NORTHWESTERN UNIVERSITY

Analogy in Learning by Reading

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Computer Science

By

David M. Barbella

EVANSTON, ILLINOIS

September 2016

Abstract

Natural language understanding is an important problem in artificial intelligence, and a challenging one. Analogy is a powerful tool that has been applied to a number of important AI tasks. This thesis describes three contributions that integrate language understanding and analogical reasoning. The first contribution is to demonstrate that analogical retrieval of previously stored sentence choice cases can be used for semantic disambiguation. This includes both word sense disambiguation and other forms of semantic disambiguation. This allows a system to avoid introducing error while reading. The second contribution is to show that by exploiting the connectivity properties of the semantic interpretation of a text, a cognitive system can extract subsets of that interpretation that are useful for analogical reasoning tasks. The third contribution is an ontology of a*nalogical dialogue acts* that can be used to improve understanding of text passages that contain explicit analogies. This allows a system to make use of textual analogies, rather than having them be a source of noise. All three contributions are supported by experiments that demonstrate their utility in improving understanding. The work described in this dissertation is integrated into a larger cognitive architecture.

Acknowledgements

First, I would like to thank Ken Forbus. Without his guidance as my advisor, this work would not exist. I feel truly fortunate to have had the opportunity to be his student. I would also like to thank my committee members, Larry Birnbaum, Doug Downey, and Chris Riesbeck for their support and feedback. I am grateful to the Office of Naval Research (Grant N00014-14-1-0111) for supporting this research.

I owe an incredible debt to my fellow QRG members as well. Everybody who I shared time with in the lab was a valuable collaborator. Special thanks to Tom Hinrichs, who acted as a constant sanity check and whose deep understanding of both the principles and implementation of the systems we use was invaluable. Thanks also to CJ McFate, whose linguistics knowledge and insight was especially critical. Thanks to Maria Chang, Subu Kandaswamay, Matt McLure, and Jon Wetzel for their camaraderie on the drive to the finish, and to my seniors in the lab for their advice and candor.

I am grateful to my friends in the Northwestern Graduate Christian Fellowship, in my church families, and elsewhere. I love my work, but it's important to have a life outside of it, and you provided that.

To my parents, Peter Barbella and Trina Tiffany, it isn't possible to encompass everything you've done for me in this space. Thank you for always encouraging me to pursue my education and for going out of your way to cultivate my interest in learning. Thank you for your continued support as I have worked on this dissertation. Thanks also to my sister, Lisa, for her friendship and support.

Contents

1 Introduction

1.1 Motivation

This work explores uses for analogy in learning by reading. Analogy is a powerful tool, and we present multiple ways to apply it to an important problem in artificial intelligence.

One of the greatest challenges for artificial intelligence is that many reasoning tasks require extensive background knowledge. Many current reasoning systems are able to operate with a minimal set of knowledge by constraining their domain or by offering only a limited level of explanatory depth. However, as we build systems that operate in broader domains, those systems will need extensive knowledge to ground their reasoning. Unfortunately, manual knowledge engineering is expensive, scales poorly and requires considerable AI expertise. The ability to build up knowledge without a human in the loop is critical if cognitive systems are to expand their capabilities. The difficulty of acquiring formally represented knowledge is known as the *knowledge acquisition bottleneck* (Feigenbaum, 1980).

The vast collection of natural language text that exists in the world is one of the most potent resources available to cognitive agents. Not only do sources exist detailing all manner of well-defined physical systems, but descriptions of everyday and common-sense scenarios could potentially be used to learn about those areas as well. A cognitive agent able to exploit those resources would have a huge advantage. In 2011, IBM earned global attention when Watson, a question-answering system they developed, successfully defeated two human former champions on *Jeopardy!*, a trivia game show. Watson's victory was due in part to its ability to synthesize evidence from multiple sources and over multiple strategies, its ability to estimate its confidence in its answers, and its general game-playing

strategy. One of the key resources available to the system was the Prismatic knowledge base, which was derived from natural language text (Fan et al., 2010; Fan et al., 2011). This suggests that *learning by reading* (LBR) as a source of knowledge holds a great deal of potential.

Some aspects of learning by reading have been extensively explored. Unfortunately, despite recent progress, it remains an open problem. *Natural language understanding* (NLU), the process of using text to produce precise representations usable for reasoning, requires parsing, word sense disambiguation, coreference resolution, and other difficult tasks to be performed at a high level of accuracy. *Knowledge integration*, the process of connecting new knowledge with existing knowledge in order to facilitate reasoning, brings challenges with it as well. Some systems – including Watson – make use of very large quantities of source text and simple representations to partially work around some of these issues. However, humans are able to learn from small numbers of textual examples, often just a single source. This suggests that it is possible to make do with far less if reading occurs at a deeper level and existing knowledge can be brought to bear. Learning Reader (Forbus, Lockwood, & Sharma, 2009) and MMKCap (Lockwood & Forbus, 2009) have demonstrated some of the promise of this approach.

Analogy is the process of comparing two structurally represented cases. It is fundamental to human cognition, playing a role in processes ranging from high-level vision to problem solving (Gentner, 1983; Gentner, 1989; Gentner, 2003). Overt analogies appear in instructional texts as learning aids as well. This suggests than analogy could be a powerful tool for cognitive agents. In my work, I have used analogy to assist in overcoming difficulties in learning by reading. This thesis describes three contributions that integrate language understanding and analogical reasoning.

1.2 Overview of Analogy and Reading

Building a useful body of stored knowledge from what a system reads is a process that stands to benefit from analogical processing in several ways. Dealing with texts that contain explicit analogies is perhaps the most obvious use for analogical processing, but analogical processing can play roles in word sense disambiguation and knowledge organization as well.

Analogical reasoning has been demonstrated to be useful in a wide variety of contexts, including moral reasoning (Blass & Forbus, 2015; Dehghani et al., 2008), inference learning (Forbus et al., 1997; Liang & Forbus, 2015), categorizing sketched concepts (Chang & Forbus, 2013; McLure, Friedman & Forbus., 2015), and learning spatial prepositions (Lockwood, Lovett, & Forbus, 2008). It has been used to learn from sketches (Forbus, Usher, & Tomai, 2005) and to solve physical reasoning problems (Klenk et al., 2005). In some work, analogical processing has been used to modeling human processing (Gentner & Forbus, 2011).

In this work, I apply analogy in novel ways to problems within learning by reading. Claim 1 deals with the application of analogy to word sense disambiguation, and Claim 2 deals with organizing knowledge in ways that are amenable to analogy. As humans benefit from analogies in instructional texts, we might expect the same to be true of cognitive systems, if they have the tools to identify and process analogies in text. Claim 3 deals with this task. The knowledge organization strategies developed as part of Claim 2 play a role as well.

1.3 Claim 1: Analogical Word Sense Disambiguation

My first contribution is to demonstrate that analogical retrieval of previously stored sentence choice cases can be used for word sense disambiguation.

Word sense disambiguation (2.1.2) is the task of selecting a lexical interpretation for a word or phrase in a source text. Many word sense disambiguation strategies work by looking in the training corpus to find similar antecedents for the current ambiguity. These antecedents are then used as evidence for disambiguation. The work presented here is part of that tradition. In our case, the method used to find the antecedents is analogical retrieval. The system generates cases that represent an ambiguity and the context in which it appears. It then uses MAC/FAC (2.4.3) and SAGE (2.4.4) to find the previously-generated cases with the greatest structural similarity. It is evaluated based on its ability to select senses that match a gold standard. This contribution is covered in Chapter 3.

1.4 Claim 2: Connection-Based Case Construction

My second contribution is to show that by exploiting the connectivity properties of the semantic interpretation of a text, a cognitive system can extract subsets of that interpretation that are useful for analogical reasoning tasks.

Large-scale reading tasks produce a very large number of statements. Working with the corpora used in one of the experiments, the reading system we used generated nearly as many statements as there are words in the source text. (Some words generate none, but some generate several). Even for a relatively short book chapter or web article, that represents hundreds of facts, some of which are more tightly related than others. The ability to build smaller cases from larger ones is useful for case-based reasoning tasks. For example, a chapter might discuss several types of heating system. If we want a system to perform a comparison between two of the types being discussed, it needs to be able to automatically create separate cases that each contain the information about one of the types.

By taking advantage of relationships between the facts in an interpretation and the properties of the sentences that generate them, we are able to build smaller (non-exclusive) cases that are consistent and

coherent. This is important, because extra information acts as a distractor in many applications. The evaluation of this claim focuses on a realistic application for the cases produced by the case construction process. This contribution is covered in Chapter 4.

1.5 Claim 3: Analogical Dialogue Acts

My third contribution is an ontology of a*nalogical dialogue acts* (ADA) that can be used to improve understanding of text passages that contain explicit analogies.

Dialogue Act Theory (Allen & Perrault, 1980; Traum, 2000) is concerned with the roles utterances play in discourse. An utterance identified as a *Requesting Information*, for example, might take the syntactic form of a question that makes the information requested explicit, such as "What time is it?," but does not necessarily do so. The surface manifestation could instead be a statement, or it could be an indirect question, such as "Do you have the time?" Analogical dialogue acts are an extension of this idea. Like other dialogue acts, they have criteria by which they can be recognized and a set of implied commitments and obligations for the dialogue participants.

Statements are classified in the ADA ontology based on the role they play in introducing or extending an analogy. For example, "The Earth's atmosphere is like the roof of a greenhouse" introduces an analogical comparison between those two things. "The roof admits sunlight" extends the base of the analogy – one of the two cases being compared. Identifying the types of ADA being employed in different statements allows the system to construct a representation of the base and the target of each analogy. This in turn allows the system to actually align the parts of the analogy as a human would, which allows it to better understand what it has read, as measured by performance on a question-answering task. As part of evaluation, a corpus of textual analogies was created by excerpting from a variety of educational materials. This contribution is covered in Chapter 5.

1.6 Organization of Thesis

Chapter 2 covers background relevant to the later sections, including the learning by reading system used in the work, an overview of the model of analogy used, and the question answering strategies employed for evaluation. Chapters 3-5 cover the individual contributions, as described above. Chapter 6 is related work and conclusions. Chapter 7 describes future work. This is followed by appendices for references, charts, tables, and selections from the source corpora.

2 Background

2.1 Learning by Reading

Learning by reading is the process of taking in natural-language text and producing forms useful for reasoning. This involves *natural language understanding*, the process of building formally-represented knowledge from text. It also involves some method for organizing this information for later use and integrating it with existing knowledge (Allen, 1994).

2.1.1 Natural Language Understanding

Natural language understanding can be decomposed into several smaller tasks. These include parsing, semantic disambiguation, and coreference, although not all systems that extract information from text attempt all of these. For example, some systems use the literal words from the text as symbols in the knowledge (Fan et al, 2010). These systems can ignore word sense disambiguation. While this may be acceptable for certain applications, it is important for systems that are designed to do intensive reasoning on what they read to be precise and to avoid introducing error by conflating things with the same name. Extra error potentially interferes with reasoning tasks. It also makes it more difficult to diagnose less avoidable errors. For example, by representing that there are multiple senses of the word

"bank," the system can make distinctions that stem from the fact that a banking institution can intentionally take actions, while a river bank cannot. When the system is reasoning about the sort of things that a river bank might be capable of, knowing that it is the sort of bank that cannot take intentional actions eliminates many possibilities.

One of the aims of learning by reading in a cognitive agent is to accrue knowledge useful for reasoning without having to have a human expert hand-author that knowledge. Because of this, it is important to avoid introducing error wherever possible. While having a human identify errors as they crop up may be cheaper than having that human author the knowledge in formal representations from scratch (Clark et al, 2005), it still severely affects scalability. Some error is all but inevitable - even humans make mistakes while reading and while integrating what they read into their existing knowledge, and the state of the art in natural language systems lags behind humans in many important areas (Navigli, 2009). While it is not possible to catch every error, however, minimization and correction of error are still important for learning by reading.

Our system builds well-connected knowledge to support reasoning. Because of this, it is useful to be able to achieve a high degree of recall on individual source texts. Many information extraction systems produce a large number of factoids by scanning over a large number of sources (e.g., Banko et al, 2007). This is effective for information that appears many times over many sources. Across the entire internet, for example, information about national capitals appears many times. In some cases, these facts are connected to other facts, so some systems of facts can be constructed in that way. In other cases, however, information may only be present in a very small number of distinct documents. A specific type of heating system might only be described once or twice. A novel system that somebody is still designing is almost certainly only described in one place. In cases like these, a high degree of recall is critical in

producing a representation for reasoning. The system cannot rely on there being many phrasings of the same information if they do not exist. Even where a multiple sources are available, generalization and integration of the representations derived from those sources should work better if the representations are more complete and more accurate to begin with. For this reason, the reading strategies we use focus on building as complete a representation as possible from source texts, even where there are additional computational costs.

2.1.2 Knowledge Organization and Integration

Organizing knowledge that is derived from natural language texts is a task that has not been as thoroughly explored than deriving that knowledge to begin with, but is still very important for reasoning. Even for simple lookup tasks, knowing when and under what circumstances facts hold is important. Any task that involves using larger amounts of information at once, as a case, requires additional organization. For example, if the system is to compare one mechanical system to another, it needs representations of those systems. Automatically constructing such representations is a challenge addressed in Chapter 4.

Knowledge integration is the task of connecting new knowledge to knowledge that the system already has. This is important if the system is to be able to effectively use knowledge from multiple sources. While knowledge integration is not a task directly tested by any of the evaluation strategies in the work in this thesis, it is a future goal for my work in case construction, and informs the representation decisions.

2.2 EA NLU

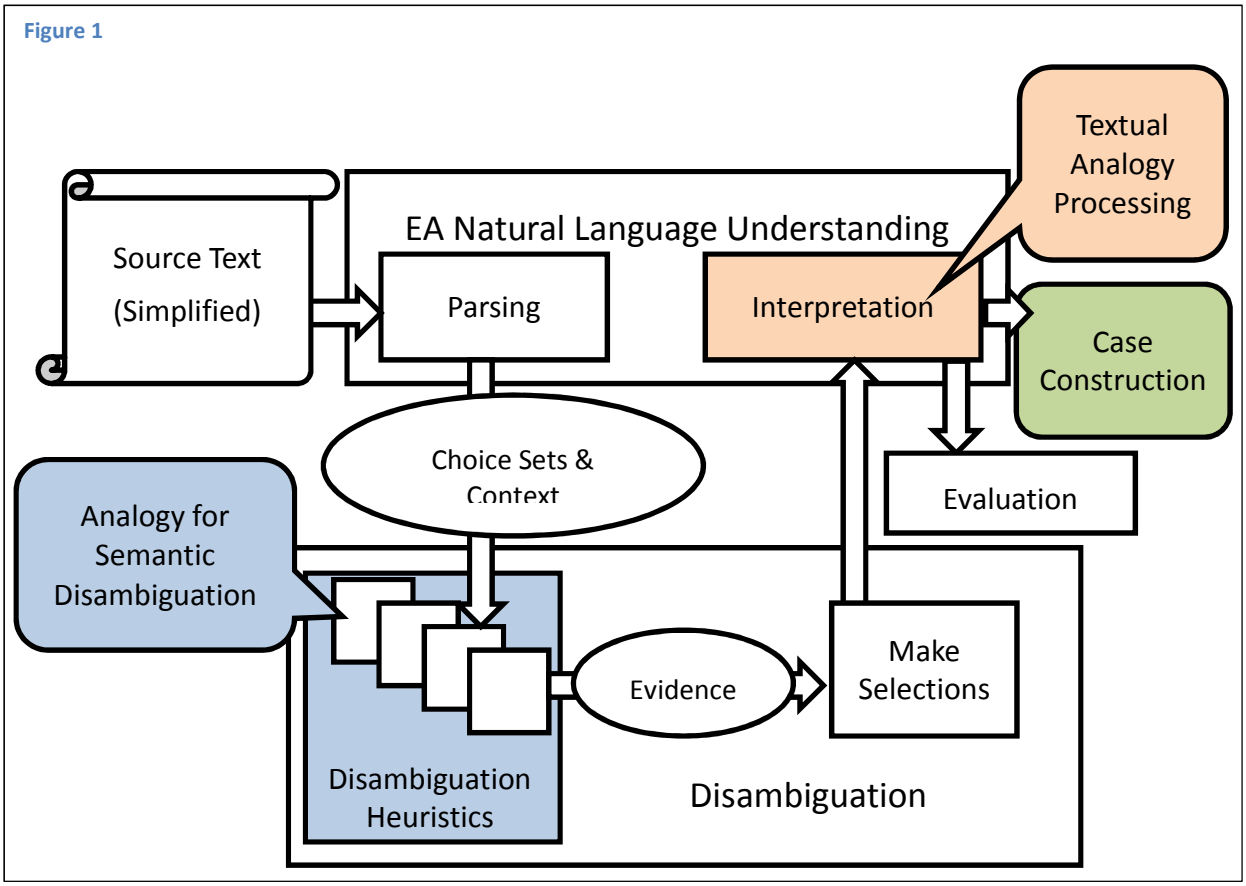
We use the Explanation Agent Natural Language Understanding system (EA NLU; Tomai & Forbus, 2009) as our natural language understanding system. Syntax is handled using Allen's TRAINS (1994) parser, a

bottom-up chart parser. The syntactic analysis produced by the parser is stored declaratively by the reasoning system using predicates that extend ResearchCyc. The relevant concepts of the Cyc ontology are described in Section 2.2.1.

EA NLU represents parse, word sense, and coreference ambiguities as choice sets (Section 2.2.4). Once choices in these choice sets have been selected, they are used to build a *sentence interpretation* for the sentence. The interpretation process uses Discourse Representation Theory (Kamp & Reyle, 1993), as described in Section 2.2.2.

Figure 1 illustrates the reading pipeline used by the system, and where my major contributions occur within it. The colored areas represent

Figure 1



2.2.1 Cyc and EA NLU Representations

EA NLU generates semantic interpretations as structured representations using Cyc collections and predicates. For example, from the sentence "A dolphin eats a fish," the system can generate the fact `(isa dolphin5332 Dolphin)`, among others. In that fact, `dolphin5332` is a *discourse variable* generated by the system to represent the dolphin. `isa` is a *predicate*. Predicates take one or more arguments and relate them. `isa` indicates that the entity in the first argument position - `dolphin5332` - is a member of the *collection* `Dolphin`. A collection is a group of things. All members of the collection `Dolphin` are dolphins, and all dolphins are members of that collection. Collections are arranged in a hierarchy, connected by `genls` statements. For example, `(genls Dolphin SeaMammal)` means that `Dolphin` is a subcollection of `SeaMammal`. Any entity that is a member of the collection `Dolphin` is a member of the collection `SeaMammal`. Colloquially, we say that `Dolphin` genls to `SeaMammal`. The opposite of genls is spec; `Dolphin` is a spec of `SeaMammal`.

The system extends the representations provided by Cyc slightly to represent things like parse features, which Cyc does not have built-in representations for. These predicates are used by the system to reason about things like whether a semantic choice is compatible with a particular parse.

Discourse variables are generated for each of the entities introduced in the text the system reads. This includes events, which are reified in the interpretation. For example, `(isa eat5642 EatingEvent)` appears in the interpretation of the sentence, with `eat5642` representing the eating event. An exception to this occurs when the system has lexical information connecting the word or phrase to a unique entity that it already knows about. For example, for "Plato eats a fish," the system

will not generate a new discourse variable for “Plato.” Rather, it will use `Plato`, as that symbol already exists in the knowledge base and there is lexical information connecting it to the word “Plato.”

Most of the lexical rules for verbs in Cyc use Davidsonian representations. Davidsonian representations reify the event, and then 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 representation, `eat5642` is the eating event. The participants are `dolphin5637` and `fish5672`. The roles they play in the event are described by their role relations. `performedBy` indicates that the actor intentionally performed the action, and `consumedObject` indicates that the object was affected and destroyed. Cyc has an extensive hierarchy of role relations, but about ten role relations account for a heavy majority of the facts produced by the language system.

EA NLU produces interpretations first on the sentence level. For each sentence in a text, the interpretations of each of the words are combined into a *sentence interpretation case*. In many cases, the parse contributes additional structure, often in the form of organizing the word-level lexical interpretations into discourse representation structures (DRS; see 2.2.2). From there, *discourse-level interpretation* processes combine the sentence level interpretations into one large interpretation, called a *discourse interpretation*. At this point, the system also performs coreference resolution (see 2.2.3). Discourse-level interpretation processes can also be used to do operations such as identifying qualitative relationships between quantities.

The names generated by the system for discourse variables, for reading trials, and for sentence, discourse, and DRS case names are slightly complex. The system needs to ensure that the names of these objects are unique, while still providing enough English in the name that the nature of the symbol can be determined by looking at it, for debugging purposes. This results in somewhat lengthy names in some cases, such as `Discourse-3623363883-2172`. Throughout this document, these names have been simplified slightly for readability.

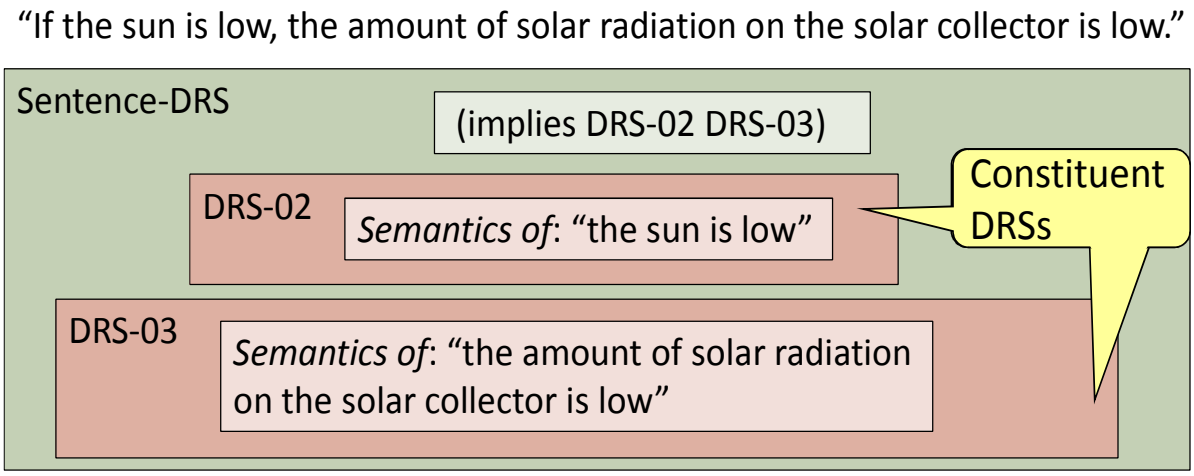
One of the goals of EA NLU is that it remain reasonably fast and general-purpose. For this reason, it does not bring complex reasoning to bear on the parsing process. It produces parses and semantic choice sets that can be ruled out by looking for type contradictions or with more sophisticated reasoning, but does not do that automatically while parsing or building choice sets. Disambiguation heuristics (Section 2.2.6) allow the system to disprefer choices that result in sematic contradictions.

2.2.2 Discourse Representation Theory

Discourse representation theory (DRT; Kamp & Reyle, 1993) forms the basis for EA NLU's semantic interpretation process. The system uses the parse and semantic information to build up *discourse representation structures* (DRS). Each DRS is a case containing one or more facts. These facts describe relationships between entities, collections that those entities belong to, and other DRSs. Each sentence has a *sentence DRS* that contains facts that represent the semantics of that sentence. Sentence DRSs can also have *constituent DRSs*. These handle scoping, including conditionals and counterfactuals. For example, the sentence "If the sun is low, the amount of solar radiation on the solar collector is low" mentions a height for the Sun and an amount of solar radiation. It does not say that the Sun actually is low or that the amount of solar radiation is low, however. If the system produced facts that said that Sun actually is low, it would reach incorrect conclusions. For that reason, 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 contains a fact of the form `(implies DRS-02 DRS-03)`. This is illustrated in Figure 2. Constituent DRSs are also used to handle negation and hypotheticals.

Figure 2. Discourse Representation Theory



While DRT is a core part of how the language system operates, and thus plays a role in all of the work described in this thesis, it is particularly relevant for case construction, as described in Chapter 4. As we discuss in Section 4.3.2, the organization of facts into DRSs, if handled properly, allows the system to avoid introducing certain types of error during case construction.

2.2.3 Coreference

Determining which entities in a text refer to the same thing is a difficult challenge, but one that is important to address. Without the ability to identify something across multiple sentences, it is difficult to build up structures of well-connected knowledge. This is important for connection-based case construction (see Chapter 4), as the construction strategies rely on connections between facts. One such connection is two facts mentioning the same entity. It is also important for analogical dialogue act

processing (see Chapter 5), as being able to keep the entities straight is important for building a good mapping between the cases. In a greenhouse effect analogy, the system would get lost if it believed that several different greenhouses were being discussed because the word “greenhouse” was mentioned several times in the text. It would similarly get lost if it believed that all instances of light being absorbed were the same event. There are three types of coreference addressed by EA NLU.

The first is *direct reference*. This is when the same object is overtly mentioned multiple times. Each mention of something in a source text triggers the system to generate a new discourse variable for that thing. For example, in the initial reading of “A penguin catches a fish. The penguin shares the fish with its young,” two separate discourse variables would be generated for the penguin, one for the first mention and one for the second. It falls to coreference resolution to determine that the two penguins mentioned are the same penguin. EA NLU looks for potential references for any common noun introduced with “the,” like “the penguin.” Its strategy for direct reference is to select the most recent valid referent, going back up to five sentences. If two possible referents appear in the same sentences, the subject is preferred over other possibilities. A small number of additional constraints that handle containment and possession also guide direct reference coreference.

