Advances in Cognitive Systems 4 (2016) 169–186 Submitted 8/2015; published 6/2016

Exploiting Connectivity for Case Construction in Learning by Reading

David Barbella BARBELLA@U.NORTHWESTERN.EDU
Kenneth D. Forbus FORBUS@NORTHWESTERN.EDU
Qualitative Reasoning Group, EECS Department, 2133 Sheridan Rd, Evanston IL 60208, USA

Abstract

One challenge faced by cognitive systems is how to organize information that is learned by reading. Analogical reasoning provides a method for immediately using learned knowledge, and analogical generalization potentially provides a means to integrate knowledge across multiple sources. To use analogy on learned material requires organizing information, represented in predicate calculus, into effective cases. This paper argues that using connectivity in semantic interpretations to organize knowledge learned by reading into overlapping cases can support analogical reasoning with these structures. We describe two connectivity-based methods and compare their performance with two baselines for the task of comparing and contrasting topics included in material the system has read.

1. Introduction and Motivation

Cognitive systems need to learn by reading in order to acquire knowledge in a scalable way. In natural language understanding and knowledge representation, solid progress has been made on extracting complex structured information from text (e.g., Barker et al., 2007; Fan et al., 2012). A new challenge is how such extracted knowledge can be organized for reasoning and for integration into what a cognitive system already knows. Analogical reasoning has been shown to be useful for robust learning by reading, for decoding instructional analogies (Barbella & Forbus, 2011), and for allowing a system to ask itself questions based on prior knowledge (Forbus et al., 2007). In this paper, we focus on organizing knowledge in ways that supports analogical reasoning.

Case comparison has been used or proposed for use in a variety of reasoning applications (Brüninghaus & Ashley, 2001; Peterson, Mahesh, & Goel, 1994; Chaudhri et al., 2014). This suggests that one way of organizing newly-read knowledge is to group it into *cases* – sets of facts, treated as units – so they can be used in analogical reasoning and learning. Analogy works best with interconnected relational structures, where relevant information is in the same case. This suggests going beyond the natural boundaries provided by language – paragraphs and sentences – and focusing instead on interconnected facts within the conceptual representations produced by semantic interpretation.

This paper describes two connectivity-based case construction mechanisms and evaluates them by comparing them to sentence-level and paragraph-level algorithms. We begin by summarizing the components and representations that the system uses. We then introduce four methods for organizing facts into cases. We follow with a description of an experiment and its results, and then close with related and future work.

2. Background

This section summarizes our learning by reading system, including its representations and its mechanisms for analogical matching and case construction. We discuss each in turn.

The system is based on the Companion cognitive architecture (Forbus, Klenk, & Hinrichs, 2009), which provides reasoning facilities (including analogy) that are used during language processing. It uses the Cyc representation language, ontology, and knowledge base contents,[1] which provide a large vocabulary of concepts (called *collections*), predicates, and several million facts constraining them. Here we use fixed-width font to denote symbols and expressions from the system. For example, `(isa solar-panel02 SolarCollector)` says that the entity `solar-panel02` is an instance of the collection `SolarCollector`. Cyc's language also provides a notion of logical environment via *microtheories*, local contexts linked via inheritance relationships. Its use of rich type-level representations, as well as microtheories for representing contexts, makes it a natural choice for expressing the kinds of complex information often communicated via language. Most of the lexical representations for verbs in Cyc are Davidsonian. They reify the event and use role relation predicates to connect the event to the actors that participate in it. For example, the full representation of "eats" in "A dolphin eats a fish" is:

```
(and
   (isa eat5642 EatingEvent)
   (performedBy eat5642 dolphin5637)
   (consumedObject eat5642 fish5672))
```

In this notation, `eat5642` is the eating event. The participants are `dolphin5637` and `fish5672`. The roles they play in the event are described by their relations: `performedBy` indicates that the actor intentionally performed the action, and `consumedObject` indicates that the object was affected and destroyed as part of the event. Cyc has an extensive hierarchy of role relations, but about ten such relations account for a majority of those used by the language system.

Our natural language system uses Allen's (1994) parser for syntactic analysis. Our semantic interpretation process is based on *Discourse Representation Theory* (DRT; Kamp & Reyle, 1993), which provides an account of scoping, including conditions and counterfactuals. Semantic interpretation builds up *discourse representation structures* (DRSs), each of which is a case containing one or more facts. These facts describe relationships among entities, collections to which those entities belong (i.e., category information), and other DRSs. Each sentence has an associated sentence DRS that contains the facts that represent the semantics of that sentence. For example, the sentence "The solar panel cools" has a sentence DRS that contains three facts:

```
(isa solar-panel02 SolarCollector)
(isa cool05 CoolingEvent)
(objectOfStateChange cool05 solar-panel02)
```

Sentence DRSs can also have *constituent DRSs*. For example, the sentence "If the valve closes, the flow of water stops" mentions a closing event and a stopping event, but does not say that either actually happened. If the system produced facts that said that a stopping event occurred and that the flow of water was what stopped, it might reach incorrect conclusions. For that reason,

[1] The URL `http://www.cyc.com` provides details about ResearchCyc.

rather than placing that information directly in the sentence DRS, the system creates two new constituent DRSs, one for the antecedent of the statement and one for the consequent. The sentence DRS then introduces a fact of the form (`implies DRS-01 DRS-02`). Constituent DRSs are also used to handle negation and hypotheticals, as well as other higher-order relationships between sets of facts.

Syntactic and lexical ambiguities are explicitly encoded as choice sets. The process of semantic interpretation involves constructing DRSs by selecting among choices from these alternatives. To factor out any domain or task specific influences, ambiguities were resolved automatically using a small set of general-purpose heuristics. For example, the reading system prefers interpretations that treat compound noun phrases like "solar panel" as atomic referents when they are available, such as `SolarPanel`. An interpretation that treated "solar panel" as a generic panel that is in some way related to the sun would be less preferred. Another heuristic prefers interpretations that include more facts. If the heuristics do not favor any of the choices in a set, the system chooses randomly. A truth maintenance system ensures that the system does not select incompatible choices.

