In The Fifth International Conference on Knowledge Capture (KCAP2009) 2009.

Knowledge Integration Across Multiple Texts

Doo Soon Kim
Dept. of Computer Science
University of Texas
Austin, TX, 78712
onue5@cs.utexas.edu

Ken Barker
Dept. of Computer Science
University of Texas
Austin, TX, 78712
kbarker@cs.utexas.edu

Bruce Porter
Dept. of Computer Science
University of Texas
Austin, TX, 78712
porter@cs.utexas.edu

ABSTRACT
One of the grand challenges of AI is to build systems that learn by reading. The ideal system would construct a rich knowledge base capable of automated reasoning. We have built a Learning-by-Reading system and this paper focuses on one aspect of it: the task of integrating together snippets of knowledge drawn from multiple texts to build a single coherent knowledge base. Our evaluation shows that our approach to the knowledge integration is both feasible and promising.

Categories and Subject Descriptors
I.2.6 [**Artificial Intelligence**]: Learning—*Knowledge Acquisition*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Discourse*

General Terms
Algorithms

Keywords
knowledge acquisition, knowledge integration, natural language understanding, learning-by-reading

1. INTRODUCTION
An ideal solution to the problem of knowledge capture is to build systems that automatically construct a knowledge base by reading texts. However, this requires solving another formidable problem: Natural Language Understanding. NLU is the AI task of reading texts to build a formal representation of their content in order to support a variety of tasks, such as automated reasoning and question answering. Despite considerable progress, full NLU remains challenging, largely because of the ambiguity of natural language and the omission of information that readers easily infer.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
K-CAP'09, Septemper 1–4,2009,Redondo Beach, California, USA.
Copyright 2009 ACM 978-1-60558-658-8/09/09 ...$10.00

Nevertheless, there has been encouraging progress on the subtasks of NLU, including parsing and semantic interpretation. Much of this progress is due to corpus-based, statistical methods that nicely handle ambiguity and uncertainty. We believe the time is ripe to tackle the challenge of NLU afresh.

We bring to this challenge a new approach which is based on our experience with the Learning by Reading project [3][1] . Rather than trying to build systems that extract the *full* content of a text, our goal is to build systems that extract a partial understanding from *many* texts (all on the same topic), then integrates these into a single, coherent knowledge base. The advantage of this approach, when applied to a corpus of texts with redundant content, is obvious: it allows a system to extract only those text segments that the system can interpret with high confidence.

This new approach partially shifts the burden of NLU from language processing tasks, which are known to be hard, to a new AI challenge: *knowledge integration*, the task of combining fragments of information drawn from multiple texts – along with background knowledge – into a single coherent knowledge base. Our preliminary results indicate that the knowledge integration task is tractable and worth investigating further.

Mobius [3], a prototype Learning-by-Reading system, showed the importance of knowledge integration. It was able to extract, on average, about 30% of the information content of texts. However, the resulting knowledge base was highly fragmented – a large set of unrelated triples. We have considerable experience manually building knowledge bases that support sophisticated reasoning and question answering (e.g. Halo [7]); we can attest that the Mobius-built knowledge base is too fragmented to be useful for such tasks.

Based on Mobius, we built a new experimental Learning-by-Reading system, Kleo, which has a sophisticated

[1]For this and other insights, we appreciate the lively discussions with our team members on the Learning by Reading project: Noah Friedland, Jerry Hobbs, Ed Hovy, Paul Martin, Ralph Weischedel and David Israel.

knowledge integration facility. This paper presents Kleo's knowledge integration component. Because the KI component is the primary difference between Mobius and Kleo, we can evaluate the contribution of knowledge integration in Kleo by comparing the two systems' ability to read a corpus of texts.

It is important to note that our goal in building Kleo is to extract from texts the domain-specific, rich representations (graphical representations). Consequently, Kleo is unlike other information extraction systems [1] that extract relatively simple facts (e.g. relations between two entities) by skimming millions of open-domain documents.

2. NL PROCESSING IN KLEO

Kleo consists of two main components: the NL component, which produces semantic representations for the sentences in the texts, and the KI component, which combines the semantic representations produced by the NL component.