The second type of reference is *pronominal reference*. In most instances where a pronoun is used, it refers to another entity or case also present in the discourse. For example, in “A dolphin gives birth to live young. It nurses the young dolphins,” the first “it” refers to the same thing as “a dolphin.” The system’s strategy for pronominal reference is to pick the first valid referent in the previous sentence, and only in the previous sentence. Gendered pronouns – “he” and “she” – will not refer to things that are known to be the other gender. Pronouns prefer to match subjects where possible. Pronouns also need to match their antecedents in number. “It” will not refer to a plural referent, like “dolphins,” and

"they" will not refer to a singular referent. (The system does not currently recognize the singular "they.") "This" and "that," when used as a noun (as in "This causes the temperature to fall," but not as in "This act impressed this judges," where "this" is a determiner) are handled specially. In these cases, "this" and "that" refer to the case for the entire previous sentence.

The final type of reference handled by the system is verb reference. Verbs corefer to verbs in the previous sentence if they are the same type of event and there are no role relation conflicts. For example, in "Water flows from the bucket. The water flows to the ground," the second instance of "flows" refers to the first. The object filling the `objectMoving` slot in both cases is the same, because the second "water" refers to the first. The first sentence puts the bucket in the `fromLocation-Underspecified` role, and the second puts the ground in the `toLocation-Underspecified role`. If the second sentence was "The water flows from the faucet," the two instances of "flows" would not corefer, as there are two different things in the `fromLocation-Underspecified` slot.

Reference is transitive. If any sort of coreference would cause an entity to refer to second entity that already refers to a third, the first entity instead refers to the third.

It is easy to think of examples where the strategies outlined above will fail. In addition to linguistic oddities like the dummy pronoun in constructions like "It is raining," there are plenty of standard cases where they do not work. Sometimes pronouns refer more than one sentence back, or refer to something other than the subject, even when the subject matches in number and gender. Full coreference is a difficult problem, and one that humans do imperfectly. The goal with our coreference system, as with all of our version of simplified English, QRG-CE (see Section 2.2.4), is to ensure that a variety of common cases are handled, that there are reasonable ways to say most things, and that the system is simple enough that the output is not astonishing for someone familiar with the system. The

coreference system as described here is not intended to be complete or foolproof, nor is it what is under primary evaluation in any of the contributions. Applying knowledge or more sophisticated reasoning to the coreference task might allow for more accurate coreference, but that is outside the scope of this thesis. The coreference mechanism used by the system was extended for the purposes of this work, but only within the general setup of the existing system. It is not tailored to be specifically suitable for the tasks described here. A more complete specification of how the coreference system works is found in Section A of the appendix.

Anaphora resolution in general is a problem with many facets (Elango 2005). Full language understanding would require the ability to handle reference to things that are implied but not explicitly mentioned, phrases whose referents change with time ("the president"), and so on. The coreference system used in the work described here is sufficient for experimental purposes and is relatively simple, covering several very common and important types of reference. A coreference subsystem that takes greater advantage of the reasoning capabilities of the system would be a valuable addition to the system.

2.2.4 QRG-CE and Simplification

For our work we have chosen to use texts written in simplified English. The English we use is called QRG-CE. Our lexicon is not limited, but our grammar covers only certain constructions. This requires simplifying many sentences. Sentences of greater length or those that employ less common constructions are more likely to require simplification. The simplified English texts are still fully readable English; the simplification process simply reconfigures and splits sentences into ones which use supported grammatical structures. 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 16 in

the appendix contains a paragraph from the simplified corpus. Importantly, simplification does *not* eliminate ambiguities. Certain heavily controlled languages, such as ACE (Fuchs, Kaljurand, & Kuhn , 2008) and PENG (Schwitter, 2002) are designed to eliminate ambiguity, but QRG-CE is not of that type.

For practical purposes, simplification has several benefits. It allows us to explore deep understanding without having to tackle the other open-ended NLP problems that usually come along with that. It also allows for end-to-end learning by reading. Simplification of texts prior to processing is not uncommon in the field. Many practical applications make use of controlled language; commercial machine translation systems for equipment manuals are one example. Simplified text can allow for authoring of knowledge in a form that is much simpler and more robust than writing propositions by hand. For example, Clark et al. (2005) describes Computer Processable Language (CPL), a set of restrictions on sentence construction that simplify machine reading while still allowing relatively natural input. Our language system is considerably more flexible than CPL. It can process a greater variety of sentence structures, allows pronouns, and does not make use of specific keywords within sentences. QRG-CE, like CPL, takes a *naturalist* approach (Clark et al., 2010). That is, it is intended to be a simplified version of uncontrolled natural language processing, rather than a formal language that maps unambiguously onto propositional logic. Both word sense and parse ambiguities are present in our simplified texts. In fact, the simplification process does not directly seek to eliminate them at all – it simply converts from unsupported grammatical structures to supported ones.

One goal of system development has been to reduce to degree to which simplification is required and to reduce the specificity of what simplification is required. The better the system can do with less simplification, the less effort and expertise is required to simplify. It also allows for better performance

on texts which have not been simplified. Additionally, the simplification process introduces some degree of doubt into the results due to the hand-processing that must be done by the experimenter.

My early work with analogical dialogue acts (Chapter 5) was particularly demanding of the texts it works over. It required fairly complete and fairly specific constructions for detection. As our attempts to make these steps more robust proceed, the need for specificity in simplification is reduced.

2.2.5 Representing Ambiguity

The system generates *choice sets* to represent ambiguities. EA NLU creates choice sets for three types of ambiguity. The first type is the *semantic choice set*. These are ambiguities about which semantics should go in the sentence interpretation. Word sense ambiguities are one source of semantic choice sets. For example, the word "star," when used as a noun, generates a choice set with four choices, each of which is a fact. One choice is the fact `(isa star5323 FamousHuman)`, where `star5323` is a discourse variable generated to represent that entity. This indicates that there is some famous human in the interpretation of the sentence. Other choices in the set are similar, but for other senses of the word "star." Semantic choice sets might have multiple choices in them even when there is no word sense ambiguity, if there is ambiguity stemming from which combinations of role relations should be included. This is most common with verbs. A semantic choice set normally covers a span of a single word, but they can also cover recognized multi-word strings, such as "Bill Gates" or "solar energy."

The second type of choice set generated by the system is the *parse tree choice set*. Each sentence with a complete parse has one such choice set. It contains all of the parses that were generated for the sentence. Parse ambiguities include prepositional phrase attachment and other scoping issues, along with structural ambiguity. ("Heat flows", for example, could be a statement indicating that heat streams from one place to another, or a command to warm up a group of instances of something moving from

one place to another. The latter is semantically improbable, but the parser does not know that.) Multiple parse ambiguities are often present in a single sentence. When this occurs, the number of possible combinations causes the number of choices in a parse tree choice set to grow very large. Parse choices and semantic choices are interdependent. Selecting a parse can disable certain semantic choices and vice versa.

It is important to note that while the primary motivating factor for the choice set-based architecture is representing ambiguities, the system does represent unambiguous things, including monosemous words (such as "porpoise") and the parses of sentences with only one valid parse (such as "The porpoise sleeps.") as choice sets. We refer to choice sets with only one choice as *singleton choice sets*. Singleton choice sets are generated for two reasons. First, it allows for a consistent architecture across both ambiguous and unambiguous words and parses. Second, a singleton choice set does not necessarily mean that that choice is correct. For example, in the sentence "A child plays with a beach ball," there will be a choice set generated that includes only `(isa beach2856 Beach)`. It would not be correct to select that choice; there is no beach under discussion. The correct interpretation involves selecting `(isa beach-ball3217 Ball)` for the two-word span "beach ball," which has its own choice set.

The system also represents coreference choices (Section 2.2.3) as choice sets, where the potentially valid referents form the choices. However, because the strategies used to generate and select from these choice sets are so different than the strategies used to generate and select from parse tree and semantic choice sets, generally "choice set" will be used in this thesis to refer only to parse and semantic choice sets.

2.2.6 Resolving Ambiguity

Resolving semantic ambiguity is done primarily through a system built into the Companion (Section 2.3) called the *disambiguation harness*. The disambiguation harness was created to support the work presented in this thesis. Briefly, the disambiguation harness works by applying a set of *disambiguation heuristics* to parse or semantic choice sets. Each heuristic assigns a score to each choice in each choice set. The scores from each heuristic are then summed. The system then creates a list of all of the choices in the sentence from all of the parse and semantic choice sets. It moves through each choice in order from highest score to lowest, and makes that choice if that choice has not yet been ruled out. A choice might be ruled out because another choice in its choice set is already selected. It might also be ruled out if an incompatible parse tree is already selected, or if all compatible parse trees have been ruled out. The disambiguation harness is designed for easy extensibility, and supports both general-purpose and targeted heuristics. All of the heuristics used in this thesis are designed to be general-purpose.

Heuristics vary in complexity. An example of a very simple heuristic is `preferPossesion`. This heuristic votes weakly in favor of semantic choices that contain expressions that denote possession, such as `(possessiveRelation dolphin3124 head3342)`. This is not a particularly reliable heuristic, which is why it gets only a weak vote by default. For semantic choices that do not contain possession expressions, it simply does not vote. The result is that possession expressions are more likely to be chosen. Heuristics can be much more complex, however. The analogical word sense disambiguation scheme described in Chapter 3, for example, is implemented as a disambiguation heuristic.

The disambiguation harness not only allows the system to define which heuristics will be applied to a disambiguation task, it allows for individual weighting of heuristics. It also stores the votes each heuristic

made on each choice. This allows heuristic weighting to potentially be learnable. This is outside the scope of this thesis – in all experiments described here, the disambiguation harness was run with the default weights. The system also allows choices to be manually selected by the user. This is used to generate the gold standard interpretations for some of the evaluation metrics used here.

2.2.7 EA TMS

In order to support the work presented in this thesis, I reconfigured EA to run on a truth maintenance system (TMS; Forbus & de Kleer, 1993). This makes it such that the system automatically detects incompatibilities between semantic choices and other semantic choices, and between semantic choices and parse tree choices. The incompatibilities it detects are "hard" incompatibilities. For example, in the sentence "The man saw a bird with a telescope," the TMS prohibits the semantic choice that the bird has the telescope if there is a parse tree selected that attaches the prepositional phrase "with a telescope" to "saw." In instances where every valid option but one has been ruled out, it infers that the last one must be true.

It is important to note that the EA TMS only prevents the system from making *syntactically* incompatible choices. It does not rule out "soft" incompatibilities that come from the semantics themselves. For example, it permits an interpretation of "Bats eat insects" that has the "baseball bat" sense of the word "bats." This is despite the fact that with further reasoning it might be able to determine that, in most models of reality, baseball bats do not literally eat anything. This sort of more extensive reasoning would be useful for detecting errors, but it is more expensive, and we want things that operate during the parsing process to be fast.

2.2.8 Readings and Source Texts

As part of the development of the techniques described in this thesis, we extended the reading plans used by Companions (Section 2.3) to make use of an organized system for tracking what the system reads. Formally, a *source text* is a particular section of text. A web article or a chapter of a physics textbook might be a source text. Two different sections of text are considered different source texts, even if they overlap. For example, a textual analogy excerpted from the physics textbook chapter is a separate source text from the chapter taken as a whole.

A *reading* is a single attempt to process and interpret a source text. Multiple readings of a source text may produce different interpretations if different parameters are applied to the reading. For example, if a different set of disambiguation heuristics are used or different discourse-level interpretation processes are run, the results will be different even if the source text remains the same. Changes to the grammar, lexicon, or coreference rules can also affect the final interpretation.

When the system stores the information it learns by reading a source text, that information is not simply placed into the knowledge base to be retrieved by anything that might be searching the KB. It is tied to the reading that generated it. This is important, as information learned by reading sometimes contains errors, most often as a result of parsing or word sense disambiguation failures. Storing information this way also allows the system or a human working with it to easily compare what is learned during different readings.

In addition to storing the interpretation itself, the system also stores information about the parameters under which the reading was run. While it is impractical to preserve the entire state of the reading system, the system stores many of the more flexible parameters along with timing data and accuracy

information (if a gold standard is available). This allows for analysis of the effects of different parameters.

2.3 Companion Cognitive Architecture

A *cognitive architecture* is a system designed to integrate multiple cognitive processes. The end goal is to build (and sometimes to model, depending on the architecture) a mind. A wide variety of cognitive architectures exist, employing a range of strategies and operating at a range of levels, from very low-level reflexive response tasks through sophisticated reasoning and planning, all the way to social reasoning. A key part of what defines cognitive architectures is that they are single systems designed to handle many tasks. This distinguishes them from many machine learning systems that focus on a single task or a very narrow set of tasks.

Our goal with the work described in this thesis is to enable and bootstrap a software social organism. With that end in mind, all of the mechanisms have been built into the Companion cognitive architecture (Forbus, Klenk, & Hinrichs, 2009). The Companion architecture is an agent-based system. There are several advantages to integrating the methods we describe into a system that incorporates multiple reasoning processes, and in particular into a Companion.

First, doing so gives the methods access to powerful reasoning tools, including the FIRE reasoning engine (Forbus et al., 2010) and an HTN planning system. EA NLU is integrated into Companions, and the architecture includes plans that coordinate reading, disambiguation, and storing away knowledge that is learned. Over the course of designing the methods and running the experiments described here, I extended many of these plans and procedures to add additional support. This included the creation of features like the disambiguation harness and EA TMS. This is a second advantage of working in an integrated system used in multiple lines of research – the systems that are designed to support the work

can be used in those other projects. Multiple lines of inquiry making use of component systems encourages general-purpose design over specialized single-purpose units.

The Companion architecture offers several specific advantages that make it a good fit for this research. First, it provides support for analogy. It contains plans that support both basic analogical mapping through SME (Section 2.4.2) and more sophisticated analogical reasoning processes such as generalization (through SAGE; Section 2.4.4) and retrieval (through MAC/FAC; Section 2.4.3). It is structured around the idea that analogy is a very important part of cognition.

One goal of the Companion system is *end-to-end reading* – going from a source text to an interpretation that can be integrated with existing knowledge and evaluated or used for practical tasks. This provides a motivation for learning by reading techniques. Rather than being designed in a vacuum only to maximize evaluation metrics in isolation, methods that are built into full reading systems can be applied to realistic, challenging tasks. This can supplement traditional evaluation metrics or provide means of evaluating efficacy on problems that cannot be measured directly.

All of the methods described in this thesis are modular. None specifically require a Companion, and could be ported to other architectures. They are contingent on having a parser that produces structured representations available, and both AWSD and ADA make use of SME. However, those things could be (and in some cases already are) included in other cognitive systems, so the methods described here are not strictly tied to Companions.

2.3.1 Architecture Structure

The Companion architecture supports a wide variety of cognition tasks, and detailing its structure and capacities in full is beyond the scope of this thesis.

A Companion uses a distributed agent architecture. The agents communicate with each other using KQML messages. Each has access to its own *working memory*, which holds the results of short-term processing and to a shared *knowledge base*, the core of which (for the work presented here) is the ResearchCyc KB. A relatively small number of agents are used in most Companion configurations. The agent types that play a role in reading are the following:

The *Session Manager* provides a user interface to the other agents and tracks their health. While it does not do any of the heavy lifting involved in reading text or evaluating the system's understanding, it receives the user commands to do those things.

The *Executive* coordinates the other agents. It takes the commands from the session manager and then dispatches other agents to do them.

The *Interaction Manager* is responsible for the bulk of the reading process. It has access to EA NLU, so all of the parsing, choice set construction, and interpretation creation is its responsibility. Its working memory is where the parse, choice set, and interpretation information is stored, so if any of those things are needed by other agents, they must be communicated to those agents or stored in a shared knowledge base. The interaction manager also runs the disambiguation harness (Section 2.2.6), making it responsible for word sense disambiguation as well.

The *Session Reasoner* is used for domain reasoning. Most relevantly for the work described in this thesis, it is responsible for the question answering system used for evaluation, including parameterized questions (Section 2.5.1).

2.4 Analogy

All of the methods described in this thesis make use of analogy. This section introduces the model of analogy used in this work as well as computational implementations of analogical mapping (SME), generalization (SAGE), and retrieval (MAC/FAC).

2.4.1 Structure mapping theory

The model of analogy we use is *structure mapping theory* (Gentner, 1983). Structure mapping builds correspondences between two structure cases. It aligns the entities in the base with entities in the target based on the similarity of the higher-order structure surrounding them, as well as their simple features. Mappings are made according to three constraints:

The *tiered identicality* constraint ensures that two statements can match only if their predicates are identical. For example, `(largerThan Nucleus Electron)` can map to `(largerThan Sun Earth)`, but not `(brighterThan Sun Earth)`, as `largerThan` and `brighterThan` are different predicates.

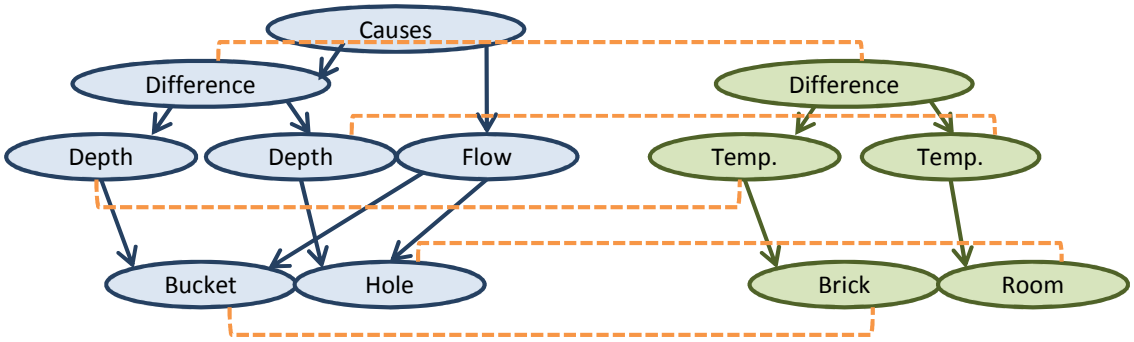
The *parallel connectivity* constraint ensures that if two statements match, then their arguments match. For example, `(largerThan Nucleus Electron)` can match `(largerThan Sun Earth)` if and only if `Nucleus` maps to `Sun` and `Electron` maps to `Earth`.

The *one-to-one mapping* constraint ensures that entities can only match one other entity, so if `Electron` maps to `Earth`, it cannot also map to `Moon`.

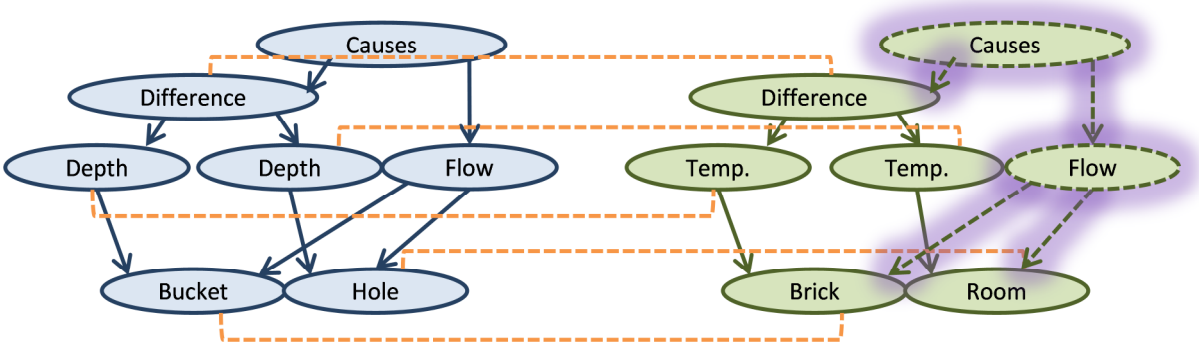
Within these constraints, mappings are given a score based on their systematicity, which is the extent to which higher-order structures in the description can be mapped to each other and their structural

similarity.

Figure 3. A mapping between "The difference between the depth of the bucket and the depth of the hole causes a flow from the bucket to the hole," and "There is a difference between the temperature of the brick and the temperature of the room."



The system draws candidate inferences about the target based on the structure of the base.

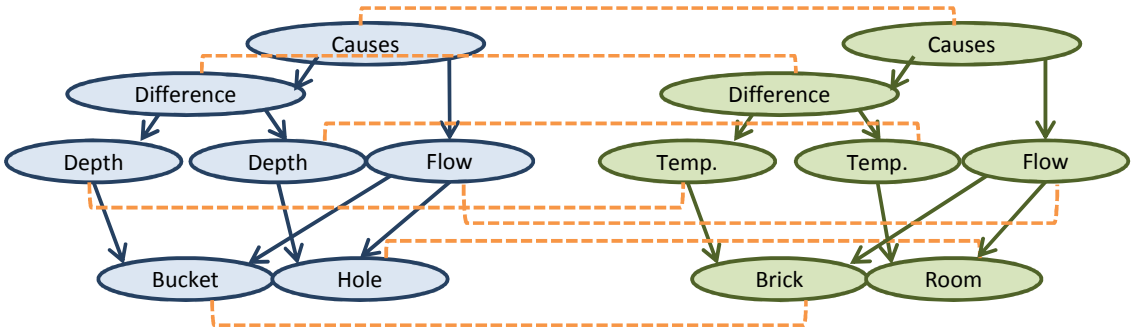


Candidate inferences about the target are then drawn to fill in structure that is present in the base but not the target. A candidate inference is estimated to be more likely if there is greater structural support for it. Candidate inferences are allowed to hypothesize the existence of new entities, if necessary. In the example in Figure 3, structure mapping allows for the hypothesis that the difference between the temperatures of the two objects in the target causes a flow between them. There is a large body of psychological evidence for structure mapping. It is a cognitively grounded theory. However, the work described in this thesis does not attempt to model human cognition; it only uses structure mapping as a

tool. Nevertheless, the fact that structure mapping appears to play a role in human reasoning suggests that it may be a good fit for AI tasks.

Structure mapping prefers structural matches – matches where higher-level relationships match to each other. An example of this is shown in Figure 4. The *Causes* relationship is a higher-level relationship, as it coordinates multiple other relationships.

Figure 4. A mapping between "The difference between the depth of the bucket and the depth of the hole causes a flow from the bucket to the hole," and "The difference between the temperature of the brick and the temperature of the room causes a flow from the brick to the room."



The specific choice of structure mapping as a model of analogy does affect the design of all of the methods described in this thesis, especially Analogical Dialogue Acts, and underpins SME, MAC/FAC and SAGE. If a different model of analogy were to be used, the methods might look very different.

The *analogy ontology* (Forbus, Mostek, & Ferguson, 2002) is a formal ontology of analogy based on Structure Mapping Theory. It extends ResearchCyc, and provides a vocabulary for describing and reasoning about analogy. It is used in my Analogical Dialogue Act theory, as described in Chapter 5.

2.4.2 SME

The Structure Mapping Engine (Falkenhainer, Forbus, & Gentner, 1989; Forbus, Ferguson, & Gentner, 1994) is an implementation of structure mapping theory. It takes as input a *base case* and a *target case*,

each expressed declaratively. It produces one to three *mappings*. A mapping consists of three parts. The first is a set of *correspondences* between elements of the base and elements of the target. The second is a *similarity score* based on systematicity. The third is a set of candidate inferences. SME can draw candidate inferences bidirectionally – both from the base to the target and from the target to the base, and does so in all of the work described in this thesis. This means that even if a reading system using SME accidentally misidentifies which case is the base and which is the target, it can still produce candidate inferences about the domain the system is learning about. SME uses a greedy merge algorithm to produce the mappings in polynomial time ($O(n^2(\log n))$).

Candidate inferences can represent information the system learns by having worked through the analogy. In the example in Figure 3, the system learns that the temperature difference causes a flow between the two objects. Candidate inference can also be thought of as salient differences between two cases. They are things that are true in the base that may or may not be true in the target. We use candidate inferences in this latter way to identify salient differences between cases in the evaluation of the case construction algorithms in Chapter 4. Candidate inferences are allowed to hypothesize the existence of new entities that are parallel to ones in the other case. These hypothesized entities are called *skolems*. For example, in the solar system/atom analogy, the system might hypothesize the existence of a force in the atom that corresponds to gravity, and that has the same relationships with the nucleus and electrons that gravity does with the sun and the planets. It would be correct to hypothesize the existence of that force, as it (electromagnetic force) actually exists. It would be represented as `<skolem gravity524>` in the match.

In addition to the base and target, SME can optionally accept a set of *match constraints*. The work described in this thesis makes use of two such constraints.

- *Required correspondence* constraints are expressed as `(requiredCorrespondence ?base-element ?target-element)`. These constrain the match by only permitting mappings in which those two elements are in correspondence. For example, `(requiredCorrespondence Nucleus Sun)` will only permit mappings where `Nucleus` and `Sun` correspond.
- *Excluded correspondence* constraints are expressed as `(excludedCorrespondence ?base-element ?target-element)`. These constrain the match by only permitting mappings in which those two elements are *not* in correspondence. For example, `(ExcludedCorrespondence Electron Moon)` will only permit mappings where `Nucleus` and `Moon` do *not* correspond.

Match constraints are important for analogical dialogue act processing, as one of the benefits of identifying ADAs is the generation of these constraints. They are also used in the evaluation of connection-based case construction.

SME is a component in the models of analogical retrieval and generalization, described below, which are used in our methods. These models have all been used to simulate a variety of psychological phenomena (e.g., Dehghani et al., 2008, Friedman & Forbus, 2008), and they have also been used in performance systems (e.g., Hinrichs & Forbus, 2012; Lockwood & Forbus, 2009). This makes them reasonable starting points for the work we describe here.

