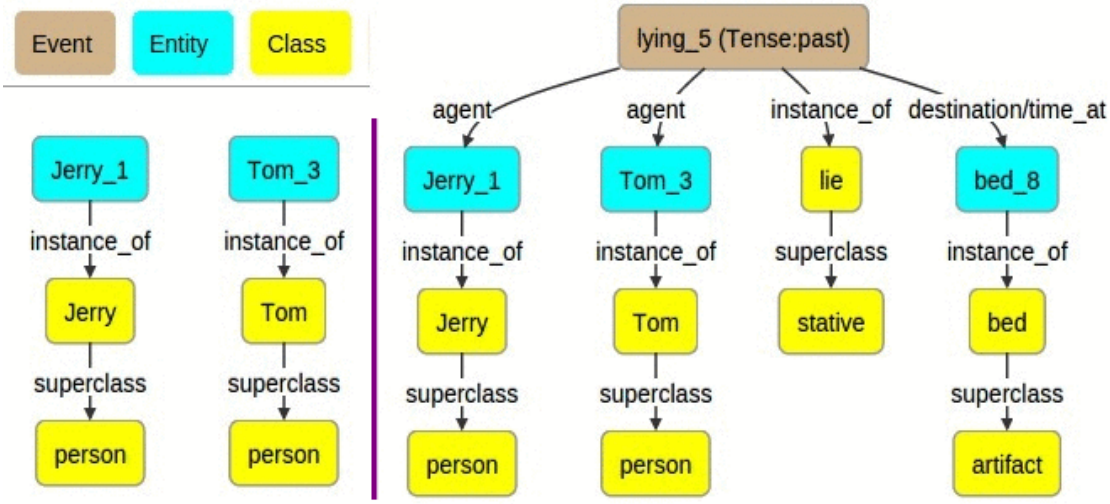

Figure 2: K-Parser output for *"Jerry and Tom"*(left) and *"Jerry and Tom were lying in the bed"*(right)

Table 1: Event Types and Example Sentences

Event type	Example Sentences
Process or Activity	*1(a) Mary walked.* *1(b) John ran.*
Accomplishment	*2(a) Mary walked to the store.*
Achievement	*3(a) Tim ran two miles.* *3(b) John arrived at his destination.*
State	*4(a) John loves Mia.* *4(b) I knew about the incident.* *4(c) He fell asleep during the meeting.*
Complex Events	*5(a) The knife was used to kill the dog.* *5(b) George was bullying Tim so we rescued him.*
Simple Events	*6(a) John loves Mia, and Mia hates John.* *6(b) Tom killed John before Tom and Jane ran away.*
Temporal Events	*7(a) Tom killed John before Tom and Jane ran away.* *7(b) She sat opposite him and looked into his eyes.*

uated the K-Parser output based on the types of events identified. This is done by manually defining gold standard representation for a corpus of 282 Winograd Schema Challenge (WSC) (Levesque et al., 2011) sentences (there is no overlap between test and training corpus). WSC is a well accepted corpus known to demonstrate complex semantics. We identified some important categories to assess the accuracy of event mentions and relations between events in the output of K-Parser. The categories are *number of Events, number of Entities, number of Classes, number of Event-Event relations* and *number of Event-Entity relations*. Each of the categories are compared with the gold standard based on measures mentioned below.

t_1 = identified and relevant and the label is correct.
t_2 = identified and relevant and the label is wrong.
t_3 = identified, but not relevant.
t_4 = not identified, but relevant.

Table 2: Evaluation Results

	Precision	Recall
Events	0.94	0.92
Entities	0.97	0.96
Classes	0.86	0.79
Event-Event Relations	0.91	0.79
Event-Entity Relations	0.94	0.89

We defined Precision and Recall of our system based on the above terms

Precision = $t_1/(t_1 + t_2 + t_3)$
Recall = $t_1/(t_1 + t_2 + t_4)$

Table 2 shows the evaluation results. We have also used the output of our system in solving a subsection of the Winograd Schema Challenge (Sharma et al., 2015).

