Kleo: A Bootstrapping Learning-by-Reading System

Doo Soon Kim and Bruce Porter
The University of Texas at Austin
Computer Science Department
{onue5, porter}@cs.utexas.edu

Abstract

KLEO is a bootstrapping learning-by-reading system that builds a knowledge base in a fully automated way by reading texts for a domain. KLEO's initial knowledge base is a small knowledge base that consists of domain independent knowledge and KLEO expands the knowledge base with the information extracted from texts. A key facility in KLEO is knowledge integration which combines new information gleaned from individual sentences of the texts, along with prior knowledge, to form a comprehensive and computationally useful knowledge base. This paper introduces the architecture of KLEO, especially the knowledge integration facility, and presents our evaluation plan.

The *knowledge acquisition bottleneck* has been the major obstacle to building large-scale knowledge bases. Despite enormous past efforts, it is still costly and tedious to build knowledge bases manually. As a solution to this problem, a new approach has been gaining much attention due to the advance of natural language processing and the proliferation of texts on the Internet. The approach is to construct a knowledge base with knowledge extracted from texts.

KLEO [1] is a such Learning-by-Reading system which operates in the following steps :

1. It reads a text to form a semantic representation. The knowledge base provides the information required to understand the text.
2. It adds the semantic representation to the knowledge base.

Kleo repeats these steps with a corpus of texts. Note that these two steps constitute a bootstrapping cycle in which reading extends the knowledge base (step2) and the extended knowledge base in turn improves the reading performance (step1). A key to this approach is knowledge integration - the task of (1) combining semantic representations for individual sentences to form a coherent representation for the text and (2) combining the new information with the prior knowledge base. Knowledge integration is an important facility in the Learning-by-Reading task because, without it, the learned knowledge base is often fragmented, incoherent and hence computationally useless.

This paper is not intended to provide technical details on the KLEO system. Rather, it will present an overview of the challenges in building a Learning-by-Reading system, along with a system architecture and evaluation plan for KLEO, focusing on the Knowledge Integration facility.

History of Kleo

KLEO is a descendent of MÖBIUS (Barker *et al.* 2007), which was an early proof-of-concept Learning-by-Reading system. The original MÖBIUS system was built during a short period (6 month) by assembling off-the-shelf components including a parser from ISI [2] and, a semantic interpreter, knowledge base, and a question-answering facility from University of Texas at Austin [3]. Then, with an expanded team of researchers from BBN and Boeing, MÖBIUS was further extended with a new parser and a rudimentary form of a bootstrapped learning.

The key differences between KLEO and MÖBIUS are :

1. All the components in KLEO are publically available, and therefore KLEO can be freely distributed, whereas some components of MÖBIUS are proprietary.
2. KLEO attempts to learn by reading a sequence of texts, whereas MÖBIUS learns from single texts. To support multi-text reading, KLEO provides new functions such as (a) integrating information gleaned from multiple texts and (b) using knowledge from previous reading to interpret subsequent texts.

Challenges in Knowledge Integration

This section presents challenging issues in knowledge integration for the Learning-by-Reading task.

- **Aligning semantic representations** An important task in knowledge integration is aligning representations correctly, which is at the core of many AI problems such as ontology alignment and model-based reasoning. It entails several challenging problems such as identifying coreferenced entities, resolving granularity differences, and resolving viewpoint differences. For example, in fig. 1,

[1] The web interface to KLEO is available at http://www.cs.utexas.edu/users/onue5

[2] http://www.isi.edu/natural-language/index.html

[3] http://www.cs.utexas.edu/users/mfkb

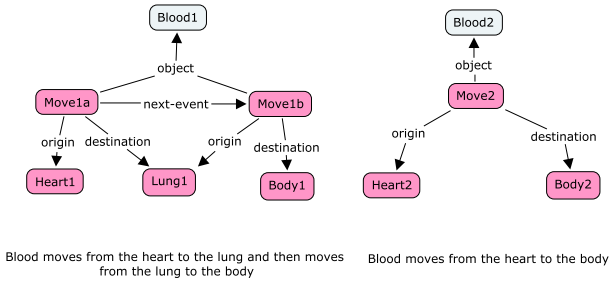


Figure 1: Two representations which express the same knowledge at different levels of granularity

combining the two representations requires identifying that the concepts, BLOOD, HEART, and BODY are co-referred in the two representations and that the two representations express the same knowledge at different levels of granularity.