This section briefly describes Kleo's NL component (see [9] for details) to provide background of Kleo's KI. Given a text, Kleo parses individual sentences into dependency parses using the Stanford Parser [10]. A dependency parse is a set of triples in which the first and the third elements are words occurring in a sentence and the second is a syntactic relation between the two words (e.g. `(move nsubj ball)` [2] for *"A ball moves"*). From these syntactic triples, Kleo produces semantic triples in which the words in the first and the third position are assigned ontological types and the second element expresses a semantic relation between the two words. For example, the left figure in fig. 1 shows a semantic representation for the sentence *"The engine's piston compresses the gasoline."*. The representation indicates that `piston-4` is a kind of DEVICE and is the *instrument* of `compresses-5`.

The ontological types and the semantic relations are drawn from a formal ontology, the Component Library [2], which defines about 800 upper-ontology concepts and 80 semantic relations. Because these concepts and relations are domain-independent, they can be used to represent knowledge of any domain. Our experiment [3] shows that the NL component extracts about 30% of useful information, on average, from a single document. [3]

3. KNOWLEDGE INTEGRATION IN KLEO

Kleo and Mobius perform two types of knowledge integration: sentence-to-sentence knowledge integration (SKI) and text-to-text knowledge integration (TKI). SKI integrates semantic representations for individual sentences within a text to create a representation for the text, and TKI assimilates the representation of the text into the knowledge base of a domain.

[2](X nsubj Y) means that a noun, Y, is a subject of X.

[3]The experiment was performed on Mobius whose NL component is almost equal to the one in Kleo

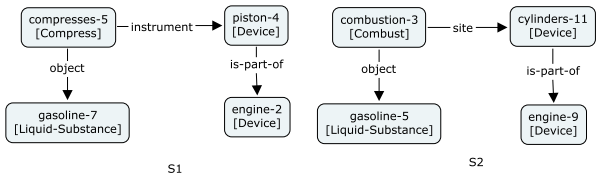


Figure 1: The semantic representations for S1 and S2

Kleo's knowledge integration capability is considerably more sophisticated than Mobius in three ways: First, Kleo is able to integrate multiple representations of an entity or event, even when the representations differ in their level of detail. Second, Kleo uses more background knowledge than Mobius in that it uses knowledge it learned from reading previous texts. Third, Kleo is considerably more scalable than Mobius in that it can combine knowledge from many more texts.

This section presents the KI capabilities in Kleo, using the example sentences below. The semantic representations for these sentences are shown in fig. 1. We skip the explanation of how the NL component produces these representations because it is not the focus of this paper.

S1: The engine's piston compresses the gasoline.
S2: Then, combustion of gasoline occurs inside the engine's cylinder.

The core component of SKI and TKI in Kleo is a graph matcher, which is described in Section 4. In addition, SKI uses the knowledge base – the one being built by Kleo – to provide background information for a current text, and TKI uses a technique called partitioning which addresses the scalability issue.

3.1 Sentence-to-Sentence Knowledge Integration

The goal of SKI is to produce representations of a text (text-KR) that make explicit the connections among the entities and events mentioned in the text. These connections are often implicit in the text, but are essential for capturing its coherence. To assess how well SKI identifies these connections, we will measure the improvement it causes in the density (connectivity) of the graphical representations of the text.

Initially, the text-KR consists of only the representation of the first sentence. As Kleo reads each subsequent sentence, SKI integrates the representation of the sentence with its text-KR. SKI consists of two steps: *Stitch* and *Elaborate*. *Stitch* aligns the representation of each sentence with the text-KR, and *Elaborate* makes explicit its implied content.

3.1.1 Stitch

Stitch aligns two semantic representations by graph matching, as described in Section 4 below. The process starts by resolving some direct anaphoric references among the sentences (using simple heuristics), then the matcher aligns the whole graphical representations, which includes the step of merging co-referential events that are described in different sentences.

The matcher combines S1 with S2 in fig. 1 using the following inferences: (1) `engine-8` is joined [13] with `engine-2` and `gasoline-4` is joined with `gasoline-6` (because they derive from the same word class); (2) `cylinders-10` is related to `piston-3` (because they are parts of the same engine); (3) *"then"* implies that `combustion-1` occurs after `compresses-4`. See fig. 2 for the result of this combination. These inference steps are described further in Section 4.

