

Project Möbius: A study on the feasibility of Learning by Reading

Noah S Friedland	David Israel	Peter Clark	Eduard Hovy, Jerry Hobbs, Rutu Mulkar	Bruce Porter, Ken Barker	Ralph Weischedel, Paul Martin
The Friedland Group, Inc.	SRI International	The Boeing Company	ISI	University of Texas	BBN Technologies

Abstract

Much of human knowledge is expressed in documents. Our ability to express ideas in written language and disseminate those ideas in documents has been key to many endeavors, especially to the development of science and technology. With the dawn of the information age and the Internet, the number of documents within easy reach has skyrocketed, yet, beyond the use of keyword search, we lack the ability to truly automatically harness the power of this textual knowledge. Project Möbius was a two year effort aimed at ascertaining the feasibility of a class of systems capable of automatically extracting knowledge from documents and utilizing this knowledge, again automatically, to expand existing knowledge bases. The approach used in the study was to develop a demonstration system that, for the first time, integrated state-of-the-art Knowledge Representation and Reasoning (KR&R) and Natural Language Understanding (NLU) technologies in the automated extension of an existing knowledge base. The year one effort targeted texts on the topic of the form and function of the human heart. Automated knowledge acquisition was evaluated through the inspection of the system's knowledge base before and after processing a previously unseen text. Year two transitioned to a far more rigorous evaluation – measuring the performance of the system answering previously unseen questions before and after the reading of previously unseen texts. The year two domain was also far more complex– the form and function of engines. The year two system was able to answer a significant number of challenge questions, achieving a score of 32.4% correct on an objective test upon which the system had scored 0% before reading. This evaluation was followed by a detailed failure analysis which helped the researchers to form a better and more fully quantitative understanding of the challenges facing a future Learning by Reading (LbR) system. The conclusion of our study is that Learning by Reading produced statistically significant improvements in the problem solving abilities of the target knowledge, and that, with a major research effort, substantial progress could be made in the general application of LbR.

Keywords: Learning by Reading, automatic knowledge acquisition from text

Introduction

The information age has seen an explosion in the number of documents now easily available on everyone's desktop. Search techniques have made this information more accessible to human readers, but for machines, the vast amounts of linguistically encoded knowledge on the Web are mostly opaque. Research in natural language understanding (NLU) has primarily been focused on "text in; text out" approaches, such as summarization and translation. Other efforts, for example those involved in MUC and ACE, that have focused on "text in; formalized representation of knowledge out", have targeted information extraction against fairly simple pre-defined schemata, e.g. looking for relationships between people, places and things. While there have also been a few NL systems which generate logical forms from text, they have not addressed the task of assembling those pieces together into a coherent knowledge base.

The Knowledge Representation and Reasoning (KR&R) community, on the other hand, has dedicated its efforts towards human-driven knowledge formulation. The CYC project produced a very large knowledge base, with millions of concepts and instances and hundreds of thousands of rules/axioms. The University of Texas advocated a different approach, focusing on hundreds of high level modules – mini theories – that could be used compositionally to formulate more specific and domain dependent knowledge. Formulating knowledge into such representational systems has been a significant challenge – a problem known as the knowledge acquisition (KA) bottleneck. The RKF program and Project Halo have been the latest major thrusts in this line of research.

Despite good progress over the last five years in KA techniques, getting humans to formulate knowledge is still very expensive and the results, very brittle – especially when faced with the construction of a sizeable KB. The expense is due to the painstaking effort needed to create an effective KB – one capable of executing one or more tasks at a pre-specified level of performance. Once constructed and tested, the KB is essentially frozen and remains functional until either the knowledge in it is obsolete or the context it is used in changes. Then, additional human labor is needed to maintain or revamp the knowledge.

Thus, the stage has been set for Learning by Reading (LbR). Automatically acquiring knowledge from electronic documents could enable the fully automatic formulation of large effective KBs. Solving the problem of automatically producing such KBs would, in turn, open up a new frontier of knowledge exploitation algorithms, which would be both robust and economical, and provide revolutionary new ways of querying and problem solving against this knowledge.

Attacking this problem poses new challenges for the NLU and KR&R communities. The former need to consider the challenge of aligning and guiding representations achievable by automated extraction from text, with all its inherent ambiguity and logical gaps, to representations that are usable by KR&R systems; while the latter should be considering how to modify KR&R systems and approaches, which have historically been designed to work with human-crafted knowledge, to work with large amounts of noisy logical form derived automatically from text.

The deep integration of KR&R and NLU techniques, suggested above, may have a substantial impact on how each community might consider approaching this problem. For example, the creation of logical form from text may be significantly impacted by the availability of a large KB, which could provide the rich knowledge context that might help constrain the ambiguities encountered during the extraction, and serve to guide further extraction of knowledge. Conversely, the availability of a large amount of text-derived logical form could potentially help vet portions of the initial KB.

The Notional Möbius Architecture

The Möbius project began as a thought exercise to formulate a feasible approach to addressing the Learning by Reading challenge. The first phase of this effort focused on the creation of a notional architecture to show how KR&R, Machine Learning (ML) and NLU techniques could be brought together to perform LbR tasks.

Figure 1, below, depicts the notional Möbius architecture. The system is initialized with a significant KB, which represents its prior knowledge. The KB provides a context in which newly acquired concepts and relations can be integrated. The KB is used, along with a learning directive, i.e. “Learn about X!”, to launch the acquisition loop. The acquisition loop derives knowledge in response to the directive, by identifying texts that may contain the desired knowledge, applying NLP techniques to extract logical form from those texts, and integrating the extracted knowledge into the KB. The consolidation loop operates on the incrementally assembled KB to verify that the knowledge being constructed is coherent and can sustain the required set of tasks, e.g. question-answering and problem solving, at some specified level of performance. The consolidation does this by performing a set of knowledge driven tests, and examining the outcome. Consolidation may result in new acquisition loop activities to either correct existing knowledge or plug gaps identified in the knowledge collected to date.

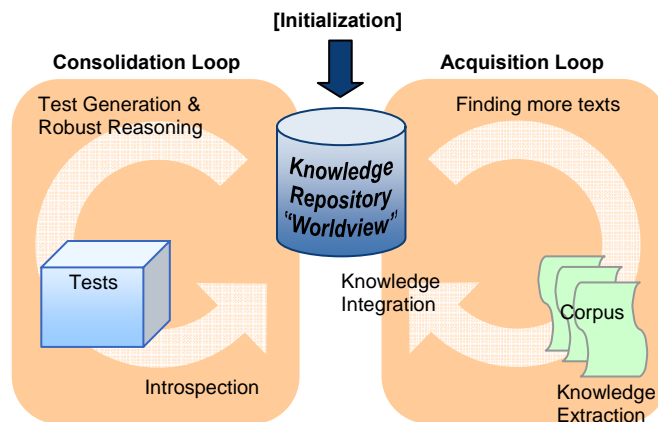


Figure 1: The notional Möbius architecture

The system stabilizes when a certain level of performance is achieved on a given task set, for a given corpus of documents. The addition of new documents or tasks may kick off a new round of Acquisition/Consolidation until a new equilibrium is achieved. Note how the notional architecture overcomes the brittleness issues of a static, human built KB. In

contrast to the Möbius vision, human constructed KBs cannot self-adjust to handle new data or tasks without significant additional knowledge engineering – making their upkeep and maintenance time consuming and expensive.

Year 1: Showing that a KB can be augmented from text

The notional architecture provided a framework of how an LbR system might function. The next task was to determine the feasibility of this notional approach. A 9 month proof of concept was launched. It called for a demonstration system to be rapidly assembled by a small team of researchers and engineers. The first challenge was to decide, given the limitations of time and scope, what the focus of the proof of concept demonstration should be, and what metrics should be used in the evaluation.

The year 1 team, which consisted of the Friedland Group as project management, SRI International as integrator, University of Texas at Austin (for KR&R) and USC ISI (for NLU), decided to focus on a component of the acquisition loop – namely the process by which text was processed to create logical form, and showing how that logical form could be inserted into an existing KB. To further restrict the scope the problem, a small domain was selected: the form and function of the human heart. The evaluation would include processing one or more previously unseen texts, and then inspecting the knowledge hierarchy to see that the logical form had been correctly produced from the text and inserted into the KB.