Given that these heuristics do not involve any learned statistics, they perform quite well. Overall, they selected a correct answer for 86.6 percent of the 886 lexical choice sets generated by the source texts used in this paper. Errors generated by the heuristics tend to be systematic, which helps reduce analogical processing mismatches. For example, the system consistently selected `Pipe-SmokingDevice` over `Pipe-GenericConduit` as the interpretation of the word "pipe." This was incorrect; the corpus discusses rainwater collection systems and solar heating, and it does not discuss smoking. In the first occurrence of "pipe," the only applicable heuristic was random choice, and smoking device was selected. Subsequent choices were influenced by a heuristic that prefers concepts already used in the interpretation. Hence very similar, albeit partially incorrect, structures were created, facilitating analogical comparison.

A *discourse* is a multi-sentence section of a source text considered together. After parsing a source text, the system runs a discourse-level interpretation process that handles coreference resolution. The basic coreference strategy is to resolve pronouns, definite references ("the dolphin"), and verbs to the most recent valid referent, as determined by common collection membership and a few other factors (such as shared arguments, in the case of verbs). After coreference resolution, the system places the contents of the sentence DRSs from the text into a *discourse interpretation* DRS, with coreferent items resolved to the same symbol. This DRS frequently has many constituent DRSs, as every constituent DRS from one of the component sentences becomes a constituent DRS in the discourse interpretation.

For our experiments, the source text was simplified syntactically (Kuehne & Forbus, 2004). This process converts unsupported grammatical structures into supported ones. It does not completely eliminate syntactic ambiguity. For example, the source sentence "As for the solar heater, the sun has not yet risen" was simplified to "The sun has not yet risen on the solar heating system." Table 1 contains a paragraph from the simplified corpus. The simplification process places no broad-ranging restrictions on the lexicon, although there are occasional coverage gaps, particularly where compound nouns are concerned. For example, the unsupported "solar heater" became "solar heating system" in the sentence above.

Our evaluation involves comparing and contrasting generated cases using analogy. This uses the Structure Mapping Engine (SME; Falkenhainer, Forbus, & Gentner, 1989), a computational implementation of Gentner's (1983) structure mapping theory. SME takes two structured representations, the base of the analogy and the target, as input. These are the two cases that will be aligned with each other. It produces one or more *mappings*, which consist of three parts. First,

Table 1. The simplified version of one paragraph from the corpus.

At the start of each day, the solar heating system is semifull. At the start of each day, the rainwater collection system is semifull. The rainwater collection system could have some rainwater in it. The solar heating system's heat storage is warm because the solar heating system collected heat during the previous day. In the rainwater collection system, the valve closes. This prevents the flow of the stored water. The rain is not falling. The water in the pipe has leaked out. The sun has not yet risen on the solar heating system. The solar collector was exposed to the cold air in the night. The heat storage contains heat. Because of this, the heat storage's temperature is greater than the air's temperature. The control device sensed that the heat storage's temperature was greater than the solar collector's temperature. Because of this, the control device shut off the pump. This prevented the cooling process of the heat storage.

mappings include a set of *correspondences* between elements of the base and elements of the target. Second, they include a *score*, an estimate of match quality. SME attempts to produce the largest mappings it can that satisfy the constraints of structure-mapping theory. Each mapping also includes *candidate inferences*, hypotheses formed by filling out the target with parts of the base not represented in the target and vice versa. For our evaluation, the score is used to determine which mapping is the best, and only that mapping is used. The correspondences of that mapping can be thought of as things that the cases have in common, and the candidate inferences can be thought of as salient differences between the two cases. Because SME can project inferences in both directions, which case is the base and which is the target is immaterial here.

Dynamic case construction (Mostek, Forbus, & Meverden, 2000) is the process of building cases automatically from a body of knowledge. Almost all case-based reasoning systems require cases to be constructed by some external process. These are sometimes hand-curated, but that approach does not scale well. The ability to build focused cases from larger knowledge sources avoids manual curation, especially when combined with natural language understanding. The methods we describe in this paper can be thought of as extensions of dynamic case construction for learning by reading.

3. Case Construction Methods

The process of case construction in the context of learning by reading can be viewed as a problem of *segmentation*: how to divide the facts generated by the natural language system into useful cases. We have developed two connection-based methods that make use of properties of the knowledge in the interpretation, and two simpler techniques that use only naturally-occurring boundaries in text to serve as baselines. All four methods take the same inputs. The system reads a chapter from a source text and produces the discourse interpretation. Each case is built around a *seed*, which is a single mention of a single entity in the source text. For example, the term "solar heating system" in the sentence "At the start of each day, the solar heating system is semifull" was one such seed. We use this seed as an example throughout this section. The system can generate cases for each possible seed in the source text, or it can postpone generating cases until prompted to do so externally, such as when it is posed a question about a particular entity.

We start with the baseline methods, as they are simpler. The first algorithm is *local sentence interpretation* (LSI). This method takes the sentence where the seed occurs and uses its sentence interpretation – including its constituent DRSs – as the case. This makes intuitive sense as a baseline; most English-language sentences are about a single thing, so it is generally likely that a

Table 2. The sentence-based segmentation (SBS) method.

```
Procedure SBS.
Inputs: s, a seed entity.
Outputs: c, a set of facts for a case
  Let c = {}
  Let s.coref = {entities coreferent with s}∪{s}
     For each entity e in s.coref:
       Let t be the sentence that e appears in.
       For each fact f that was derived from t:
         Let c = adjoin(c, f)
       For each constituent DRS k derived from t:
         For each fact f in k:
           Let c = adjoin (c, f)
Return c.
```

sentence case contains mostly things that are directly relevant to reasoning about the things it mentions. For our example sentence, the information learned from the sentence interpretation is that there is a solar heating system, and at the start of a day it is partially full. The method is inexpensive, as no additional computation is required beyond what goes into understanding the sentence in the first place. A limitation of this method is that important information about an entity is often spread over multiple sentences. For our example seed, all of the facts derived from the sentence "At the start of each day, the solar heating system is semifull" are added to the case, and only those facts are adde. This is six facts in total.