2.4.3 MAC/FAC

Many are Called/Few are Chosen (MAC/FAC; Forbus, Gentner, & Law, 1995) is a model of similarity-based retrieval. It takes a *probe case* and a *case library* and finds the cases in the case library that are most similar to the probe. A case library is simply a group of cases; every SAGE generalization context is

a case library. MAC/FAC operates in two stages, as its name suggests. The first stage, MAC, is a coarse but fast method. Content vectors are automatically computed for each case, consisting of frequency information about the predicates in the case. MAC finds the cases from the case library whose content vectors are most similar to the probe case's content vector. It returns the three best cases. FAC, the second stage, compares each of those three cases to the probe case using SME. It returns the one with the highest similarity score, although it can return multiple cases if their similarity scores are very close. In the work described in this thesis, the methods only ever use the top result returned by MAC/FAC.

MAC/FAC is used directly in analogical word sense disambiguation (Section 3), where it provides the primary mechanism by which the method selects which prior case to use as a precedent. While MAC/FAC is not used directly in the case construction strategies described in Chapter 4, the utility of analogical retrieval for reasoning motivates the task.

2.4.4 SAGE

The Sequential Analogical Generalization of Examples (SAGE) model is a generalization system. It is an extension of SEQL (Kuehne et al., 2000). SAGE takes a stream of cases as input and organizes them within a *generalization context*. A generalization context contains *generalizations* of similar cases. When cases are combined into a generalization, SAGE replaces non-identical entities with symbols that denote arbitrary individuals and starts to accumulate frequency information for each matching statement. For example, if a case that contains the statements

```
(performedBy eat15 dolphin12)
(isa eat15 EatingEvent)
(isa dolphin12 Dolphin)
```

indicating that a dolphin ate something, is combined with a case that contains the statements

```
(performedBy eat19 porpoise16)
(isa eat19 EatingEvent)
(isa porpoise16 Porpoise)
```

the resulting generalization will contain the statements

```
(performedBy <GenEnt 1>  <GenEnt 2>)     Probability = 1.0
(isa <GenEnt 1> EatingEvent)             Probability = 1.0
(isa <GenEnt 2> Porpoise)                Probability = 0.5
(isa <GenEnt 2> Dolphin)                 Probability = 0.5
```

A `GenEnt` is a *generalized entity*. These are entities that differed between different parallel statements in the generalization. Statements not present in every case in the generalization have probabilities less than 1. When SAGE adds additional to a generalization, it updates the frequency information for statements in that generalization (Halstead & Forbus, 2005). Statements with probabilities below a set filtering threshold (unrelated to the generalization threshold) are filtered out of the generalization.

The generalization context also contains *ungeneralized examples*, single cases that are not part of any generalization. When a new case is encountered, SAGE uses analogical retrieval to find the most similar generalizations and ungeneralized examples among those already in the generalization context. If it is most analogically similar to a generalization and that similarity score is above a certain *assimilation threshold*, it is added to that generalization. If it is most similar to an ungeneralized example and the similarity score is above the threshold, it is combined with that example into a new generalization. If it is not sufficiently similar to any of the generalizations or examples, it is just added as a new ungeneralized example.

There may be a large number of generalizations in the generalization context. The ability to maintain multiple generalizations is useful for handling disjunctive concepts, and the ability to maintain unassimilated examples is useful for dealing with exceptions and edge cases. A novel and important

feature of SAGE is that it typically achieves robust performance after only a dozen cases for each concept, making it faster (in terms of number of examples required) than most statistical learners.

SAGE is used in the Analogical Word Sense Disambiguation work described in Chapter 3. Generalizing across training examples allows for more compact storage of the training data with no loss in disambiguation performance.

2.5 Evaluation with Question Answering

Question answering is the task of inferring an answer from a knowledge store, given a question. If the question is posed in natural language, question answering has a natural language understanding element as well. Question answering is a very common method for evaluating the performance of AI systems as a means of evaluating learning.

Question answering is itself a full avenue of research, and comes with its own set of challenges. I use question answering for evaluation in both the case construction work and the analogical dialogue acts work, as described in Sections 3.7 and 5.4, respectively. I use open-ended question answering, rather than multiple-choice or true/false questions.

2.5.1 Parameterized Questions

Parameterized question answering is a special question answering task where the questions are generated by filling in slots in question templates (Forbus et al., 2007). It can be used to generate a large number of questions for testing purposes. Parameterized question generation requires a set of *topics*. This topic list can be generated by collecting all of the entities in a reading, or that list can be automatically or manually filtered. For the purposes of the work we discuss here, topics are entities mentioned in what has been read by the system. For example, if the system reads about a solar heating

system, a topic might be the Sun, the solar heating system at a particular time of the day, or a particular flow in the system.

The parameterized question generation system has a set of *parameterized question templates*, which are manually authored. These templates are used to generate questions from the topics. Each template has one or more collections as *slots*. A topic that belongs to that collection can fill that slot. For example, the "What causes this?" template has one slot, for an `Event`. For each `Event` in the set of topics, the system asks what caused the event. Because many events are caused by something, this is a useful way to test comprehension and look for holes in the system's knowledge.

A parameterized question template may have multiple slots. For example, the "Compare and contrast" template has two slots, both for members of the collection `Thing`. (Events and abstract concepts are still members of that collection, so this is less restrictive than it may sound.) For each pair of things in the topic list, the system will compare and contrast those things. For a large topic list, this is a huge number of comparisons; for this reason, the system supports using a list of *topic pairs*, rather than a list of individual topics. Only topics that are paired with each other are used together by the parameterized question generation system. For example, consider a task where the system has read about dolphins and porpoises. [dolphin, porpoise] is an interesting topic pair for a compare and contrast task; there are interesting similarities and differences between those two concepts. [ocean, tail] is less likely to produce interesting similarities and differences, and should probably not be used as a topic pair even though both of those concepts appear in the source text.

Each parameterized question template has one or more *solve goals* associated with it. These use the Solve system built into fire. Solve is an incremental and/or problem solver. It uses *solve suggestions* that

define the nodes in the and/or tree. These can use the reasoning capabilities of the FIRE engine, as well as special outsourced predicates that are handled programmatically.

Parameterized questions have several benefits as an evaluation metric for reading comprehension. They can be used to measure where there is no clear gold standard, when few or no external questions are available, or when the questions that are available require knowledge that is out of scope. They also have some challenges as an evaluation metric. The correct answer still has to be authored. This can be time-consuming and imprecise for questions with complex answers, such as compare and contrast questions. The ceiling performance is not well-defined, because in many instances, the answer to a particular parameterized question is not present in the text. For example, the parameterized question generation system knows that a `Person` has biological parents. When it knows that something the system has read about is a `Person`, it generates a question asking who they are. However, that information is not found in most texts talking about most `Person` entities. Even a human reader would not be able to tell you who the parents of a person mentioned in a text are if the text does not mention those parents.

Connection-based case construction uses parameterized questions for evaluation, as described in Section 3.7. It uses an enumerated list of topic pairs, which it uses for a compare and contrast task, and a curated list of similarities and differences to be found. This avoids the ambiguous ceiling effect described above, as all of the similarities and differences to be found are things a human reader could find from the passage. Because Solve can access SME through outsourced predicates, it is capable of analogical comparison.

3 Analogical Word Sense Disambiguation

3.1 Goals

Word sense disambiguation is the task of selecting the correct sense of a word from a sense inventory that specifies alternate meanings for the word. Most existing WSD techniques rely on massive labeled training corpora, which are time-consuming to build and which do not necessarily generalize to other domains (Navigli, 2009). A WSD scheme that can operate at a satisfactory level with much less training data has the potential to allow for better scalability to new domains. Additionally, some domains may have a limited amount of training data available to begin with, so making the most of that is important for WSD.

Humans do not require anything like the quantity of training data required by most modern WSD systems to achieve much better performance. We believe that using cognitively-inspired strategies can allow a system to produce good results with dramatically less data. Analogical Word Sense Disambiguation (AWSD) is a demonstration of this. The work detailed in this chaper was first described in Barbella and Forbus (2013).

3.2 Word Sense Disambiguation

Word sense disambiguation is the task of selecting the correct semantics for a given word in a natural language text. For example, consider the word “star” in the sentence “He quickly emerged as a league star and his scoring entertained crowds.” This sentence is drawn from a Wikipedia article about Michael Jordan, the legendary basketball player. In this sentence, the intended meaning is the “famous person” sense of the word “star.” A system that selected the “astronomical object” sense of the word might run into trouble when it attempted to do further reasoning on the interpretation. Integrating the interpretation with existing knowledge would also be more difficult.

Word sense disambiguation is a difficult task, even for people: Navigli (2009) reports that, even for instances of "coarse-grained, possibly binary, sense inventories," interannotator agreement only reaches about 90 percent for human annotators, and on more fine-grained sense inventories it is between 67 percent and 80 percent.

Senses are usually drawn from a sense inventory – a collection of possible senses for a given word. The most widely used sense inventory is WordNet (Gale, Church, & Yarowsky, 1992; Miller, 1995; Navigli, 2009). Most work in word sense disambiguation involves making selections on the simple sense level. That is, the sense options for each word are just the type of thing or event the word represents. This strategy has some limitations when applied to a learning by reading task. It does not naturally produce or use the kind of rich conceptual information needed to support reasoning. Structured, relational representations are important to extract from language for reasoning. For example, understanding a verb in a sentence involves inferring how it connects the constituents to the event or relationship expressed by the verb.

For these reasons, our work uses the ResearchCyc ontology as a sense inventory, rather than WordNet. ResearchCyc was not designed specifically for this purpose, but offers several advantages. The senses associated with words in ResearchCyc are complete fact statements, rather than simple senses. Particularly for verbs, the senses in ResearchCyc are extended interpretations, which support reasoning. For example, the interpretation for "entertained" in "He quickly emerged as a league star and his scoring entertained crowds" includes not only `(isa entertain3767 EntertainmentEvent)`, but also `(spectators entertain3767 crowd4077)` and `(doneBy entertain3767 score3575)`. These latter two statements relate the entertainment event to the actors participating

in it. An interpretation choice that lacked those statements would be incomplete. We use this broader definition of word sense disambiguation in the work presented here.

The second limitation we wish to overcome is that most existing techniques rely on large hand-annotated corpora for training. For example, the Brown Corpus (Francis & Kučera, 1964) contains one million words, and the Wall Street Journal Corpus (Paul & Baker, 1992) contains 30 million words. In contrast, people manage to learn to read without using massive amounts of input labeled in terms of their internal representations. Much of the difference can certainly be accounted for by reasoning and application of real-world knowledge, something that current WSD systems do very little of, especially in a domain-general way. Those capabilities are likely necessary to achieve true human-level performance. Understanding how to learn to read without such large annotated corpora is an important problem, both scientifically and to make the construction of large-scale cognitive systems truly practical.

Sense inventories can be more or less fine-grained. Oftentimes words have a cluster of closely related meanings, and different sense inventories might lump these together or split them apart. For example, the word “star” can mean “famous person,” but has a closely related meaning in “principal actor in a production.” ResearchCyc is moderately fine-grained. It contains fewer total (atomic) senses on average than WordNet, but still contains a very large number of concepts.

3.2.1 Benefits of Word Sense Disambiguation

Word sense disambiguation serves several important roles in a learning by reading context. The first is to distinguish concepts that have the same name in natural language. For example “star” can mean “famous person” or “astronomical object.” A system that hopes to do reasoning benefits from distinguishing those, as they have very different properties. A famous person can make decisions and do other things that involve intentionality, while a star in space cannot. That is important if a system is

trying to determine why an event involving a star occurred. It is also important for learning what concepts are related – an agent is related to a famous person, while a planet is related to an astronomical object.

The second role that WSD plays is unification. Just as there can be several senses for a single word, there can be several words that map to a single sense in the sense inventory. For example, both "star" and "celebrity" can refer to a famous person. In general, things that we learn about stars (in that sense) are true about celebrities and vice versa, because the words are nearly synonymous, at least in their "famous person" senses.

Resolving words to concepts in an existing ontology assists with reasoning in other ways. For example, ResearchCyc knows that all instances of `FamousHuman` are also instances of `Person`. It also knows a lot of information about the collection `Person`, so knowing that something is a `FamousHuman` nets the system a lot of free additional information about that thing. This can be used for operations like error detection. If the interpretation indicates that a star is playing a role in an event that must be played by a human, but it is a `Star` rather than a `FamousHuman`, something has likely gone wrong with either the interpretation of which sense of "star" was meant or with the system's interpretation of the event. WSD is also important for knowledge integration. When new knowledge about topics is derived from text, it is important that it can be related to existing knowledge on those topics.

3.2.2 Strategies for Word Sense Disambiguation

Word sense disambiguation has been extensively explored. Its utility, its importance, and the relative ease with which it can be evaluated fairly objectively has made it a popular problem within the field of NLU. Navigli (2009) provides an excellent overview of the state of the field; we will only cover the strategy distinctions here that are most relevant to the work presented in this thesis.

One large division in WSD strategies is between supervised and unsupervised techniques. Supervised techniques use labeled training data to train a classifier, and then use the trained model to make new disambiguation choices. The biggest advantage of this is that these methods generally achieve the best performance. The disadvantage is that producing labeled training data in the quantities demanded by many WSD algorithms, as mentioned above, is extremely labor-intensive and requires some expertise. Furthermore, training data prepared using one type of corpus – news articles, blog posts, medical research articles – may not generalize to other types of text. Unsupervised strategies do not require this training, but also tend to perform at a lower level. Additionally, unsupervised strategies that build their own classes, rather than using a predefined sense inventory, can make it more difficult to hook the semantic representations into an existing knowledge store. This can be worked around by seeding the system with classes, however.

Within the category of supervised techniques is a smaller category of techniques that operate by preserving all of the training data and disambiguating by finding the most similar prior example, and then making the same disambiguation choice as that example. We will call these *example-preserving strategies*. These contrast with systems that use the training data to build a less penetrable classifier, such as a neural network or a support vector machine. There are two ways in which example-preserving strategies can vary. The most important one is the similarity measure. For example, a simple example-preserving strategy might look at which prior case has the most nearby words in common with the current ambiguity. A more complex one might integrate that measure into a larger similarity function that also takes other factors into account. The second axis along which example-preserving strategies can vary is the manner in which they choose which of the most similar examples to use. The simplest option is to choose only the one that is most similar, and make the same choice that it did. A k-nearest neighbor strategy lets the top *k* prior examples vote.

Example-preserving strategies have a few advantages. The first is that they are easily inspectable. The reason that a disambiguation choice was made will always be that a certain set of examples were the most similar. Provided that the similarity metric itself is reasonably comprehensible, this allows for human-readable explanations of why a choice was made. The second advantage of example-preserving strategies is that they are easily extensible. When new training data is encountered, extending the training set is as simple as just adding the new examples to the library of cases. It is not necessary to train a new classifier. This makes them appropriate for systems that slowly accumulate new training information over time, as a cognitive agent might.

3.3 Approach

The fundamental idea behind analogical word sense disambiguation is that choices made in similar prior circumstances are good sources of evidence for what to do in understanding new sentences. As discussed above, this is a class of WSD strategies that is somewhat common. At a high level, the system uses analogy to construct generalizations of training examples and analogical retrieval to choose the most similar from among those generalizations and examples in order to disambiguate new choices later. The analogical processing techniques we employ are based on structure-mapping theory (Section 2.4.1). Analogical matching is performed by the Structure-Mapping Engine (Section 2.4.2), which takes two structured, relational inputs, a base and target. The system uses MAC/FAC (Section 2.4.3) and SAGE (Section 2.4.4) for analogical retrieval and generalization, respectively. In training, cases are constructed representing the ambiguities and the choices made for those ambiguities. These cases contain information about the choice and the context in which it appears. During operation, cases are constructed for new ambiguities, and the system seeks the most structurally similar cases from the training set.

Our work differs from prior research in that it adopts conceptual representations for its sense inventories, and uses relational representations instead of only feature-based representations for its cases. Additionally, it is unique in making use of structure mapping. We assume there are other methods of disambiguation to fall back on when analogies are not available, and that successful instances of disambiguation are used to incrementally provide cases that can be used in analogical learning. For evaluation, we use hand-made disambiguation choices to prime the pump, to focus on what analogy can contribute.

The AWSD system is built into the Companion cognitive architecture (Section 2.3). Specifically, it is implemented as a disambiguation heuristic within the disambiguation harness (Section 2.2.6). It is the most complex and ambitious heuristic currently developed for that system, but it is also capable of handling disambiguation tasks that simple no-training heuristics cannot handle at all.

3.4 Method

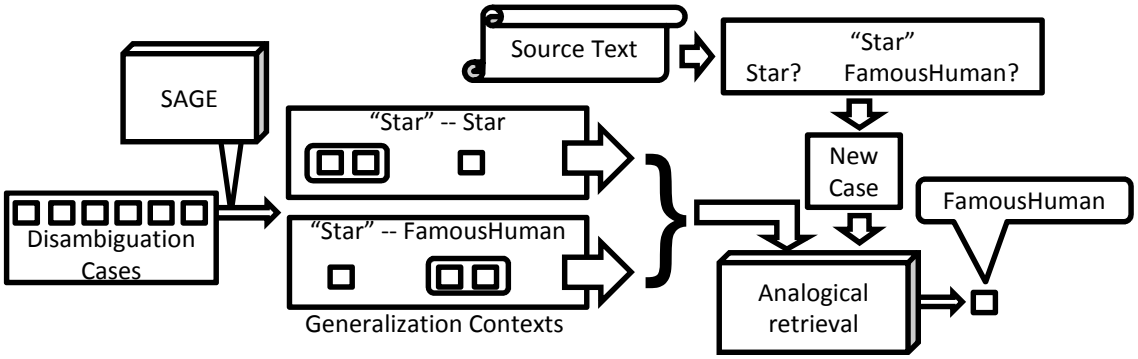
The method begins with a training corpus that has already been accurately disambiguated (such as one disambiguated by hand or with abduction or other WSD methods, and then checked and corrected.) It reads through each sentence in this corpus.

Ambiguities it encounters while reading are represented as choice sets (Section 2.2.4). For each choice set, the system constructs a case. It populates the case with information about the choice and the context in which it appears. The information contained in these representations and the form they take is the greatest factor in determining how the system operates. These are discussed in Section 3.5.

The cases are then added to generalization contexts using SAGE (Section 2.4.4). For word sense generalization contexts, each generalization context contains all of the examples built around choice sets that share a word and a sense. For example, all of the choice sets over the word “star” for which the

sense FamousHuman was the correct choice will appear in a generalization context together. The choice sets for which the correct choice was Star-AstronomicalObject will appear in a separate generalization context, even though the word was the same. Similarly, cases built around choice sets attached to the word "celebrity" will appear in their own generalization context even if the selected choice is FamousHuman. These generalization contexts together form a *generalization context family* (GCF), one for each word. That is, the word "star" has a GCF, "can" has a separate GCF, and so on. A diagram of the system operation is show in Figure 5.

Figure 5. A schematic diagram of how the AWSD method operates.



Generalizing from experience is an important form of information compression. While one of the goals of the AWSD system is to enable satisfactory performance with relatively little training data, it is also important that, as the system acquires more training data over time, it can incorporate that information in a compact way. While the average generalization size produced by the system in our experiments means that generalization only compresses the population library by a modest amount (see Table 4 in Section 3.8.1), it should help with scalability somewhat over longer periods.

An important parameter that influences the system's generalization strategy is the *generalization threshold*. Recall that the generalization threshold is the degree of similarity that two cases must have before SAGE will merge them. The higher the generalization threshold is, the more similar two cases have to be before they are merged. The system was tested with three different generalization thresholds. With a generalization threshold of 0.2, the system eagerly merges cases. With a threshold of 0.4, it is more conservative with merging. A generalization threshold of 1.0 means that the system will only merge identical cases. These cutoffs were determined to produce interestingly varied results though experiments on pilot data, but there is nothing exceptional about the exact numbers used. The effects of these strategies are discussed in Section 3.8.2.

Once we have accumulated a set of generalization contexts from a training corpus, the system can attempt to use them to disambiguate new texts. When a fresh choice set is encountered in a new sentence, a case is constructed for that choice set. This is done using the same method that was used to generate the training cases. This case is used as a probe to MAC/FAC (Section 2.4.3), as described in Section 3.6.

3.5 Representations

As mentioned above, the largest factor in the system's performance is what facts are chosen to represent an ambiguity. Throughout this section, we will use the example sentence "The heat flows to the object with the lower temperature," constructing a case for the word "flows" in that sentence. Deciding the form such a case takes involves determining what information will be represented and how that information will be represented. In particular, many sorts of information can be represented as either features or relations. For example, consider a fact that represents that "heat" is the subject of the sentence. One possible representation is `(subjectOfSentence heat)`. That representation is nice

and simple, but that is not the only possible representation, nor is it necessarily optimal. In particular, it may not be very well-suited for use with MAC/FAC. In the MAC stage, it will make almost no difference at all. If every case has exactly one `subjectOfSentence` fact, that does not help MAC distinguish between them. In the FAC stage, it potentially helps a little, but not very strongly. Because `heat` is just an entity in that representation, `(subjectOfSentence heat)` can match to any subjectOfSentence fact in SME, provided that heat is not matched to any other incompatible symbol.

Instead, *logical functions* are used to reify information about constants in order to improve sensitivity in analogical retrieval. Neither SME nor MAC/FAC, by design, is sensitive to specific constants that are the arguments to relations, such as `heat`. By redundantly encoding relational information as attributes (e.g., `(subjectOfSentenceOfChoiceSetFn (TheList heat)`), analogical retrieval picks up on information it would otherwise miss. The representation `(isa ChoiceSet-227 (subjectOfSentenceOfChoiceSetFn (TheList heat)))` brings in the name of the choice set, but more importantly it makes the fact a feature. This means that during the MAC stage, other cases where "heat" was the subject of the sentence are more likely to be retrieved, because they will also have facts that look like `(isa ?choice-set (subjectOfSentenceOfChoiceSetFn (TheList heat)))`.

The reason that "heat" is represented as `(TheList heat)`, rather than just `heat`, is that the subject of a sentence can be multiple words long. For example, if "Bill Gates" was the subject of the sentence, the representation would be `(isa ChoiceSet-227 (subjectOfSentenceOfChoiceSetFn (TheList bill gates)))`. The actual words are used here, rather than the semantic interpretation of the words, because the interpretation is not available when building the probe cases for disambiguation.

There are a large number of possible features that could be represented. Information about the parse tree the ambiguity appears in, the semantics of the other words in the sentence, and notable sentence structure features – such as whether the ambiguous word has a prepositional phrase attached to it – are all things that might be stored. Similar information about nearby sentences could also potentially be stored. Given that there are multiple possible ways to represent the facts that we do choose to include, the space of possible representations is very large.

It is likely that there are diminishing returns on including additional things in the case. For example, it is unlikely that detailed parse information for sentences many sentences away in the source text will be relevant. Very large cases also risk including a lot of things that are just noise, which has the potential to interfere with both retrieval and generalization.

In the method described here, we only consider features local to the sentence that the ambiguity appears in. There are a few advantages to this. First, it allows the cases to be constructed from isolated sentences, which is useful for corpora that consist of those. Second, it allows for disambiguation on sentences that appear without context.

As mentioned above, probe cases for new ambiguities are constructed in the same fashion as training cases are. Some statements in the case – including the one that indicates the correct choice, for example – may not be computable without knowledge of what the correct choice is. This means that the probe cases and the training cases look a little different. This does not pose a problem for the system. The system does not rely on the presence of every possible feature. In many cases, certain features simply do not exist. For example, the feature that records when a prepositional phrase is attached to the ambiguous word is simply omitted if there is not one.

The representations used were arrived at by starting with a small set of features drawn from prior work – in particular, the inclusion of the other words in the sentence as features was motivated by the frequency with which this is incorporated into other WSD techniques. From there, we added additional features that form a structured representation of the context that the choice appears in. These were tested on a small set of sample data. It is not the intent of the system that different features should be used for different tasks, and the set of features presented here is designed to be general-purpose.

The parameters that control which features are included in the case are stored in the knowledge base. This allows for easy experimentation and extensibility. Additionally, this setup will allow the system to someday reason about the strategies it is using.

3.5.1 Features Used

In our example sentence, "The heat flows to the object with the lower temperature," each word in the sentence that generates a choice set will have a case generated for it. Consider the construction of the case for the word "flows". This case includes a set of statements that describe the context in which this decision appears, which incorporates semantic, surface, and parse information. These statements include:

- A statement indicating the word sense used. For this example, the word sense includes both the sense itself, `FluidFlow-Translation`, and a role relation statement that connects the action to one of its actors, `(primaryObjectMoving flow-22523 heat-22505)`. In a case for a noun, this will often be simpler, and only indicate that the word belongs to a particular collection (e.g., `FamousHuman`);
- Statements describing the choices selected for the other words in the sentence (e.g., `ThermalEnergy`). These statements encode the semantics of the sentence. Most nouns

generate statements describing what something is, such as `(isa heat-22505 ThermalEnergy)`, while other words generate more complex statements, such as `(possessiveRelation object-22573 temperature-22651)`.

This provides a very simple form of semantic information about the context. Other facts record surface information:

- A statement indicating that the word covered by the choice set is `flows`;
- Statements indicating the other words in the sentence (`heat`, `lower`, etc.). Note that we do *not* exclude words from a list of stop words, as common words such as prepositions may be correlated with particular senses of a word.