5 Related Works

There are many semantic parsers available, such as the SEMAFOR parser (Das et al., 2010). While it assigns semantic roles to entities and verbs in the text, they lack in defining event mentions and relations between them. Furthermore, these systems do not correctly process the implications, quantifications and conceptual class information about the text (eg. *John* is an instance of *person* class). Among the others, there is Boxer system (Bos, 2008b) that translates English sentences into first order logic. Despite its many advantages, this parser fails to represent the event-event and event-entity re-

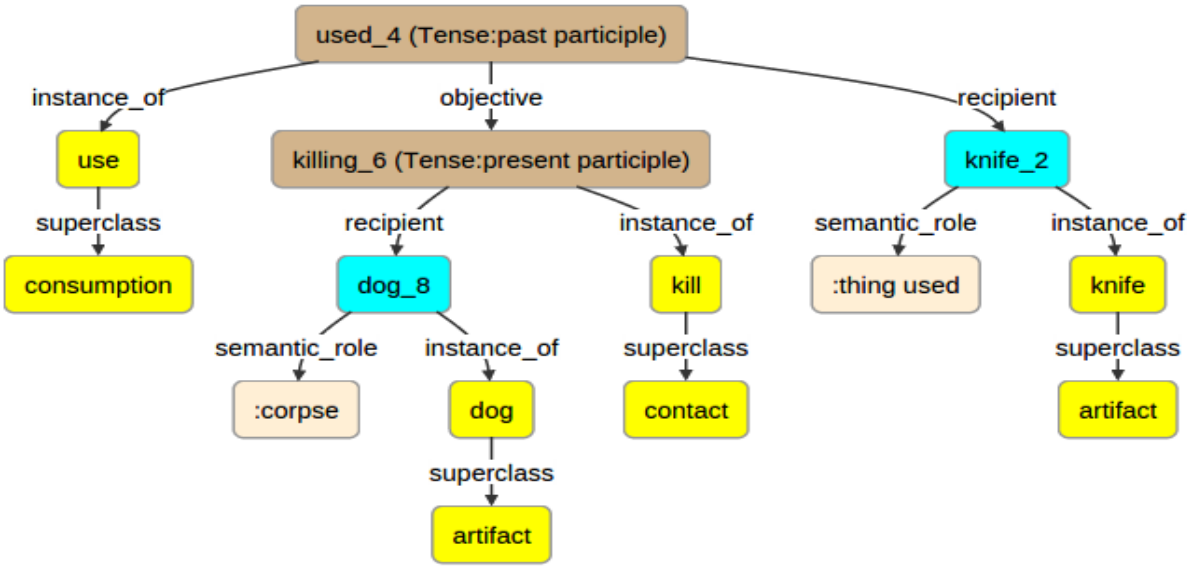

Figure 3: K-Parser output for *"The knife was used for killing the dog."*

lations in the text. The inclusion of the homonym-hypernym information and resolution of identical meaning words are important for downstream reasoning. Boxer system does not capture such ontological information about entities or similarities between connectives. Carbonell et al., (2014) presents another semantic parser that translates natural language strings into Abstract Meaning Representation (Banarescu et al., 2013). Similar to K-Parser, TRIPS (Allen et al., 2007) translates text into a semantic graph. The system encodes the features such as the conceptual classes of the words, quantification of entities and representation of the participants of an event. However, it does not have event-event relations in the text. These relations are required to specify the causality and dependency of events in a particular context. Another parser that participated in STEP 2008 shared task (Bos, 2008a) is the TEXTCAP semantic interpreter (Callaway, 2008). It translates the input text into a list of co-indexed semantic triples that represent the explicitly recoverable semantic content in the input text. Though it uses Word Sense Disambiguation on the WordNet data (like K-Parser) to extract the classes of events and entities, it does not label the specific relationship between events and their participants. For example in the sentence *My dog quickly chased rabbits yesterday.*(from TEXTCAP paper), the triple *(DOG492,CHASING141,RABBIT#n1)* represents the relation between two entities *the dog* and *the rabbits* in the form of the event *chasing*. In the output of K-Parser for the above sentence, there is an event node *chasing* which has an agent *dog* and the recipient *rabbits*. The other meaningful words in the sentence (such as quickly and yesterday) are also identified by K-Parser.