Year 1 Architecture

The system architecture is depicted in Figure 2. A text is loaded into the system. The ISI technology is depicted by the three boxes, labeled “Parser”, “L(ogical) Form” and “NL Triples” on the lower left hand side of the figure. The UT technology, depicted by the boxes “KR Triples”, “Integrate” and “Knowledge Repository”, appear on the lower right hand side of the figure. The repository itself consists of two elements. The first is the UT Component Library (CLIB), a collection of several hundred logical models, each representing a basic, high-level concept, like the event “Move”, the role “Fuel” or the entity “Device”. These components constitute the system’s prior knowledge. The second element in the repository is the collection of text-derived models, which represent a domain-specific extension of components in the CLIB, for example, a human heart is a type of “Internal Organ”.

At the beginning of processing, the text undergoes parsing, using the ISI CONTEXT parser. The parse tree is then processed to extract logical form. The final step takes the logical form and formulates “NL triples” in the form <word relation-term word>. The lexical NL triples are converted into conceptual KR triples of the form <concept relation concept> and then placed, by the “Integrate” module, into the new models area in the knowledge repository for manual inspection.

Newly acquired knowledge can take the form of concepts or axioms. Concepts, e.g. Heart – an “Internal Organ”, Blood – a “Liquid Substance”, etc. may be either newly learned, i.e. a new concept that did not exist prior to reading the given passage, or it may represent a domain specific specialization of an already existing CLIB component. New concepts,

like those in the example above, are attributed (as children) to one or more CLIB components, effectively inheriting the parents' attributes. This illustrates the power of having a large KB available during learning, as these children potentially can inherit a significant amount of knowledge from their parents that may not otherwise have been explicitly stated in the text. For example, knowing that Blood is a Liquid Substance allows the system to infer that Blood has liquid properties, like the ability to flow, or be the subject of pumping.

Axioms describe relations between concepts. Take, as an example, the sentence "The heart pumps blood." There are two concepts, Heart and Blood, and one action/event, Pump. This sentence could produce the following two axioms: <Heart agent-of Pumping> and <Blood object-of Pumping>. Thus, instances of Heart and Blood are connected logically via an instance of a Pumping event.

Domain models are graphs assembled from combinations of new or specialized concept instances, linked together by axioms. Models can be as small as a single KR triple, and potentially contain hundreds of concepts derived from multiple texts. For example, a domain model of a human heart may indicate that the heart has four chambers, and is attached to the human body's vascular system by veins and arteries. Models can vary in their level of detail, depending upon how detailed the originating texts are. Ultimately, the level of detail in the domain models should be sufficient to support question answering or problem solving tasks at a pre-specified level of performance.

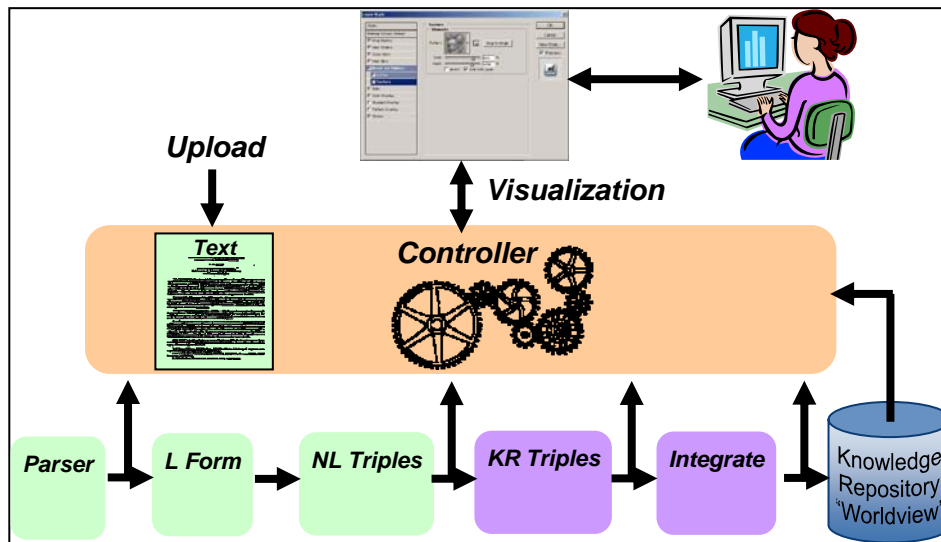


Figure 2: Year 1 System Architecture

Figure 3 depicts an example of how the NL processes text into NL triples. The pipeline begins with a parse produced by ISI's CONTEX parser. Next, the parse tree is converted into Hobbs Normal Form (HNF), which is then converted into NL triples. A significant number of rules were required in the LF Toolkit module to handle the diversity of forms that parse trees could produce. In many cases rules were either malformed or missing, which resulted in missing connections (triples) between a verb and one or more of its

arguments. We call these attachment problems “representational fragmentation”. Fragmentation manifested in NL triples that lacked a specific target term. Take the sentence “The heart pumps blood.” described above. Representational fragmentation might result in an inability to properly attach concepts in these triples, producing: <pump agent entity/thing>, <pump object entity/thing>.

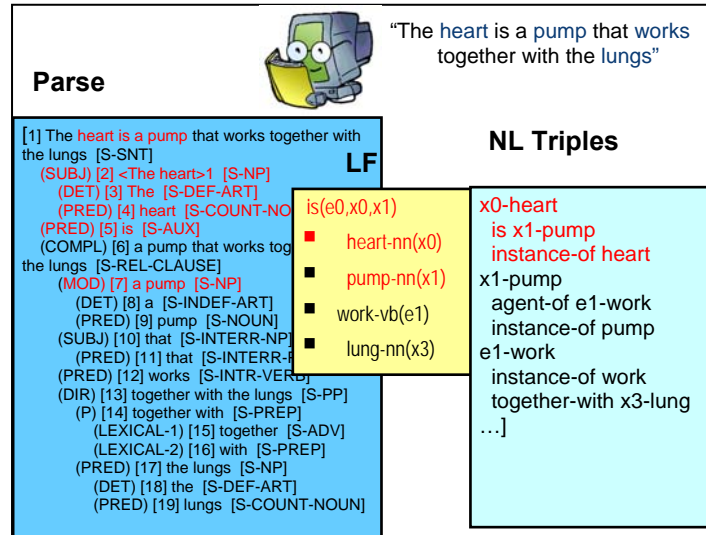


Figure 3: NL triple formulation

The next challenge involved word sense disambiguation (WSD), i.e., the mapping from lexical terms in NL triples to conceptual terms in KR triples. To do this, we followed the simple process depicted in Figure 4. As part of the prior process of constructing the CLIB, each concept is tagged with the WordNet synset(s) it corresponds to. Given a term to disambiguate, here “pump”, Mobius reduces it to its root form and then sees if directly matches a CLIB concept, i.e., is a member of one of the synsets attached to the concept. If concepts are found they are set aside as potential matches. If no concepts are found, the algorithm “climbs” the Wordnet2.0 hypernym structure until all matches are found. A heuristic is then left to pick a best match from all the candidates, factoring in the considerations, like the generality of the word sense and of the candidate conceptual match, or the distance traveled in the Wordnet2.0 hypernym hierarchy. It is this mechanism that allows the placement of both new and specialized concepts derived from text. This example again demonstrates how critical it is to have a significant existing KB to be able to facilitate the correct capture of text-derived concepts, as well as a good mechanism for mapping its conceptual hierarchy into lexical terms.