Information from the syntactic analysis of the sentence is also included:

- The subject of the sentence (`heat`);
- The part of speech of the word (`verb`);
- If the word appears in a compound noun phrase, the word(s) with which it appears are also recorded. In the sentence "He emerged as a league star and his scoring entertained crowds," for the word "star" this would be (`league`);
- If the word has a prepositional phrase attached to it, the preposition; in this example, "to the object…" is attached to "flows", so (`to`) is recorded.
- If the word appears in a prepositional phrase, the preposition.

Table 1: A portion of a disambiguation case; only 10 out of a total of 32 facts are shown. The first three facts shown concern conceptual information. Nine facts out of 32 concern semantics, with the rest encoding syntactic or lexical information. Full case in Section 0.

```
;;; Subset of the case generated for "The heat flows to the object with the lower
;;; temperature."
;;; The word sense used, FluidFlow-Translation, and a role relation connecting the
;;; action to one of the actors.

(selectedChoice ChoiceSet-22757 Choice-22758
  (and (isa flow-22523 FluidFlow-Translation)
       (primaryObjectMoving flow-22523 heat-22505)))

;;; Other choices made in the sentence.
(otherSentenceSelectedChoice (isa heat-22505 ThermalEnergy))
(otherSentenceSelectedChoice   (possessiveRelation object-22573 temperature-22651))

;;; The word covered by the choice set
(isa ChoiceSet-22757 (ChoiceSetForPhraseFn (TheList flows)))

;;; Other words that are present in the sentence - "heat" and "lower".
(isa ChoiceSet-22757 (wordInSentenceOfChoiceFn heat))
(isa ChoiceSet-22757 (wordInSentenceOfChoiceFn lower))

;;; The subject of the sentence
(isa ChoiceSet-22757 (subjectOfSentenceOfChoiceSetFn (TheList heat)))

;;; The part of speech of the word
(isa ChoiceSet-22757 (partOfSpeechOfChoiceSetFn verb))

;;; Preposition attached to the word
(isa ChoiceSet-22757 (hasPPWithLeadingPrepositionFn to))
```

Table 1 illustrates a portion of the case generated for the decision about "flows" made in our example sentence, "The heat flows to the object with the lower temperature."

3.6 Retrieval and Evidence Production

Given a new occurrence of a word to disambiguate, our technique uses MAC/FAC to retrieve relevant prior cases and/or generalizations, if any. Recall that each generalization context covers a word plus a particular sense for that word. The union of all generalization contexts involving that word, for all of the senses seen so far, constitutes the case library. In the example above, the system uses a separate generalization context not only for different senses of the word "flows", including `<(TheList flows), (isa ?entity FluidFlow-Translation)>`, but for different combinations of role relations, such as `<(TheList flows), (and (isa ?entity FluidFlow-Translation)`

(primaryObjectMoving ?entity ?object)>. In our example, the second of these is the correct sense; the system will be successful if it retrieves a case or generalization that had been part of that generalization context.

The candidate inferences generated by SME during retrieval are examined to determine which word sense choice is best supported by experience. Given the way that cases are encoded, these inferences will include the decision associated with that case or generalization. The system uses that prior decision to generate evidence for the current decision. Returning to our example, when the occurrence of "flows" in "The heat flows to the object with the lower temperature" is used as a probe, the system retrieves a generalization of four previously seen cases, including "Despite this, volume flows toward the can with the low depth" and "Volume flows to the lower depth." This generalization is retrieved because of all of the generalizations and examples in the GCF, it is the most similar. The analogy between these generates a candidate inference of the form

```
(selectedChoice
ChoiceSet-16374
(:skolem Choice-35146)
(and
 (isa (:skolem flow34459) FluidFlow-Translation)
 (primaryObjectMoving ?entity (:skolem heat34459)))
```

suggesting the FluidFlow-Translation sense that also includes primaryObjectMoving, which is correct in this case. The reasons that the generalization is chosen include not only a shared subject and several shared words, but a similar structure – the sentence and the generalization both have a prepositional phrase starting with "to" attached to the word "flows". Table 2 shows a portion of the match between the two cases.

Table 2: A portion of the match from the retrieval made with the probe case for "The heat flows to the object with the lower temperature" in experiment 2. `(GenEntFn 10 flows)` is a generalized entity that was created when cases were merged during training. Some representations have been simplified for space and readability.

Base Case	Retrieved Target
((hasPPWithLeadingPrepositionFn to) ChoiceSet-3583789884-50713)	((hasPPWithLeadingPrepositionFn t0) (GenEntFn 10 flows))
(possessiveRelation object6515 temperature6596)	(possessiveRelation can6515 temperature6596)
((ChoiceSetForPhraseFn (TheList flows)) ChoiceSet-3583789884-50713)	((ChoiceSetForPhraseFn (TheList flows)) (GenEntFn 10 flows))
((wordInSentenceOfChoiceFn lower) ChoiceSet-3583789884-50713)	((wordInSentenceOfChoiceFn lower) (GenEntFn 10 flows))

In some situations, the system retrieves an incorrect example; for example, while attempting to disambiguate the first "depth" in the sentence "The depth of the can differs from the depth of the other can," the system retrieves the sentence "Depth differs from volume." While those sentences are similar structurally and have other features in common, the first refers to a specific depth of a particular object, while the second refers to the general concept of depth. The distinction is relatively subtle, but as the representations are different internally, the system does not select the best choice based on this retrieval.

Because a sense that appears more frequently will have more examples in its generalization context, this process automatically incorporates sense frequency as a factor. That is, if a sense is very common in the training corpus, there will be more generalizations and examples available for it to match.

3.7 Evaluation

Evaluating WSD is fairly straightforward. The senses chosen by the system can be compared to a gold standard. Generating such gold standards can be time-consuming and requires familiarity with the representations being used, but they provide a sensible and objective measurement. The gold standards

used for evaluation are created by having the reading system process the text to be used as a test corpus, and then manually selecting appropriate semantic choices. It is necessary to use human-verified selections for this to maximize the accuracy of the gold standard.

There are some subtle challenges involved in using gold standards as an evaluation, however. The first is defining what constitutes satisfactory performance on a particular disambiguation task. It is easy to compare methods to each other by comparing their performance against each other, but that does not measure whether any or all of the methods are actually good enough to be fit for a particular purpose.

Another complication is that in many cases there is not a single unique correct disambiguation choice, particularly in sense inventories that are fine-grained or hierarchical, as ResearchCyc is. For example, the English word "cat" can refer to either specifically domestic house cats or to the broader cat family that includes big cats, wildcats, and cheetahs as well. In some cases, the distinction may not matter, but in other cases it may. Even trickier are instances where one of the senses is a more specific version of another. For example, consider the word "flow" in "Water flows into the storage tank." Suppose two of the semantic choices are `FluidFlow-Translation` and `Movement-TranslationEvent`. The former is better, as it is more specific, but the latter is by no means incorrect. In a case like this, it might be correct to award credit for either answer, or it might be correct to award partial credit for the worse-but-still-correct answer. Similarly, there are cases where one choice subsumes another. For example, two of the choices for "temperature" in "The water's temperature rises" are

```
(isa temperature6206 Temperature)
```

and

```
(and
  (temperatureOfObject water6185 temperature6206)
  (isa temperature6206 Temperature))
```

The first choice is not *wrong* per se, but it is significantly less useful than the second. It is true that the water's temperature is an instance of `Temperature`, but for most reasoning purposes, the fact that it is the temperature of the water is likely to be relevant. In a case like this, we would likely want to only credit the system if it selected the second of the two choices. Cases like these mean that it is best to frame WSD not as a task of selecting a single correct option from a group of otherwise incorrect ones, but as a task of selecting better options from a group of choices of varying quality. To avoid introducing additional layers of human judgment into the evaluation, the system was only given credit for choosing the very best choice from the choice set, and received no credit for selections that are technically correct but incomplete, or only partially correct.

An additional difficulty is that, for many ambiguous words, one sense dominates the others in frequency, even over broad topic areas. In many corpora, for example, the "clear atmospheric gas" sense of the word "air," used as a noun, might be many times more common than the "impression or manner" sense of the word (as in "He had an air of mischief about him.") A system that simply always guessed the "atmospheric gas" sense of the word would likely achieve a quite high overall precision, but is of little use as a disambiguation system.

A final difficulty that occurs when evaluating WSD is that human judgments about word sense are not completely consistent. As noted above (Navigli ,2009), interannotator agreement ranges from 60% to 90% depending on the granularity of the sense inventory. In addition to raising questions about what it means for a machine to do better than this, it also introduces a degree of uncertainty into gold standard comparisons as a metric.

3.7.1 Baselines

The most basic possible baseline that WSD performance can be compared to is chance. At a minimum, a WSD algorithm should be able to do better than a system selecting senses uniformly at random from the sense inventory. A much tougher baseline is one where the system always selects the most common sense of the word in the training corpus. As described above, many word sense distributions are very lopsided, so this is often a very difficult baseline to beat in terms of absolute precision, despite the fact that it is not very helpful as an actual disambiguation strategy. Finally, we can compare to the performance of systems like CatchWord (Hearst, 1991), which operate under similar constraints.

3.8 Experiments

Our primary hypothesis is that analogical processing over relational representations can provide accuracies comparable to other word sense disambiguation algorithms. Since prior research has focused on very different sense inventories, representations, and texts, performance comparisons must be made with care. Nevertheless, they provide a useful component-level benchmark, since accuracy is a desirable property in all cases. To provide evidence for this hypothesis, we conducted two experiments.

3.8.1 Experiment 1: Binary Word-Sense Task

The first study tests the ability of our technique to distinguish between senses of a specific word. As noted above, our word sense disambiguation task uses a very different kind of sense inventory than traditionally used, but distinguishing noun senses provides the closest comparison.

To test performance on a traditional word sense disambiguation task, we created a test corpus of 104 sentences that used the noun “star” in either the sense of a famous person or an astronomical object, as per our running example. The sentences were drawn from the English and Simple English Wikipedias,

and simplified when necessary to fit the syntax that EA NLU handles (Section 2.2.4). The sentences were roughly evenly divided between the two word senses. The full corpus appears in the appendix.

We used five-fold cross validation, with approximately ten examples of each word sense in the test set, with the remainder of each corpus providing the set of cases used as input for training.[1] This experiment is not intended to model a normal reading experience; the senses of "star" do not occur with the same frequencies in this corpus as they do in more natural corpora, and are effectively drawn from many different documents. Indeed, in a normal reading situation, the fact that polysemous words that recur within a document typically carry the same sense (Gale, Church, & Yarowsky, 1992) can be exploited. This experiment is intended to test performance in situations where a comparable number of examples of two different senses exist in the training set.

Table 3 summarizes the results of three conditions, varying only in assimilation threshold. By setting the threshold to 1.0, SAGE only stores examples, as otherwise cases must identical to be assimilated at that threshold. The 0.2 condition is the opposite extreme, leading it to form generalizations more readily, while the 0.4 condition is more conservative. Importantly, there is no significant difference in accuracy between these three conditions, suggesting that, for this task, more aggressive generalization strategies can be used without impacting performance. Accuracy is significantly and substantially different from chance (50%) in all conditions.

[1] The "astronomical body" sense appears with slightly greater frequency in the corpus because a greater number of sentences that use that sense also use it more than once.

Table 3: Accuracy for binary disambiguation of *star*, mean of five-fold cross-validation. Results significant vs. chance; $p < .05$. n = 113 in all conditions. No significant difference between generalization and retrieval.

Sense	Generalization: Threshold 0.2	Generalization: Threshold 0.4	Retrieval Only
FamousHuman	60%	75%	71%
Star	95%	85%	85%
Total	79%	81%	79%

The accuracy achieved is in the range reported for state of the art algorithms with more traditional word sense inventories. Hearst (1991) also used a method involving storing syntactic and semantic information from training data for disambiguation, and hence is the most comparable system to ours. Of the six words that Hearst used, accuracies after 40 examples ranged from 75% to 85%,[2] and from 82% to 88% on the two words for which 50 or more examples were used. Thus although our results are not directly comparable due to differences in training data, test data, and representations, our accuracy is comparable to Hearst's.

Our results are also similar to those reported by other systems. GAMBL, which employs example-based learning and a genetic algorithm for selecting which features to use, achieves an accuracy of 74% on binary course-grained word senses (Decadt et al., 2004). The IMS system (Zhang & Ng, 2010), which uses support vector machines and cross-linguistic parallel corpora, achieved a 73% accuracy rating on a lexical sample test corpus. The best-performing systems on Semeval-2007, which featured a lexical sample task,[3] achieved an f-score of about 89% (Pradhan et al., 2007). There are several differences between that task and ours. One hundred words were tested, the number of training examples ranged from 32 to 1009 (mean 223), and the number of senses present ranged from one to 13 (one to eight

[2] This leaves out "bass", which hit 100% after only 25 examples, and thus was an outlier.
[3] http://nlp.cs.swarthmore.edu/semeval/tasks/task17/description.shtml

when senses with fewer than three instances are excluded). The corpus is not sense-balanced, with the most-frequent-sense baseline achieving 78% accuracy.

There are word sense disambiguation methods that provide higher accuracy, but at the cost of vastly more training data. For example, Yarowsky (1995) achieves 94 to 98% accuracy on similar tasks using a semi-supervised method. However, that method used an (unlabeled) corpus of 400 to 11,000 examples per word, whereas ours requires relatively little labeled training data and does not require a large corpus to operate.

Acquisition rate is also an important concern for learning by reading, as word sense disambiguation is just one component of a larger system. Consequently, we also measured how few examples sufficed to achieve reasonable levels of performance. Keeping the assimilation threshold at 0.4, when we varied the number of training examples from 20 to 80, the accuracy ranged from 73% to 81%, a difference that is not statistically significant (n = 113). In other words, by twenty examples the system is already performing reasonably well.

3.8.2 Comparing Generalization Strategies

We noted earlier the lack of significant difference between the accuracy for the retrieval-only and generalization conditions. One possible explanation for this is that cases are not being generalized, or that only ungeneralized examples are being used in retrieval. To test this, we examined how the method works under each condition in more detail. Table 4 illustrates, for one randomly-chosen fold, the number of generalizations and examples produced, where Gen Size indicates the number of examples assimilated to produce a generalization. As expected, a lower assimilation threshold leads to more and larger generalizations. In the 0.4 condition, generalizations provided evidence 23% of the time, and in the 0.2 condition, generalizations provided evidence 21% of the time. This difference is not statistically

significant. As the system accumulates yet more experience, some of what are now cases would become generalizations, as sufficiently similar sentences are encountered. Thus we believe that the full benefit of generalization may only be apparent when used in the intended context of learning by reading, where a system will be reading very large amounts of text. We expect that generalizations will be used most of the time as a system becomes more experienced, and the information compression they provide will help keep memory loads reasonable.

Table 4: Generalizations created on first fold of Experiment 1.

Trial	Examples	Ungenl'd Examples	Gens Created (Star)	Gens Created (FamousHuman)	Min Gen Size	Max Gen Size
Generalization: Threshold 0.2	92	34	16	12	2	3
Generalization: Threshold 0.4	92	44	14	10	2	2
Retrieval Only	92	92	0	0	N/A	N/A

3.8.3 Experiment 2: Progressive Learning Task

Experiment 1 served to examine accuracy in the kind of test traditionally used in word sense disambiguation research. In this subsection we introduce a new kind of test, involving connected text, that is designed to be a more natural evaluation for word sense disambiguation in the context of learning by reading. Recall that our hypotheses are that (1) analogical word sense disambiguation will have reasonably high accuracy, compared to other systems, while reading connected text about a topic and that (2) its rate of learning will be rapid. Experiment 2 provides evidence for both hypotheses.

We divided Chapter 2 of a popular science book (Buckley, 1979) into contiguous sections, each ten sentences long. There were a total of 90 sentences after simplification, leading to nine subsections. For each section, the disambiguation procedure was run using the generalization contexts generated from gold standards of the previous sections as the source of cases for disambiguation. (This means that for

the first section there will be no correct answers.) We note that this is an idealization: mistakes in understanding commonly occur when people read, and they will be even more likely in reading systems that have a much weaker grasp on the world. After recording accuracy, cases for that section were automatically generated, based on manually selected word sense choices, to extend the generalization contexts accordingly.

There are several things worth noting about this experimental setup. First, the system gets many choices correct simply because they are monosemous within the text, so the only sense that has been seen before is the correct one. The reminding will produce the correct sense regardless of which generalization or example it retrieves. For example, the word “pan”, within the text used in this experiment, always refers to a cooking pan. We refer to such choice sets as *easy*, because any system with access to the set of training data could get them correct by simply choosing the only attested sense. Note that a singleton choice set is not automatically easy, as the correct thing to do may be to select no choice; a singleton choice set for a word that has never been seen before will not have a selection made, as the system is not allowed to guess.

There are also situations where a previously seen word is now being used in a novel sense, and hence there are no cases with the correct sense. We refer to such choice sets as *impossible*. Choice sets for words that have never been seen before are impossible, even if it is a singleton choice set.

The most interesting situations are where multiple senses have previously been encountered and one of them is actually correct. We refer to these as *interesting* choice sets. Every choice set is either easy, impossible, or interesting. We focus on the interesting choice sets in measuring the accuracy of analogical disambiguation, but we tabulate the others because they help provide upper and lower bounds on how much can be obtained by any learning system that accumulates information while

reading.

Table 5: Results from Experiment 2– Progressive performance on sections of a text, with generalization at an assimilation threshold of 0.4.

Section	Total	Total Accuracy	Impossible	Easy	Interesting	Interesting Accuracy
1	40	0%	40 (100%)	0 (0%)	0 (0%)	N/A
2	61	30%	43 (70%)	18 (30%)	0 (0%)	N/A
3	51	39%	28 (55%)	18 (35%)	5 (10%)	80%
4	66	59%	22 (33%)	25 (38%)	19 (29%)	95%
5	68	32%	41 (60%)	17 (25%)	10 (13%)	60%
6	57	35%	31 (54%)	10 (18%)	16 (28%)	63%
7	62	61%	20 (32%)	11 (18%)	31(50%)	87%
8	69	49%	23 (33%)	27 (39%)	19 (28%)	37%
9	54	48%	17 (31%)	6 (11%)	31(57%)	65%

Generally, most words are used with only one word sense within a given document (Daelemans, Van Den Bosch, & Zavrel, 1999). There are 131 interesting choice sets in our experiment, out of 528 total (24.8%). This is higher than might be expected, but they represent only 18 distinct words. Most of the 130 words that generate choice sets are monosemous within the text. The 18 words that generate at least one interesting choice set are those which are fairly common in the text, including “of”, “can” and “temperature”. Only 52 out of 131 interesting choice sets are choices between word senses alone: the 79 others involve the same word sense but differ in semantic frames, i.e., in role relations that interconnect constituents. Thus this problem is substantially different than what is normally addressed in research on word sense disambiguation.

Table 5 summarizes the results. For each section, Total is the total number of choice sets generated by the NLU system. In general there are fewer choice sets than there are words, as there are some words (e.g., “an”, “the”) for which the language system does not generate choice sets. Total Accuracy is the number that the procedure got correct. Impossible, Easy, and Interesting indicate how many such choice

sets that section contained. The first section had no prior examples, and the second section had no interesting choice sets, so Interesting is zero in both sections. The last column gives the system's accuracy on interesting choice sets. For them, the mean number of senses previously seen is 3.13. The total accuracy on interesting choice sets is 70.2 percent. Note that this number cannot be directly compared to accuracy numbers from most other disambiguation experiments (including from our Experiment 1), as it covers only disambiguation cases where multiple senses of the word have previously been seen, which is not what experiments typically measure. However, it is still encouraging that it is comparable in accuracy to other disambiguation techniques applied to what we view as simpler disambiguation problems.

The total accuracy on all choice sets is 41.1 percent. Note that this includes section 1, where all choice sets were impossible. The total accuracy on words seen before (easy, interesting, and impossible choice sets that previously appeared in the text with a different sense) is 71.3 percent. Again, owing the nature of the task, we cannot directly compare these numbers to most results on traditional word sense disambiguation, but it does suggest that analogical disambiguation can be useful for learning by reading.

Table 5 also highlights how local properties of the text can affect the performance. Sudden shifts in sense use can cause temporary accuracy drops. For example, there is a topic shift at section 5 that is responsible not only for a drop in the accuracy over interesting choice sets, but also for the rise in the number of impossible choice sets. Note also that there is not a monotonic upward trend in accuracy on interesting words from section to section. This might seem surprising, but new words continue to be introduced throughout the text, and words that are not introduced until later have no precedents.

To examine learning rate on interesting choice sets, Figure 6 tabulates accuracy versus the number of available prior cases. Importantly, accuracy is in the range of 65% even with only a handful of examples.

This suggests that analogical word sense disambiguation can provide useful evidence, even in the early stages of becoming familiar with how words are used. The accuracy rises from 65% in the conditions with four or fewer previously seen instances to 76% in cases with 13 or more, although this increase is not statistically significant.

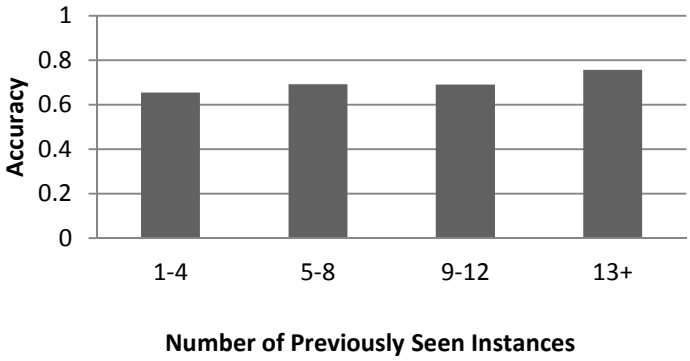


Figure 6: Accuracy on interesting choice sets for experiment 2, broken down by number of previously seen examples, with n ≥ 26 for all groups. All results are significantly above chance, but there is no significant difference between categories. In 13+ category, the maximum number of previously seen examples is 25, whereas the mean is 16.9.

3.9 Discussion

We have described a new approach to word sense disambiguation that uses human-like models of analogical processing operating over relational representations that combine linguistic and conceptual information produced in the course of natural language understanding. The system operates either with analogical retrieval alone or with analogical retrieval and generalization. It provides comparable accuracy to traditional word sense disambiguation algorithms that rely on feature-based representations, while requiring relatively little training data. It achieves 72% accuracy with just 20 training sentences. While that is not quite as accurate as a supervised system on a binary

disambiguation task, it is comparably strong for the quantity of training data. This is important for learning by reading systems, which attempt to learn without the benefit of a large annotated corpus.

We also described an experiment that tested the system's performance in a situation more like those that are faced by learning by reading systems. Our unconventional experimental setup was intended to measure the ability to disambiguate semantics with just an understanding of the previous sections of the text. Our system's performance on words with just a few previously-seen examples was surprisingly high, up to 70% even after only a few examples of a word. This is promising, as it suggests that the technique could be valuable for learning by reading.

Finally, we note that in applying analogical processing to word sense disambiguation, we did not need to modify the SME, MAC/FAC, or SAGE systems. This provides additional evidence as to the generality of analogical processing, and highlights its potential for use in other cognitive systems and other settings.

4 Connection-Based Case Construction

4.1 Goals and Motivations

Part of what makes reading powerful is that it allows a system to acquire large amounts of closely related and well-connected knowledge. One major challenge that comes with learning by reading is the task of organizing that knowledge. For example, one 80-sentence chapter of *Sun Up to Sun Down* (Buckley, 1979) allowed the system to generate 733 statements directly from the text. An 88-sentence web article comparing and contrasting dolphins and porpoises (Dolphin vs. Porpoise, 2015) allowed the system to generate 751 statements. The set of statements from each source are all loosely interrelated, as they all come from a single source, but some facts are much more closely related than others. This chapter addresses the task of organizing the facts into coherent cases.

Structure mapping, the model of analogy we use in this work (Section 2.4.1), operates over structured representations. One thing that makes analogy so useful is that these structured representations are a natural way to describe a wide variety of things, especially complex systems or processes. Creating structured representations, however, is not easy. We would like a system to be able to organize the things it learns by reading into cases that are useful for structure mapping and other forms of reasoning.

An LbR system attempting to accumulate knowledge over time benefits from being able to integrate that knowledge with its existing knowledge (see Section 2.1.2). There are several things a case construction algorithm can do to aid with knowledge integration. Analogical generalization (see Section 2.4.4) can be used to build up generalizations of situations, but only if the information is represented as structured cases. The system represents the cases using the same ontology as its existing knowledge base. This has several advantages for knowledge integration. First, it allows cases to be more easily

integrated into the knowledge base, as the terms being used are the same. Second, it allows the existing knowledge to guide how new information is integrated.

Case construction is the task of building a case, often by selecting a set of facts from a larger set. The case construction algorithms I describe here all take a discourse interpretation (Section 2.2.1) as input. They select subsets of that interpretation that are useful for reasoning. For example, in a textbook chapter about solar energy there might be a description of a solar heating system. The system selects a subset of the facts in the discourse interpretation that form a useful description of a concept of interest, such as the solar heating system. This allows the system to make useful comparisons. For example, it can compare the solar heating system to a rainwater collection system. It can also compare the state of the solar heating system at different times during the day.

Throughout, I use *case comparison* as the primary use example. Case comparison is the task of finding similarities and differences between cases. It has been used or proposed for use in a variety of reasoning applications (Brüninghaus & Ashley, 2001; Peterson, Mahesh, & Goel, 1994; Chaudhri et al., 2014).