- **Implicit information** A text omits information that readers can easily infer. It is important to identify such unspecified information and represent it explicitly because a knowledge base built without such explicit representations would be incomplete and fragmented. KLEO addresses this problem by adding knowledge from the knowledge base to the individual sentential representations, thereby filling the gaps in the text. This process entails identifying the right knowledge structures from the knowledge base that are crucial to understanding the text and aligning it with the individual sentential representations.
- **Uncertainty management** Natural Language Understanding is inherently an abductive process, and hence it is challenging to identify a correct interpretation out of many. This challenge is somewhat alleviated in multi-text reading, because knowledge extracted from one text can help inform the interpretation of the next. Therefore, it is an important research issue on knowledge integration to devise an algorithm that narrows down to a correct interpretation given evidence from multiple texts.
- **Managing a large-scale knowledge base** It is challenging to manage a large-scale knowledge base because (1) accessing and updating the large knowledge base tends to be inefficient, and (2) even a small change to the knowledge base may require a significant modification to the whole knowledge base.

Architecture

KLEO has a pipeline architecture in which the following components are serially connected: parser, semantic interpreter, sentence-to-sentence knowledge integrator (S2S-KI), and text-to-text knowledge integrator (T2T-KI) (fig. 2). Given a text, the parser and the semantic interpreter build sentence-level semantic forms. S2S-KI integrates these forms into a coherent representation of the whole text, and T2T-KI integrates that representation with the knowledge base.

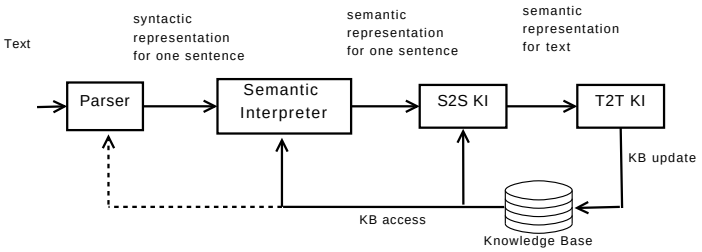


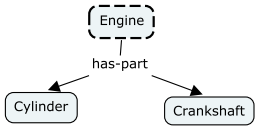
Figure 2: Architecture of KLEO: The dotted line from the knowledge base to the parser indicates that the current parser in KLEO does not use the knowledge base. It is our future plan to make the parser exploit the knowledge base.

The lines in fig. 2 represent the data flow, indicating that the semantic interpreter and S2S-KI uses the knowledge base to interpret the texts (see the arrow labeled *KB access*) and T2T-KI updates the knowledge base with the new information from reading (see the arrow labeled *KB update*). Notice that these two arrows form a cycle indicating that the knowledge base is extended in a bootstrapping manner.

Knowledge base

KLEO's initial knowledge base is the upper ontology of the Component Library (Barker, Porter, & Clark 2001), which consists of about 700 generic concepts such as MOVE and CONTAINER and about 75 relations including temporal, spatial, structural, partonomic and causal relations.