6 Conclusion

In this paper we showed how our parser i.e. K-Parser, is able to identify various kinds of events that are present in the input text. We also explained how the output of K-Parser can be further used to differentiate between the types of events (*processes, achievements, accomplishments* and *states*). Furthermore, we showed that the event mentions can be identified by extracting the verbs from the text and connecting the entities that participate in those verbs (using appropriate relations). This is an ongoing research and an on-line demo of our system is available at *www.kparser.org*.

Acknowledgements

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

References

James Allen, Mehdi Manshadi, Myroslava Dzikovska, and Mary Swift. 2007. Deep linguistic processing for spoken dialogue systems. In *Proceedings of the Workshop on Deep Linguistic Processing*, pages 49–56. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking.

Ken Barker, Bruce Porter, and Peter Clark. 2001. A library of generic concepts for composing knowledge bases. In *Proceedings of the 1st international conference on Knowledge capture*, pages 14–21. ACM.

Pierpaolo Basile, Marco Degemmis, Anna Lisa Gentile, Pasquale Lops, and Giovanni Semeraro. 2007. The jigsaw algorithm for word sense disambiguation and semantic indexing of documents. In *AI* IA 2007: Artificial Intelligence and Human-Oriented Computing*, pages 314–325. Springer.

Marzieh Bazrafshan and Daniel Gildea. 2013. Semantic roles for string to tree machine translation. In *ACL (2)*, pages 419–423.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of ACL*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, pages 1533–1544.

Claire Bonial, Jena Hwang, Julia Bonn, Kathryn Conger, Olga Babko-Malaya, and Martha Palmer. 2012. English propbank annotation guidelines. *Center for Computational Language and Education Research Institute of Cognitive Science University of Colorado at Boulder*.

Johan Bos. 2008a. Introduction to the shared task on comparing semantic representations. In *Proceedings of the 2008 Conference on Semantics in Text Processing*, pages 257–261. Association for Computational Linguistics.

Johan Bos. 2008b. Wide-coverage semantic analysis with boxer. In *Proceedings of the 2008 Conference on Semantics in Text Processing*, pages 277–286. Association for Computational Linguistics.

Charles B Callaway. 2008. The textcap semantic interpreter. In *Proceedings of the 2008 Conference on Semantics in Text Processing*, pages 327–342. Association for Computational Linguistics.

Jeffrey Flanigan Sam Thomson Jaime Carbonell and Chris Dyer Noah A Smith. 2014. A discriminative graph-based parser for the abstract meaning representation.

Peter Clark, Bruce Porter, and Boeing Phantom Works. 2004. Kmthe knowledge machine 2.0: Users manual. *Department of Computer Science, University of Texas at Austin*.

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A Smith. 2010. Semafor 1.0: A probabilistic frame-semantic parser. *Language Technologies Institute, School of Computer Science, Carnegie Mellon University*.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. Stanford typed dependencies manual. *URL http://nlp. stanford. edu/software/dependencies manual. pdf*.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Myroslava O Dzikovska, James F Allen, and Mary D Swift. 2003. Integrating linguistic and domain knowledge for spoken dialogue systems in multiple domains. In *Proc. of IJCAI-03 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012. Semantics-based machine translation with hyperedge replacement grammars. In *COLING*, pages 1359–1376.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*.

Ken Litkowski. 2013. The preposition project corpora. Technical report, Technical Report 13-01. Damascus, MD: CL Research.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

James Pustejovsky. 1991. The syntax of event structure. *Cognition*, 41(1):47–81.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 492–501. ACL.

Arpit Sharma, Nguyen H Vo, Shruti Gaur, and Chitta Baral. 2015. An approach to solve winograd schema challenge using automatically extracted commonsense knowledge. In *2015 AAAI Spring Symposium Series*.

Ian H Witten, Eibe Frank, Leonard E Trigg, Mark A Hall, Geoffrey Holmes, and Sally Jo Cunningham. 1999. Weka: Practical machine learning tools and techniques with java implementations.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of ACL*.