The second baseline algorithm, *local paragraph interpretation* (LPI), is similar, but it uses all of the facts derived from all of the sentences in the paragraph of the source text in which the seed appears. It uses facts from the discourse interpretation case, rather than the individual sentence cases, so that coreferent symbols are resolved to each other, but otherwise operates like local sentence interpretation. Local paragraph interpretation is much less likely to miss important information about the seed, but it has two potential disadvantages. First, a paragraph can cover multiple topics, which increases the amount of noise in the case. Second, it is may be less useful for comparing or contrasting two seeds that are in the same paragraph, because the cases will be identical. We investigate this potential drawback in Section 4. Because the system enforces alignment between the seeds, there can still be some candidate inferences drawn, but structural alignment will result in many statements matching with themselves. For our example seed, the case created by this algorithm is very large, 178 facts, and covers a broad range of topics, as it contains the facts built from all 15 sentences in the paragraph.

The last two methods we propose exploit connectivity properties of the conceptual representation for the semantics of the sentence. Table 2 describes *sentence-based segmentation* (SBS), which creates a case by adding all of the facts derived from sentences where the seed is mentioned. This makes intuitive sense, as these facts are likely to be related to the topic in question, since they come from the same sentences. One major advantage of this strategy is that it includes facts from the discourse interpretation that are otherwise disconnected from the rest of the graph, but does not simply include everything. Facts that are in the same sentence as relevant ones are likely to be relevant themselves. Table 3 contains the case produced by sentence-based segmentation for our example seed. Note that these facts were drawn from four sentences, as there are four references to the seed in the source text, which coreference identifies.

Table 3. Example of facts in a case built using sentence-based segmentation. The case consists of 28 facts across four DRSs. Representations are simplified for space and readability. Note that some facts are incorrect, due to errors in automatic word sense disambiguation. Four sentences contained or referred to the seed: "At the start of each day, the solar heating system is semifull," "We examined the elements of a solar heating system," "We also compare the rainwater collection system and the solar heating system," and "We will examine the solar heating system operating during a typical day."

Holds in `Discourse-DRS-01`:

```
(evaluee-Direct compare02 rainwater-collection-system03)
(evaluee-Direct compare02 solar-heating-system01)
(evaluee-Direct examine04 element05)
(fullnessOfContainer solar-heating-system01 PartiallyFull)
(implies-DrsDrs DRS-02 DRS-03)
(isa compare02 Comparing)
(isa day06 ObservanceDay) ;; WSD error – "day"
(isa examine04 Inspecting)
(isa group-of-element05 Set-Mathematical)
(isa solar-heating-system01 SolarHeatingSystem)
(performedBy compare02 we07)
(performedBy examine04 we08
(possessiveRelation day06 start09)
(startingPoint day06 start09)
(temporallyIntersects start09 (StartFn be10))
(willBe DRS-04)
```

Holds in `DRS-02`:

```
(member element05 group-of-element05)
```

Holds in `DRS-03`:

```
(isa element05 ElementStuffTypeByNumberOfProtons) ;; WSD error – "element"
(isa solar-heating-system01 SolarHeatingSystem)
(possessiveRelation solar-heating-system01 element05)
```

Holds in `DRS-04`:

```
(conceptuallyRelated day11 Normal-Usual)
(evaluee-Direct examine04 solar-heating-system01)
(isa solar-heating-system01 SolarHeatingSystem)
(performedBy examine04 we08)
(temporallySubsumes day11 operate12)
(isa day11 ObservanceDay) ;; Word sense disambiguation error
(isa examine04 Inspecting)
(isa operate12 Surgery) ;; Word sense disambiguation error
```

The second connection-based case construction method that we developed, *fact-based segmentation* (FBS; see Table 4), is inspired by the method described in Mostek et al. (2000), but is adapted to use the interpretations produced by the language system. Fact-based segmentation

Table 4: The fact-based segmentation (FBS) method.

```
Procedure FBS.
Inputs: s, a seed entity; dMax, a maximum depth.
Outputs: c, a set of facts for a case
 Let eExtended = {s} ;; The entities that have already been used.
 Let c = gatherFactsForEntity(s, 0, dMax)
 Return c.

Procedure gatherFactsForEntity.
Inputs: eCur, the current extension entity; depthCur, the current depth;
        dMax, a maximum depth.
Outputs: cExtend, a set of facts.
 Let cExtSameDepth = {} ;; The facts added at the same depth
 Let cExtNextDepth = {} ;; The facts added at the next depth
 Let cExtend = {}
 For each fi in {isa facts that mention eCur}:
  Let cExtSameDepth = adjoin(cExtSameDepth, fi)
  For each ji in {isa facts in fi's paragraph that use fi's collection}:
   Let cExtNextDepth = adjoin(cExtNextDepth, ji) ;; Add at the next depth.
 For each fn in {non-isa facts that mention eCur}:
  Let cExtNextDepth = adjoin(cExtNextDepth, fn) ;; Add at the next depth.
 Let cExtSameDepth = cExtSameDepth∪getDRSFacts(cExtSameDepth)
 Let cExtNextDepth = cExtNextDepth∪getDRSFacts(cExtNextDepth)
 Let cExtend = cExtNextDepth∪cExtSameDepth
 For each fc in cExtSameDepth:
  For each eNext in {entities mentioned in fc}:
   Unless member(eNext, eExtended):
    Let eExtended = adjoin(eExtended, eNext)
    Let cExtend = cExtend∪gatherFactsForEntity(eNext, depthCur, dMax)
 Unless depthCur = dMax:
  For each fc in cExtNextDepth:
   For each eNext in {entities mentioned in fc}:
    Unless member(eNext, eExtended):
     Let eExtended = adjoin(eExtended, eNext)
     Let cExtend = cExtend∪gatherFactsForEntity(eNext, depthCur + 1, dMax)
 Return cExtend.

Procedure getDRSFacts.
Inputs: extendCase, a set of facts.
Outputs: DRSFacts, a set of facts.
 Let DRSFacts = {}
  For each fc in extendCase:
   If fc is contained within a constituent DRS cDrs:
    For each fccDrs in {facts in cDrs}:
     Let DRSFacts = adjoin(DRSFacts, fccDrs)
    For each fmcDrs in {facts that mention cDrs}:
     Let DRSFacts = adjoin(DRSFacts, fmcDrs)
 Return DRSFacts.
```

starts with a seed entity that is passed to the `gatherFactsForEntity` procedure. Facts are added to the case at the same level if they are `isa` facts that mention that entity. They are added to the case at the next level down either if they are `isa` facts that do not mention the entity but

Table 5. A subset of the facts found using the FBS algorithm, which added 63 facts in total across six DRSs. Representations are simplified for space and readability.