Stitch also performs a simple mapping (similar to [4]) from cue phrases to semantic relations between the consecutive sentences. For example, *"because"*, *"and so"* are mapped to a casual relations, and *"then"*, *"after"*, *"before"* are mapped to a temporal relations.

3.1.2 Elaborate

To produce a complete and coherent representation for the text, the omitted information in the text should be represented explicitly. Kleo does this by augmenting the representation produced by *Stitch* with relevant background knowledge. The augmentation may relate together two or more nodes in the graphical representation with semantic paths, which could resolve indirect anaphora and add temporal and causal relations among the events mentioned in the text.

Kleo draws background information from two sources. One is the Component Library which provides rich representations of general events and entities, such as ENTER and CONTAINER. For example, if Kleo reads *"fuel enters the cylinder"*, it draws on knowledge of ENTER to infer that a cylinder is a container (the base of all ENTER events) and that the fuel passed through a portal of the cylinder enroute from the outside of the cylinder to the inside.

The other source is information that Kleo learned by previously reading other texts on the same topic (i.e. the knowledge base constructed with the previous texts). This provides a bootstrapping capability in which Kleo exploits pre-learned knowledge.

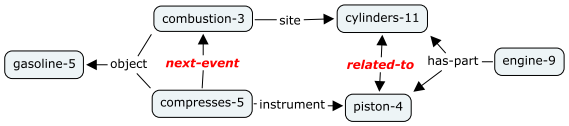


Figure 2: The *stitched* representation

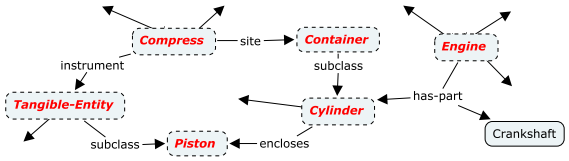


Figure 3: The example knowledge base

Because the information sources contain a vast amount of background knowledge, it is challenging to retrieve only the information that is relevant to the text. Kleo meets this challenge by performing spreading activation through the information sources. Specifically, the activation starts from the concepts referenced in the representations produced by *Stitch*. For example, suppose that the knowledge base - i.e. the Component Library or pre-learned knowledge - contains the representation shown in fig. 3. For the representation in fig. 2, activation starts from ENGINE, CYLINDER, PISTON, and COMPRESS which are referenced by the types of engine-9, cylinders-11, piston-4 and compresses-5. Currently, we use a simple termination condition whereby activation stops when it meets another activation (success) or it travels a certain distance (failure). Thus, the italicized parts - the region explored by the spreading activation (*"The compression occurs inside the cylinder"* and *"The cylinder encloses the piston"*) - are returned as background information and aligned with the representation in fig. 2 using the graph matcher. Fig. 4 shows the resulting representation.

3.2 Text-to-Text Knowledge Integration

The goal of TKI is to combine knowledge extracted from multiple texts into a single knowledge base. Initially, the knowledge base consists of only the representation of the first text. As Kleo reads each subsequent text, its representation is integrated with the knowledge base.

Typically, the texts have overlapping content. TKI attempts to identify the points of overlap, which is challenging because the texts differ in their presentation. Commonly, for example, the texts present overlapping content, but at different levels of granularity.

The graph matcher (in Section 4) is the main component of TKI. However, before it can be applied, we must address an efficiency concern because TKI attempts to match graphs that are large. (Contrast this with SKI, which attempts to match graphs that represent mere

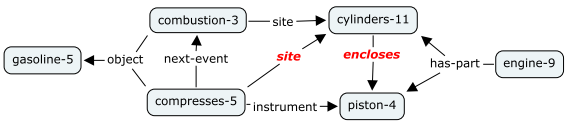


Figure 4: The *elaborated* representation

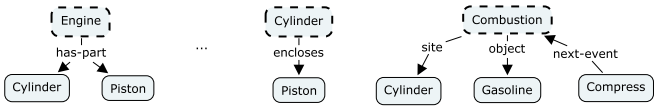


Figure 5: K-units produced from the representation in fig. 4 by partitioning. The dotted nodes are roots.

sentences). Our approach is to partition the graphs – and the background knowledge base that will be used to align them – into a set of coherent subgraphs, that we will call K-units. Fig. 5 shows examples of K-units for ENGINE, CYLINDER, and COMBUSTION.