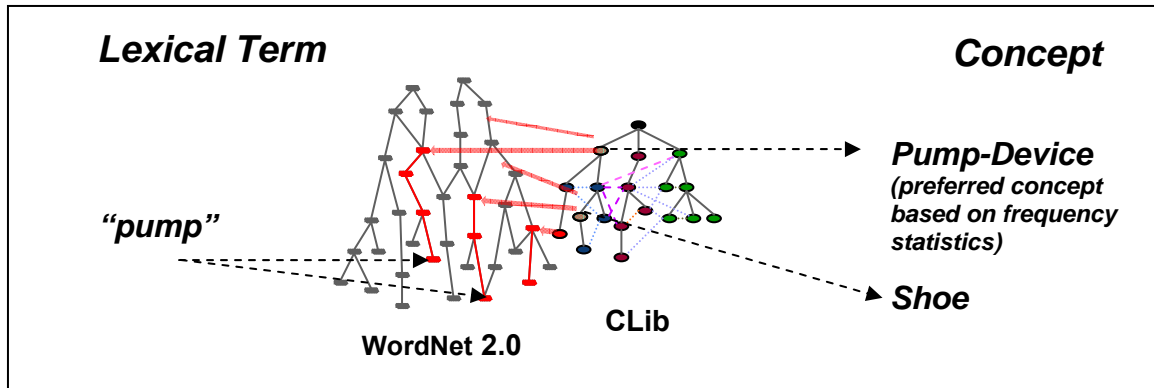


Figure 4: Aligning CLib with Wordnet 2.0 for lexical to conceptual mapping

Year 1 Results

Table 1 depicts example output from the Year 1 system. The texts used in the analysis were provided by a third party at the time of the live demonstration – so the team did not have access or prior knowledge of the text before the proof of concept exercise. Texts were gleaned, by the third party, from readily available Web resources. The left hand column depicts the sentences, numbered sequentially, in the passage being processed. Below each sentence is a bulleted list of all the concepts identified for that sentence. The concepts could be either new concepts defined by the system, or previously existing concepts that were modified by the system. The right hand column depicts the learned axioms, listed under their associated concepts. Concepts are listed alphabetically. Concepts highlighted in red were newly learned from the text, while other concepts were specialized from concepts already known to the system. The parent concept appears in parentheses, e.g. Blood (Liquid-Substance); in other words, Blood is a kind of Liquid-Substance. Each learned axiom includes its provenance, i.e. the sentence it originated from, in square brackets to its right. The top of the Right hand column provides a summary of the number of newly learned concepts, the number of unique, non-isa axioms learned and the average number of axioms learned per sentence. Is-a axioms generally result in new concept definitions and are, thus, handled separately.

For example, the axiom (KR triple) <Blood agent-of Enter> was learned from sentence 3: “Blood enters the heart through two input tubes.”, and <Blood agent-of Exit> was learned from sentence 5: “Blood exits the heart through tubes called arteries.” We learn that <Blood is-inside Body> and <Blood object-of Flow> from sentence 2: “The heart circulates blood through the body.”

Note that there are plenty of errors in this example as well. Artery is mislabeled as a type of Heart. Note the number of axioms describing Entity, Event and Thing, which indicate representational fragmentation. There are also axiomatic errors, like <Heart is-inside Tube> from sentence 5: “Blood exits the heart through tubes called arteries.” There are also plenty of missing axioms, like <Tube path-of Enter> from sentence 3. The system also failed to recover the fact that there were two input tubes. Many of these failures were due to the limitations of the year 1 system in recovering details from the sentences, primarily due to errors in the formulation of logical form.

Table 1: Example of Year 1 Results	
Learned or Specialized Concepts by Sentence	Learned Axioms
<p>1. A heart is an internal organ</p> <ul style="list-style-type: none"> Heart: A heart is an internal-organ. <p>2. The heart circulates blood through the body</p> <ul style="list-style-type: none"> Blood: Blood is liquid-substance. Heart: A heart is an internal-organ. <p>3. Blood enters the heart through two input tubes</p> <ul style="list-style-type: none"> Tube: A tube is a body-part. Blood: Blood is liquid-substance. Heart: A heart is an internal-organ. <p>4. The input tubes are called veins</p> <ul style="list-style-type: none"> Vein: A vein is an entity. Tube: A tube is a body-part. Blood: Blood is liquid-substance. Heart: A heart is an internal-organ. <p>5. The blood exits the heart through tubes called arteries</p> <ul style="list-style-type: none"> Artery: An artery is a heart. Vein: A vein is an entity. Tube: A tube is a body-part. Blood: Blood is liquid-substance. Heart: A heart is an internal-organ. <p>6. The heart consists of two lower chambers and two upper chambers</p> <ul style="list-style-type: none"> Artery: An artery is a heart. Vein: A vein is an entity. Tube: A tube is a body-part. Blood: Blood is liquid-substance. Heart: A heart is an internal-organ. <p>7. The veins are connected into the upper chambers</p> <ul style="list-style-type: none"> Artery: An artery is a heart. Vein: A vein is an entity. Tube: A tube is a body-part. Blood: Blood is liquid-substance. Heart: A heart is an internal-organ. <p>8. The arteries are connected to the lower chambers</p> <ul style="list-style-type: none"> Artery: An artery is a heart. 	<p>Concepts learned: 8 Unique, non-isa axioms learned: 72 Average unique learned axioms per sentence: 5.5</p> <p>Artery (Heart) agent-of Event [13]</p> <p>Blood (Liquid-Substance) agent-of Enter [3] agent-of Exit [5] agent-of Move [11] is-inside Body [2] object-of Event [13] object-of Flow [2] object-of Pumping [11]</p> <p>Body destination-of Event [11] encloses Blood [2]</p> <p>Chamber agent-of Contract [13] agent-of Move [12] destination-of Event [12] is-part-of Heart [6] object-of Event [12] origin-of Event [12] position (*low) [13] position (*upper) [12]</p> <p>Contract agent Chamber [13] agent Entity [13]</p> <p>Deliver agent Entity [11] object Oxygen [11]</p> <p>Device has-part Entity [13]</p> <p>Enter agent Blood [3] object Heart [3]</p> <p>Entity agent-of Contract [13] agent-of Deliver [11] agent-of Event [11] agent-of Exit [5] agent-of Move [12] is-part-of Device [13] object-of Exit [5] object-of Move [12]</p> <p>Event agent Artery [13] agent Entity [11] destination Body [11] destination Chamber [12] destination Heart [11] destination Lung [9] object Blood [13]</p>

<ul style="list-style-type: none"> • Vein: A vein is an entity. • Tube: A tube is a body-part. • Blood: Blood is liquid-substance. • Heart: A heart is an internal-organ. <p>9. One of the lower chambers pumps blood into the lungs</p> <ul style="list-style-type: none"> • Pump: A pump is a pumping-device. • Lung: A lung is an internal-organ. • Artery: An artery is a heart. • Vein: A vein is an entity. • Tube: A tube is a body-part. • Blood: Blood is liquid-substance. • Heart: A heart is an internal-organ. <p>10. The blood is enriched with oxygen in the lungs</p> <ul style="list-style-type: none"> • Oxygen: Oxygen is gas-substance. • Pump: A pump is a pumping-device. • Lung: A lung is an internal-organ. • Artery: An artery is a heart. • Vein: A vein is an entity. • Tube: A tube is a body-part. • Blood: Blood is liquid-substance. • Heart: A heart is an internal-organ. <p>11. The enriched blood returns to the heart and is then pumped to deliver the oxygen to the entire body</p> <ul style="list-style-type: none"> • Oxygen: Oxygen is gas-substance. • Pump: A pump is a pumping-device. • Lung: A lung is an internal-organ. • Artery: An artery is a heart. • Vein: A vein is an entity. • Tube: A tube is a body-part. • Blood: Blood is liquid-substance. • Heart: A heart is an internal-organ. <p>12. When the chambers compress, blood is forced from the upper chambers to the lower chambers</p> <ul style="list-style-type: none"> • Oxygen: Oxygen is gas-substance. • Pump: A pump is a pumping-device. • Lung: A lung is an internal-organ. • Artery: An artery is a heart. • Vein: A vein is an entity. • Tube: A tube is a body-part. • Blood: Blood is liquid-substance. • Heart: A heart is an internal-organ. <p>13. When the lower chambers contract, they force the blood into the arteries</p> <ul style="list-style-type: none"> • Oxygen: Oxygen is gas-substance. 	<p>object Chamber [12] object Oxygen [10] origin Chamber [12]</p> <p>Exit agent Blood [5] agent Entity [5] object Entity [5]</p> <p>Flow agent Heart [2] object Blood [2]</p> <p>Heart (Internal-Organ) agent-of Event [1] agent-of Flow [2] agent-of Pumping [11] destination-of Event [11] has-part Chamber [6] has-part Tissue [1] is-inside Tube [5] is-part-of Body [1] is-part-of Multicellular-Organism [1] object-of Enter [3] object-of Move [11]</p> <p>Lung (Internal-Organ) destination-of Event [9]</p> <p>Move agent Blood [11] agent Chamber [12] agent Entity [12] object Entity [12] object Heart [11]</p> <p>Oxygen (Gas-Substance) object-of Deliver [11] object-of Event [10]</p> <p>Pump (Pumping-Device)</p> <p>Pumping agent Heart [11] object Blood [11]</p> <p>Spatial-Entity path-of Thing [3]</p> <p>Thing position (*upper) [6]</p> <p>Tube (Body-Part) encloses Heart [5]</p> <p>Vein (Entity)</p>
--	--

<ul style="list-style-type: none"> • Pump: A pump is a pumping-device. • Lung: A lung is an internal-organ. • Artery: An artery is a heart. • Vein: A vein is an entity. • Tube: A tube is a body-part. • Blood: Blood is liquid-substance. • Heart: A heart is an internal-organ. 	
---	--

Year 1 Summary

The Year 1 system demonstrated qualitatively that a software system could extract logical forms from a small collection of previously unseen texts on the form and function of the human heart. The system accumulated that knowledge in an archive that would allow, under fairly significant assumptions, the cross-sentential aggregation of that knowledge. Although the system was able to produce some logical form correctly, it also failed to extract substantial amounts of logical form from these texts. Since the live study did not afford the possibility of doing a detailed failure analysis, we could only speculate on the reasons why the system failed to produce logical form – widespread fragmentation of the derived logical form, due, in part, to verb participant attachment problems. It was also unclear, from a quantitative standpoint, what impact the acquired knowledge had on the performance of the KB.