The case construction strategies I describe here exploit connections between fact statements in interpretations generated while reading. Some connections are based on the parts of the source used to produce the facts. For example, two facts might be connected if they were derived from the same sentence. Other connections are derived from the facts themselves. For example, two facts might be connected if they mention the same entity.

This chapter presents four methods for building cases from discourse interpretations and compares their performance. Two make use of only natural divisions in the text, such as paragraph breaks. Two also take advantage of connectivity properties of the entities mentioned in the text. All are constructed using a *seed* entity as a starting point. For example, the term “solar heating system” in the sentence “At

the start of each day, the solar heating system is semifull" was one seed that the system used. This seed was drawn from a simplified chapter of *Sun Up to Sun Down* used in the corpus. It will be used as an example throughout this chapter. I begin with a brief summary of the four methods used, and then move on to what motivates them. I then describe how they are constructed. I close with an evaluation of the methods based on their performance on a practical task, and discuss this performance, including an analysis of the types of error produced by each method. The work detailed in this chapter was first described in Barbella and Forbus (2015).

4.2 Overview of Case Construction Methods

In this chapter I describe four methods for building cases from collections of facts derived through reading. The first method, Local Sentence Interpretation (LSI; Section 4.5.1), simply uses all of the facts derived from the sentence where the seed appears. The second, Local Paragraph interpretation (LPI; Section 4.5.2) uses all of the facts from the paragraph where the seed appears. The third, Sentence-Based Segmentation (SBS; Section 4.5.3), uses all of the facts from *all* of the sentences where the seed appears. The fourth, Fact-Based Segmentation (FBS; Section 4.5.4), adds facts in iterative waves. At each step, it adds facts that share entities with facts already in the case. Before discussing the implementation of the methods, I discuss what motivates and constrains them.

4.3 Desired Features for Cases

4.3.1 Coherence

One of the most basic goals for a system that builds cases using information learned while reading is that those cases should be *coherent* – that is, they should be a collection of information that is usefully related, rather than an unrelated collection of facts. This goal is so basic that it can almost go unstated.

There may be *some* use for a truly arbitrary group of facts that happen to all have been pulled from the same source, but such uses are not common in reasoning.

There is no simple way to measure coherence. While some automatic measures exist, such as graph connectivity measures, those do not necessarily capture utility in any meaningful way. One common strategy is to use performance on external measures as the measure of quality, as in Wei and Croft (2006). They use the topics for retrieval, and we follow suit by choosing an external measure to evaluate the quality of the cases the system produces. Our task is one for which this is especially appropriate. First, the end goal for the algorithms presented here is for use in reasoning tasks, so it makes sense to measure those uses directly. Second, the case construction algorithms produce sets of related facts, rather than clusters of words, as some similar work does. This makes measures based on things like the relatedness of the Wikipedia articles for the contents of the cases less useful.

In the case of our example seed, a coherent case likely contains other facts about the solar heating system operating at the beginning of the day, and as few facts as possible that are not related to that.

4.3.2 Interpretation-Consistency

A second desirable feature for case construction methods that build from discourse interpretations is that they should not introduce new errors. That is, if the source interpretation is true, the constructed case should also be true. This is not a trivial constraint. Not every statement in a discourse interpretation is true when removed from that discourse interpretation. This means arbitrary statements from a discourse interpretation cannot just be added to a case without potentially producing an untrue case. I will refer to cases that are true, given that the discourse interpretation is true, as *interpretation-consistent*.

One way such errors can be introduced under our representation scheme involves causation and constituent DRS cases. Suppose two constituent DRSs are involved in a causation relationship that looks like this:

```
(causes-SitSit DRS-01 DRS-02)
```

`causes-SitSit` is a causation predicate that connects two situations. It indicates that the situation described by the facts in `DRS-01` causes the situation described by the facts in `DRS-02`. However, if in the process of creating the case, the system includes that causation statement but fails to include some of the facts in `DRS-01`, the statement might not be true. This means that extra care needs to be taken when including facts that mention constituent DRSs or facts that are themselves within a constituent DRS.

For example, consider the sentence “Angry dogs attack people.” The interpretation is as follows:

```
In Discourse-2172:

     a) (implies-DRS-DRS DRS-18 DRS-19)

In DRS-18:

     b) (isa dog3348 Dog)
     c) (feelsEmotion dog3348 (MediumToVeryHighAmountFn Anger))
     d) (isa people3409 Person)

In DRS-19:

     e) (performedBy attack3358 dog3348)
     f) (isa attack3358 AttackOnObject)
```

```
g) (intendedAttackTargets attack3358 people3409)
```

This interpretation means “if a dog is angry, and there is a person, that dog will attack that person.” Notice that if fact *c* is somehow omitted from the case, however, then the meaning becomes “if there is a dog and a person, the dog will attack that person.” That is not a conclusion that follows from the discourse interpretation. It is not always the case that failure to include a fact from a constituent DRS results in interpretation-inconsistency. For example, fact *g* could be omitted, and what remains is still consistent with the discourse interpretation. However, reasoning about which facts within a DRS can be removed without changing the meaning is outside the scope of a case construction algorithm. In the algorithms described in this thesis, any time part of a DRS is included in a case, the entire DRS is included.

I introduce the term interpretation-consistent, rather than just using *true*, because there are many reasons other than case construction that the case might be untrue. One major reason is that the interpretation might have errors introduced during parsing or semantic interpretation. Dealing with errors introduced in this way is outside the scope of the case construction methods described here, although this is part of the broader task of knowledge integration. Additionally, the source text might be erroneous, out of date or intentionally misleading. It might be fictional, or be describing a hypothetical without identifying individual statements as hypothetical. Reasoning about the nature of the source text in those terms is similarly outside the scope of the case construction processes. My method for keeping constructed cases interpretation-consistent involves treating certain discourse representation structures as atomic.

4.3.3 Amenability to Analogy

Another desirable feature for constructed cases is that they are amenable to analogical processing, including comparison and retrieval. As described in Section 2.4, analogy undergirds many cognitive processes in humans, and has been used for a variety of computational tasks. Given this, a case construction strategy that builds cases that allow analogy to maximize its potential is preferable to ones that do not. As noted above, the model of analogy used for the work described here is structure mapping theory (Gentner, 1983; see Section 2.4.1). This model of analogy – and, most relevant for our purposes, SME (Section 2.4.2) – prioritizes structural matches, and is able to draw more (and more interesting) inferences from representations that are more hierarchical and better-connected. Producing cases that are amenable to analogical reasoning allows for more effective analogical comparison and retrieval. This can be supported by using case construction strategies and representation schemes that favor facts that mention the same entities and higher-level facts that connect multiple other facts together. The Fact-based Segmentation construction method introduced in Section 4.5.4 is designed to attempt to capture these sorts of relationships. With our example seed, a good case would be one that allows for comparisons to the operation of a rainwater collection system, as the source text suggests.

One special scenario for building structured cases from discourse interpretations is when the text uses an explicit analogy. We describe a method for extracting analogy bases and targets from text, along with relationships between their entities, in Chapter 5. A case construction method that produces cases amenable to analogy can serve as a starting place for this functionality.

4.3.4 Size Tradeoffs

Different case construction strategies result in cases of different average sizes. Adding a larger number of facts to a case increases the odds that the algorithm will capture everything important, but it comes at the cost of including more facts that may not be relevant.

There are clear costs associated with not including enough facts. A case that is missing important facts is an incomplete description of a situation, which impacts a system's ability to reason about that situation. Moreover, critical missing facts can reduce the value of analogical comparison. For example, consider a case built around our example seed, the term "solar heating system" in the sentence "At the beginning of the day, the solar heating system is semifull." Suppose that case is compared to a case built around a seed representing the solar heating system later in the day. There is a difference in how much heat the system contains at different points in time. This difference is an important one. To detect that as a difference, the cases need to actually contain information about how much heat the system contains. If they do not, that difference cannot be detected. A case construction method that added facts too conservatively might miss the facts that represent the amount of heat the system contains. If this happens, the comparison system would not report that difference.

Including some extra facts is not fatal. SME is capable pulling out matching structured representations from pairs of cases even if there is a considerable quantity of extra facts (Ferguson, 1994). Nevertheless, if a case contains more irrelevant facts, is is more likely it is that noise will compromise an analogical comparison. Additionally, a case with a large number of facts is more difficult for a human working with the system to quickly understand. Depending on the use case, this last drawback might be a major design constraint or it might be largely irrelevant.

In our error analysis for Experiment 3, we evaluate the tradeoffs between including more facts and including fewer.

4.4 Approach

The case construction algorithms described here work by exploiting naturally-occurring boundaries in text, such as sentence and paragraph breaks. Two also exploit the connectivity properties of knowledge learned by reading. The most important resource available is the discourse interpretation itself. For a particular entity, facts that mention that entity are likely to be relevant to a description of that entity than facts that do not mention that entity. Similarly, two entities that are mentioned in the same fact are probably related to each other. For example, `cat15` and `eat27` in `(doneBy eat27 cat15)`, which indicates that the `eat27` was performed by the `cat15`, are likely related, as that fact mentions both of them. If we are looking for information about the eating event `eat27`, knowing that it was done by `cat15` is likely to be relevant. The dynamic case construction algorithm described in Mostek, Forbus, and Meverden (2000) takes advantage of this. It begins with a *seed* entity. It then adds to the case any facts that mention that entity, subject to filtering based on predicates or other factors. It proceeds by then adding any facts that mention entities mentioned in those facts, and so on. That strategy forms the foundation for the *Fact-based Segmentation* approach, described below.

The discourse interpretations produced by EA NLU use DRT (Section 2.2.2). This means that the facts are contextualized. Some are directly in the discourse interpretation case. Others are in constituent DRS cases. Facts can also be asserted in multiple contexts. For example, a fact might be in both the discourse interpretation one of its constituent DRSs. When this occurs, the system treats the instances as separate facts, becasue each being true means something different semantically. Having conditional facts necessitates taking steps to ensure interpretation-consistency (Section 4.3.2).

The methods can get the sentence and paragraph boundaries from the source text itself. These directly or indirectly guide for all four case construction algorithms described in this chapter. In general, a single sentence is generally about a single topic, and if one fact derived from a sentence is relevant, it is likely that other facts derived from that sentence are relevant as well. Paragraphs are similar, although the effect is weaker.

4.5 Case Construction Algorithms

Described here are two baseline methods and two connection-based methods. I will continue to use our example seed, the term “solar heating system” in the sentence “At the start of each day, the solar heating system is semifull.”

4.5.1 Local Sentence Interpretation

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 generally about a single thing. Therefore, it is likely that a 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. This 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 be added to the case, and only those facts are added to the case. This is six facts in total.

4.5.2 Local Paragraph Interpretation

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 that the seed appears in. It uses facts from the discourse interpretation case, rather than the individual sentence cases, so that it can take advantage of coreference resolution, but otherwise operates like local sentence interpretation. The local paragraph interpretation method 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 specifically in Experiment 3, below. 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 and covers a broader range of topics, as it contains the facts built from all 15 sentences in the paragraph. This is 178 facts in total.

4.5.3 Sentence-Based Segmentation

Sentence-based segmentation (SBS) is described in Figure 7 . It creates a case by adding all of the facts derived from sentences where the seed is mentioned. This makes intuitive sense. 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. Because SBS adds entire sentences at once, including their constituent DRSs, interpretation-inconsistency is avoided.

Figure 7. A pseudocode representation of how Sentence-based Segmentation works.

```
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}U{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.
```

The system pulls facts from the discourse interpretation rather than just using the union of the sentence interpretations. This is done because the discourse interpretation unifies coreferent entities into a single discourse variable. For example, if coreference determines that two mentions of “the solar heating system” refer to the same thing, they need to have the same symbol in any case that describes the situation. When the discourse interpretation is constructed from the sentence interpretations, coreferent entities are all replaced by the same symbol, providing a more integrated description. For our example seed, the case includes information from all three sentences in the discourse that mention the solar heating system – 28 facts in total. Those facts are shown in Table 15, in Appendix B.

Table 6. A subset of the facts that were added to the case using the fact-based segmentation method. The method added 63 facts in total, across 6 DRSs. Representations 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)
```

The method is reliant on effective coreference (Section 2.2.3). That is the system's primary way of recognizing that the same entity is being used in multiple sentences[4]. Additionally, topics that are mentioned once and then elaborated on without further direct reference to the topic itself can result in very limited cases. For example, consider the sentences, "The dolphin often has one calf. The calf is weaned after one year." If "dolphin" in the first sentence is the seed, the second sentence will not be included, as it does not mention the dolphin, even though it is likely relevant. On the positive side, the method's relative conservatism in including facts means that information that is included in the case is likely to generally be relevant to the topic.

4.5.4 Fact-Based Segmentation

The second connection-based case construction method I developed is *fact-based segmentation* (FBS). It operates as described in Figure 8. This method is inspired by the method described in Mostek, Forbus and Meverden (2000). However, it has been adapted to use the interpretations produced by the language system. The big-picture idea behind FBS is that facts from the discourse interpretation are

[4] The system also recognizes when a properly named entity, such as "Canada" or "Bill Gates," is mentioned multiple times, and assumes that those refer to the same thing.

added to the case if they share an entity, collection, or DRS with a fact already in the case. This process happens iteratively until a maximum *depth* is reached.

Like the other methods, fact-based segmentation starts with a seed entity. This entity is passed to the `gatherFactsForEntity` procedure. Facts are added to the case at the same depth if they are isa facts that mention that entity. They are added to the case at the next depth down if they are either isa facts that do not mention the entity, but use the same collection as one of the entity's isa facts or non-isa facts that mention that entity. If any of these facts mention a constituent DRS or are contained in one, the facts in that DRS are added at the same depth. This is done by the procedure `getDRSFacts`. From there, the system goes through each entity that is mentioned in a fact that added to the case that round. Each of these is extended from, using `gatherFactsFromEntity`, just as the seed was extended from. Entities at the maximum depth are not extended. Entities that have already been extended from are not extended, which prevents loops.

Figure 8: A pseudocode representation of how Fact-based Segmentation works.

```
Procedure FBS.
Inputs: s, a seed entity; depthMax, 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, depthMax)
 Return c.