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 CYLINDER K-unit in fig. 5 represents $\forall x.Cylinder(x) \rightarrow \exists y.Piston(y) \wedge encloses(x, y)$. Notice that the K-units are implicitly associated to one another because the root nodes are universally quantified. For example, CYLINDER in the ENGINE K-unit in fig. 5 is connected to the root node of the CYLINDER K-unit.

TKI partitions learned knowledge into K-units with the following procedure: Given a text-KR from SKI (e.g. the representation in fig. 4), TKI chooses nodes that can serve as root nodes - typically, key concepts in the domain of the texts. The heuristic used by TKI is selecting nodes whose input words frequently appear in the texts or nodes that are an instance of an EVENT because frequently occurring words or events are generally important concepts. Once the root nodes are identified, TKI performs spreading activation from each root node until the activation arrives at other root nodes. The region explored by the activation becomes a K-unit for the root node. Fig. 5 shows the K-units resulting from partitioning the representation in fig. 4.

For each new K-unit, Kleo retrieves a relevant K-Unit from the knowledge base. Two K-units are relevant if (1) their roots are instances of the same entity or (2) their root nodes are events and their case roles are taxonomically related. The two relevant K-units are aligned by the graph matcher and then the new integrated K-unit is stored in the knowledge base.

4. GRAPH MATCHING

The central issue in both SKI and TKI is combining multiple representations of knowledge into a coherent whole. The basic operation is graph join [13], but complications commonly arise because the representations do not align perfectly. Our graph matcher attempts to resolve mismatches.

Based on our earlier "semantic matcher" [15] that handles a variety of common cases, such as the transitivity of causality, the graph matcher in Kleo handles a common source of mismatch: shifts in granularity such as the example in fig. 6.

The matcher operates in two steps: (1) First, it 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.

4.1 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 are entities and originate from the same word class.*

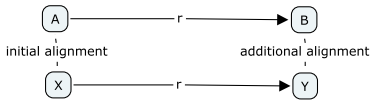
HEURISTIC 2. *Node1 is mapped with node2 if both are events, node1 subsumes node2 (or vice versa) and the case roles of node1 subsumes ones of node2 (or vice versa).*

As an example of heuristic2, consider two representations, *"Fuel moves into an engine"* (Move-Into1 object Fuel1) (Move-Into1 destination Engine1) and *"Engine takes in gasoline"* (Take-In2 object Gasoline2) (Take-In2 destination Engine1). Heuristic2 chooses (Move-Into1, Take-In2) as a pair of seed nodes because, in CLib, TAKE-IN and GASOLINE are subclass of MOVE-IN and FUEL, respectively. It is important to check the case roles because the case roles restrict the meaning of the events. To identify more seed nodes We plan to incorporate state-of-the-art methods for resolving anaphoric references [12].

4.2 Extending mappings

The mappings are recursively extended from the seed nodes by *extension pattern* rules. In the description of the patterns, G1 and G2 refer to the two input graphs, and A and X refer to 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 B subsumes Y (or vice versa). Then, B is mapped with Y.*

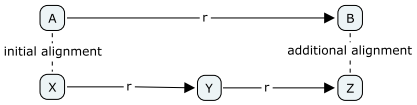


The pattern can align, for example, (Engine1 has-part Piston1) with (Gasoline-Engine2 has-part Piston2) if the system knows (Engine subclass Gasoline-Engine).

Patterns 2 through 6 resolve several types of granularity mismatches.

PATTERN 2. **(transitivity-based alignment)** *There are triples (A r B) in G1 and (X r Y) (Y r Z) in G2*

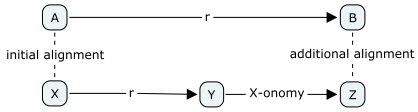
such that B subsumes Z (or vice versa) and r is a transitive relation such as causes, next-event. Then, B is mapped with Z.



A transitive relation often causes a granularity mismatch such as (Engine1 has-part Engine-Block1) (Engine-Block1 has-part Cylinder1) and (Engine2 has-part Cylinder2).