In `Discourse-DRS-01`:
`(isa solar-heating-system01 SolarHeatingSystem)`
`(fullnessOfContainer solar-heating-system01 PartiallyFull)`
`(isa flow16 FluidFlow-Translation)`
`(isa prevent15 (PreventingFn flow16))`
`(primaryObjectMoving flow16 water17)`
`(not DRS-08)`
In `DRS-08`:
`(isa rise13 AscendingEvent)`
`(isa solar-heating-system01 SolarHeatingSystem)`
`(objectMoving rise13 sun14)`
`(on-UnderspecifiedSurface sun14 solar-heating-system01)`

use the same collection as one of the entity's `isa` facts, or if they are non-`isa` facts that mention that entity. If any of these facts mention a constituent DRS, the facts in that DRS are added at the same level. If any facts are contained in a constituent DRS, the facts in that DRS are added at the same level. This is done by the procedure `getDRSFacts`. From there, the system goes through each entity mentioned in a fact that it added to the case on that round. Each of these is extended, using `gatherFactsFromEntity`, just as the seed was extended. Entities at the maximum depth are not extended, and nor are ones that have already been extended, which prevents loops.

Working through an example of the fact-based segmentation method in action can be useful. Recall our example seed, the term "solar heating system" in the sentence "At the start of each day, the solar heating system is semifull." The method begins by selecting the discourse variable derived from that seed, `solar-heating-system01`. When the case mentions an entity, FBS adds facts that mention the entity to the case. This means that the facts `(fullnessOf-Container solar-heating-system01 PartiallyFull)` and `(isa solar-heating-system01 SolarHeatingSystem)` are added, in the contexts in which they appear. The first appears in the top-level DRS, `Discourse-DRS-01`, whereas the second appears in several constituent DRSs, including `DRS-08`. When FBS adds an `isa` fact to the case, other `isa` facts in the paragraph that use the same collection are also added. For example, if there were other instances of `SolarHeatingSystem` in the paragraph, such as `(isa solar-heating-system35 SolarHeatingSystem)`, the facts indicating this would be added. This lets the system include multiple instances of the same type of thing and produce better cases by providing a mechanism for quickly reaching related facts. When a constituent DRS is mentioned in a fact in the case or a fact contained in a constituent DRS is added, facts that mention that constituent DRS and facts inside that constituent DRS are added to the case. Here, when FBS adds a fact that is contained in `DRS-08` to the case, it adds `(not DRS-08)` to the case. When a fact contained in or mentioning a constituent DRS is in the case, the rest of the facts in that constituent DRS are added. Because the case contains a fact in `DRS-08`, this rule adds the other facts in `DRS-08`. A fact present in multiple DRSs may appear in the final case more than once, contextualized in different DRSs.

All facts in a given DRS (except for the top-level sentence) are added to the case at the same depth. Adding a complete constituent DRS is crucial for accuracy. For example, consider the sentence "When the temperature of the heat storage equals the temperature of the solar panel, the

Table 6. Potential strengths and weaknesses of each of the four case construction methods.

Method	Avoids Noise	Captures Relevant Statements	Other
LSI	Excellent	Poor	Misses any information not in the single sentence
LPI	Poor	Excellent	Produces identical cases for seeds in the same paragraph Produces very large cases
SBS	Excellent	Average	Can miss relevant information even from nearby sentences
FBS	Average	Excellent	Most CPU time-intensive to produce

heat does not flow." Leaving out any part of the DRS constituent structure, such as the antecedent information or the negation, would change the meaning of the sentence. We chose a maximum depth of three, based on examination of pilot data. For our example seed, the resulting case had 63 facts. With a depth of two, the case produced had only 60 facts, and for a depth of four, the case produced had 115 facts. Table 5 shows some of the facts that were added to the case for our example seed.

One apparent advantage of this method is that it incorporates connections across multiple sentences. This should be useful in cases where a topic is only mentioned once but is elaborated upon across several sentences. While some sentences that elaborate on a topic may continue to refer to it directly, there may be relevant information in sentences that do not. For example, in "The dolphin often has one calf. The calf is weaned after one year," if the seed being used is the instance of "dolphin" in the first sentence, sentence-based segmentation will not include any information from the second sentence, as it does not explicitly mention the dolphin. Fact-based segmentation will include information from the second sentence, as it chains through facts that share entities.

This section introduced four methods for building cases from a discourse interpretation. Two of these, local sentence interpretation and local paragraph interpretation, use only natural boundaries in the text. Sentence-based segmentation and fact-based segmentation take advantage of other connections between things in the interpretation. The methods have different strengths and weaknesses, as summarized in Table 6. One of the greatest differences is size of the cases they produce. Table 7 shows the number of facts that are added to a case created by each of the methods when used on the example seed.

Table 7. The number of facts in a case constructed around the example seed for each of the four case construction methods.

Method	No. of Facts
Local Sentence Interpretation	6
Local Paragraph Interpretation	178
Sentence-based Segmentation	28
Fact-based Segmentation	63

4. Experimental Design and Evaluation

Given the different potential strengths and weaknesses of the four methods, we designed an experiment to evaluate them. The hypothesis tested was that the connection-based construction methods, fact-based segmentation and sentence-based segmentation, will produce more effective

cases, as measured by their performance on a compare and contrast task, and produce more compact cases as well. Local sentence interpretation and local paragraph interpretation are the baseline methods. We use a compare and contrast task for evaluation because it is one of the most straightforward kinds of analogical reasoning, and it has interesting potential applications (Brüninghaus & Ashley, 2001; Peterson et al., 1994; Chaudhri et al., 2014). Given two entities in the text, the system compares and contrasts cases generated using those entities as seeds. For the study, we created two corpora from preexisting source texts, described next.