KLEO's knowledge base consists of a set of structures called K-units; each one is a coherent set of triples that represent a single concept (e.g. engine, combustion). In contrast, the knowledge base used in other Learning-by-Reading systems such as MÖBIUS (Barker *et al.* 2007) and TEXTRUNNER (Banko *et al.* 2007) consists of collection of triples, representing knowledge of many concepts. A K-unit has a root node, which is universally quantified. The other nodes in the K-unit are existentially quantified in the scope of the root node. For example, the following K-unit represents $\forall x.Engine(x) \rightarrow \exists y, z.Cylinder(y) \wedge Crankshaft(z) \wedge has\text{-}part(x, y) \wedge has\text{-}part(x, z)$



The advantage of K-units is in the process of knowledge integration - matching representations of a new text with the knowledge base (This process is described more fully in the section titled *Text-to-Text Knowledge Integrator*). Because most of the knowledge base is irrelevant to the text, it is inefficient to consider the whole knowledge base during integration. In contrast, KLEO focuses its integration effort on only K-units referenced by the text, ignoring irrelevant portions. This enables the fast and efficient update of the knowledge base.

The next subsections explain the components of Kleo in detail with an example, showing how a sentence is processed in each component. Let's suppose that KLEO attempts to read two consecutive sentences:

S1: Combustion of gasoline occurs inside the engine's cylinders.

S2: The engine's piston compresses the gasoline.

Parser

KLEO uses the Stanford Parser (Klein & Manning 2003), which is fast, and provides broad-coverage and high accuracy. The parser produces a dependency relation parse, which is easier to convert to the corresponding semantic representation than a phrase structure parse. The following is the parsing result for S1. [4]

```
(occurs-5 nsubj Combustion-1)
(gasoline-4 det the-3)
(Combustion-1 prep_of gasoline-4)
(engine-8 det the-7)
(cylinders-10 poss engine-8)
(occurs-5 prep_inside cylinders-10)
```

Semantic interpreter (SI)

KLEO's semantic interpreter is derived from the one used in MÖBIUS (Barker *et al.* 2007) and further extended as follows.

For each word in the dependency triples - that is, the first and the third items of the dependency triples-, the semantic interpreter identifies a concept from the Component Library (Barker, Porter, & Clark 2001) in the following manner:

1. For each synset of the word, SI identifies a corresponding concept from the Component Library. This is performed using the mappings from the Component Library to synsets in Wordnet (The mappings were manually constructed as a part of the Component Library). If the synset does not have a mapping to a concept [5], SI climbs up the hypernym hierarchy in Wordnet until it finds a synset that has a corresponding concept in the Component Library.
2. Out of the candidate concepts for each synset, SI chooses the best one based on several factors. For example, one factor is the specificity of the candidate concepts as measured by the distance between the concept and the root concept of the Component Library.

If a word occurs frequently in the text, KLEO regards it as an important word, which is worth reifying as a new concept in the knowledge base. For such words, SI creates a new class named after the word as a subclass of the assigned concept.

Another important task in SI is to convert dependency relations into conceptual relations defined in the Component Library. Currently, this is performed by manually written rules. For example, the rules convert (Compresses nsubj Piston) into (Compress instrument Piston).

The semantic interpreter handles a variety of linguistic forms, including:

- Auxiliary verbs : SI identifies a main verb and replaces the auxiliary verb with the main verb. e.g. For S1, SI identifies that *combustion* is a main verb and *gasoline* is its semantic object.
- Noun-noun compound : SI identifies a semantic relation between the head noun and the modifier noun. e.g. *combustion engine* is translated into (Combustion site Engine)
- Causal verbs : SI identifies a causing event and a caused event and connects them through a causal relation. e.g. For "*Combustion of gasoline causes an automobile to move.*", SI produces (Combustion **causes** Move)
- Definitional phrases : SI identifies a subsumption relation through pre-defined lexico-syntactic patterns such as *A such as B, A called B, A is B, A known as B*, appositive.