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

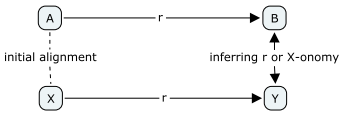
PATTERN 3. **(Co-reference across a granularity shift)** *There are (A r B) in G1 and (X r Y) (Y X-onomy Z) in G2 such that B subsumes Z (or vice versa). Then, B is mapped with Z.*



This pattern handles a case that two expressions reference the same entity at different granularities. For example, two representations, *"The engine takes in gasoline"*(Move1 object Gasoline1) (Move1 destination Engine1) and *"The cylinder in the engine takes in gasoline"* (Move2 object Gasoline2) (Move2 destination Cylinder2) (Engine2 has-part Cylinder2), co-references at the different granularity the location which gasoline is taken into. Notice that the patterns3 can align (Move1 destination Engine1) with (Move2 destination Cylinder2) (Engine2 has-part Cylinder2). In this case, X-onomy is a *has-part* relation.

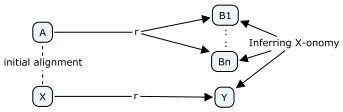
Patterns 4 through 6 introduce assumptions - additional triples - without which the alignment would fail.

PATTERN 4. **(Abductive transfer-thru alignment)** *There are triples (A r B) in G1 and (X r Y) in G2 such that B does not subsumes Y (or vice versa). 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, too. (In the implementation of Kleo, we abduce related-to).*



This pattern is an abductive version of pattern 2 and pattern 3. For example, for S1 and S2 in fig. 1, this pattern abduces (`cylinder-11` related-to `piston-4`) because both `cylinder-11` and `piston-4` are parts of the same engine.

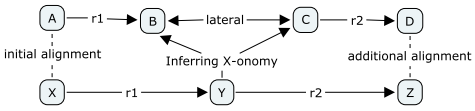
PATTERN 5. **(Generalization-based alignment)** *There are (A r B1) .. (A r Bn) in G1 and (X r Y) in G2 such that each of B1, ..., Bn does not subsumes Y (or vice versa). Then, X-onomy is abduced between Bi and Y for i = 1,2, ..., n.*



This pattern handles a case that several pieces of similar information in a fine-grained representation are generalized together to form a coarse-grained representation. For example, given two representations, (Engine1 has-part Cylinder1) (Engine1 has-part Piston1) and (Engine2 has-part Device2), this pattern makes an inference that Cylinder1 and Device2 are in a X-onomy relationship and that Piston1 and Device2 are in a X-onomy relationship.

The following pattern uses a relation called *lateral relation*, which is a general relation that connects two consecutive things (e.g. *next-event*, *beside*) - whether ENTITY or EVENT

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.*



This pattern handles a case that an aggregation of small things, whether ENTITIES or EVENTS, is viewed as one thing. Fig. 6 shows an example of this pattern which infers that Move2 is in a X-onomy relationship with Move1a and Move1b.

All these patterns are inherently uncertain; That is, a pattern could make a wrong alignment even if its precondition holds true (especially, pattern 4-6). Part of our future research is to deal with the uncertain nature of *knowledge integration*.

5. EVALUATION

We evaluate the contribution of knowledge integration by comparing Kleo with Mobius, two systems that differ primarily in their knowledge integration capabilities [4]. Specifically, they differ in three ways: (1) Kleo's graph matcher handles mismatches in granularity, while Mobius's does not (i.e. Kleo uses patterns 1 - 6, described

[4]The original Mobius system used BBN's Serif parser [11]; but, for the evaluation purpose, we replaced the parser with the one used in Kleo, the Stanford parser.

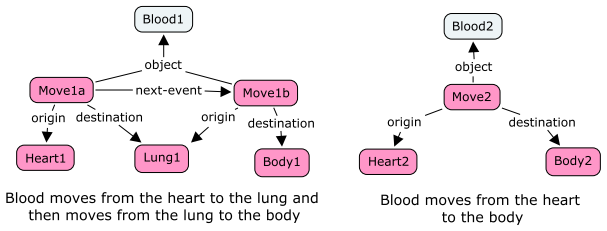


Figure 6: Example for the extension pattern 6

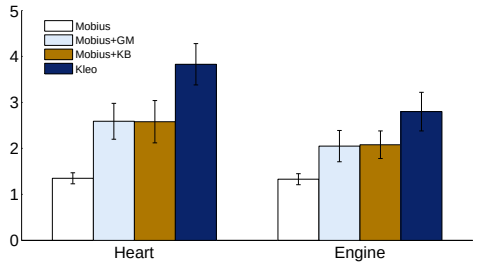


Figure 7: The average density-improvement over 25 texts in each domain (95% confidence level).

in Section 4. Mobius, however, uses only pattern 1). (2) During knowledge integration, Kleo uses both CLib and knowledge derived from reading previous texts. Mobius uses only CLib. (3) To improve efficiency, Kleo uses partitioning (see Section 3.2), but Mobius does not.