Year 2: A Quantitative Proof of Concept

The focus of Year 2 was to create quantitative, scientifically rigorous evidence that a system could learn by reading text, by examining the impact of the learned knowledge on an objective test of the KB. A second goal was to perform a rigorous failure analysis to use the demo system to better understand the larger technological challenges facing LbR. The core task in the evaluation was using the KB to answer comprehension-style questions. Assume that a third party produced one or more previously unseen texts on one or more topics, and a set of questions, with gold standard answers, to go along with those texts. A system could demonstrate its ability to learn by reading if it were able to do significantly better on answering the questions *after* reading the text than it could *before* those texts were read.

Again, the Möbius team needed to decide, given limited time and resources, and in light of the given challenge, where to put the emphasis of the development effort. Four critical areas were identified:

1. Strengthening the NL pipeline – creating a more robust mechanism for formulating NL triples and converting them into KR triples.
2. Adding a question-answering mechanism to facilitate the evaluation.
3. Improving the ways in which knowledge extracted from text is integrated into an already existing knowledge base.
4. Targeted Reading. The notional Möbius architecture called for a mechanism that would allow the system to identify new texts which might contribute to knowledge formulation.

Given the ambitious nature of the Year 2 effort, the Möbius team was augmented by two new team members: Boeing Phantom Works joined to assist in development of a question-answering capability and in the overall evaluation of the system, and BBN Technologies came on board to assist in strengthening the NL pipeline and assisting in experimentation on Targeted Reading.

The Y2 effort also took on a new, more complex domain – the form and function of engines. There are dozens of engine types, ranging from internal combustion, to gas turbine, jet and electric engines – each with a variety of possible uses and diverse internal functionality. The scope of the effort was oriented towards components and their roles, and less towards more advanced capabilities, such as sequences of events or diagnostics.

Year 2 Architecture

The Y2 architecture is depicted in Figure 5. The main thrust components appear in shaded areas. The NL pipeline is on the right hand side, question-answering is the region in the center of the figure, directly above the targeted reading (TR) module, on the lower portion of the figure. Knowledge integration (KI) is the region on the left hand side of the figure.

During learning by reading, a document enters the controller (center top of Figure 5) and is broken into its constituent sentences. Next, the sentences are fed through the NL pipeline, producing logical form, NL triples and finally, KR triples. KR triples are then sent to the KI module and used to build up dynamic domain models. The KR triples are also sent to a Triple Store to be used for problem solving.

The TR system is intended to provide an additional source of logical form. In many instances, the logical form that can be extracted from a given text can be incomplete. This is often due to failures in the formulation process, because the original English was unclear or ambiguous, or because the author, writing for a human audience, omitted important details that a machine reader could not infer. The idea behind TR is to utilize the redundancy in a large corpus, like the Web, to build, offline, a repository of logical form, i.e. “graph fragments”, and then to use these structures to “fill in gaps” in the current text. When the TR system is engaged, the semantic elaboration module at the end of the NL pipeline calls on the TR index to find fragments of logical form created from a corpus consisting of 200-500 text passages, to help in the correct formulation of KR triples.

During question answering, the system receives a question formulated in CPL. The question is passed into the Q-A module, where it is converted into logical form, much like a read sentence is. Then the logical form is used to match sentence-based KR triples in the Triple Store. The possible matches are then used to derive possible answers, which are, in turn, sent back to the controller.

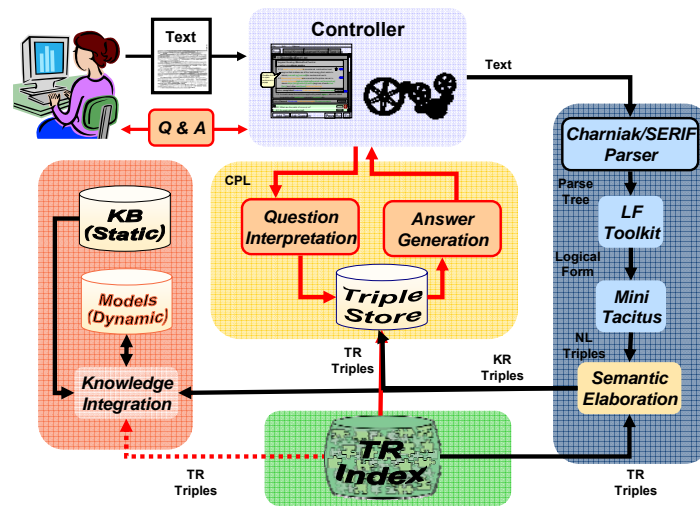


Figure 5: Y2 Möbius Architecture

The NL Pipeline

Figure 6 depicts the details of the NL pipeline. The modules appear as rectangular elements and the rule base for those components appear as ovals. The components are color coded to reflect their degree of generality. Three significant improvements are evident:

1. The pipeline is utilizing two state-of-the-art statistical parsers – the Charniak parser from Brown University and BBN’s SERIF parser. Both are trained against the Penn Treebank marked parses of WSJ text. BBN added 38 trees from the engine domain (less than 1%) to this training set, to address domain specific issues, like compound nouns – e.g. “spark plug”.
2. Parse tree binarization: Transformation of the parse trees (augmented with 306K dictionary rules) into binary form greatly reduced the number of rules required to map trees to logical form. This step virtually eliminated the participant verb attachment problems experienced in Year1.
3. A new triples generator, the final step before passing the NL triples to the KR component, dramatically reduced the number of errors in the NL triples.

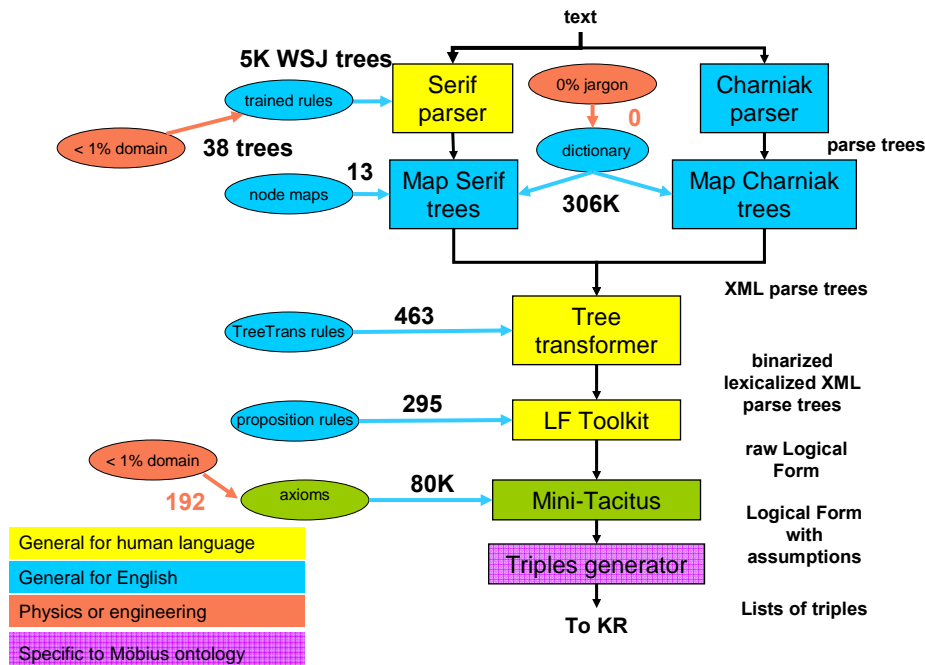


Figure 6: The NL pipeline

The Question-Answering Algorithm

The question-answering algorithm is depicted in Figure 7. The algorithm is intended to maximize the recall from the system's knowledge, at the cost of lower precision. The process involves the following steps:

1. The original question is manually rewritten in a simplified form of English called CPL (Computer-Processable Language), for ease of processing (we treated understanding of the full English questions as out of scope, as it was not our primary project focus).
2. The CPL is automatically interpreted to generate a set of triples representing the question. This triple set, like the triples produced by Möbius when reading, can be thought of visually as a graph of concepts (nodes) and relations (arcs). Initially the CPL interpreter assigns word senses based on the original CLib ontology. Subsequently, the word senses may be refined to more specific concepts learned by Möbius during reading, if there is lexical evidence to do so. In the example in Figure 7, the word "dampening" is initially mapped to "Activity", but is revised to the more specific "Restrained".
3. The question graph is used to search the Triple Store for similar graph structures. In this case, a good match was found in text6 sentence 23.
4. The question graph, including the variable node denoting the answer to the question, is unified with the graphs in the Triple Store found in the search process, and hence the answer is found.