On the dependency parse for the example sentence,

- SI assigns a concept, DEVICE, both to engine-8, and cylinders-10. Also, by assuming that the words occur frequently, SI creates new classes, ENGINE and CYLINDER, as subclasses of DEVICE.
- SI assigns COMBUST to combustion-1. Because occurs-5 is an auxiliary verb, combustion-1 becomes a main verb.
- SI converts dependency relations, *poss* and *prep_inside*, into *has-part* and *site*, respectively.

and produces

```
(Engine superclasses Device)
(Cylinder superclasses Device)
(Combustion-1 instance-of Combust)
(engine-8 instance-of Engine)
(cylinders-10 instance-of Cylinder)
(Combustion-1 object gasoline-4)
(engine-8 has-part cylinders-10)
(Combustion-1 site cylinders-10)
```

Sentence-to-Sentence Knowledge Integrator (S2S-KI)

S2S-KI forms a coherent representation by integrating individual representations from each sentence. This representation for the text is more useful than a simple aggregation of individual representations because the coherently organized knowledge structure can enable high-level reasoning.

S2S-KI integrates knowledge fragments in two steps: *Stitch* and *Elaborate*. *Stitch* identifies mappings between the knowledge fragments, and *Elaborate* adds additional knowledge structures from the knowledge base to the knowledge fragments.

Stitch *Stitch* aligns two knowledge fragments by graph matching. The matcher used in KLEO is more advanced than traditional matchers in the following ways: (1) It can resolve granularity mismatches between the knowledge fragments

[4] The number in a variable represents the location of the variable in the sentence. For example, occur-5 represents that the variable is from the 5th word from the beginning.

[5] The mappings exist for all concepts in Component Library, not for all synsets in Wordnet.

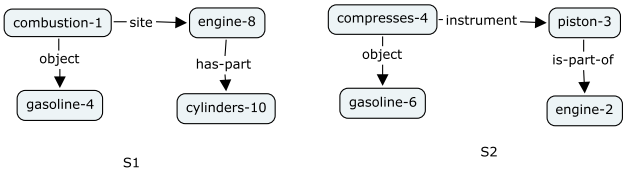


Figure 3: The semantic representations for S1 and S2

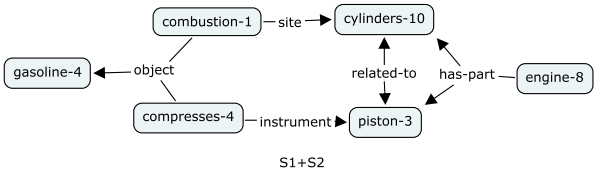


Figure 4: stitched representation

and (2) it abduces assumptions if the assumptions lead to discovering more mappings. The next section describes the graph matcher in detail.

For S1 and S2 in the example (shown in fig. 3 as a graph representation), *Stitch* aligns the two representations as shown in fig. 4 by identifying

- engine-8 co-references engine-2 and gasoline-4 co-references gasoline-6.
- cylinders-10 and piston-3 are related to each other.

Elaborate A text omits a great deal of information that human readers would easily infer. It is important to explicitly represent the unspecified information in a representation for a text to establish coherence among the sentences. For this task, *Elaborate* draws a knowledge structure from the knowledge base, and aligns it with the sentential representations.

KLEO uses spreading activation through the knowledge base to build the knowledge structure. The activation starts from the concepts references in the sentential representations. For example, let's suppose that the knowledge base contains the representation shown in fig. 5. For S1 and S2 in fig. 3, activation starts from ENGINE, CYLINDER, PISTON, COMPRESS, and COMBUST and spreads through the italicized parts of fig. 5. Currently, we use a simple termination condition whereby activation stops when it meets another activation (success) or it travels a certain distance (failure). The italicized parts are returned as a knowledge structure aligned with the representation in fig. 4. Fig. 6 shows the representation after *elaborate*.