The first source text was chapter 16 of *Sun Up to Sun Down* (SUSD; Buckley, 1979), a book about solar energy and solar heating that makes extensive use of analogies. We selected chapter 16 because it uses an extended analogy to explain a solar heating system in terms of rainwater collection. The simplified version of the chapter consists of 80 sentences in 11 paragraphs; Table 1 shows a sample. The interpretation process produced 733 facts across 54 DRSs. The second source text was a Diffen article that discusses dolphins and porpoises (Dolphin vs. Porpoise). Diffen is an online, user-editable encyclopedia. Its articles compare and contrast similar topics. We chose this article because it differed in both style and subject matter from the other source text. After simplification, the article was eight paragraphs and 88 sentences long. The interpretation process produced 751 facts across 150 DRSs.

We selected pairs of seeds for which the similarities and differences would be illuminating, based on the information available in the texts. For example, after reading the SUSD text, we tasked the system with contrasting the state of a solar heating system at different points during the day and comparing it to an analogous rainwater collection system. All of the tasks related to the Dolphin/Porpoise text involved comparing different aspects of the two creatures, such as their physical anatomy or mating habits. Although the system is capable of comparing any pair of objects or events, in most cases arbitrary comparisons are not very interesting. In total, we asked the system for 24 comparisons, 11 from the SUSD text and 13 from the Dolphin/Porpoise text. In every comparison, the seeds in the topic pair placed an additional constraint on the analogy mapping: SME was only allowed to produce mappings where the seeds correspond to each other.

We used two primary evaluation metrics. The first is *recall* on identifying similarities and differences between the things being compared. For each comparison, we wrote two to 15 *goal facts*, prior to running any of the methods over them. Across the 24 comparisons, 118 goal facts were used in total, with 64 of these coming from the SUSD text and 54 from the Dolphin/Porpoise text. We discarded seven additional goal facts because they relied on context not given directly in the text, meaning that no case construction method could generate them. Each goal fact represented one similarity or difference in the text. The score for a method, given a pair of cases, was equal to the number of goal facts that it found. Similarities were scored if the similarity was among the correspondences produced by SME. Differences were scored if the difference was among the candidate inferences produced by SME. For example, when cases produced from our example seed were compared to a rainwater collector system seed, all of the methods produced a correspondence between

```
(fullnessOfContainer solar-heating-system01 PartiallyFull)
```

and

```
(fullnessOfContainer rainwater-collection-system01 PartiallyFull).
```

The system derived these facts from “At the start of the day, the solar heating system is semifull” and “At the start of the day, the rainwater collection system is semifull,” respectively. This indicates that the system could tell that one commonality between the rainwater collection system and the solar heating system, in the scenario being described, is that they are both partially full.

Table 8. Experimental results for four methods: local sentence interpretation (LSI), local paragraph interpretation (LPI), sentence-based segmentation (SBS), and fact-based segmentation (FBS).

Method	**LSI**	**LPI**	**SBS**	**FBS**
Goal Facts Found	27	81	59	88
Goal Facts Found (%)	22.9	68.6	50	74.5
Generation Efficiency (%)	8.4	2.5	8.3	3.7
Unique Correct	0	8	3	8
Average Case Size	8.9	107.2	16.1	66.8
Average CIs	8.4	49.9	19.5	52.0
Average Correspondences	3.4	69.7	7	35.3

Goal facts are compositional. This rewards more complete answers while still providing some credit for partial answers. For example, rather than using "In the second case, heat flows from the solar collector to the storage tank" as a single example of a difference between two cases, we break it into three statements: One saying that a flow of heat exists, which appears as `(objectMoving flow01 heat15)`, one saying that that flow leaves from the solar collector, which appears as `(fromLocation flow01 solar-collector24)`, and one saying that flow goes to the storage tank, which appears as `(toLocation flow01 storage-tank35)`. We did not include simple features as separate goal facts to be found. For example, when comparing a solar heating system's operation at different times of the day, the fact that it is a solar heating system in both cases was not among the goal facts, all of which were relationships between two entities.

The second evaluation metric is *generation efficiency*. This is the percentage of fact-level correspondences and candidate inferences that were used to produce goal facts. It penalizes the inclusion of facts that were not useful for this compare and contrast task, although such facts might be useful for other tasks. This is important because larger cases are more computationally expensive to reason over. A secondary benefit of smaller cases is that they are easier for humans to read.

We did consider alternate evaluation metrics. Simply looking for the presence of important facts in cases would not be effective, because facts must show up in both cases in a comparison and must be alignable to be useful. Identifying goal facts involving similarities and differences is more objective, providing a gold standard for the task of comparing learned knowledge.

We ran each of the four methods on both of the texts, measuring the number of goal facts each found and the generation efficiency. The results appear in Table 8. The results from the two source texts are combined, as they are generally comparable across the board. Each column presents results for one of the methods described earlier – local sentence interpretation (LSI), local paragraph interpretation (LPI), sentence-based segmentation (SBS), and fact-based segmentation (FBS). *Goal Facts Found* is the number of the goal facts the method produced, whereas *Goal Facts Found (%)* is the percentage of the goal facts.

Unique Correct is the number of goal facts for which the method was the only one that correctly produced it, making it a measure of each method's ability to produce interesting conclusions that the others did not. In some instances, including most in which LPI was the only method to produce a goal fact, it is because the others did not include the relevant facts. In other instances, such as when SBS was the only method to produce the goal fact, one or more of the

Table 9. Experimental results on cases in which the base and target seeds appear in the same paragraph.

Method	**LSI**	**LPI**	**SBS**	**FBS**
Total Correct	20	35	33	41
Correct (%)	32.3	56.5	53.2	66.1

other methods did include the relevant facts, but failed to produce the proper correspondence or candidate inference.

Average Case Size is the average size of the bases and targets produced by the system when constructing cases based on the seeds. As there is no functional difference between bases and targets for this task (candidate inferences are produced in both directions), their sizes are simply averaged in the table. *Average CIs* is the number of candidate inferences produced by the system when comparing cases generated using the method. *Average Corrs* is the average number of fact-level correspondences.

The accuracies of FBS and of LPI are very similar. While the two methods found different sets of goal facts, the difference in their overall performance was not statistically significant. The difference between the performance of LSI and the other three methods was statistically significant ($p < 0.005$), as was the difference between SBS and the other three. In total, 100 of the 118 goal facts (84.7%) were produced by at least one of the methods.