It is conceivable that many graphs in the Triple Store might match the question graph, which may result in multiple answers. The potential for bad matches increases when the answer is not in the learned knowledge (either because the answer was not stated in the original text or Möbius failed to understand the appropriate part of the original text). When that happens, matches might be much more general. Say the concept “Muffler” was not found in the learned knowledge, the next, more general correspondence the Q-A engine might have used would have been “Device”. The general term Device may have resulted in many, many potential hits in the learned knowledge – sometimes producing hundreds of spurious responses.

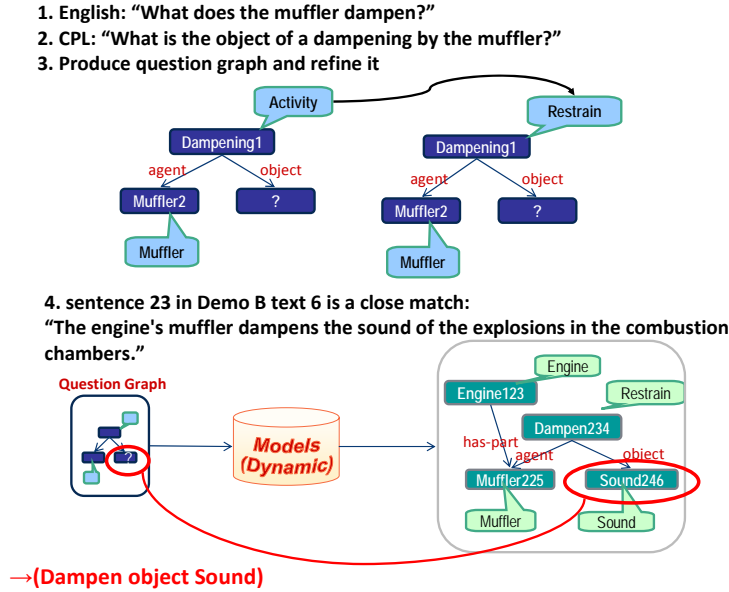


Figure 7: An example depicting how the Q-A algorithm works

Knowledge Integration

The purpose of the Knowledge Integration (KI) module in the Mobius system is to stitch together the information that is acquired during reading, and to relate it to the background knowledge in the initial knowledge base. The KI module’s performance can be evaluated by looking at the coherence of the knowledge that Mobius learns. It should be tightly knit together into a coherent whole, not a collection of loosely connected “islands” of knowledge. We will use a metric, called relatedness, to measure KI’s contribution:

$$relatedness = \frac{\sum_1^n n'(Gi) + e'(Gi)}{\sum_1^n n(Gi) + e(Gi)}$$

where

- Gi input representations
- n(G) The number of total nodes in G
- e(G) The number of total nodes in G
- n'(G) The number of total nodes in G related with other representations by KI

e'(G) The number of total nodes in G related with other representations by KI

There are numerous challenges in automating knowledge integration. In Year 2 we have focused on two of them: 1) resolving granularity differences among fragments of knowledge, e.g. one passage may say something like “the engine drives the car”, while another would go into minute detail about the engine, its components and their function, and 2) incorporating background knowledge into a learned representation.

Resolving granularity differences

One of the primary challenges in knowledge integration is resolving granularity differences among fragments of information. Consider, for example, this pair of sentences:

S1: An engine converts fuel into motion.

S2: An air-gas mixture combusts in the cylinder, which drives the piston, turns the crankshaft, and moves the vehicle.

S1 is a typical topic sentence – it provides an overview. S2 elaborates the topic sentence by providing additional details. The reader’s job, which the KI module attempts to automate, is to find the connections between S1 and S2 that make these sentences coherent. This requires resolving the granularity difference: S1 is relatively coarse grained, and S2 is relatively fine grained.

To automate this process, we need to understand the ways in which granularity differences can arise in text, and the ways in which they can be resolved. We selected about 35 general texts and manually performed knowledge integration on the sentences of the texts to identify common granularity mismatches. This analysis revealed four major types of granularity differences, and we developed graph alignment methods that resolve each type.

Incorporating background knowledge

The Möbius system starts with a small amount of background knowledge about Engines, consisting of these concepts: Chamber, Chemical-Entity, Combust, Cycle, Expand, Flow, Protect, Valve, and (a very skeletal version of) Engine. Integrating this initial knowledge with knowledge learned by reading can improve the coherence of the resulting knowledge base.

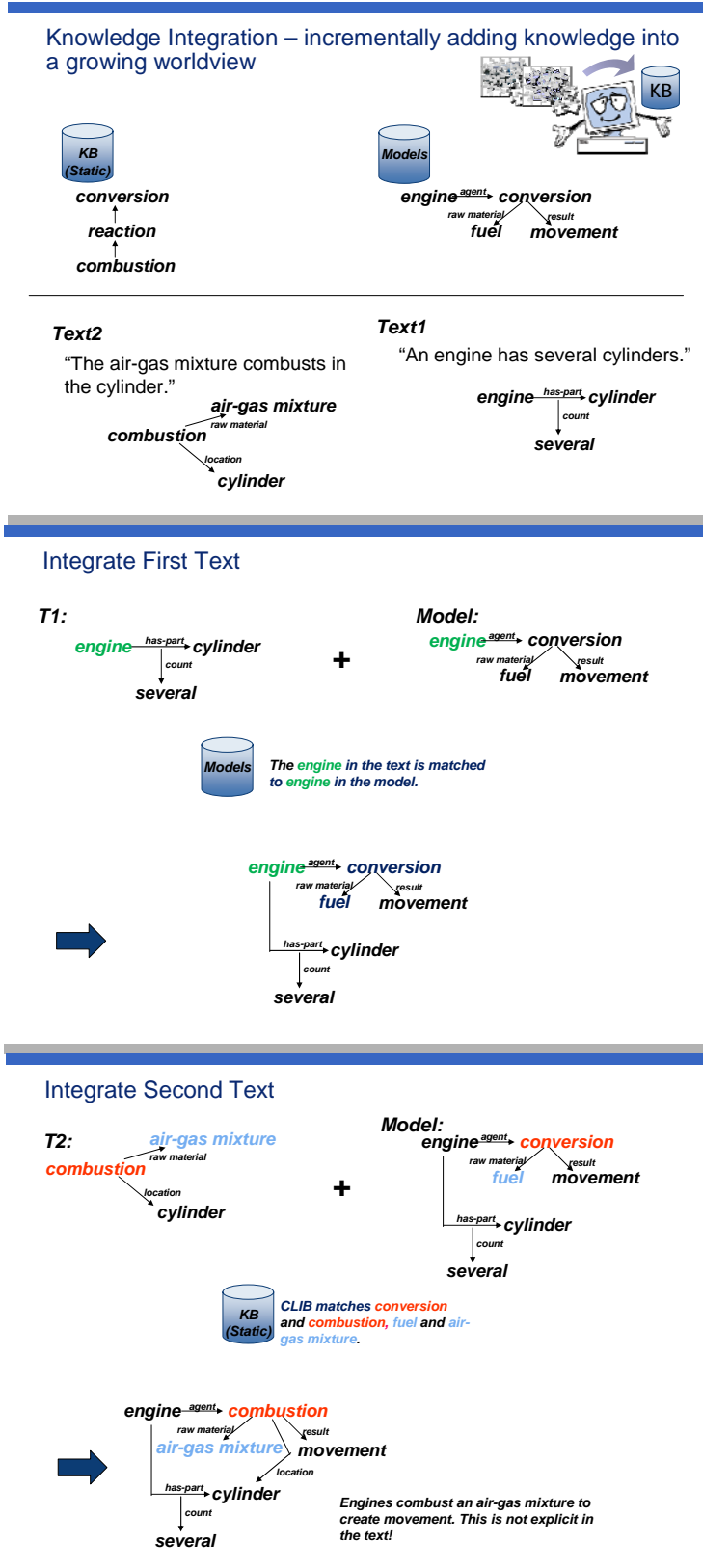


Figure 8: A graphical depiction of the KI process

Targeted Reading

Learning by Reading (LbR) is an attempt to formulate inferentially correct logical structures directly from text. This requirement is often at odds with the formats used by human authors. For example, consider the following text:

“A gasoline engine is a type of internal combustion engine. The piston moves inside the cylinder, causing the crankshaft to rotate.”