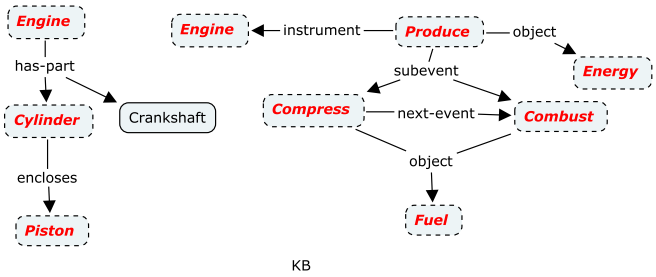


Figure 5: The italicized nodes denote a knowledge structure added to the representation in fig. 4

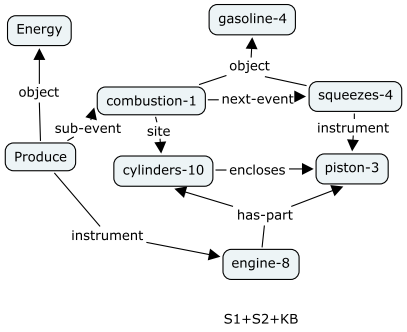


Figure 6: The representation produced from S2S-KI

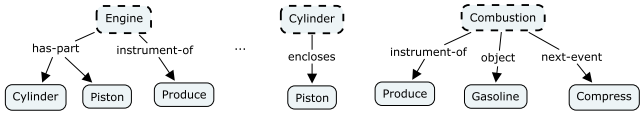


Figure 7: K-units resulting from partitioning of the representation in fig. 6. Due to the limitation of the space, we show only some of them.

Text-to-Text Knowledge Integrator (T2T-KI)

T2T-KI updates the knowledge base with the knowledge structure produced by S2S-KI. T2T-KI first partitions the knowledge structure into several K-units. As we stated earlier, this partitioning makes management of knowledge base efficient (e.g. fast update and retrieval of knowledge base) The partitioning begins with selecting nodes in the knowledge structure that can serve as root nodes. The heuristic used in KLEO is to select a node that is an instance of a new class or an event. Once the root nodes are identified, for each root node, T2T-KI performs spreading activation from the root node until the activation arrives at another root node. The knowledge structure which the activation explored becomes a K-unit for the root node. Fig. 7 shows some of the K-units resulting from partitioning the textual representation in fig. 6.

Then, for each K-unit, KLEO identifies a K-unit from the knowledge base that is relevant. Two K-units are relevant to each other if (1) their root nodes are instances of the same entity or (2) their root nodes are instances of an EVENT and their case roles are aligned to each other. Then, using the graph matcher, the two K-units are integrated.

Graph-matching: Aligning two representations

This section presents the technical detail of our graph matcher because the matcher is an important piece in S2S-

KI and T2T-KI, even if the paper is intended to be a general overview on KLEO.

Our graph matcher operates in two steps: (1) It first identifies seed nodes in the two graphs that can be mapped to each other, and (2) from the pairs of seed nodes, the matcher extends the mappings to identify more.

Unlike traditional matchers, our matcher can resolve granularity differences. It is important both in S2S-KI and T2T-KI to resolve granularity because representations can differ in their level of detail: (1) between two sentences and (2) between a text and a knowledge base. For example, a major source of granularity mismatches is between the topic sentence and the body of the paragraph. In general, the body is a fine-grained representation of the topic sentence. Also, the granularity of knowledge structures in a knowledge base could be different from the one of the text.

Identifying seed nodes

The matcher uses two heuristics to identify seed nodes. In the description of the heuristics, let *node1* be a node from the first input graph, and *node2* from the second.

Heuristic 1 *Node1 is mapped with node2 if both originate from the same noun.*

This heuristic works well because, within a paragraph, when the same word is used twice, it usually references the same object. (Grice 1975)

Heuristic 2 *Node1 is mapped with node2 if node1 and node2 are an event of a same type and their case roles are aligned to each other.*

As an example of heuristic2, consider two sentences, *"Fuel is sucked into an engine"* and *"Engine takes in gasoline"*. Assume that, for the first sentence, the semantic interpreter produces (Move1 object Fuel1) (Move1 destination Engine1) and, for the second, (Move2 object Gasoline2) (Move1 destination Engine1). Because both Move1 and Move2 are Move events and their case roles are aligned to each other - Fuel1 with Gasoline2, Engine1 with Engine2 - heuristic2 chooses (Move1, Move2) as a pair of seed nodes. It is important to check the case roles because events are often generic. For example, for two sentences, *"Piston moves"* and *"Engine moves a car"*, the two Move events are different because their case roles are different.