Both systems read about 25 texts in two domains: *the blood circulation in the human heart* and *the cycle in the internal-combustion engine*. Each text consisted of a paragraph drawn from a variety of sources – encyclopedia, web pages, textbooks – and were roughly at the same complexity as the Wikipedia articles on the topics. On average, the texts contained 5.67 sentences, and the sentences averaged 16 words in length.

5.1 Evaluation of SKI

We evaluate the two main features of SKI: (1) the ability to align representations, even when they differ in their level of granularity. This is performed by pattern rules 2 through 6 in the graph matcher (see Section 4.2) (2) the use of background knowledge learned by reading previous texts (see *Elaborate* in Section 3.1).

To measure the contribution of each feature, we compared 4 systems: Kleo (uses both features), Mobius+GM (uses only the extension rules), Mobius+KB (uses only the pre-learned knowledge), Mobius (uses neither). The graph matcher used in Mobius and Mobius+KB is the same as the one in Kleo except that it uses only rule pattern 1, and none of the rules that handle granularity mismatches.

The main purpose of SKI is to reduce the fragmentation of knowledge among the sentence-level representations. To measure the contribution of SKI, we use a metric called *density-improvement*. This metric is based on the *density* of the graph, which measures the connectivity of a graphical representation:

$$density = \frac{\text{number of edges}}{\text{number of edges in a fully connected graph of n nodes}} = \frac{e}{\binom{n}{2}}$$

where n is the number of nodes in the graph and e is the number of edges. The maximum of *density* is 1 in the case that the graph is fully connected.

Density-improvement is defined as

$$\frac{\text{density of the graph after SKI}}{\text{density of the graph before SKI}}$$

This metric measures SKI's contribution to increase the density of the graphical representation. Fig. 7 shows the results. SKI improved the density of the graph of learned knowledge by a factor of 2.5-3.5 (the *density-improvement* of Kleo is 2.5-3.5 times higher than the *density-improvement* of Mobius). The two main features of SKI – handling granularity mismatches and using background knowledge from previous reading – contributed equally to this improvement (the *density-improvement* of Mobius+GM and Mobius+KB are about equal). We also found that this result is independent to the order of reading texts.

Further, we investigated why SKI contributed more in the heart domain than in the engine domain. We found that the major reason for this difference is that Kleo drew more background knowledge for the texts about the heart because they have more overlapping content, as measured by the number of words in common across the texts.

5.2 Failure Analysis on SKI

To prioritize the future work on knowledge integration, we analyzed the failure cases of SKI during the experiment in Section 5.1. We randomly chose 15 texts used in the experiment, and closely examined the following knowledge integration operations – merging two knowledge structures and inferring additional semantic relations.

First, we used a simple scoring function to measure how often the operations contribute to combining representations coherently: 1 if an operation contributes; .5 if an operation partially contributes; 0 if an operation is not justified. Examples of the operations that partially contribute are : (1) Background knowledge used during reading is only partially correct (2) Merging two nodes is only partially justified.[5]

[5]For example, let's assume that node3 should be merged with node1 but not with node2 and that knowledge integration merges node1 and node2 to produce node4 and then merges node4 with node3. Merging of node4 with node3

The average score was .56, indicating that roughly half of the operations contribute to the goal of coherently combining information [6]. To investigate the other inappropriate operations, we categorized the failures and measured their frequencies:

1. Incorrect semantic representations from previous reading (36%) During knowledge integration, Kleo draws upon knowledge acquired from reading previous texts. In errors of this type, the previously acquired knowledge was incorrect, typically due to errors in the NL component.