It is clear to human readers that the terms “piston”, “cylinder” and “crankshaft” all refer to components of the gasoline engine. A simplistic machine reader would require the author to explicitly call out these relationships, i.e. <engine has-part piston>, <engine has-part cylinder>, <engine has-part crankshaft>, to be able to make the necessary connections in the underlying logical model. This is just one example of the types of gaps and limitations human authored texts might contain that impede in-depth model construction from a single text. In other cases, important sentences might be so complex or ambiguous that the system may fail to recover key logical elements. The solution to these problems lies in the ability of the system to access other knowledge sources – be those background knowledge, or additional texts. This latter case is what we mean by “Targeted Reading”.

The initial attempt at Targeted Reading (TR) tried to perform the following sequence:

1. Identify a potential gap in the model
2. Create a query from keywords and synthesized clauses
3. Use keyword search to find passages relevant to the query, and rank them
4. Produce logical forms from the top ranking passages
5. Attempt to use the logical form produced to fill in the gaps in the current model

This approach had several shortcomings:

1. It was difficult to produce the “right” queries for the text-based search
2. The search results had extremely low recall and precision
3. Run-time production of logical forms from the top ranking passages was slow

Together, the overall problem with this approach was that Möbius makes “two transitions through the lexical-conceptual boundary”, i.e., from concepts, to words, to concepts again, each transition being error-prone. The first pass went from requirements derived from observed gaps in the knowledge model, to a set of words to be matched. The second pass was interpreting the candidate passages from lexical to conceptual representation. Each of these passes involved great computational expense and introduced significant inaccuracies into the results. Once these limitations were clearly understood, another approach to TR was attempted. Pre-extracting logical form from a large corpus of documents into a repository would enable high-speed, accurate and complex querying. Thus, an offline variant of the usual Möbius process was run over a large number of texts. Each text was automatically converted into logical forms. These forms were placed in a high-speed, indexed database that would facilitate quick recall. Since the index was

represented in concept space, high precision queries could be formulated. As a result, the index could support hundreds of such queries a second.

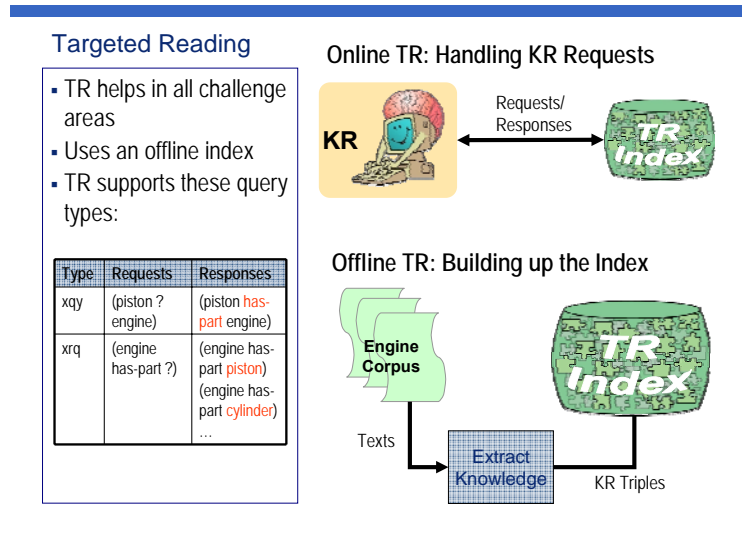


Figure 9: Targeted Reading

The logical elements captured in the index consisted of one or more KR triples. The index consisted of a single relational database table. SQL queries were derived from two main query types:

1. XqY, e.g. provide all graphs that contain a path between the concept “Engine” and the concept “Piston”, subject to constraints. Such a path may be a direct link (axiom) or a sequence of such links.
2. WRq, e.g. provide all graphs that contain a relation of the pattern “Engine has-part ?”, subject to constraints.

Year 2 Results

The goal of the Year 2 effort was to produce quantitative evidence of the feasibility of LbR, as measured by an objective test on the KB. In contrast to text-driven techniques (see sidebar), used in TREC and other question-answering challenges, the question-answering used in Möbius, employed pure KB techniques, operating upon KR triples in the Triple Store. So, although some of the questions asked may appear to be TREC-like, correct answer derivation in this case was an indication that the entire knowledge acquisition and integration process (and Q-A) worked perfectly end-to-end.

A second goal was to perform a comprehensive failure analysis, which might suggest potential avenues of LbR research. To this end, two demonstrations were produced. Demo A, in mid October of 2007, and demo B, in early March 2008. Both demos used question answering as a metric and followed the protocol specified in Figure 10, below.

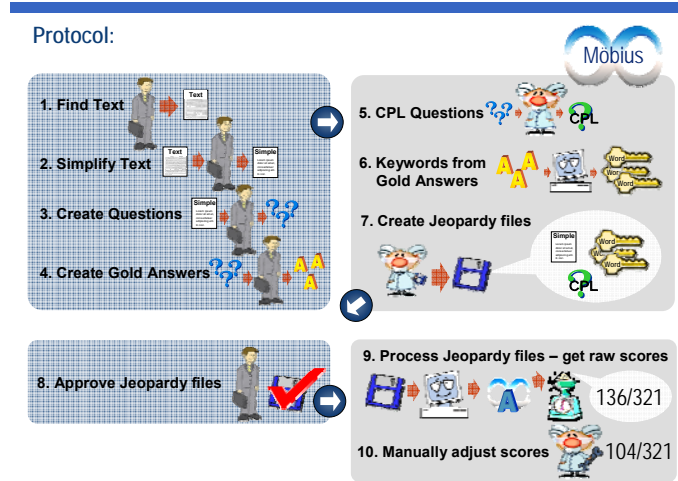


Figure 10: The evaluation protocol for Year 2 Möbius – both demos.

The demonstration protocol involved the following steps:

1. A third party developed a set of texts, questions, and gold standard answers. These texts were simplified by the third party to remove anaphora, negation, most conjunction/disjunction statements (“and”/”or”) and break up complex sentences – phenomenology that the Möbius Year 2 demonstration system was not equipped to address for a variety of technical reasons, but might be addressed in future work.
2. The Möbius team received this material and produced “Jeopardy” files – files used by the Möbius system to perform batch-oriented learning by reading and question answering. The most labor intensive part of the Jeopardy file creation was rewriting the original English questions in CPL simplified English.
3. The third party inspected the Jeopardy files to verify their correctness.
4. The Möbius team submitted the files to the system which performed the experiment and auto-graded the results. Auto-grading was performed by comparing the system output to keywords derived from the gold standard answers, Möbius scoring 1 point for each gold keyword included in its answer.
5. Manual verification of the auto-grading to make sure that spurious outputs were not counted as correct, in particular canceling points awarded when a keyword had been correctly found, but the answer triple containing it was incorrect or nonsensical. This manual verification process was essential to correct for the obvious approximate nature of keyword-style scoring.