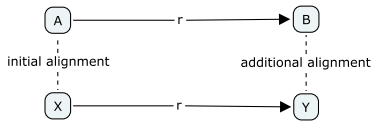
State-of-the-art methods for resolving pronouns and definite noun phrases could identify more seed nodes. We plan to incorporate these methods in the future.

Extending mappings

The mappings are further extended by several rules that we call *extension patterns*. In the description of the patterns, G1 and G2 refer to the two input graphs, and A and X are nodes in G1 and G2 that are already mapped to each other.

Pattern 1 (simple alignment) *There are triples (A r B) in G1 and (X r Y) in G2 such that A is taxonomically aligned with X. Then, B is mapped with Y.*

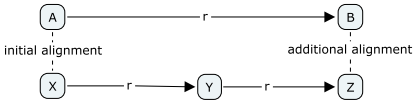
This pattern has been used in many traditional graph matchers to align, for example, (Human1 has-part Limb1)



with (Human2 has-part Arm2)

Patterns 2 through 6 resolve several types of granularity mismatches. Among them, patterns 4 through 6 introduce assumptions without which the alignment would fail.

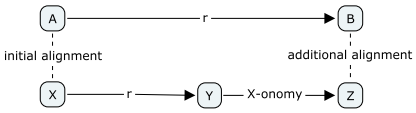
Pattern 2 (transitivity-based alignment) *There are triples (A r B) in G1 and (X r Y) (Y r Z) in G2 such that B is taxonomically aligned with Y and r is a transitive relation, such as causes, next-event, etc. Then, B is mapped with Z.*



A transitive relation often causes a granularity mismatch such as (Human1 has-part Face1) (Face1 has-part Nose1) and (Human2 has-part Nose2). Pattern 2 can resolve this type of granularity mismatch.

The following pattern uses a relation called X-onomy. X-onomy is a general relation that includes all relations that involve hierarchy, such as has-part and has-region (partonomy), isa (taxonomy), contains, etc.

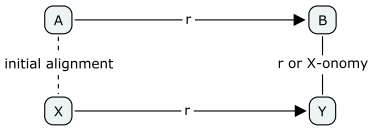
Pattern 3 (Co-reference across a granularity shift) *There are triples (A r B) in G1 and (X r Y) (Y X-onomy Z) in G2 such that B is taxonomically aligned with Z. Then, B is mapped with Z.*



This pattern handles a case in which two expressions reference the same entity at different granularities. For example, let's suppose that two representations (Move1 object Gasoline1) (Move1 destination Engine1) (*"The engine takes in gasoline"*) and (Move2 object Gasoline2) (Move2 destination Cylinder2) (Engine2 has-part Cylinder2) (*"The cylinder in the engine takes in gasoline"*) should be aligned to each other. This pattern can align the two representations even though the destination of the two TAKE-IN events are different in their level of detail. In this case, X-onomy is a *has-part* relation.

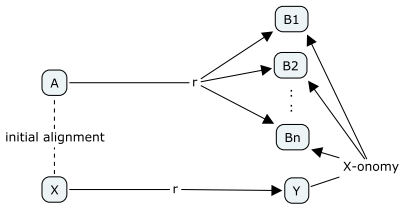
The following patterns introduce assumptions.

Pattern 4 (Abductive transfer-thru alignment) *There are triples (A r B) in G1 and (X r Y) in G2 such that B is not taxonomically aligned with Y. Then, X-onomy can be abduced between B and Y. Additionally, if r is a transitive relation, r can be abduced between B and Y.*



Note that pattern 4 is an abductive version of pattern 2 and pattern 3. For example, given two sentences, *"Gasoline moves to an engine"* and *"Gasoline moves to a cylinder"*, it abduces an assumption that Engine and Cylinder are in a X-onomy relationship.