2. Incorrect application of background knowledge (18%) In errors of this type, Kleo pulled in previously acquired knowledge that irrelevant. For example, Kleo pulled in information about *a jet engine* for a text about *gasoline engine*. To avoid this error, KLEO should consider context when drawing information from the background knowledge base.

3. Overly general background knowledge (18%) In errors of this type, Kleo pulled in knowledge that is overly general and vague. For example, Kleo pulled in the triple (Engine related-to Piston), which fails to specify the relationship.

4. Inappropriate application of Heuristic1 in Section 4.1 (14%) In errors of this type, heuristic1 joined two entities that share a common head word, but are otherwise dissimilar. For example, Kleo inappropriately joined *"right atrium"* with *"left atrium"*. To avoid this type of mistake, Kleo should consider features derived from modifying phrases.

5. Language interpretation failure (8%) In errors of this type, Kleo makes incorrect knowledge-integration decisions because the NL component misread a text. For example, mapping a word to an inappropriate concept can cause errors in graph matching.

6. Mishandling transient properties (5%) In errors of this type, Kleo fails to recognize that a property is transient, causing inappropriate matches. For example, Kleo incorrectly matched *"oxygen-rich blood"* with *"oxygen-poor blood"*

The above categories show that the failure of the NL component (44% from error1 and 5) is a main bottleneck of knowledge integration (In our experiment, only 30% of the triples produced by the NL component were correct [3]). We plan to address this brittleness of Natural Language Processing by reading multiple texts with

is partially justified because node1 should be merged with node3 but node2 should not.

[6]Because the erroneous operations are often caused by the failure of the NL, not KI's algorithm (see the categories of the failure cases), the score will increase independent to the KI component with more advanced NL technology.

patterns	frequency(%)	patterns	frequency(%)
pattern1	52.4	pattern2	0.3
pattern3	8.2	pattern4	31.9
pattern5	7.2	pattern6	0

Table 1: The frequency of invocation of the extension pattern rules

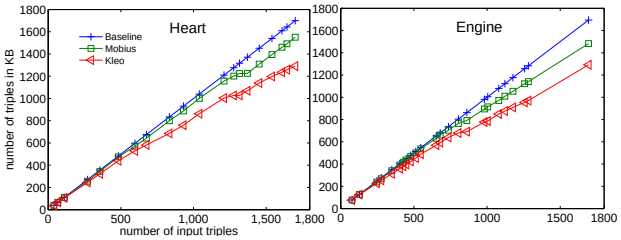


Figure 8: Increase in knowledge-base size with presentation of triples to TKI

redundant contents. Specifically, we will widen the funnel between the NL and the KI component, which currently passes forward only one interpretation for a single text (while pruning the other possible interpretations which may contain the right one). We recently developed a compact representation for representing a set of candidate interpretations. Then, the KI component finds the most plausible one by comparing different sets of the candidate interpretations from multiple texts on the same topic.

5.3 Evaluation of TKI

The main purpose of TKI is to combine information learned across multiple texts. In this evaluation, we use the metric of knowledge base size – the number of triples in the final knowledge base of information learned by reading a corpus of texts. Because TKI attempts to find overlapping content, to avoid redundancy and improve coherence, the more TKI identifies and merges the overlapping contents the smaller the resulting KB will be.

Specifically, we evaluate the contribution of the graph matcher, especially the pattern rules that handle granularity mismatches, by comparing three systems: Kleo (uses all the pattern rules described in Section 4), Mobius (uses only pattern rule 1, not rules 2 through 6, which are intended to handle granularity mismatches), and a baseline (simply appends knowledge structures with no attempt to match them).

Fig. 8 shows the result. The x-axis is the total number of input triples presented to TKI and the y-axis is the total number of output triples produced by TKI (i.e. the total number of triples in the final knowledge base). In both domains, TKI discovered a significant amount of overlapping content across the texts – the knowledge base built by Kleo was 30% smaller than the one built

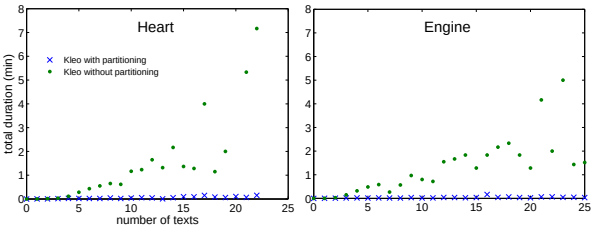


Figure 9: With partitioning, the run time of updating a knowledge base is almost negligible

by the baseline. The graph-matcher rules that handle granularity mismatches account for about 45% of this contribution – the knowledge base built by Kleo was 18% smaller than the one built by Mobius.