Procedure gatherFactsForEntity.
Inputs: eCur, the current extension entity; depthCur, the current depth;
depthMax, 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 the same
                      collection as fi}:
   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 = cExtSameDepthUgetDRSFacts(cExtSameDepth)
 Let cExtNextDepth = cExtNextDepthUgetDRSFacts(cExtNextDepth)
 Let cExtend = cExtNextDepthUcExtSameDepth
 For each fc in cExtSameDepth:
  For each eNext in {entities mentioned in fc}:
   Unless member(eNext, eExtended):
    Let eExtended = adjoin(eExtended, eNext)
    Let cExtend = cExtendUgatherFactsForEntity(eNext, depthCur, depthMax)
 Unless depthCur = depthMax:
  For each fc in cExtNextDepth:
   For each eNext in {entities mentioned in fc}:
    Unless member(eNext, eExtended):
     Let eExtended = adjoin(eExtended, eNext)
     Let cExtend = cExtendUgatherFactsForEntity(eNext, depthCur + 1,
                                                depthMax)
 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.
```

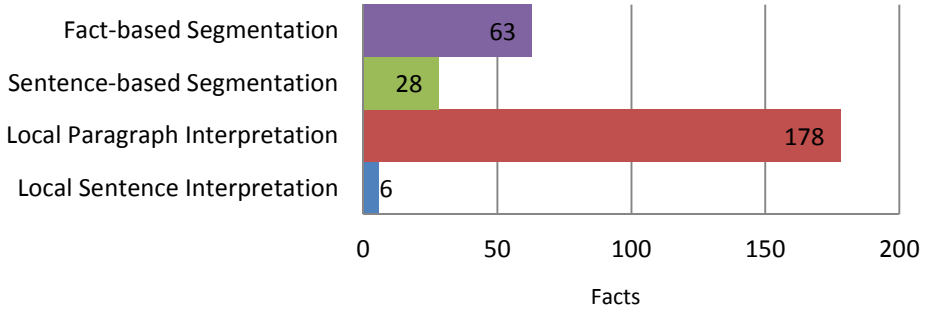
Working through an example of the fact-based segmentation method in action can be useful. Recall our example seed, the term "solar heating system" in the sentence "At the start of each day, the solar heating system is semifull." The method begins by selecting the discourse variable derived from that seed, `solar-heating-system01`. When an entity is mentioned in the case, facts that mention that entity are added to the case. This means that the facts `(fullnessOfContainer solar-heating-system01 PartiallyFull)` and `(isa solar-heating-system01 SolarHeatingSystem)` are added to the case, in the contexts they appear in. The first appears in the top-level DRS, `Discourse-DRS-01`. The second appears in several constituent DRSs, including `DRS-08`. When an isa fact collection is added the case, other isa facts in the paragraph that use the same collection are added to the case. For example, if there were other instances of `SolarHeatingSystem` in the paragraph, such as `(isa solar-heating-system-35 SolarHeatingSystem)`, the facts indicating this would be added. This lets the system include multiple instances of the same type of thing in the case 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, facts that mention that constituent DRS and facts inside that constituent DRS are added to the case. Here, when `DRS-08` is added to the case, `(not DRS-08)` is added to the case. When a fact contained in a constituent DRS is in the case, the rest of that constituent DRS is added to the case. 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. Some of the facts that were added to the case for our example seed are displayed in Table 6; a full list is available in the appendix, in Appendix D.

All of the facts in a given constituent DRS are added to the case at the same depth. Adding a complete constituent DRS is crucial for interpretation-consistency. For example, consider the sentence "When the

temperature of the heat storage equals the temperature of the solar panel, the 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.

I 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, it produced had 115 facts. Table 6 shows some of the facts that were added to the case for our example seed.

Figure 9. The number of facts in a case constructed around the example seed for each of the four case construction methods.



Error! Reference source not found. shows the number of facts in the case produced by each of the four methods, working over the example seed. Table 7 summarizes the theoretical strengths and weaknesses of the methods.

Table 7 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 compute time-intensive to produce

4.6 Evaluation

Given the different potential strengths and weaknesses of the four methods, I designed an experiment to evaluate them. The hypothesis tested with our experiment is 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.

Direct evaluation of case construction is difficult. This is because there is no gold standard for what constitutes a correct case. Not only is it task-dependent, but there may be variation in human judgment regarding what the best case configuration is, even for a specific task. My evaluation metric uses measures that are less direct, but more objective. They are focused on the sorts of tasks that such a system will actually be used for.

I 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, Mahesh & Goel, 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, I created two corpora from pre-existing source texts, described next.

Structural alignment is useful for identifying alignable differences (Gentner & Gunn, 2001). These are differences between cases that are conceptually related to their commonalities. Cases that describe the same system in different conditions or at different points of operation naturally have a lot of commonalities, so finding places where they differ is a natural task for structure mapping.

This task is a useful evaluation for several reasons. First, it is realistic, providing a more objective basis for evaluation. Second, it rewards case construction methods for including useful facts. It also rewards methods that do not include large amounts of irrelevant information. As the results below demonstrate, however, including more information generally led to improved performance. While there is no perfectly objective gold standard for what constitutes the correct conclusions to draw about similarities and differences between cases, the task is more specific and has more objective desired results than a simple “produce a good case” task. The differences and similarities between the solar heating system at the start of the day and the solar heating system at the end of the day are relatively well-defined. The same is true for the similarities and differences between the solar heating system at the start of the day and a rainwater collection system to which it is compared. This means that the evaluation criteria are less arbitrary.

4.6.1 Corpora

The first source text was chapter 16 of *Sun Up to Sun Down* (SUSD16; Buckley, 1979), a book about solar energy and solar heating that makes extensive use of analogies. I selected chapter 16 because it contains descriptions of multiple complex systems. This includes an extended analogy explaining a solar heating system in terms of a rainwater collection system. The simplified version of the chapter consists of 80 sentences in 11 paragraphs. A paragraph from the simplified version appears in Table 1. The interpretation process produced 733 fact statements 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. I chose this article because it differed in both style and subject matter from the other source text. After simplification, the article was 8 paragraphs and 88 sentences long. The interpretation process produced 751 fact statements across 150 DRSs. That the texts produced a very similar number of fact statements is not unexpected, given that they are of similar length. Both texts were simplified syntactically, as described in Section 2.2.4. The SUSD16 text was simplified prior to the design of the algorithms, while the Dolphin/Porpoise text was simplified later.

4.6.2 Baselines

In this work, I use the Local Sentence Interpretation and the Local Paragraph Interpretation as baselines. I considered several other potential baselines for the experiment. A truly random baseline – that is, one that contains a random set of facts from the source – is very unlikely to perform well, and would be trivial to beat. Another potential baseline is to use the entire interpretation as a case. This has clear disadvantages, especially for very large sources. While it has the unique advantage that it is guaranteed to contain any fact you might want (provided that that fact is mentioned at all), entire discourse cases

are impractical for many tasks. (This is what motivates this type of case construction in the first place.) The more practical units of information are the sentence and the paragraph.

4.6.3 Reading and Disambiguation

Reading was performed by EA NLU (Section 2.2), embedded in the Companion Cognitive Architecture (Section 2.3), using the standard pipeline as described in those sections.

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. EA TMS (Section 2.2.7) makes TMS assertions that ensure that the system does not select incompatible choices.

Given that these heuristics do not involve any learned statistics and are task independent, 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 chapter. Errors generated by the heuristics tend to be systematic, which helps prevent 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. Note that this set of heuristics

does not do well on specifically challenging WSD tasks; when used on the Stars corpus (Section 3.8.1; Appendix 0), for example, it performs no better than chance.

4.6.4 Comparison and Contrast with Topic Pairs

Comparisons between cases built around arbitrary seeds are not very illuminating. For example, consider cases built around these two seeds:

a) The flow of water through a pipe in a solar heating system

b) Earth's atmosphere

A comparison between those cases might produce a few matches, but the similarities and differences between those two things are unlikely to be of much interest. For this reason, we use comparisons between *topic pairs*. A topic pair is a set of two seeds where we might expect the cases generated around the seeds to have interesting similarities and differences. They might be descriptions of a system at two different points of operation, two concepts that are similar or easily confused, or the base and target of an explicit analogy. For our evaluation, the topic pairs were hand-selected before the system read the source texts. The topic pairs were things that were natural to compare, such as the solar heating system at different points of operation, or parallel aspects of the anatomy or behaviors of dolphins and porpoises.

4.6.5 Experiment 3 Setup

In total, the experiment used 24 topic pairs, meaning that 24 comparisons were made. 11 of these were from the SUSD text. 13 were 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. For example, for example, when comparing and contrasting a solar heating system and a rainwater collection system, it started with those two things

already in alignment. This allowed the local paragraph interpretation method to operate over cases built from seeds in the same paragraph, as otherwise it would simply map every entity and fact to itself.

Two primary evaluation metrics are used. 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. 64 of these came from the SUSD text, and 54 came from the Dolphin/Porpoise text. We discarded 7 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. 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 do 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 goal facts 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. This penalizes including 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 with. A secondary benefit of smaller cases is that they are easier for humans to read.

I did consider alternate evaluation metrics. Simply looking for the presence of important facts in cases would not be effective; 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.

4.6.6 Experiment 3 Results

I 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. Each column presents results for one of the

methods described earlier. The results from the two source texts are combined, as they were generally comparable for all methods.

- Local Sentence Interpretation (LSI)
- Local Paragraph Interpretation (LPI)
- Sentence-Based Segmentation (SBS)
- Fact-Based Segmentation (FBS)

Table 8. Experiment 3 Results

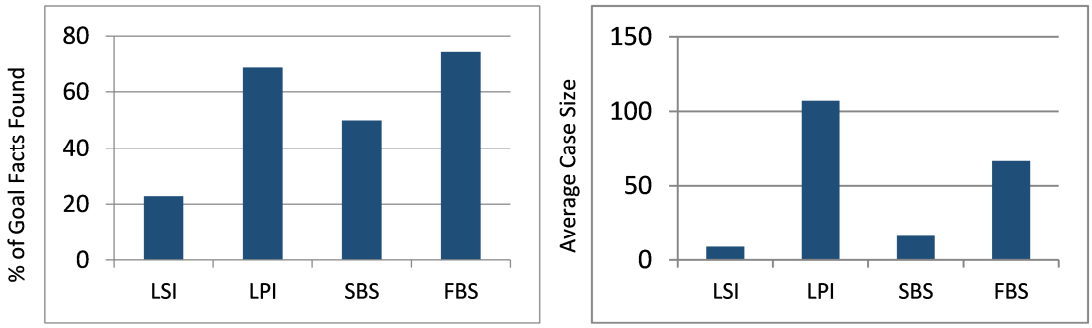
Method	LSI	LPI	SBS	FBS
Goal Facts Found	27	81	59	88
Goal Facts Found (%)	22.9	68.6	50	74.5
Generation Efficiency (%)	8.4	2.5	8.3	3.7
Unique Correct	0	8	3	8
Avg. Case Size	8.9	107.2	16.1	66.8
Average CIs	8.4	49.9	19.5	52.0
Average Corrs.	3.4	69.7	7.0	35.3

Goal Facts Found is the number of the goal facts the method produced. *Goal Facts Found* (%) is the percent of the possible goal facts that the method produced

Unique Correct is the number of goal facts where the method was the only one of the four that correctly produced it. It is a measure of each method's ability to produce interesting conclusions that the others did not. In some situations, including most instances when LPI was the only method to produce a goal fact, it is because the other methods 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 other methods did include the relevant facts, but failed to produce the proper correspondence or candidate inferences. This occurs when extra facts in the case interfered with the match.

Avg. Case Size is the average size of the bases and targets produced by the system when constructing cases based on the seeds. There is no functional difference between bases and targets for this task, as candidate inferences are produced in both directions. Because of this, their sizes are simply averaged together in the table. *Average CIs* is the average 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.

Figure 10. Performance and the average case size of each method.



The accuracy of Fact-Based Segmentation and of Local Paragraph Interpretation 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. **Error! Reference source not found.** illustrates the performance and the average case size of each method.

Table 9. Experimental results on cases where the base and target seeds are in the same paragraph.

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

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 be disadvantaged. To test this, we looked at the results on only the comparisons made between entities in the same paragraph. In total, 65 goal facts came from comparisons of this type. The accuracy on just those goal facts is shown in Table 9. 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 is also still not significant.

4.6.7 Error Analysis

One challenge in evaluating the system's overall performance is that assigning blame for failures is not easy. I 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 by including the corresponding facts, but they do not align when the cases are compared via SME. Some types of errors can occlude other types. 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.

To evaluate the performance of the case construction methods in more detail, we performed a comprehensive error analysis. Table 10 summarizes the sources of error for each of the methods. Each goal fact missed by a method was assigned a source of error, which might vary from method to method. For example, it is possible for LSI to miss a similarity goal fact by failing to include the facts in the cases

to begin with, while FBS misses the same goal fact by including the corresponding facts, but failing to have them align when the cases are mapped with SME. It is possible for some types of errors to occlude other types. For example, if one of the case construction methods failed 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. Sources of error in experiment 3.

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

Facts Not In Case refers to errors where key facts were in the global interpretation, but not included by the case construction method. Unsurprisingly, the methods that build larger cases were much less likely to produce this type of error. This error accounts for the vast majority of Local Sentence Interpretation's errors. For that method, any goal fact that needs information from another sentence will fail as a result of this error. 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. These facts are in different sentences than the one that introduces the comparison between the anatomy of the two creatures. Because of this, the cases produced bys Local Sentence Interpretation lack those facts.

SME Representation Mismatch refers to errors where facts representing a similarity were present in the base and the target, but were sufficiently different representationally that SME did not match them together. 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 occurred was 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 where the mismatch can be traced to the disambiguation heuristics making an incorrect choice. As noted above, the heuristics used were not perfect; 13.4 percent of the semantic ambiguities were resolved incorrectly. However, such disambiguation errors often do not affect the SME matches. Only disambiguation errors leading to mismatches were coded as this type of error.

SME Alignment Mismatch refers to errors where 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 both 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 did the same with the porpoise's nose. As a result, the system drew no candidate inferences.

4.7 Discussion

The results suggest that Fact-Based Segmentation and Local Paragraph interpretation are the best of the four methods tested on the primary measure, partially falsifying the hypothesis that FBS would produce superior performance. FBS had the advantage that it produced the best accuracy results while achieving significantly higher generation efficiency than LPI. This comes at the cost of a small amount of additional overhead during the case construction step, as producing cases with FBS requires extra computation. In the evaluation, LPI had very little trouble with Facts Not In Case errors. This is because the goal facts were all local. In a comparison where facts are spread more evenly across a source text, it would be less effective.

Local Sentence Interpretation fared the worst. This is not surprising. While it produced relatively little noise, as indicated by its high generation efficiency, 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. There are instances where 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 rarer.

That Fact-based Segmentation produces smaller cases than Local Paragraph Interpretation gives it several advantages. First, smaller cases are easier for a human to read and understand. Even with the aid of visualization tools or pidginization functions, large cases can be difficult for a human reader to make sense of, 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 – it operates in $O(n^2 \log(n))$ time – but it still operates faster over smaller cases.

5 Analogical Dialogue Acts

Instructional texts and other written explanations often use analogy to convey new concepts and systems of related ideas to learners. Some of the best-known ways of explaining certain concepts is via analogy. For example, the greenhouse effect takes its name from the analogy most often used to explain it, and the Rutherford-Bohr model of the atom is often explained via analogy to the solar system. Heat and electricity are sometimes explained by comparing them to water. Analogy is useful for human learners (Gentner & Smith, 2013). The analogy in Figure 11, for example, is from a book on solar heating, designed to help the reader understand heat by analogy to water. The behavior of water is likely familiar to the book's middle school audience. It is the base of the analogy, the side that is already known. The behavior of heat is what is being learned about. It is the target of the analogy. We will use this analogy as an example throughout this section.

A learning by reading system that simply applies ordinary reading processes to such analogies when they appear in text will miss the point of that text. At best, it might end up with irrelevant information about the base of the analogy in the interpretation. At worst, the analogy introduces noise or even confusion into the interpretation that interferes with later reasoning. This work combines Gentner's (1983) structure-mapping theory with ideas from dialogue act theory (e.g. Traum, 2000) to describe a catalog of analogical dialogue acts (ADAs) which capture the functional roles that discourse elements play in instructional analogies.

Our goal is ultimately to provide human-like capabilities for understanding instructional analogies, in order to understand human learning better, and to improve learning by reading. We view these goals as closely aligned, since, after all, natural language texts are intended for human readers. We outline criteria for identifying ADAs in text, but we mainly focus on what operations they imply for discourse

processing. We provide evidence that this model captures important aspects of understanding instructional analogies using a simulation that uses knowledge gleaned from reading instructional analogies to answer questions.

Specifically, analogical dialogue acts are used to:

- Build structured base and target cases from a textual analogy
- Provide constraints on which elements of the base and the target must (or must not) correspond

This allows an LBR system to process analogies. The candidate inferences gleaned though this process provide additional information. As a result, the system demonstrates improved understanding, as measured by question answering performance. The work detailed in this chapter was first described in Barbella and Forbus (2011).

Figure 11. A paragraph read by the system, with the ADA labels applied to each of its sentences.

A house is like a bucket with holes in the bottom. *Introduce Comparison*
Insulating the house is like shrinking the holes in the bucket. *Introduce Correspondence*
If the house is insulated, a furnace uses less heat to warm the house. *Extend Target*
Similarly, if we shrink the holes, a faucet uses less water to fill the bucket. *Introduce Candidate Inference, Extend Base*
The furnace adds an amount of heat. The amount of the heat added matches the amount of the heat that leaks out. *Extend Target*
Similarly, the faucet adds an amount of water. *Introduce Candidate Inference, Extend Base*
The amount of the water added matches the amount of the water that leaks out. *Extend Base*
If the amount of the heat that is leaking is decreased, the amount of the heat that the furnace adds is decreased. *Extend Target*
If we shrink the holes in the bucket, the amount of the water that is leaking is decreased. Because of this decrease, less water is needed from the faucet. *Extend Base*

5.1 Dialogue Act Theory

Dialogue act theories (Allen & Perrault, 1980) are concerned with the roles utterances play in discourse and the effects they have on the world or on understanding. An utterance identified as a *Requesting Information*, for example, might take the syntactic form of a question that makes the information requested explicit, as in “What time is it?”, but does not have to do so. The surface manifestation might

instead be a statement, or an indirect question, as in “Do you have the time?” Dialogue acts are part of all linguistic communication, not just verbal communication.

5.2 ADAs Defined

We claim that there exists a set of *analogical dialogue acts* that are used in communicating analogies. Like other dialogue acts, they have criteria by which they can be recognized and a set of implied commitments and obligations for the dialogue participants. There are a wide variety of dialogue act models, but all include some variation of *Inform* (Traum, 2000), which indicates the intent to describe the state of the world. Analogical dialogue acts can be viewed as specializations of *Inform*. Each specifically deals with the role that an utterance plays in building up or extending an analogy. The ADAs are motivated by the analogy ontology described in Forbus, Mostek, and Ferguson (2002). This ontology is derived from structure-mapping theory (Gentner, 1983; Section 2.4.1). Each of the ADAs correspond to an operation that informs or controls the structure mapping process. This includes both building the base and the target and constraining the mapping process. The ADAs are as follows:

- **Introduce Comparison**: Identifies the base and target to be compared. For example, in “A house is like a bucket with holes in the bottom,” the base is a situation where water leaks from a bucket, and the target is a situation where heat leaks from a house. While this signals that an analogy is being used, it is not necessarily the first statement in an analogy. In Figure 11, the comparison is introduced explicitly in a single sentence, but more complex cases might involve combining information across multiple sentences - parallel sentence structure in subsequent sentences, for example. Determining which of the domains is the base and which is the target requires a non-local assessment about what the topic of the text is. (This particular example is drawn from a book on solar energy, and the rest of the chapter makes clear that heat is the

domain being taught.) Since candidate inferences can be constructed in both directions, an incorrect assessment is not fatal. Processing an Introduce Comparison act requires producing appropriate cases for the base and target. The target is constrained by what has already been introduced in the text. The base, unless it has been used before in the same text and is being used in a consistent manner, must be constructed from the reader's knowledge. Often the most relevant elements of the base and target are highlighted explicitly, but frequently information from the reader's memory is required as well. Because it places two entities in correspondence, an Introduce Comparison act is also an Introduce Correspondence act.

- **Introduce Correspondence**: These acts provide clues as to the author's intended mapping. For example, "Insulating the house is like shrinking the holes in the bucket" indicates that those two entities correspond. Sometimes Introduce Correspondence acts are expressed as identity statements – for example, "The glass is the atmosphere" in the standard greenhouse/atmosphere analogy. Sometimes these acts are signaled by pairs of sentences, one expressing a fact about the base followed immediately by one about the target, with similar syntax. These latter constructions are not currently detected by the system.

- **Block Correspondence**: This is an utterance that establishes that some element of the base and some element of the target should not be in alignment in a mapping between the base and target domains. This is typically a pair that might otherwise be tempting for a reader to align. ("The ozone layer's hole is not like an ordinary hole in the roof, however.") Block Correspondence statements are very rare in written text, although they appear in resources like Harrison & Coll (2007), which assist educators with interactive instruction.

- **Extend Base**, **Extend Target**: These acts add information to the base or target of a comparison, respectively. Such acts are identified by relationships and/or entities being mentioned in the same statement as an entity in the base or target, but which is not a statement about correspondences or candidate inferences. For example, "If we shrink the holes in the bucket, the amount of the water that is leaking is decreased" extends the base, and "The amount of the heat added matches the amount of the heat that leaks out" extends the target.

- **Introduce Candidate Inference**: This introduces a candidate inference - information about the target domain - which the reader is intended to have drawn from the analogy. ("Similarly, if we shrink the holes, a faucet uses less water to fill the bucket.")

- **Block Candidate Inference**: This is an utterance that establishes that a candidate inference that a reader might be tempted to draw should not be drawn. ("Unlike solar radiation, heat flow that occurs by conduction is unaffected by color.") Block Candidate Inference statements are very rare in written text, although like Block Correspondence they appear in resources focused on interactive instruction.

5.2.1 Strategies for Identifying Analogical Dialogue Acts

The analogical dialogue act status of any statement is contextual. This is most obviously true for Extending Base/Target statements, which nearly always look like any non-analogy statement, and are distinguished only because they relate to an entity being discussed in another analogical statement, such as an Introduce Comparison. However, it is also true for other ADA types. Introduce Comparison, for example, can be identified by statements that take forms like "An X is like a Y," but not all Introduce Comparison statements look like that, and not all statements like that are introducing a correspondence. For example, that sentence form is also used to describe literal similarities, as in "A tangerine is like an

orange." A comparison can also be introduced in sentences that do not have any particular format that suggests analogy when considered in isolation. For example, consider the sentence pair "Water leaks from a bucket when the water level gets too high. Heat leaks from a solar collector when the temperature gets too high." These sentences use parallel structure to imply a comparison should be made.

The system currently does not attempt to skim away false positives; every similarity is treated as a potential analogy. Detecting false positives is an important part of a more general area of future work where the system more carefully evaluates the facts it learns via analogy, as false analogy detection positives are one potential source of incorrect inferences.

5.2.2 Incorporating World Knowledge

In understanding instructional analogies, a human learner is expected to draw upon their existing world knowledge, particularly about the base domain. It is typical for many of the most relevant parts of the base domain to be explained in the text. For example, a text about the greenhouse effect will usually explain how a greenhouse works. Despite this, many common-sense facts about the base will generally not be explicitly asserted. For example, most descriptions of the greenhouse effect do not explicitly state that the roof of the greenhouse covers the greenhouse. That is an important part of the structure of the analogy – the roof and the greenhouse have a relationship similar to that of the planet and the atmosphere. However, the relationship between a building and its roof is one that the reader is expected to bring with them. For this reason, fully processing many instructional analogies requires retrieving and incorporating knowledge from outside the text.

In some situations, whole cases representing a prior experience may be retrieved from memory. In other situations, cases may be constructed dynamically from one's general knowledge of the world. This

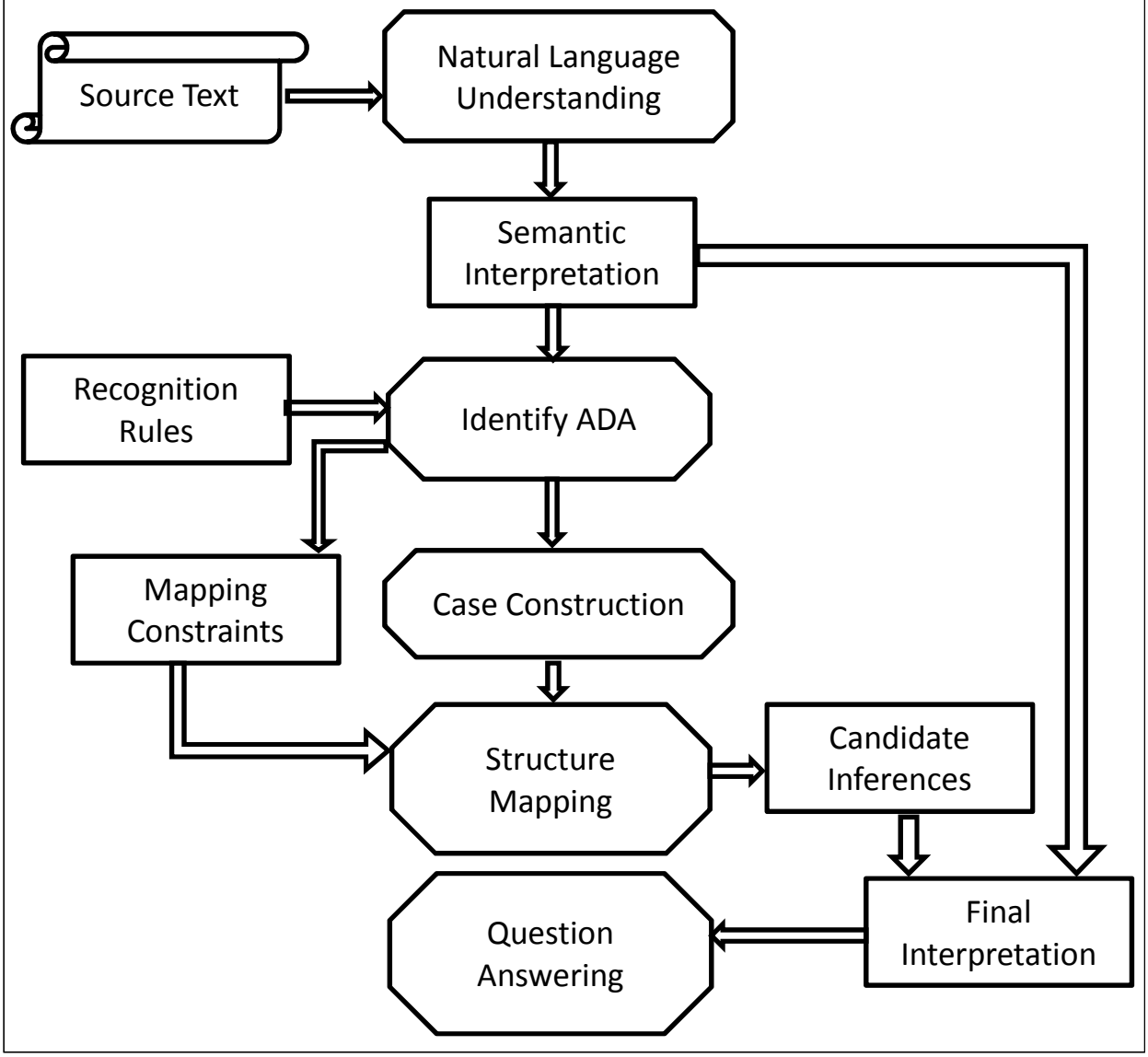
latter approach is what the system uses in the work described here. Section 5.3.4 details the full strategy the system uses.

One challenge in this area is accommodating representational differences. Operating with a language system that uses representations from the Cyc knowledge base is a huge advantage here, as it ensures some uniformity of representation. However, Cyc is sufficiently expressive and complex that there are often multiple reasonable ways to represent something. This is especially true when starting from natural language.

5.3 Method

The overall pipeline used by the system to process text that may contain analogy is shown in Figure 12. The source text is processed using EA NLU, as described in Section 2.2. In this section, we detail the specific strategies used by the implementation of analogical dialogue act processing that the evaluation uses. The system begins by parsing and disambiguating text as normal. Coreference resolution occurs as described in Section 2.2.3. Analogical dialogue act processing is an additional discourse interpretation process that occurs after coreference is complete. The system attempts to prove the analogical dialogue act status of each statement in the discourse.

Figure 12: A chart of the pipeline used in analogical dialogue act processing.



5.3.1 Detecting Analogical Dialogue Acts

Detecting different types of analogical dialogue acts is not trivial. In the current implementation of the system, a limited set of heuristics is used to detect each act. Three types of ADA, Introduce Comparison, Introduce Correspondence, and Block Correspondence, are detected using primarily local features.

- **Introduce Comparison, Introduce Correspondence**: These acts are detected by the presence of `similarTo` facts in the discourse interpretation. `similarTo` is a relation that says that two things are similar. The first statement that generates a `similarTo` fact is an Introduce Comparison. Any subsequent facts that do so are Introduce Correspondence. The first argument to `similarTo` is taken to be an element of the target, and the second is taken to be an element of the base. For example, the statement "A house is like a bucket with holes in the bottom," from our example, produces a fact of the form `(similarTo house2294 bucket2372).`

- **Block Correspondence**: These acts are detected by looking for `similarTo` fact in negated sub-DRSs. Any statement that produces a negated `similarTo` fact is Block Correspondence.

The other types of ADA cannot be locally detected. The search for these types is triggered by the appearance of Introduce Comparison. All of these types rely on knowing that elements they mention are in the base or in the target already. For this reason, they are searched for iteratively.

- **Introduce Candidate Inference**: These are currently detected by looking for sentences that begin with the "Similarly,..." construction and that mention an element already known to be in the base or in the target. Specifically, the detection rules check the interpretation for facts of the form that that construction produces.

- **Block Candidate Inference**: These are currently detected by looking for sentences with the "However,…" construction and that mention an element already known to be in the base or in the target. As with Introduce Candidate Inference, the detection rules check the interpretation for facts of the form that that construction produces.

- **Extend Base**, **Extend Target**: These acts are detected according to two criteria. In order to be included as an Extend Base dialogue act, a statement must both:
    - Mention an entity already known to be in the base. This may be an entity that is part of another Extend Base statement, or it may be an entity known to be in the base for other reasons.
    - NOT be an Introducing Correspondence or Introducing Candidate Inference dialogue act or an Extend act of the other type. Extend Base and Extend Target are the last type of analogical dialogue act considered for each statement. A statement must be found to be none of the others before it will be classified as Extend Base or Extend Target.

Further development of the set of detection heuristics is an important part of my future work with ADAs. These heuristics are both too narrow to catch all common formulations of analogy and broad enough to produce false positives. There is also a certain degree of brittleness in the detection heuristics. For example, the heuristics assume that in all statements of the form "The [base-element] is like the [target-element]," the base and the target remain in a consistent order. No harm is done if the base and target are consistently reversed, but if the base is sometimes first and is sometimes second, the heuristics will fail.

It is unlikely that any simple set of heuristics can account for the breadth of possible ways of expressing analogy. Learning to better recognize occurrences of analogy is an important avenue for future work. This includes being able to identify and separate multiple analogies within a single discourse.

5.3.2 Using ADAs

Each type of analogical dialogue act creates certain obligations for the system. These obligations can affect what is included in the base or the target. They can also affect the classification of other statements in the discourse. Finally, they can be used to place additional constraints on the analogy.