One of the theoretical weaknesses of LPI is that it might suffer in instances where both of the seeds appear in the same paragraph. This is because it produces pairs of cases that contain all of the same facts in those instances. Because the seeds (which are different) are automatically mapped to each other, it still produces some useful correspondences and candidate inferences. However, we might suspect that it may still be disadvantaged, as the system is likely to map many entities to themselves. To test this, we looked at the results on only the comparisons made between entities in the same paragraph. In total, 62 goal facts came from comparisons of this type. Table 9 shows the accuracy on just those goal facts. LPI's performance is worse than on the full set, but the difference is not statistically significant, and the difference between LPI and FBS on this limited set remains insignificant.

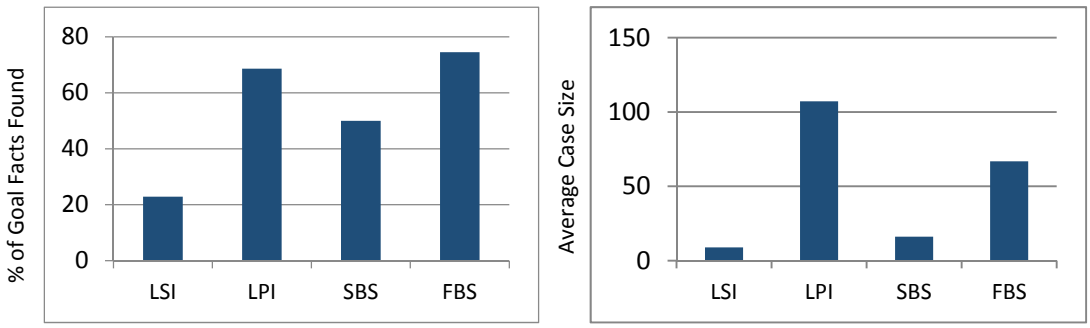

Figure 1. Percent of goal facts found and the average case size of each method.

In summary, the experiment compared four methods for generating cases from a discourse interpretation produced by a natural language understanding system, evaluating them on a realistic task. We found that fact-based segmentation and local paragraph interpretation are the

best of the four methods on the primary measure. This partially falsified the hypothesis that fact-based segmentation and sentence-based segmentation would produce superior performance. Fact-based segmentation had the advantage that it produced the best accuracy results while achieving significantly higher generation efficiency than local paragraph interpretation. This is valuable, as smaller cases are more efficient for analogical comparison and other reasoning processes. This comes at the cost of a small amount of additional overhead during the case construction step, as producing cases with fact-based segmentation requires extra computation. Figure 1 presents the percent of goal facts found and the average case size for each method.

That local sentence interpretation fared the worst is not surprising. It produced relatively little noise, as indicated by its high generation efficiency, but information is spread across multiple sentences too frequently for it to get the information required to make broader comparisons between two entities. Even in situations where smaller cases are desirable, sentence-based segmentation produced much better results with only moderately larger cases.

There appears to be a tradeoff between accuracy and generation efficiency. This result is reasonable; methods that produce more facts are less likely to produce the Facts Not In Case error, provided that the additional facts being added are at least potentially useful. In some instances, including additional facts in the case produced SME Alignment Mismatch errors, but at the case sizes produced by the four methods described here, those are rare.

There are several advantages to producing smaller cases, as fact-based segmentation does. First, they are easier for a human to read and understand. Even with the aid of visualization tools or natural language generalization, large cases can be difficult for a human reader to interpret, especially if they contain a great deal of superfluous information. Second, smaller cases can be processed more quickly in many case-based reasoning tasks. SME is efficient, operating in $O(n^2 \log(n))$ time, but it still operates faster over smaller cases.

5. Error Analysis

One challenge in evaluating the system's overall performance is that assigning blame for failures is not easy. We conducted an error analysis to explore this issue. This provides additional insight into the strengths and weaknesses of each method. Each time a method missed a goal fact, we classified the source of the error. For example, LSI can miss a similarity goal fact by failing to include the facts in the cases to begin with, while FBS can miss the same goal fact if it includes the corresponding facts but they do not align when the cases are compared using SME. Some types of errors can occlude others. For instance, if a case construction method fails to include the relevant facts when building the case, it is impossible for those facts to end up being mismatched in the SME mapping. Table 10 summarizes the sources of error for each method.

Facts Not In Case refers to errors in which key facts were in the global interpretation, but not included by the case construction method. Unsurprisingly, the methods that, on average, build larger cases were much less likely to produce such errors. This type of error accounts for the vast majority of the instances where local sentence interpretation failed to produce a result, as any goal fact that needed information from another sentence would fail. For example, in the Dolphin/ Porpoise corpus, one of the goal facts when comparing the anatomy of the two creatures is that dolphins have conical teeth, while porpoises have flat teeth. Because these facts are in different sentences from the one that introduces the comparison between the two anatomies, the cases produced by local sentence interpretation lack them.

Table 10. Sources of error for each of the four methods.

Method	LSI	LPI	SBS	FBS
Facts Not In Case	85	1	49	11
SME Representation Mismatch	2	4	2	4
Wrong Interpretation Choice	2	2	4	3
SME Alignment Mismatch	2	27	6	12

SME Representation Mismatch refers to errors in which facts representing a similarity were present in the base and the target, but were sufficiently different representationally that SME did not match them. Although local paragraph interpretation and fact-based segmentation missed more goal facts as a result of this error, they are not more likely to produce representation mismatches. All four case construction methods start with the same global interpretation, and all use the same representations as a result. The reason that LPI and FBS missed more goal facts as a result of this error is that, in some instances, LSI and SBS did not include any relevant facts at all. An example of where this error occurred is when the system compared the rainwater collection system to the solar heating system while both are operating. The rainwater collection system is collecting rainwater, and the solar heating system is collecting heat. One of the similarity goal facts was that both the rainwater and the heat are being collected. However, the way the source text phrases the sentences that provide this information resulted in different representations. The text says that the solar collector "absorbs the sunlight" ("solar radiation is absorbed," in the presimplified version.) Because the statements the system produces for absorption events are not similar to the statements for falling events, the falling of rainwater onto the system's collection tray does not align with the absorption of heat by the solar panel. Repairing such issues would require rerepresentation (Yan, Forbus, & Gentner, 2003).