Table. 1 shows the number of frequencies in which the extension pattern rules are invoked during reading the 25 texts in two domains. It shows that the rules for resolving granularity mismatches (pattern 2-6) make a significant contribution (about 50% of total invocation) and that Pattern4 is used most whereas Pattern2 and 6 are rarely used.

Lastly, we evaluated the contribution of partitioning. The purpose of partitioning is to dampen the increase in the cost of graph matching as the knowledge base grows. We compare two systems: Kleo and Kleo-Partioning (i.e. Kleo without the partitioning capability).

Fig. 9 shows the results. The time required to read a text by Kleo-Partitioning grows rapidly with the size of the knowledge base (as measured here by the number of texts read so far). In contrast, the time required by Kleo remains almost constant. Partitioning seems crucial for the scalability of knowledge integration.

6. RELATED WORK

Learning Reader [6] and Kleo are similar in that both attempt to relate information across texts. Their goals, however, differ. Learning Reader, which reads a corpus of newswire, attempts to identify a same incident mentioned across articles and merge the information about the incident together. In contrast, Kleo, which reads texts about general concepts such as the engine or the heart, uses the other texts to imply unstated information in a text (see *Elaborate* in Section 3.1).

Most information extraction systems extract particular types of knowledge such as relations between two entities [1] or named-entities [5]. Because their representations are simple (e.g. triples or named-entities), they generally do not employ a sophisticated knowledge integration such as the graph matching performed by Kleo. Some machine reading systems perform a simple type of knowledge integration – mostly, named-entity resolution, using, for example, spreading activation [8] or semantic matching [14].

7. ACKNOWLEDGEMENT

Support for this research was provided by a 2008 IBM Open Collaborative Faculty Award.

8. REFERENCES

[1] M. Banko and *et al.* Open information extraction from the web. In *IJCAI*, Hyderabad, 2007.

[2] K. Barker and *et al.* A library of generic concepts for composing knowledge bases. In *Proc. of K-Cap*, Victoria, 2001.

[3] K. Barker and *et al.* Learning by reading: A prototype system, performance baseline and lessons learned. In *Proc. of 21nd National Conference on Artificial Intelligence*, 2007.

[4] K. Barker and S. Szpakowicz. Interactive semantic analysis of clause-level relationships. In *Proc. of PACLING*, pages 22–30, 1995.

[5] O. Etzioni and *et al.* Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134, 2005.

[6] K. Forbus and *et al.* Integrating natural language, knowledge representation and reasoning, and analogical processing to learn by reading. In *Proc. of AAAI*, pages 1542–1547, 2007.

[7] N. Friedland and *et al.* Project Halo: Towards a digital Aristotle. *AI Magazine*, 25(4):29–48, 2004.

[8] B. Harrington and S. Clark. Asknet: Automated semantic knowledge network. In *Proc. of AAAI*, 2007.

[9] D. S. Kim and B. Porter. Kleo: A bootstrapping learning-by-reading system. In *Proc. of the Spring AAAI Symposium Series*, 2009.

[10] D. Klein and C. Manning. Accurate unlexicalized parsing. In *Proc. of ACL*, 2003.

[11] S. Miller and *et al.* A novel use of statistical parsing to extract information from text. In *Proc. of the 1st conf. on NAACL*, pages 226–233, 2000.

[12] R. Mitkov, B. Boguraev, and S. Lappin. Introduction to the special issue on computational anaphora resolution. *Computational Linguistics*, 27(4):473–477, 2001.

[13] J. Sowa. *Conceptual Structures*. Addison-Wesley, 1984.

[14] M. Yatskevich and *et al.* Coreference resolution on rdf graphs generated from information extraction: first results. In *ISWC Workshop*, 2006.

[15] P. Z. Yeh, B. Porter, and K. Barker. Using transformations to improve semantic matching. In *K-CAP'03*, pages 180–189, 2003.