- **Introduce Comparison**: When an Introduce Correspondence act is detected, the base and target are checked to see if they already contain the entities or relationships mentioned. If they do not, then the descriptions are extended to include them. When the case construction process builds the base and the target, it also includes a *required correspondence* relation between the two entities (Section 2.4.2). The system begins its search for other analogical dialogue acts in the discourse.

- **Introduce Correspondence**: These acts are handled the same way that Introduce Comparison acts are, except they do not trigger the search for other types of ADA. This is because Introduce Comparison has already done that.

- **Extend Base**, **Extend Target**: The base or the target (as appropriate) is checked to see if it already contains the relationships introduced in the statement. If it does not, it is extended to include those relationships.

- **Block Correspondence**: The base and target are checked to see if they already contain the entities or relationships mentioned. If they do not, then the descriptions are extended to include them. When the case construction process builds the base and the target, it also includes an *excluded correspondence* relation between the two entities (Section 2.4.2). The system begins its search for other analogical dialogue acts in the discourse.

- **Introduce Candidate Inference, Block Candidate Inference**: These are handled as Extend Base or Extend Target, as appropriate. More sophisticated handling of the entailments created by these acts is an area of future work.

5.3.3 Case Construction and Constraining Analogy

The analogical dialogue act classifications are used to construct a base case and a target case. After a statement is classified as an analogical dialogue act, facts that are derived from that statement are added to the base or the target, as appropriate. From our example, the base would include all of the facts in the statements:

- Similarly, if we shrink the holes, a faucet uses less water to fill the bucket.
- Similarly, the faucet adds an amount of water.
- The amount of the water added matches the amount of the water that leaks out.
- If we shrink the holes in the bucket, the amount of the water that is leaking is decreased.
- Because of this decrease, less water is needed from the faucet

These statements are all classified as Extend Base statements. The base would also include select facts from the following statements:

- A house is like a bucket with holes in the bottom.
- Insulating a house is like reducing the size of the holes in the bucket.

These statements are classified as either Introduce Comparison or Introduce Correspondence. Only facts derived from the part of the statement describing the base are added to the base case. The target case is constructed in a similar fashion.

The Introduce Comparison, Introduce Correspondence, and Block Correspondence analogical dialogue acts all place additional constraints on the analogical match. For the first two, this constraint is a *required correspondence* between the entities that are directly compared to each other in the statement. For example, the statement "A house is like a bucket with holes in the bottom" is an Introduce Comparison statement. As part of the processing, a constraint of the form `(requiredCorrespondence bucket1324 house1132)` is produced. While the statement also mentions holes and the bottom of the bucket, no particular match obligations are produced for those entities. Block Correspondence acts are handled in a similar way, except that the system produces an `excludedCorrespondence` constraint instead.

Table 11 shows some of the entities put into correspondence as a result of reading and doing analogical dialogue act processing on the text from Figure 11. Two of the correspondences were the result of `requiredCorrespondence` statements that the system produced as part of analogical dialogue act processing.

Table 11: Example correspondences from the example in Figure 11, along with their match hypothesis score. Bolded rows represent correspondences that were required in the match because they were in Introduce Comparison or Introduce Correspondence statements.

Base	Target	Match Hypothesis Score
use93130	use92620	0.216
amount106103	amount93921	0.244
add106075	add93894	0.108
water106136	heat93953	0.14
faucet93123	furnace92615	0.108
leak106882	leak94694	0.14
add106287	add94099	0.108
water93210	heat92691	0.036
shrink92216	**insulate92115**	**0.036**
match106344	match94156	0.036
bucket91861	**house91787**	**0**

In our example, several useful candidate inferences are generated by the system. Table 12 shows some of these, along with glosses of their meaning. Individually, these candidate inferences – even the correct ones – are not useful on their own. They can be combined with other information, however, to allow the system to answer questions it could not otherwise answer.

Table 12: A selection of the candidate inferences produced by the system in our example case, shown in Figure 11. The inferences in regular type are factually correct; the one in italics are supported by the match, but not factually correct. The glosses are approximate English translations of the inferences. As produced, the inferences are contextualized. This context has been omitted here for space.

Inference	**Gloss**
(causes-Underspecified insulate92115 (skolem decrease119615))	"Insulation causes a decrease in the amount (of heat)."
(isa (skolem decrease119615) DecreaseEvent)	
(possessiveRelation amount93921 (skolem decrease119615))	
(isa furnace92615 Tap-PlumbingFixture)	*"The furnace is a faucet."*
(isa house91787 Bucket)	*"The house is a bucket."*

For example, consider the question "What reduces the loss of heat from the house?" This question cannot be answered directly from the text. The system knows that there is a loss of some amount of heat from the house, but it doesn't know anything about a decrease in that amount. Using the first three facts in Table 12 combined with this information, the system is able to answer the question.

Not all of the candidate inferences generated by the system are necessarily correct. As Table 12 shows, some are spurious. For example, one of the candidate inferences produced is `(isa furnace92615 Tap-PlumbingFixture)`. This was produced because the furnace and the faucet were placed into alignment. One of the features of the faucet is that it is a `Tap-PlumbingFixture`. The system hypothesized that this was a feature of the furnace as well.

5.3.4 Retrieving World Knowledge

As discussed earlier (Section 5.2.2), analogies in text are frequently incomplete. They often rely on structure that is assumed to be familiar to the reader, but which is not restated explicitly in the text. This poses a problem for any system that relies exclusively on the discourse interpretation to populate the base and the target.

The system makes use of *rule macro predicates* to augment the base and target cases after constructing them. Rule macro predicates are defined as part of the Cyc ontology. The rule macro predicates used by the system relate three things:

- Two collections or entities
- A predicate that relates those two arguments

The nature of the relationship is defined by the specific rule macro predicate. For example, one rule macro predicate is `relationAllAll`. In addition to a predicate, `relationAllAll` takes two collections. `(relationAllAll ?pred ?collection1 ?collection2)` means that for ALL members of `?collection1` and ALL members of `?collection2`, `(?pred ?member1 ?member2)` holds. For example, `(relationAllExists covers-Generic Greenhouse RoofOfAConstruction)` means that all greenhouses are covered by a roof (not necessarily the same roof, of course). The system can use this knowledge to expand a case in two ways:

- If the system knows that a greenhouse and a roof have been mentioned in a discourse, it can hypothesize that a `covers-Generic` relationship holds between them. (*Closed rule macro predicate elaboration.*)

- If a greenhouse has been mentioned, but no roof has been mentioned, the system can introduce a roof entity, and create a `covers-Generic` relation between the greenhouse and that roof. (*Open rule macro predicate elaboration.*)

This is extremely powerful for elaborating cases. If a text mentions "the greenhouse" and later mentions "the roof," the system can build a connection between the two like a human would, even if no explicit relationships are drawn between them in the text. As the experiment below demonstrates, it is also important for understanding many textual analogies. The system always prefers closed rule macro predicate elaboration to open rule macro predicate elaboration, and will not hypothesize the existence of a new entity if an appropriate one is available in the context already. The information in the rule macro predicates used by the system was sourced from Simple English Wikipedia articles on the topics in question.

Rule macro predicate elaboration is not infallible. It runs the risk of hypothesizing relationships that aren't true. For example, consider the statement "If we go up onto the roof, we should be able to see the greenhouse." In that sentence, it is very unlikely that the roof is the roof the greenhouse, so there is not likely a `covers-Generic` relationship between the roof and the greenhouse. Avoiding this type of error requires sophisticated general-purpose reasoning, and is outside the scope of this dissertation.

Another issue with rule macro predicate elaboration – particularly open rule macro predicate elaboration – is that it potentially introduces irrelevant information. For example, a rule macro predicate might specify that all rain falls from a cloud. While this is true, it may not be relevant to every scenario that involves rain. Fortunately, small amounts of irrelevant information do not typically interfere with matching.

The elaboration occurs as part of a discourse interpretation process (Section 2.2.1). This triggers a set of general-purpose reasoning queries that were designed for the sake of the work described here, but which have since been used elsewhere.

5.4 Evaluation

When evaluating the ADA handling strategies used by the system, it is important to distinguish what is actually tested by different evaluation strategies. Examination of how the system classifies different statements is a good evaluation metric for the ADA detection process, but does not actually demonstrate that the constructed cases are useful. For this reason, we turn to question-answering performance as a means of evaluating the system. The primary hypothesis tested by the experiment is that classifying analogical dialogue acts and using them to construct a match between the cases can be used to improve question answering performance. Specifically, the extra information gleaned through candidate inferences that result from the analogy match can be used to improve question answering. A secondary hypothesis is that, in situations where only a partial description can be built directly from the discourse interpretation, bringing in additional background knowledge will allow the system to further improve its question answering performance.

While looking at how the system classifies different utterances is not a complete evaluation metric on its own, it does serve as a sanity check. It is also important for distinguishing at which point in the process errors arise as well. Because complete evaluation of the system's overall performance requires the integration of several systems, there are several stages in the process where errors might arise.

The first place errors might occur is in language processing. Natural language understanding is far from a solved problem. Certain types of error in parsing, word sense disambiguation, and coreference can potentially interfere with identification of analogical dialogue acts. They can also potentially interfere

with the quality of the match. In order to factor out parsing issues as a source of error, the texts used as part of the corpus for the experiment were manually simplified into forms that the parser can handle, as described in Section 2.2.4.

Errors can also potentially be the result of representational differences. If the base and the target are described in different language, that can result in different representations in the discourse interpretation. This has the potential to interfere with the match, producing incorrect candidate inferences. As it is not possible to factor out this source of error without excessive tailoring, this is a potential source of error in the evaluation

5.4.1 Corpus

The corpus used for this work consists of paragraph-length chunks of text excerpted from various instructional materials, covering a variety of science domains. While there is no part of the algorithm that we expect to be domain-specific, this demonstrates a degree of generality. Because we are not primarily concerned with evaluating detection in these experiments, the corpus does not include things that look like they might be analogies, but are not, such as instances where literal similarity between two things is described. If it did, those would be detected as potential analogies, as the system does not currently attempt to detect false positives. The corpus was simplified as described in Section 2.2.4, with one additional restriction: Introduce Comparison and Introduce Correspondence statements were simplified into one of the forms supported by the detection system, in the instances where that was not already the case. Table 13 summarizes the corpus.

Table 13: Corpus Information. #B/#A = # sentences before/after conversion to QRG-CE

Example
Rubber Ball/Sound Echo
Gold mine/Collecting Solar Energy
Bucket of water/Hot brick
Water storage/Heat storage
Phase change materials
Faucet/Thermostat and furnace
Stopping a leak/Insulation
Rain/Sunlight
Stagnation depth/Stagnation temperature
Intensity of rain/sunlight
Power plant/Mitochondrion
Greenhouse/Atmosphere (Text 1)
Greenhouse/Atmosphere (Text 2)

5.4.2 Manual Question Generation

To test the effectiveness of knowledge capture, comprehension questions similar to those found in middle-school science texts were generated by independent readers of the texts (see Figure 13 for an example). The questions were generated by independent readers, instructed to design questions that require understanding the analogy in order to answer them. The readers were provided with the simplified versions of the texts. Some of the questions require combining pre-existing information from the knowledge base with knowledge gleaned from the text. Figure 13 contains an example of a question generated in this way; the remainder appear in Appendix E. In total, the experiment used 22 questions, which were manually translated to predicate calculus.

Figure 13: Question for analogy shown in **Figure 11**.

Question: What reduces the loss of heat from the house?

5.4.3 Experimental Conditions

I ran four experimental conditions, based on a 2x2 design. The first factor is whether the system used analogical dialogue acts at all. In the positive case (+A), it automatically detected ADAs, used them to construct an analogy, and included the candidate inferences in the context where questions were asked. In the negative case (-A), it did not do these things. The second factor is whether what was learned from the text was augmented with information from the knowledge base (+K) or not (-K).

The system asks questions in a special case containing the target case of the analogy along with all of the candidate inferences that are produced. The system automatically generates this case as part of ADA processing.

5.5 Results

The results appear in Table 14. The information from the text alone is sufficient to answer only one question, with or without information from the KB (-A, -K/+K). Understanding the analogy using just knowledge from the text enables just over a third of the questions to be answered (+A, -K). Allowing the system to add background knowledge from the knowledge base to the analogy raises this to over 80% (+A, +K). This demonstrates that ADAs can help a system learn from an analogy, including harnessing existing knowledge better to improve performance. These differences are all significant ($p < .01$). By incorporating existing knowledge – relational knowledge in particular – the system is able to build a more complete picture of the base domain.

Table 14: Results for question answering. +/- means with/without; A means analogy; K means facts retrieved from knowledge base.

Condition	-K	+K
-A	1 0.45	1 .045
+A	8 .364	18 .818

In all cases, questions that are answered in a condition that answered more questions correctly are a superset of the questions answered in conditions that answered fewer. In other words, there are no questions where the conditions that produced worse performance did better than any condition that produced better performance. The differences between all conditions (except for [-A,-K] and [-A,+K], which performed identically) are statistically significant. The complete question-by-question results appear in Appendix E.

5.6 Discussion

Instructional analogies are very useful to human readers. Replicating their advantages for a human reader, however, requires marshaling a variety of resources and capabilities. In particular, both accurate analysis of the analogy itself and the ability to retrieve knowledge about the domains under discussion are vital. In our corpus of examples, the system was only able to achieve robust performance when both of those capabilities are in place. It is important to note that in no case did the system simply retrieve the correct answer from memory. In all cases where using background knowledge helped, it was because background knowledge allowed for a more complete picture of the case, permitting a better or more complete analogy.

Even with the ability to draw on background knowledge and comprehend an analogy, the system still misses 4 of the 22 questions. In each of these cases, representational differences between the base and the target prevented the system from producing a match that was complete and correct enough to produce the necessary candidate inferences. Automatically adapting the representations to account for this is a major challenge for the system going forward. I discuss this further in Section 7.5.3.

As mentioned in Section 5.4.3, as part of analogical dialogue act processing the system produces a case that contains all of the facts in the target domain as well as all of the candidate inferences from the analogies. Candidate inferences are not guaranteed to be valid, and they frequently are not, particularly for cross-domain analogy. A reasonable concern is that by using all of the candidate inferences without filtering, performance on practical tasks might suffer. For the question-answering task used in the experiment, the inclusion of spurious candidate inferences was harmless. Incorrect candidate inferences can only interfere with such a task if they match the pattern the query is looking for, which did not occur in any of the examples. For other uses, however, filtering the inferences in some way will be necessary. In particular, the inferences should never just be incorporated into a long-term knowledge store without some form of validation. I discuss this further in Section 7.5.5.

In conclusion, the work here demonstrates that a system can learn from instructional analogies to improve its understanding as measured on a question answering test. The system would benefit from additional robustness in several stages of processing, but even at its current state of refinement it produces strong improvements in comprehension.

6 Related Work

6.1 Other Cognitive Architectures

Companions is not the only cognitive architecture which has been used for language tasks. Because of the importance of language, many cognitive architectures (and similar systems) have some strategies for handling it. In this section, I briefly summarize some of the work that has been done with language in other cognitive architectures. Compared to Companions, most cognitive architectures are relatively skill-oriented. Companions also supports analogy at a more foundational level than other cognitive architectures.

6.1.1 ACT-R

The ACT-R architecture (Anderson & Lebiere, 1998) is one of the most widely used and developed-for cognitive architectures available. This cognitive architecture traces its history back over thirty years, and as part of its long history there have been numerous attempts to integrate language. The core ACT-R system does not privilege language in any fashion; there is no inherent accommodation for it, and it must be implemented using ACT-R's standard rules for cognition (Anderson et al., 2004). Efforts have been made to model language at many levels, from learning individual rules (Taatgen & Anderson, 2002) to more comprehensive systems. ACT-R processing uses *productions*. These are rules that fire when their prerequisites are met and produce some output. Information is stored a "chunks", bits of declarative memory which can be placed in the system's buffers. These give the central processing module and the other modules access to each other. ACT-R is largely symbolic, but it uses subsymbolic processes for rule activation and functions like decay. ACT-R, in its core formulation, has a general perception module through which all input to the system comes.

ACT-R permits and supports language systems that are only loosely integrated into the system as a whole, allowing a wide variety of such systems to be potentially integrated into a fuller model. Ball (2003), for example, adapts the Double R Model, a model of grammar developed for this purpose, as an ACT-R module. Because ACT-R lacks the powerful unification and backtracking capabilities possessed by systems like Prolog, accounts of language understanding implemented in the cognitive architecture cannot rely on these capabilities. Ball (2004) attempts to use these limitations as an advantage, bounding the potential search space based on what isn't feasible. In this regard, Double R Theory is in keeping with a general ACT-R philosophy of keeping things cognitively plausible.

6.1.2 SOAR

Like ACT-R, SOAR (Laird, 2008) is a widely-used cognitive architecture that has been put to a wide variety of uses. Efforts in SOAR to handle natural language include NL-SOAR (Lehman et al., 1991). SOAR focuses on presenting a unified model of cognition, and NL-SOAR is part of this tradition. SOAR's model of cognition is built around applying operators to reach a goal state from a current state. Production rules are used to both make goals and determine which operators to apply. The dependency trees built by NL-SOAR can be used to perform simple question answering and inference tasks (including multimodal ones), but are less focused on long-term learning.

6.2 Other NLU Strategies

Learning by reading – particularly the natural language understanding part – is an extensively explored field, so we will be covering it only at a very high level here, aside from the approaches that most closely resemble our own. The most direct antecedent to our work in LBR is Learning Reader (Forbus et al., 2007). It has the same focus on producing facts from reading that are useful for reasoning, uses the ResearchCyc contents in its knowledge base, and executes operations using some of the same systems,

such as MAC/FAC (see section 2.3.4). It also uses automatically generated parameterized questions for evaluation. Möbius (Barker et al., 2007) produces concept-relation-concept triples and integrates them with a knowledge base. Kleo, Möbius's descendant, (Kim & Porter, 2009) starts with a small knowledge base, which it expands by integrating into it information learned by reading. It then uses that knowledge base to assist in further reading, forming a bootstrapping cycle. Both systems rely on fairly simple NLP techniques and use a system of generating new concepts based on the word used which led to some duplicate concepts, reducing the connectivity of the resulting representations. Never-Ending Language Learner (Carlson et al., 2010), or NELL, also generates facts by reading. It promotes facts to the level of knowledge based on the number of sources in which they appear and the believed reliability of those sources. Unlike our system, it does not parse sentences or attempt to map words it reads to conceptual representations. It uses word patterns such as "mayor of X" and cooccurrence in lists, among other techniques, to learn which categories things belong to and to learn relationships between them. Keeping accuracy high when learning by reading is not an easy problem, as work on NELL demonstrates: accuracy falls sharply as the system continues to add knowledge. Computer Processable Language (CPL; Clark et al., 2005) defines a set of restrictions on sentence construction that simplify machine reading while still allowing relatively natural input. Our language system is considerably more flexible than CPL. It can process a greater variety of sentence structures, allows pronouns, and does not make use of specific keywords within sentences. Some systems handle similar issues from another direction. HALO, for example (Friedland et al., 2004; Clark et al., 2007) provided a means for experts to construct knowledge through an interface without having to go directly to the level of knowledge engineering, and used controlled language for question input.

6.2.1 Machine Learning and Big Data

One of the dominant paradigms in natural language understanding today is machine learning, using large labeled training corpora. Such systems are very powerful, and are often behind state-of-the-art performance in a variety of natural language understanding tasks. Despite access to a superhuman amount of training data, however, current performance lags significantly behind humans even on relatively straightforward tasks such as word sense disambiguation (Navigli, 2009). That this is the case is not altogether surprising. Humans have the ability to marshal together a wide variety of resources for the sake of understanding language, and semantics and pragmatics almost certainly play some role. (It is also worth noting that, especially for fine-grained semantic choices, humans do not perform this task perfectly.) We hope that methods like the ones presented in this paper can serve as a complement to such methods.

6.2.2 Open Information Extraction

Open Information extraction systems such as TextRunner (Banko et al., 2007) seek to derive knowledge from a source such as the web, much like learning by reading systems do. These systems take a very different approach, however. Rather than extracting as much information as possible from a limited number of sources, they extract relational tuples from a very wide corpus (such as the entire web), and rely on redundancy to compute confidence. These have the advantage of generating a large number of tuples very efficiently, but the disadvantage of being less well-suited to capturing unique or interrelated knowledge. Additionally, they do not support reasoning over the learned information.

6.2.3 Watson

While not a cognitive architecture in a conventional sense, IBM's Watson system (Fan et al., 2010; Fan et al., 2011) has been used to perform question answering tasks using a large knowledge base as a resource. The PRISMATIC knowledge base (Fan et al., 2012) generated as part of the IBM Watson effort

is a landmark in learning by reading research, having accumulated close to a billion syntactic sentence frames; these were shown to be useful in factoid question answering that went well beyond the prior state of the art. It sidesteps word sense disambiguation entirely, by treating words themselves as predicates.

6.3 Related Work in Word Sense Disambiguation

Our goal is a system that performs word sense disambiguation in the context of a cognitive agent that learns by reading. Thus the sense inventories we use are drawn from conceptual representations, rather than language-level vocabularies such as WordNet synsets. Nevertheless, the challenges are similar, as noted above. Our system most closely resembles CatchWord (Hearst, 1991) in that it incorporates a mix of semantic and syntactic information from the local context of a word in a set of labeled training data. However, CatchWord does not use analogy for retrieval over structured relational representations constructed from the training data.

We start with the contents of the ResearchCyc KB, plus our own extensions, which are extremely accurate because they are vetted via a careful (and laborious) knowledge engineering process. Machine reading research typically tries to start with less. Some efforts (e.g., Etzioni et al., 2005) only generate word-level descriptions, such as triples of co-occurring words. While relatively straightforward to gather, such representations do not provide the kind of conceptual precision needed for reasoning. Systems like DIRT (Pantel et al., 2007) go further by extracting rules involving triples from dependency trees used in parsing. As Clark and Harrison (2010) point out, DIRT's 12 million inference rules are quite noisy, with longer chains being less reliable. DART (Clark & Harrison, 2009) and KNEXT (Van Durme & Schubert, 2008) have improved on accuracy somewhat while still being fully automatically extracted.

6.4 Related Work in Case Construction

Textual CBR systems have generally focused on building cases from text resources with the goal of using them as pointers to those text resources, rather than building standalone cases that are content in and of themselves. This is a sensible approach for many usage scenarios. For example, a system that retrieves legal documents – a very common task for TCBR - may as well just point off to the original source. The degree of specificity and nuance present in legal documents is such that a legal professional would almost certainly want to look at the original case in most scenarios. In many other scenarios, however, a structured case – or a generalization of structured cases – does make sense as a response. Most importantly, having a case representation represented using the same formalisms as a general-purpose reasoning system allows for the system to reason about the case and its contents, to find places where cases contradict each other, and to find other connections between cases. A case that merely acts as a pointer to a natural-language document does not have these properties. For this reason, we focus on constructing cases that are themselves useful for general reasoning.

Another area where related work might be useful to the future of the case construction work described in this thesis is in automatically measuring case quality. The work here uses a practical task as a measure of quality. However, there have been efforts in the field of topic modeling (Blei, 2012) to measure the coherence of the sets of words that topic modeling algorithms extract from documents (Minmo et al., 2011).

7 Conclusions and Future Work

7.1 Conclusions

In this dissertation, I present several roles for analogy in learning by reading. These are part of an overall agenda for learning by reading that focuses on maximizing the amount of information gleaned from a relatively small amount of text. There is nothing, in principle, keeping any of the methods described in this thesis from scaling for use with larger corpora. However, methods that do not rely on large corpora have advantages in situations where such corpora do not exist.

The methods described here are integrated into the Companions cognitive architecture, as part of a long-term goal to construct a long-lived reading and reasoning system. They take advantage of several key features of Companions, such as the support for analogy. However, all of the work could be adapted to use in other systems. The primary external resource required is the Structure Mapping Engine, along with MAC/FAC and SAGE, which make use of SME.

Analogical word sense disambiguation addresses an important problem in long-term learning by reading. As systems scale toward greater reasoning capabilities, representation systems that do not adequately distinguish between senses of a word risk compromising the ability of a system to reason about what it has read. AWSD has key advantages in that it requires relatively little training data and is capable of making distinctions in situations where simple heuristics fail.

Analogical reasoning is powerful, but relies on effective representations for the base and target. Natural language understanding can provide knowledge as grist for reasoning, but additional steps are required to get from the interpretations produced by NLU and cases for reasoning. Connection-based case construction is a powerful and flexible tool for building such cases.

Analogy is a powerful tool for educating humans, and analogical dialogue act processing makes that tool available for educating cognitive systems as well. It transforms what would otherwise be a liability in instructional texts, the use of instructional analogy, into a significant advantage. As we move forward, we plan to investigate the roles that analogy can play in other settings, such as a machine tutoring of people.

Together, these contributions, along with the engineering- and evaluation-oriented contributions that go into supporting them, advance the capabilities of the Companions system in learning by reading in a number of areas. They also lay a foundation for further improvement in learning by reading.

7.2 Analogical Word Sense Disambiguation

7.2.1 Improved Case Construction

There are two directions for future work in AWSD that are particularly important. First, we want to improve accuracy and learning rates even further. We plan to use the statistical information accumulated in the generalizations to ascertain which relationships and attributes are pulling their weight, and adapt the case construction process accordingly. This should increase the discrimination of analogical retrieval. We also plan to explore whether additional conceptual information from the discourse context can be used to speed learning. For example, in a text on solar energy, the word "heat" rarely refers to spiciness. Additionally, inspired by the use of confidence estimation in IBM's Watson (Gondek et al., 2012), we plan to explore techniques for enabling the system to provide confidence values for the choices it makes, based on numerical similarity estimates computed by SME. These confidence values can be combined with evidence from other disambiguation techniques (e.g., abduction), both to make initial choices and to help guide backtracking during semantic interpretation.

Additionally, we plan to explore letting the system use its own disambiguations as precedents, at least where its confidence in them is sufficiently high.

7.2.2 Integration into Long-term Learning by Reading

Analogical word sense disambiguation is integrated into Companions as a disambiguation heuristic. This means that it can and will be part of future long-term learning by reading experiments done using that system. One important task in scaling up to reading longer texts is identifying when and where different disambiguation strategies should be deployed. AWSD is capable of performing well on disambiguation tasks that our simpler set of heuristics used for general disambiguation is hopeless on, but it is also more expensive than those heuristics. This goal is symbiotic with improving the case construction process itself. As the system scales to reading larger and more deeply integrated things, we will have the data necessary to identify additional weaknesses that may exist in the AWSD system.