Wrong Interpretation Choice is very similar to SME Representation Mismatch, but refers to errors in which the mismatch can be traced to the disambiguation heuristics making an incorrect choice. As noted above, the heuristics used were not perfect; some 13.4 percent of the semantic ambiguities were resolved incorrectly. However, such disambiguation errors often do not affect the SME matches. Only disambiguation errors that led to mismatches were coded as this type of error.

SME Alignment Mismatch refers to errors in which the base and the target both contain the necessary fact or facts to identify a similarity or difference, but SME did not produce an appropriate correspondence or candidate inference because the match did not align appropriately. An example of this error is when the system was asked to compare the anatomy of the dolphin and the porpoise. One of the goal facts is that dolphins have long noses, while porpoises have flat noses. Because these facts appeared in the same paragraph, local paragraph interpretation produced a base and a target that each included both cases. Although it aligned dolphin with porpoise, as constrained by the question asked, it did not align the dolphin's nose in the base with the porpoise's nose in the target. Rather, it aligned the dolphin's nose in the base with the dolphin's nose in the target, and it did the same with the porpoise's nose. As a result, the system drew no candidate inferences about the creatures' noses.

6. Related Work

Several lines of research address similar issues. Case-based reasoning (Schank & Cleary, 1994) uses existing cases to solve problems and answer questions. Such systems sometimes use domain-specific retrieval and matching systems, unlike our use of a general-purpose analogical matcher, SME. Textual CBR systems have generally focused on building cases from text resources with the goal of retrieving relevant source texts, rather than building formally represented cases that can be used for reasoning, as we do. Generally this has involved minimal natural language processing. Brüninghaus and Ashley (2001) describe SMILE, which uses natural language processing in the legal domain to build more sophisticated cases that can be more accurately compared to each other, in contrast to methods that use bag-of-words techniques, but it relies on manual identification of important features. Gupta and Aha (2004) describe FACIT, a textual case-based reasoning system that uses logical forms as its representations. Like most such systems, it operates by using the generated cases to index texts, rather than reasoning directly over the cases produced. Compare&Contrast (Liu, Wagner, & Birnbaum, 2007) uses the Web as a source to find cases similar to a seed case. Rather than parsing the entire source, it builds vectors of named and non-named entities to represent the contents. Such feature-based representations cannot support the kinds of explanation generation that we can, given our use of relational representations.

One way in which our approach to learning by reading differs is in its representations. Our approach produces more comprehensive, structured representations than some other systems, at the cost of additional computational overhead. DART (Clark & Harrison, 2009), NELL (Carlson et al., 2010), and KNEXT (Van Durme & Schubert, 2009) are other efforts in knowledge extraction that produce logical forms. These systems handle a broader range of syntax than ours, but the representations they produce are simpler. The PRISMATIC knowledge base, used in IBM's Watson project (Fan et al., 2012), uses even simpler encodings, treating words themselves as predicates. This is good for factoid question answering, but less so for reasoning tasks. West et al. (2014) describe a system for targeting the Web with specific queries in order to extend Freebase (Bollacker et al., 2008), filling in certain types of missing knowledge. These systems uses less expressive representational vocabularies than our combination of DRT and Cyc provide.

KA (Peterson, Mahesh, & Goel., 1994) is a proposed system that resembles ours in that it would construct cases from text and compare them to other cases. This would let it diagnose errors in the design of physical systems. The system described here extends this idea, and is fully implemented and more general, as it is not tied to any particular domain.

Previous work in constructing cases from discourse interpretations includes analogical classification of dialogue acts (Barbella & Forbus, 2011). This approach creates cases from textual analogies by classifying sentences based on their role in establishing or extending an analogy. It then uses those classifications to determine which statements are part of the base and the target of the analogy. Like the methods described in this paper, it makes use of connectivity properties of semantic interpretations.

7. Conclusions and Future Work

In summary, we compared four methods for producing cases from a discourse interpretation produced by a natural language understanding system. We evaluated them by comparing their ability to produce cases amenable to comparison and contrast tasks. The efficiency of the cases served as a secondary evaluation metric. We also examined the types of error that the different

methods incur. We found that fact-based segmentation and local paragraph interpretation produce cases that best support comparison and contrast, with the former producing more compact cases.

Looking to the future, there are several areas where additional improvements and testing are possible. One property of the fact-based segmentation method is that it depends on coreference resolution, an area where our language system could be better. Currently, fact-based segmentation uses common collection membership to determine that topics are related even where entities are not coreferent. Incorporating additional means of discovering topic similarity could be a useful refinement. We also plan to explore whether combining our two connection-based methods would produce further improvements. Another extension is to exploit cases for analogical retrieval (Forbus et al., 1997; Forbus, Gentner, & Law, 1995) to answer other types of questions using analogy (e.g., Klenk & Forbus, 2009). The system might be able to learn models of concepts via analogical generalization (McLure & Forbus, 2012) using cases constructed via these methods.

Currently, the system either produces cases for every possible seed after reading or waits to produce cases until prompted to do so by a relevant question. It can also produce cases for a particular seed that is targeted. Another extension would automatically identify which entity is likely to be the most useful seed for a case. For example, the entity that names the topic of a paragraph may be a better seed than an arbitrary entity from later in that paragraph. Taken together, these extensions would allow the system to operate more fully autonomously and be used on a wider range of reasoning tasks.

Acknowledgements

This research was supported by Grant N00014-14-1-0111 from the Intelligent and Autonomous Systems Program of the Office of Naval Research.

References

Allen, J. F. (1994). *Natural language understanding (2nd Ed.)* Redwood City, CA: Benjamin/ Cummings.

Barbella, D., & Forbus, K. (2011). Analogical dialogue acts: Supporting learning by reading analogies in instructional texts. *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence* (pp. 1429–1435). San Francisco: AAAI Press.