Table 2: Demo Results Summary (No TR)			
	Demo A	Demo B	Comments
End-to-end Q-A	21.5/138 = 15.6%	104/321 = 32.4%	

Points lost from Q-A	unknown	55; upper bound on score: 159/321 = 49.5%	Lower bound, Q-A system recall: 104/159 = 65.4%
Data	3 topics, 122 sentences, 57 questions, total gold score 138	1 topic, 166 sentences, 127 questions, total gold score 321	

Table 2 provides a summary of the results obtained in the two demos. Demo A consisted of a total of 122 sentences on three engine topics: pulse detonation engines, 2-stroke engines and Steam Turbine engines. A total of 57 questions, worth a total of 138 points (1 point per answer keyword) , were submitted to the system in demo A. Demo B consisted of 166 sentences on one topic – the 4-stroke engine – with a total of 127 questions worth a total of 321 points. The demo A system scored 21.5 points out of a possible 138 for an overall score of 15.6%, while the demo system scored 104 points out of a total of 321, for a score of 32.4%. An analysis was made of the points lost due to poor performance of the question-answering mechanism. In demo B, up to 55 points were lost due to question-answering issues, like bad CPL encodings, problems with gold keywords, and failure of the system to produce the correct answer, despite the existence of the required triples in the Triple Store. For more, see results depicted in Figure 16. A system with “perfect” question-answering capability might have scored up to 159 points correct, or a score of 49.5%. Q-A recall therefore had a lower bound of 65.4%.

Figure 11 depicts example text and questions from the demo B evaluation. Note that the answers produced are in an Anglicized logical form, and that the forms appear in pairs, reflecting the associated axioms and their inverses.

“Sump” - Example Text and Questions




- **Text6:** ... “The sump surrounds the crankshaft. The sump contains oil which collects in the bottom of the sump. The oil pan is at the bottom of the sump.” ... 
- **Questions + Gold Answers:**
 - SUMP-1: What does the sump surround? Crankshaft.
 - SUMP-2: What does the sump contain? Oil.
- **CPL and System Answers:**
 - SUMP-1: What is the object of a surrounding by the sump?
 - be-contained/surround :: object = CRANKSHAFT.[1.145]
 - CRANKSHAFT :: object-of = be-contained/surround(tangible-entity, CRANKSHAFT).[1.145]
 - SUMP-2: The sump is the agent of a containing. What is the object of the containing?
 - be-contained/contain :: object = OIL.[1.146]
 - OIL :: object-of = be-contained/contain(tangible-entity, OIL).[1.146]

Figure 11: Example texts, questions, and system answers.

Figure 12 depicts the end-to-end question-answering results of the demo B system on the demo B data. The 166 sentences were divided into 6 separate texts. The system learned the texts one-by-one and asked all 127 questions between each text learned. The initial score on the questions before reading was 0% - so the system was unable to answer any of the questions. Note that the score gradually increased the more texts were learned, ultimately producing a score of 32.4% after learning the final text.

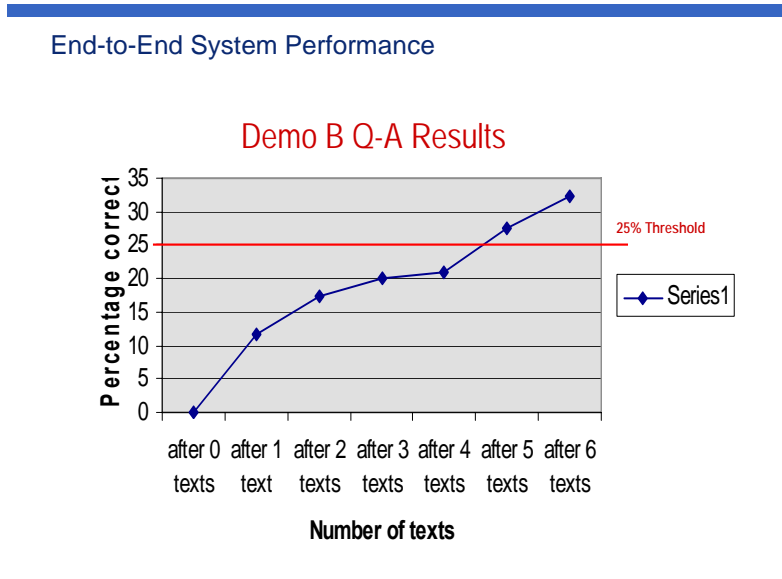


Figure 12: End-to-end system performance on demo B data.

Unfortunately, the demo A system was unable to answer the demo B questions, so we were not able to conduct a direct comparison of the scores for this data. We were, however, able to compare the results of the demo A and demo B systems on the demo A data. Figure 13, below depicts the results obtained on the three demo A engine topics. Build 694 is the demo A build system, while build 1020 is the demo B build system. Note the significant improvement achieved by the demo B system, particularly as more texts are read. There is an anomaly in the 2-Stroke topic, as the original demo A scores after reading 2 texts contained a scoring error, where the system was awarded a point for a spurious response. This scoring error was avoided in Demo B, which explains why the demo A system scored better than the demo B system for this data point.

Comparative Demo A-B Performance on Demo A Data

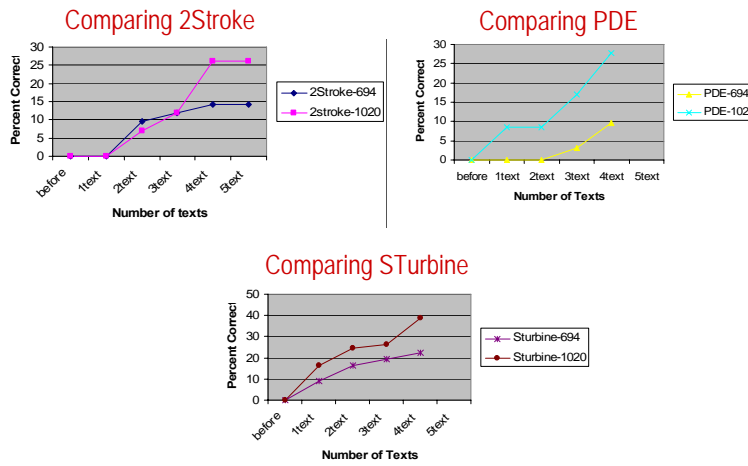




Figure 13: Comparison of demo A and demo B systems on demo A data.

A small experiment was also conducted to measure the impact of targeted reading (TR) on the end-to-end question-answering performance. The experiment was run on text 1 of the demo B data set. The correct scores went from 37/321 without TR, to 43/321 with TR – an increase of 16%. An example of one of the questions impacted by this experiment is depicted in Figure 14, below. Note the “TR” provenance of the answer axioms on the right, compared to the left, where no TR has taken place. TR allowed the system to pick up another point (2/9) over the non-TR run on text1 (1/9), at the expense of producing a lot more axioms pairs in the answer – some of which were irrelevant, like the Steam-Engine components. TR, in its current incarnation, appears to increase the score (recall) but has a negative impact on precision. This is still an open issue that will need to be tackled in future research.

The demo B experiments also included some very preliminary work on knowledge integration (KI). As this technology did not impact the question-answering performance of the system, we cannot report on its impact on the overall scoring. Thus, we will leave the reporting of the progress and impact of KI to a future publication.

TR Analysis, build 1020, demo B Text1

- Engine1: (English) What are the parts of an engine?
- Engine1:(CPL) What are the parts of an engine
- Engine1:(Gold answer): Cooling system, fuel system, exhaust system, lubrication system, piston, bearing, muffler, and cylinder.

TR=off 1/9 	TR=on 2/9 
engine :: has-part = PISTON.[1.1] PISTON :: is-part-of = engine.[1.1]	engine :: has-part = PISTON.[1.1] PISTON :: is-part-of = engine.[1.1] CYLINDER :: is-part-of = engine.[TR] engine :: has-part = CYLINDER.[TR] PISTON :: is-part-of = engine.[TR] engine :: has-part = PISTON.[TR] CYLINDER :: is-part-of = steam-engine.[TR] steam-engine :: has-part = CYLINDER.[TR] entity :: is-part-of = tangible-entity(FUEL).[TR]

TR=on for text1 resulted in a 16% improvement in the adjusted Q-A score

Figure 14: End-to-end TR results on demo B text 1.

Year 2 Failure Analysis

The demo B system produced significantly better results than the demo A system. The Möbius team wanted to investigate the reasons for this improvement. This was accomplished by a detailed, comparative failure analysis between both builds. The analysis involved a painstaking examination of the system and sub-component behaviors of the demo A and demo B systems on select subsets of the demo A and demo B data. The effort represented approximately one week of engineering effort for each of the two main Möbius modules – the NL pipeline and the KR components.