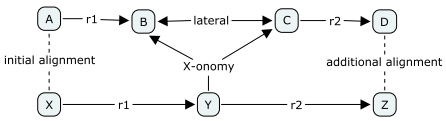
Pattern 5 (Generalization-based alignment) *There are triples (A r B1) .. (A r Bn) in G1 and (X r Z) in G2 such that each of B1, ..., Bn is aligned with Y. Then, a X-onomy relation is abduced between Bi and Y for i = 1,2, ..., n.*



This pattern handles a type of granularity mismatch in which several pieces of similar information in a fine-grained representation are generalized together to form a coarse-grained representation. For example, given two representations, (Human1 has-part Arm1) (Human1 has-part Leg1) and (Human2 has-part Limb2), this pattern abduces an assumption that Arm1 and Limb2 are in a X-onomy relationship and that Leg1 and Limb2 are in a X-onomy relationship.

The following pattern uses a relation called *lateral relation*. *Lateral relation* is a general relation that connects concepts at the same level of detail (e.g. next-event, beside, etc.).

Pattern 6 (Abstraction-based alignment) *There are triples (A r1 B) (B lateral-relation C) (C r2 D) in G1 and (X r1 Y) (Y r2 Z) in G2. Then D is mapped with Z, and two X-onomy relations are abduced between Y and B and between Z and C.*



The last pattern handles the most common case of granularity mismatch in which an aggregation of small things, whether entities or events, is viewed as one thing. For example, in fig. 1, this pattern can align the two representations by abducing an assumption that Move2 is in a X-onomy relationship with Move1a and Move1b.

All these patterns are inherently uncertain. Thus, it is possible to produce an incorrect representation using a pattern even if the precondition of the pattern holds true. Part of our future research is to deal with the uncertain nature of *knowledge integration*.

Evaluation Plan

We plan to evaluate three hypotheses for KLEO.

Hypothesis 1 *The representation formed by Kleo is both coherent and faithful to the original text. (evaluation on S2S-KI)*

Hypothesis 2 *The knowledge base formed by Kleo is computationally useful (evaluation on T2T-KI)*

It is hard to evaluate the coherence of a representation (hypothesis1) and the quality of the knowledge base (hypothesis2) automatically, because it involves human judgement. Instead, we plan to evaluate those representations through tasks such as question-answering.

Hypothesis 3 *Kleo's reading performance improves over time (evaluation on Kleo's bootstrapping capability)*

The reading performance can be measured by several metrics on the natural language components such as the accuracy of parsing, semantic interpretation and S2S-KI.

Finally, we plan to evaluate the performance of KLEO as an end-to-end Learning-by-Reading system by comparing the knowledge base produced by KLEO with the ones produced by knowledge engineers or other Learning-by-Reading systems in terms of how they perform on a question-answering task.

Acknowledgments

This research was funded by DARPA under contract NBCHD030010.

References

Banko, M.; Cafarella, M. J.; Soderland, S.; Broadhead, M.; and Etzioni, O. 2007. Open information extraction from the web. In Veloso, M. M., ed., *IJCAI*, 2670–2676.

Barker, K.; Agashe, B.; Chaw, S. Y.; Fan, J.; Glass, M.; Hobbs, J.; Hovy, E.; Israel, D.; Kim, D. S.; Mulkar, R.; Patwardhan, S.; Porter, B.; Tecuci, D.; and Yeh, P. 2007. Learning by reading: A prototype system, performance baseline and lessons learned. In *Proceedings of Twenty-Second National Conference on Artificial Intelligence*.

Barker, K.; Porter, B.; and Clark, P. 2001. A library of generic concepts for composing knowledge bases. In *Proceedings of the international conference on Knowledge capture*, 14–21. ACM Press.

Grice, H. P. 1975. Logic and conversation. In Cole, P., and Morgan, J. L., eds., *Syntax and Semantics*, volume 3: Speech Acts. New York: Academic Press. 41–58.

Klein, D., and Manning, C. D. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*.