Barker, K., Agashe, B., Chaw, S., Fan, J., Glass, M., Hobbs, J., Hovey, E., Israel, D., Kim, D., Mulkar, R., Patwardhan, S., Porter, B., Tecuci, D., & Yeh, P. (2007). Learning by reading: A prototype system, performance baseline, and lessons learned. *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence* (pp. 280–286). Vancouver, BC: AAAI Press.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., & Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. *Proceedings of the 2008 ACM SIGMOD international conference on Management of data* (pp. 1247–1250). ACM.

Brüninghaus, S., & Ashley, K. D. (2001). The role of information extraction for textual CBR. *Case-based reasoning research and development* (pp. 74–89). Vancouver, BC: Springer.

Buckley, S. (1979). *Sun up to sun down*. New York: McGraw-Hill.

Carlson, A., Betteridge, B., Kisiel, B., Settles, E. R., Hruschka, E. R., & Mitchell, T. M. (2010). Toward an architecture for never-ending language learning. *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence* (pp. 1306–1313). Atlanta, GA: AAAI Press.

Clark, P., & Harrison, P. (2009). Large-scale extraction and use of knowledge from text. *Proceedings of the Fifth International Conference on Knowledge Capture* (pp. 153–160). Redondo Beach, CA: ACM.

Chaudhri, V., Heymans, S., Spaulding, A., Overholtzer, A., & Wessel, M. (2014). Large-scale analogical reasoning. *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence* (pp. 359–365). Québec City, QC: AAAI Press.

Dolphin vs Porpoise. (n.d.). Retrieved February 9, 2015, from `http://www.diffen.com/difference/Dolphin_vs_Porpoise`.

Falkenhainer, B., Forbus, K., & Gentner, D. (1989). The Structure-Mapping Engine: Algorithm and examples. *Artificial Intelligence, 41*, 1–63.

Fan, J., Kalyanpur, A., Gondek, D. C., & Ferrucci, D. A. (2012). Automatic knowledge extraction from documents. *IBM Journal of Research & Development, 56*, 5:1–5:10.

Forbus, K., Gentner, D., Everett, J. O., & Wu, M. (1997). Towards a computational model of evaluating and using analogical inferences. *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society* (pp. 229–234). Palo Alto, CA: Lawrence Erlbaum.

Forbus, K., Gentner, D., & Law, K. (1995). MAC/FAC: A model of similarity-based retrieval. *Cognitive Science, 19*, 141–205.

Forbus, K, Klenk, M., & Hinrichs, T. (2009). Companion Cognitive Systems: Design goals and lessons learned so far. *IEEE Intelligent Systems, 24*, 36–46.

Forbus, K., Riesbeck, C., Birnbaum, L., Livingston, K., Sharma, A., & Ureel, L. (2007). Integrating natural language, knowledge representation and reasoning, and analogical processing to learn by leading. *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence* (pp. 1542–1547). Vancouver, BC: AAAI Press.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science, 7*, 155–170.

Gentner, D., & Gunn, V. (2001). Structural alignment facilitates the noticing of differences. *Memory and Cognition, 29*, 565–577.

Grishman, R., Macleod, C., & Wolff, S. (1993). *The COMLEX Syntax Project*. Ft. Belvoir, MD: Defense Technical Information Center.

Gupta, K. M., & Aha, D. W. (2004). Towards acquiring case indexing taxonomies from text. *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference* (pp. 172–177). Clearwater Beach, FL: AAAI Press.

Kamp, H., & Reyle, U. (1993). *From discourse to logic: Introduction to model-theoretic semantics of natural language*. Boston, MA: Kluwer Academic.

Klenk, M., & Forbus, K. D. (2009). Domain transfer via cross-domain analogy. *Cognitive Systems Research, 10*, 240250.

Kuehne, S., & Forbus, K. (2004). Capturing QP-relevant information from natural language text. *Proceedings of the Eighteenth International Qualitative Reasoning Workshop*. Evanston, IL.

Liu, J., Wagner, E., & Birnbaum, L. (2007). Compare&Contrast: Using the Web to discover comparable cases for news stories. *Proceedings of the Sixteenth International World Wide Web Conference* (pp. 541–551). New York: ACM.

Macleod, C., Grisham, R., & Meyers, A. (1998). *COMLEX syntax reference manual, Version 3.0.* Linguistic Data Consortium, University of Pennsylvania.

McFate, C.J., Forbus, K. and Hinrichs, T. (2014). Using narrative function to extract qualitative information from natural language texts. *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence* (pp. 373–379). Québec City, Québec.

McLure, M., & Forbus, K. (2012). Encoding strategies for learning geographical concepts via analogy. *Proceedings of the Twenty-Sixth International Workshop on Qualitative Reasoning*. Los Angeles, CA.

Mostek, T., Forbus, K. D., & Meverden, C. (2000). Dynamic case creation and expansion for analogical reasoning. *Proceedings of the Seventeenth National Conference on Artificial Intelligence* (pp. 323–329). Austin, TX: AAAI Press.

Peterson, J., Mahesh, K., & Goel, A. (1994). Situating natural language understanding within experience-based design. *International Journal of Human-Computer Studies, 41*, 881–913.

Schank, R., & Cleary, C. (1994). *Engines for education*. Hillsdale, NJ: Lawrence Erlbaum.

Tomai, E., & Forbus, K. (2009). EA NLU: Practical language understanding for cognitive modeling. *Proceedings of the Twenty-First International Florida Artificial Intelligence Research Society Conference* (pp. 117–122). Sanibel Island, FL: AAAI Press.

Van Durme, B., & Schubert, L. (2008). Open knowledge extraction through compositional language processing. *Symposium on Semantics in Systems for Text Processing* (pp. 239–254). Stroudsburg, PA: Association for Computational Linguistics.

West, R., Gabrilovich, E., Murphy, K., Sun, S., Gupta, R., & Lin, D. (2014). Knowledge base completion via search-based question answering. *Proceedings of the 23rd International Conference on World Wide Web* (pp. 515–526). International World Wide Web Conferences Steering Committee.

Yan, J., Forbus, K., & Gentner, D. (2003). A theory of rerepresentation in analogical matching. *Proceedings of the Twenty-Fifth Annual Conference of the Cognitive Science Society* (pp. 1265–1270). Mahwah, NJ: Lawrence Erlbaum.