The analysis broke down into two parts: NL and KR. The NL analysis, depicted in Figure 15, below, tabulated all the serious errors – ones that downgraded the system’s overall performance. Next, the errors were categorized into three classes. Red errors represent errors that stem from significant research challenges – ones that may take years and entirely new approaches to resolve. Note that these results were produced on simplified texts, so the list of observed failures will be somewhat abridged, compared to errors that may arise when confronting more complex texts. The two red types listed in this analysis are prepositional phrase attachment and parse errors. Clearly given more complex sentences, the number and types of red errors would have increased. The green errors are errors that can easily be fixed with a few months of engineering effort. Most of these errors pertain to the mechanical process of how NL triples are formed in the pipeline. Yellow errors lie somewhere in between the red and green types. The left hand chart shows the serious errors produced by examining all the demo B data on the demo B system. The three main causes of error were: Prepositional attachment (red); missing arguments in the rules supporting tree binarization or logical form creation (yellow); and NL triple generation errors (green), which is the last step in the NL pipeline, before Semantic Elaboration transforms NL triples to KR triples. The right hand chart in Figure 15, depicts a comparative analysis of the demo A (build 694) and demo B (build 1020) system on text2 of the demo B data set. Note the significant difference in the number of serious errors. The demo A system produced a total of 349 serious errors, while the demo

B system produced only 22. The largest decrease in serious errors was in the green category – specifically, NL triple formulation errors. Their “green” designation marks their relative ease of repair, and the months between the demos included many of the needed repairs. The demo B system also had significantly less red parse errors, partially because the demo B system employed the BBN SERIF parser which had been trained with a sprinkling of engine domain example sentences, to address domain specific compound nouns, e.g. “Spark Plug”, added to the Wall Street Journal training trees, while the demo A system was scored using Brown University’s Charniak Parser, which had not been trained with any engine data.

NL Error Analysis

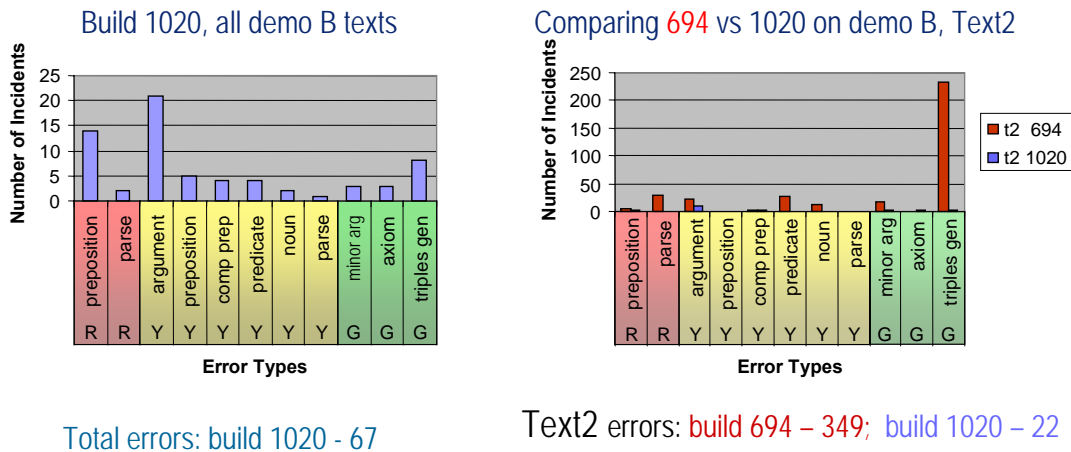


Figure 15: NL Error Analysis

Figure 16 depicts the KR and Q-A components failure analysis. The graph on the upper left hand corner depicts a comparative analysis of KR triples produced by the demo A (build 694) and demo B (build 1020) systems on text 5 of the demo B data set. The counts represent the number of “good”, “bad”, “spurious” and “missing” KR triples. “Good” triples are appropriate, given the text data. “Bad” triples are not. “Spurious” triples have nothing to do with the sentence, and “Missing” triples are triples that should have been produced, but for some reason were not. The head-on comparison between the builds yields the following facts: (i) the demo B system produced 48% more triples than the demo A system on the same text, and (ii) the demo B system produced 62% more “Good” triples than the demo A system.

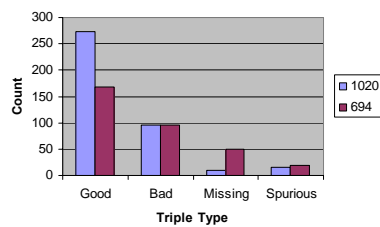
The upper right hand chart in Figure 16 depicts the KR error counts and categories for the demo B system on demo B texts 4, 5, and 6. The leading cause of KR errors are bad Word-to-Concept mappings (i.e., bad word sense disambiguation). These mappings form the core of the interpretation process between NL and KR triples in the Semantic Elaboration module. These errors have been classified by the University of Texas researchers as Red-Yellow in difficulty. The next major source of KR error is Noun Phrase Semantics. This has to do with how the system handles compound noun phrases, for example, the mishandling of “Spark Plug” or “Internal Combustion Engine”. These errors were labeled red. The third largest source of KR system errors are produced by the NL subsystem, and are then propagated into the KR subsystem.

The lower right hand chart in Figure 16 depicts a comparative analysis of KR errors between the demo B (build 1020) and demo A (build 694) systems, on demo text 5 data. Note the significant decrease in NL-related errors, which have been reduced by over 50% in the demo B system. The demo B system does experience more errors in the NP category, because the demo B NL pipeline was instrumented to recover more noun phrases than the demo A system.

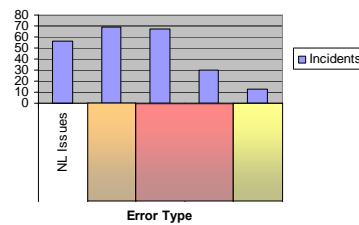
The final chart on Figure 16, on the lower left hand side, is an analysis of question-answering failures on the demo B system (Build 1020) for all 127 questions. NL and KR errors account for 162 points lost out of a total 321 points. The Q-A system itself is responsible for a loss of 37 points out of 321. Bad CPL encodings are responsible for a loss of 11 points, and incomplete gold standard keyword allocation is responsible for 7 points lost from a total of 321. Thus, the overall points lost for non-NL/KR errors stands at 55, which results in a Q/A recall score of up to 65.4%.

KR/Q-A Error Analysis

Comparison: KR Triple Count, builds 1020, 694, Text5



Serious Errors: Build 1020, demo B Texts 4, 5, 6



Q-A Error Analysis Build 1020, demo B

Failure Type	Count
NL/KR	162
QA	37
CPL	11
Keywords	7

Comparing Build 1020 and Build 694, demo B text 5

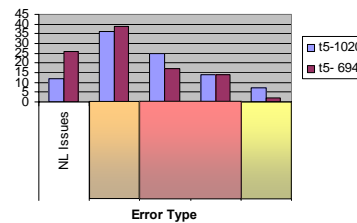


Figure 16: KR and Q-A Error Analysis

Summary and Conclusions

The goal of the two year Möbius project was to ascertain the feasibility of learning by reading. The Year 1 effort established qualitative evidence that a demo system could be assembled that would be capable of extracting logical form from text and placing that data into an existing knowledge base. The Year 2 effort produced quantitative evidence that a software system could be built that was capable of learning from text and that the knowledge learned could be demonstrably applied to an objective task: question-answering. In fact, the more the system read, the better it was able to perform on the Q-A tasks.

Also importantly, the study demonstrated the potential of the high-level architecture – one that leverages techniques and technologies in three major discipline areas in computer science: Knowledge Representation and Reasoning (KR&R), Natural Language Understanding (NLU) and Machine Learning (ML). Most of the focus centered on the acquisition loop of the Möbius notional architecture. More effort will be needed to create the capabilities needed to support the consolidation loop, which will be critical in producing correct, high performance knowledge.

The feasibility study also exposed some of the primary challenges facing LbR research:

1. Bridging the NL-KR gap: harvesting high quality KR triples from text. It will be important to expand the ability to harvest logical form from increasingly complex texts.
2. Building Robust Knowledge Models: using KR triples to incrementally construct robust models of knowledge that can support high-performance inference. The challenge will be to construct these robust models from “noisy” logical form acquired automatically from text. Machine learning techniques may be helpful in eliminating some of this noise.
3. Problem-Solving in Text-Derived Knowledge: creating problem-solving and question-answering (Q-A) methodologies to support evaluation and introspection of the system’s knowledge. Scaling problem solving techniques to large, automatically created knowledge bases, will require new approaches to isolate a few relevant assertions from thousands or tens of thousands of irrelevant ones.

The authors are confident that significant progress can be achieved in these challenge areas in the immediate future.