As the system scales into use for a greater number of reasoning tasks, these can in turn be used to further improve the system's disambiguation strategies. Performance on such tasks can supplement gold standard comparisons as a measure of the quality of a disambiguation strategy. As implemented, AWSD is a Companions disambiguation heuristic, meaning that it can easily be used alongside other such heuristics, allowing them to complement each other's strengths.

7.3 Improving Connection-Based Case Construction

The case construction algorithms as presented have utility for certain tasks, but there are a number of variations and extensions on the algorithms possible, some of which I believe to be worth pursuing further.

7.3.1 Exploiting Additional Resources

There are resources available to case construction algorithms that are not used in any of those described in Section 4. Some of these represent promising extensions to the work, while others were considered but ultimately rejected for various reasons.

One unexploited resource that the algorithm has access to is parse information. This includes part of speech information, attachment information, and other parse features. Most of the important information that can be derived from this information is already reflected in the discourse interpretation. For example, in the sentence "He eats pizza using a knife," "using" is attached to "eats," rather than "pizza." (The person eats pizza by using a knife to cut it; the pizza is not itself using a knife.) This is already represented in the discourse interpretation, which includes the fact `(instrument-Generic use2208 pizza2196)`. The pilot experiments did not turn up any particularly fertile uses for parse information in case construction, so none of the algorithms use it. Any case construction algorithm that uses parse information has an additional cost, as well, in that it has to either occur while the system has the parse in memory, or detailed parse information has to be stored long-term.

The knowledge base might also serve as a resource for case construction, above and beyond the role its lexical frames play in parsing the sentence and producing the interpretation in the first place and the role its ontology plays in providing a vocabulary of concepts and relationships. One way in which the KB can serve as a resource is by providing additional information about what concepts are related to each other. This is potentially useful in situations where the text assumes knowledge on the part of the reader. For example, in a description of the greenhouse effect that uses the well-known analogy between the earth's atmosphere and a greenhouse, the text might mention the greenhouse, its walls and roof, and the glass that they are made of without actually mentioning them together such that a

case construction algorithm working just with the text would be able to tell that they are related. A knowledge base might know that a greenhouse is a type of structure and that structures have walls and roofs, and that in the case of a greenhouse these are typically made of glass. By exploiting this knowledge, the case construction algorithm can potentially make connections between entities that are not explicitly connected in the discourse interpretation.

A second way to make use of an existing knowledge base is as a source of facts for the case itself. Rather than pulling facts only from the discourse interpretation, a case construction algorithm could in principle also include facts from the knowledge base itself, chosen according to the same strategy used to choose facts from the discourse interpretation, or chosen using a different (likely more restrictive) strategy. Analogical dialogue act processing uses a version of this to elaborate cases, as described in Section 5.3.4, and this could perhaps be extended to build better cases during connection-based case construction.

7.3.2 Additional Control Strategies

The techniques described in Section 4 treat all collections the same and all predicates (except `isa`) the same. This makes the methods very general, but if additional parameters can be cheaply specified, they can lend additional power. For example, one of the hazards of allowing a case construction algorithm to extend to isa facts when they share a collection with an isa fact already in the case is that there are some collections where this is unlikely to bring in relevant facts. Multiple instances of `SolarPanel` are more likely to be relevant to each other (in most contexts) than multiple instances of `MovementEvent`. In certain contexts, it may be valuable to specify (or learn) a set of collections that do not automatically bring in all other members of that collection. As currently implemented, the system can use `notForSegmentationPred` and `notForSegmentationCollection` to keep facts with the designated predicate and isa-facts about the designated collection (respectively) from being

included in cases altogether. The experiments described above only use these control predicates to keep bookkeeping facts and structural predicates (such as `missingSemTrans`) from being improperly extended from, but they could also be used as semantics-specific filters.

Other types of facts in the discourse interpretation cases, such as the facts that form qualitative process frames (McFate, Forbus, & Hinrichs, 2014), may be important to include in the case, but not extend from. `notForSegmentationChainingPred` and `notForSegmentationChainingCollection` can be used for this.

Another option available to a case construction algorithm is to use depth costs other than 0 and 1 in order to make certain operations more costly. For example, we might assign a cost of 3 to the operation that extends from a fact to the sentence that fact is derived from. If the depth cap is not much higher than 3, this would mean that facts that were added at a relatively high depth themselves would not be able to include their entire sentences, and facts that were added because they were in the same sentence as an existing fact would be limited in their ability to add new things.

7.3.3 Representation

One consequence of the case construction algorithms described in this thesis is that the structure of the discourse interpretation affects what sort of connections are made, on top of determining what the facts themselves look like. This means that the language system itself affects the behavior of the case construction process, as it both sets up relationships between DRSs and manages the small amount of semantic information built into the grammar. The frames it uses to generate the interpretation also affect the behavior. As noted above, the semantic frames used by our language system are heavily Davidsonian. One consequence of this is that it takes Fact-Based Segmentation two iterations to get from one actor in an event to another actor in the same event. For example, in “The cat ate a pizza,” if

the seed is "cat," the system would first add `(performedBy eat8546 cat7884)`, and then at the next depth it would add `(objectConsumed eat8546 pizza8781)`. With a different representation scheme, this information might all be added at once. In the work presented in this thesis, the representation that case construction operates over is exactly what is produced by the language system. Exploring the impact of different representation schemes on the performance of different algorithms would be part of adapting the work presented here to other language systems.

7.4 Exploiting Connection-Based Case Construction

In addition to the algorithmic refinements suggested above, there is future work in Connection-Based Case Construction that involves expanding the range of tasks to which it is applied.

7.4.1 Incorporating Multiple Sources

In the work described in this thesis, the pool of facts from which the cases are built is the discourse interpretation of a reading of a single source text. Fact-Based Segmentation, however, can be used to combine facts from multiple sources, whether that is multiple source texts or a source text and a knowledge base. This potentially makes it a valuable tool for knowledge integration. The other case construction methods do not allow for this flexibility, which is an advantage of FBS.

7.4.2 Correcting Interpretation Errors

Natural language understanding is imperfect. Parse errors, word sense disambiguation errors, coreference errors, and missing or incorrect lexical information can introduce errors into a final interpretation. Having cases that can be compared to existing cases describing similar things can potentially be used for error detection, just as it can be used to detect similarities and differences.

7.4.3 Additional Uses for Cases

The experiment in this thesis demonstrates the utility of cases constructed using the methods presented here for the task of comparison and contrast. Those are not the only tasks where such cases might be useful, however. Future work in this area might explore using cases for tasks like analogical retrieval. As discussed in Section 6.4, there has been extensive work done in textual case-based reasoning, and the cases produced using my methods could potentially be applied to some of the same problems.

7.5 Analogical Dialogue Acts

7.5.1 Less frequent ADAs

The ADA ontology is drawn from the constraints that can be placed on analogy as defined by the analogy ontology (Forbus, Mostek, & Ferguson, 2002). However, certain constraints – blocking a candidate inference and blocking a correspondence – do not appear to be common in instructional texts. One reason for this may be that they require anticipating mismatches that are sensible and likely to be made, but incorrect. It is intuitive that such mismatches might be relatively rare; if a mismatch is not sensible, there is little reason to explicitly rule it out, as common sense will do that. If an analogy naturally invites a large number of sensible but incorrect mappings, its value as an instructional tool is diminished. These acts may be more common in a more interactive environment, where the instructor can react to a student who appears to be drawing incorrect conclusions or making incorrect mappings.

7.5.2 Enhancing Detection

One of the shortcomings of the analogical dialogue acts system as currently implemented is that the detection heuristics are fairly simplistic, although in different ways. The detection heuristics for everything except Extend Base and Extend Target work by detecting one of a relatively small number of predicates in the interpretation. A simple way to enhance detection is to simply identify more such

patterns. However, this is not a complete solution. Some instances of things like Introduce Comparison rely on parallel phrasing between two sentences, neither of which has any special analogical function on its own. More sophisticated detection is necessary to correctly handle such constructions.

Extend Base and Extend Target make use of a version of Fact-Based Segmentation for detection, with no depth limit. For the corpus of instructional analogies used in the experiment described in this thesis, this works extremely well. However, if the analogies were embedded in a much longer source text passage, this could result in the entire source text being classified as these types of statements, something that may be undesirable. This is of particular concern if there are multiple analogies present in the text, as is sometimes the case. More complex bounds on the identification of these two types of ADA will likely be necessary as part of scaling up.

Another detection concern is that currently the system assumes that any pattern in the knowledge that matches the detection pattern for Introduce Comparison is an Introduce Comparison. It makes no attempt at any stage to weed out false positives. This has two potential costs. First, the inferences generated in these circumstances are unlikely to be very helpful. As the system scales up to increased autonomy, generating useless inferences and analogies risks introducing noise into the store of accumulated knowledge. A secondary concern is that, while ADA processing is not terribly expensive in terms of time or resources, building up the cases does take some time, and it would be better if that could be avoided in situations where it would not do any good.

7.5.3 Robustness

More than the other work presented here, ADA is sensitive to changes in phrasing and representation. While this is not unique to this system, ADA composes several systems, including coreference, structure mapping, its own detection, and knowledge elaboration, which are all representation-sensitive. Even

with extended detection criteria, this remains an area where future work could lead to further improvements

7.5.4 Background Knowledge and Elaboration

The current knowledge elaboration processes used by the system make use of rule macro predicates. This is a powerful and flexible system, but it does not exploit the full capabilities of the knowledge base. The system could easily be extended in the future to make use of other implication statements in the knowledge base as well.

Currently, the system runs the elaboration process exactly once, looking for all applicable rule macro predicates for all of the collections used in the case. A more sophisticated version of the system might only elaborate on certain entities, something that might be useful as the system's library of available relationships expands.

Finally, the system currently assumes that all relationships hypothesized with knowledge elaboration are correct and relevant. For some rule macro predicates, correctness is a reasonably safe assumption, although it is usually possible to come up with a counterexample. For example, in "If we go up onto the roof, we should be able to see the greenhouse," the roof is *not* the roof of the greenhouse, even though all greenhouses have roofs. Nevertheless, this rule macro predicate probably has a reasonable hit rate (and never fires incorrectly in the experimental corpus.) For many other rule macro predicates, it is much less safe to assume that they hold just because the collections they govern are present. For example, `(relationAllExists victim Murder Person)` indicates that for every murder event, there is some person who is the victim of the murder. It is not at all safe to assume that a situation that involves a murder and a person is one where the person is the victim of the murder. They might be the murderer or a detective. It does not require a contrived scenario for there to be a person

and a murder in the same text where that person is not the victim of the murder. There are two potential consequences of a false elaboration of this sort. The more serious is that it can spoil the analogy entirely. If a murderer is identified as the victim of the murder, he or she might be aligned with the victim of another crime, rather than with the perpetrator, which means that most of the candidate inferences stemming from that part of the match are likely to be bad. Less seriously, relationships that are irrelevant can still contribute to noise in the match, potentially increasing the number of bad candidate inferences that are drawn.

7.5.5 Validating Candidate Inferences

Even absent errors that might creep in earlier in the pipeline, SME does not guarantee that all of the candidate inferences drawn are correct. After a match is made, candidate inferences are drawn based on the structure of the match – semantics do not play a role. In practice, many candidate inferences drawn from a large, highly-structured cross-domain analogy are incorrect. This is because while the base and the target typically have many relational properties – and sometimes features – in common, there are usually some differences. The most common type of erroneous candidate inference is when features are mapped from the base to the target and vice versa. In a cross-domain analogy, these are nearly always false. For example, after aligning the faucet with the furnace in the example from Chapter 5, the system concludes that the faucet is a furnace and the furnace is a faucet. Spurious relationships are much less common, but still occur, especially as the result of knowledge elaboration, which sometimes brings in irrelevant relationships. For question-answering purposes, these are largely harmless (in no case in the experiment presented in the chapter did spurious inferences result in the question being missed.) For the purpose of building up a knowledge store, however, a separate vetting step for candidate inferences will be necessary. Questions can assist with this; if knowledge is useful for answering a question, it is much more likely to be relevant.

7.5.6 Generation and Interactivity

An entirely different use for analogical dialogue acts is in natural language generation. Translating an existing analogy into natural language might be done by working backwards from the analogy to determine which sort of analogical dialogue acts are necessary to convey the elements of and constraints on the analogy.

A more ambitious extension of analogical dialogue act processing is deploying it in interactive dialogue situations, where a human and a machine are communicating back and forth. If a system is attempting to tutor a human learner using an analogy, part of that process involves figuring out which ADAs are necessary to constrain a human user's misconceptions, and then generating statements that serve that function. This extends on generation as an additional application for analogical dialogue acts.

7.6 Long-term Learning by Reading

All of the work described in this thesis is aimed at eventually scaling up to longer-term learning by reading for general or targeted knowledge acquisition. There are several additional challenges that come with this. Some are specific to the techniques described in this thesis, and are described in those sections. There are other challenges, however, that are more general.

Knowledge integration is the task of connecting new knowledge with existing knowledge in order to facilitate reasoning. Part of knowledge integration is evaluating the knowledge that is learned by reading. A system that naively accepts almost everything that is learned by reading will very quickly become clogged with erroneous facts. These would interfere with both reasoning and checking the validity of new facts. One major source of such errors is errors in natural language understanding. Almost every step of NLU is imperfect, and potentially introduces errors into the interpretation. Inaccuracies in the source texts themselves can also contribute errors. While this is a relatively minor

concern, it becomes a larger factor if the internet is used as a source. Additionally, sources that are otherwise trustworthy can simply be out of date. For example, an astronomy book published before 2006 will state that there are nine planets in our solar system, because at the time Pluto was still considered a planet.

For these reasons, it is necessarily for a system to be able to estimate its confidence in what it has learned. This estimation can be based on both how confident the system is in its parsing and disambiguation and on how well the information it learned fits in with what it already knows. For example, suppose that the system believes, as a result of what it has read, that basketball player Michael Jordan was an astronomical star (rather than a star in the "famous person" sense of the word.) If this contradicts things that are already known about Michael Jordan – for example, that he is a human, and a single entity cannot be both a human and a star – the system might doubt this new information. If the system wasn't that confident in that choice to begin with – for example, if the choice of "astronomical star" over "famous person" was mostly a guess – that is further reason to lack confidence in the new fact `(isa MichaelJordan Star)`. If facts are assigned a degree of confidence, that can be used to mediate how they are used in the future.

The learning by reading system, as currently implemented, does not simply deposit learned facts into a single large repository. Rather, they are grouped by individual readings of source texts, as described in Section 2.2.8. A major goal for the system going forward is developing means for integrating the information in these readings with each other and with information already in the knowledge base.

References

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

Allen, J. F. & Perrault, C. R. (1980). *Analyzing Intention in Utterances*. Artificial Intelligence 15(3).

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review* 111, (4). 1036-1060.

Anderson, J. R. & Lebiere, C. (1998). *The atomic components of thought*. Mahwah, NJ: Erlbaum.

Ball, J. (2003). Beginnings of a language comprhension module in ACT-R 5.0. In F. Detje, D. Doerner, & H. Schaub (Eds.), *Proceedings of the Fifth International Conference on Cognitive Modeling* (pp. 231-232). Bamberg, Germany: Universitats-Verlag Bamberg.

Ball, J. (2004). A cognitively plausible model of language comprehension. *Proceedings of the 13th Conference on Behavior Representation in Modeling and Simulation* (pp. 305-316).

Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., & Etzioni, O. (2007). Open information extraction for the web. In *IJCAI* (Vol. 7, pp. 2670-2676).

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, CA: AAAI Press.

Barbella, D. & Forbus, K. (2013). Analogical Word Sense Disambiguation. *Advances in Cognitive Systems*, 2:297-315.

Barbella, D. & Forbus, K. (2015). Exploiting Connectivity for Case Construction in Learning by Reading. *Proceedings of the Third Annual Conference on Advances in Cognitive Systems.*

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.

Blass, J. & Forbus, K. (2015). Moral Decision-Making by Analogy: Generalizations vs. Exemplars. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, Austin, Texas.

Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, 55(4), 77-84.

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

Buckley, S. 1979. *Sun Up to Sun Down*. New York, NY: McGraw-Hill

Carlson, A., Betteridge, B., Kisiel, B., Settles, E.R., Hruschka, E.R., & Mitchell, T. (2010). Toward an architecture for never-ending language learning. *Proceedings of AAAI-10.*

Chang, M. D. & Forbus, K. D. (2013). Clustering Hand-Drawn Sketches via Analogical Generalization. *Proceedings of the Twenty-Fifth Innovative Applications of Artificial Intelligence* (IAAI-13), Bellevue, Washington.

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.

Clark, P., Chaw, S. Y., Barker, K., Chaudhri, V., Harrison, P., Fan, J., John, B., Porter, B., Spaulding, A., Thompson, J. & Yeh, P. (2007). Capturing and answering questions posed to a knowledge-based system. In *Proceedings of the 4th international conference on Knowledge capture* (pp. 63-70). ACM.

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.

Clark, P., & Harrison, P. (2010). BLUE-Lite: A knowledge-based lexical entailment system for RTE6. *Proceedings of the Third Text Analysis Conference.* Gaithersburg, MD: National Institute of Standards and Technology.

Clark, P., Harrison, P., Jenkins, T., Thompson, J. A., & Wojcik, R. H. (2005). Acquiring and Using World Knowledge Using a Restricted Subset of English. In FLAIRS Conference (pp. 506-511).

Clark, P., Murray, W. R., Harrison, P., & Thompson, J. (2010). Naturalness vs. predictability: A key debate in controlled languages. In Controlled Natural Language (pp. 65-81). Springer Berlin Heidelberg.

Daelemans, W., Van Den Bosch, A., & Zavrel, J. (1999). Forgetting exceptions is harmful in language learning. Machine Learning, 34, 11-41.

Decadt, B, Hoste, V., Daelemans, W., & van den Bosch, A. (2004). GAMBL, genetic algorithm optimization of memory-based WSD. *Proceedings of the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text* (pp. 108-112). Barcelona, Spain: Association for Computational Linguistics

Dehghani, M., Tomai, E., Forbus, K., Klenk, M. (2008). An Integrated Reasoning Approach to Moral Decision-Making. Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI). Chicago, IL.

de Kleer, J. (1986). An assumption-based TMS. Artificial intelligence, 28(2), 127-162.

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

Elango, P. (2005). Coreference resolution: A survey. *University of Wisconsin, Madison, WI.*

Etzioni, O., Cafarella, M. , Downey, D. , Popescu, A. , Shaked, T. , Soderland, S. , Weld, D., & Yates, A. (2005). Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165¸ 91-134.

Falkenhainer, B., Forbus, K., & Gentner, D. (1989). The structure-mapping engine: Algorithm and examples. *Artificial Intelligence, 41*, 1-63.

Fan, J., Ferrucci, D., Gondek, D., & Kalyanpur, A. (2010). *Prismatic: Inducing knowledge from a large scale lexicalized relation resource*. In Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading (pp. 122-127). Association for Computational Linguistics.

Fan, J., Kalyanpur, A., Gondek, D.C., & Ferrucci, D.A. (2012) Automatic knowledge extraction from documents. IBM J. Res. & Dev. 56(3/4).

Fan, J., Kalyanpur, A., Murdock, J. W., & Boguraev, B. K. (2011). *Mining Knowledge from Large Corpora for Type Coercion in Question Answering*.

Feigenbaum, E. A. (1980). *Knowledge Engineering: The Applied Side of Artificial Intelligence* (No. STAN-CS-80-812). STANFORD UNIV CA DEPT OF COMPUTER SCIENCE.

Ferguson, R. W. (1994). *MAGI: Analogy-based encoding using regularity and symmetry*. In A. Ram and K. Eiselt (Eds.), Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society (pp. 283-288). Atlanta, GA: Lawrence Erlbaum Associates.

Forbus, K. & de Kleer, J. (1993). *Building Problem Solvers*. MIT Press, Cambridge, MA, USA.

Forbus, K., Ferguson, R. & Gentner, D. (1994). Incremental Structure-Mapping. *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, August.

Forbus, K., Gentner, D., Everett, J. & Wu, M. (1997). Towards a computational model of evaluating and using analogical inferences. *Proceedings of CogSci97*.

Forbus, K., Gentner, D. & Law, K. (April-June, 1995). MAC/FAC: A model of Similarity-based Retrieval. *Cognitive Science*, 19(2), 141-205.

Forbus, K. D., Hinrichs, T. R., de Kleer, J., & Usher, J. M. (2010, March). FIRE: Infrastructure for Experience-Based Systems with Common Sense. In *AAAI Fall Symposium: Commonsense Knowledge*.

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., Mostek, T. & Ferguson, R. (2002). An analogy ontology for integrating analogical processing and first-principles reasoning. *Proceedings of IAAI-02*, July.

Forbus, K., Lockwood, K. & Sharma, A. (2009). Steps towards a 2nd generation learning by reading system. *AAAI Spring Symposium on Learning by Reading*, Spring 2009.

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

Forbus, K., Usher, J. & Tomai, E. (2005). Analogical learning of visual/conceptual relationships in sketches. *Proceedings of AAAI-05*.

Francis, W. N. & Kučera, H. (1964). *Manual of Information to Accompany A Standard Corpus of Present-Day Edited American English, for Use with Digital Computers*. Providence, RI: Department of Linguistics, Brown University. Revised 1971. Revised and amplified 1979.

Friedman, S., & Forbus, K. (2008). Learning causal models via progressive alignment & qualitative modeling: A simulation. *Proceedings of the Thirtieth Annual Conference of the Cognitive Science Society*. Austin, TX: Cognitive Science Society.

Friedland, N. S., Allen, P. G., Matthews, G., Witbrock, M., Baxter, D., Curtis, J., Shepard, B., Miraglia, P., Angele, J., Staab, S.& Moench, E. (2004). Project halo: Towards a digital Aristotle. *AI magazine*, 25(4), 29.

Fuchs, N. E., Kaljurand, K., & Kuhn, T. (2008). Attempto Controlled English for knowledge representation. In *Reasoning Web* (pp. 104-124). Springer Berlin Heidelberg.

Gale, W. A., Church, K., & Yarowsky, D. (1992). *Estimating upper and lower bounds on the performance of word-sense disambiguation programs*. In Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics (Newark, NJ). 249–256.

Gentner, D. (1983). *Structure-mapping: A theoretical framework for analogy*. Cognitive Science, 7, 155-170. (Reprinted in A. Collins & E. E. Smith (Eds.), Readings in cognitive science: A perspective from psychology and artificial intelligence. Palo Alto, CA: Kaufmann).

Gentner, D. (1989). *The mechanisms of analogical learning*. In S. Vosniadou & A. Ortony (Eds.), Similarity and analogical reasoning (pp. 199-241). London: Cambridge University Press. (Reprinted in Knowledge acquisition and learning, 1993, 673-694).

Gentner, D. (2003). *Why we're so smart*. In D. Gentner and S. Goldin-Meadow (Eds.), Language in mind: Advances in the study of language and thought (pp.195-235). Cambridge, MA: MIT Press.

Gentner, D., & Forbus, K. (2011). Computational models of analogy. *WIREs Cognitive Science*, 2. 266-276.

Gentner, D., & Gunn, V. (2001). Structural alignment facilitates the noticing of differences. *Memory and Cognition*, 29(4), 565-577.

Gentner, D., & Smith, L. A. (2013). *Analogical learning and reasoning*. In D. Reisberg (Ed.), The Oxford handbook of Cognitive Psychology (pp. 668-681). New York, NY: Oxford University Press.

Gondek, D., Lally, A., Kalyanpur, A. Murdock, J., Duboue, P., Zhang, L., Pan, Y., Qui, Z., & Welty, C. (2012). A framework for merging and ranking of answers in DeepQA. *IBM Journal of Research & Development,* Volume 56, Issue 3:4 (pp. 14:1-14:12).

Halstead, D., & Forbus, K. (2005). Transforming between propositions and features: Bridging the gap. *Proceedings of the Twentieth National Conference on Artificial Intelligence* (pp. 777-782). Pittsburgh, PA: AAAI Press/The MIT Press

Harrison, A. G. & Coll, R. K., Eds. (2007). *Using Analogies in Middle and Secondary Science Classrooms: The FAR Guide – An Interesting Way to Teach With Analogies*. Thousand Oaks, CA: SAGE Publications.

Hearst, M. (1991). Noun homograph disambiguation using local context in large text corpora. In *Proceedings of the Seventh Annual Conference of the UW Centre for the New OED and Text Research: Using Corpora*. Waterloo, ON: University of Waterloo.

Hinrichs, T., & Forbus, K. (2012). Learning qualitative models by demonstration. *Proceedings of the Twenty-sixth AAAI Conference on Artificial Intelligence* (pp. 207-213). Toronto, ON: AAAI Press.

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

Kim, D. & Porter, B. (2009). Kleo: A bootstrapping learning by reading system. *Proceedings of the AAAI Spring Symposium on Learning by Reading and Learning to Read* (pp. 44-49). Menlo Park, CA: AAAI Press.

Klenk, K., Forbus, K., Tomai, E., Kim, H., & Kyckelhahn, B. (2005). *Solving everyday physical reasoning problems by analogy using sketches*. Proceedings of AAAI-05.

Kuehne, S., Forbus, K., Gentner, D., & Quinn, B. (2000). SEQL: Category learning as progressive abstraction using structure mapping. *Proceedings of the Twenty-Second Annual Conference of the Cognitive Science Society* (pp. 770-775). Philadelphia, PA: Cognitive Science Society.

Laird, J. 2008. Extending the Soar Cognitive Architecture*. Artificial General Intelligence Conference*, Memphis, TN.

Lehman, J. F., Lewis, R. L., & Newell, A. (1991). Natural language comprehension in Soar: Spring 1991.

Liang, C. & Forbus, K. (2015). Learning Plausible Inferences from Semantic Web Knowledge by Combining Analogical Generalization with Structured Logistic Regression. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, Austin, Texas

Lockwood, K. & Forbus, K. (2009). Multimodal knowledge capture from text and diagrams*. Proceedings of KCAP-2009.*

Lockwood, K., Lovett, A., & Forbus, K. (2008). Automatic Classification of Containment and Support Spatial Relations in English and Dutch. *Proceedings of Spatial Cognition*.

Miller, George A. (1995). *WordNet: A Lexical Database for English*. Communications of the ACM Vol. 38, No. 11: 39-41.

McFate, C.J., Forbus, K. & Hinrichs, T. (2014). Using Narrative Function to Extract Qualitative Information from Natural Language Texts. *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, Québec City, Québec, Canada.

McLure, M.D., Friedman S.E. & Forbus, K.D. (2015). Extending Analogical Generalization with Near-Misses. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, Austin, Texas

Mimno, D., Wallach, H. M., Talley, E., Leenders, M., & McCallum, A. (2011). Optimizing semantic coherence in topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 262-272). Association for Computational Linguistics.

Mostek, T., Forbus, K. D., & Meverden, C. (2000). Dynamic case creation and expansion for analogical reasoning. In *Proceedings of AAAI-2000*, 323-329. Austin, TX.

Navigli, R. (2009). Word sense disambiguation: A Survey. *ACM Computing Surveys*, *41*.

Pantel, P., Bhagat, R., Coppola, B., Chklovski, T., & Hovy, E. (2007). ISP: Learning Inferential Selection Preferences. *Proceedings of the 2007 Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics* (pp. 131-138). Rochester, NY: Association for Computational Linguistics.

Paul, D. B., & Baker, J. M. (1992). The design for the Wall Street Journal-based CSR corpus. *Proceedings of the Workshop on Speech and Natural Language* (pp. 357-362). Stroudsburg, PA: Association for Computational Linguistics.

Peterson, J., Mahesh, K., & Goel, A. (1994). Situating natural language understanding within experience-based design. *International journal of human-computer studies*, 41(6), 881-913.

Pradhan, S., Loper, E., Dligach, D., & Palmer, M. (2007). SemEval-2007 task 17: English lexical sample, SRL and all words. *Proceedings of the Fourth International Workshop on Semantic Evaluations* (pp. 87-92). Stroudsburg, PA: Association for Computational Linguistics.

Schwitter, R. (2002). English as a formal specification language. *Proceedings of the 13th International Workshop on Database and Expert Systems Applications*. IEEE, 2002.

Taatgen, N.A. & Anderson, J.R. (2002). Why do children learn to say "broke"? A model of learning the past tense without feedback. *Cognition*, 86(2), 123-155.

Tomai, E. & Forbus, K. (2009). EA NLU: Practical Language Understanding for Cognitive Modeling. *Proceedings of the 22nd International Florida Artificial Intelligence Research Society Conference*. Sanibel Island, Florida.

Traum, D. R. (2000). 20 Questions on Dialogue Act Taxonomies. *Journal of Semantics*, 17, 7-30.

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.

Wei, X., & Croft, W. B. (2006). LDA-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 178-185). ACM.

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.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics* (189-196). Stroudsburg, PA: Association for Computational Linguistics.

Zhang, Z., & Ng, H. T. (2010). It Makes Sense: A Wide-Coverage Word Sense Disambiguation System for Free Text. *Proceedings of the ACL 2010 System Demonstrations* (pp. 78-83). Uppsala, Sweden: Association for Computational Linguistics

Appendix: Charts, Tables, Selected Sources, etc.

A. Full Coreference Specification

The coreference resolution algorithm is designed to be general purpose, and should not be altered for the sake of handling specific examples, unless those examples are emblematic of broad patterns in language. The algorithm is part of QRG-CE. This means that it is *not* intended to handle every English construction. Rather, the design goal is that there should be some way to express things we want to express. The following is a description of the way the system worked at the time the experiments described above were run. The coreference system can be used with sketching to allow for references to things in a sketch. As those details are not relevant to the work described in this thesis, they are omitted here.

A.1 Behavior

There are four things that seek reference resolution:

- Direct reference: "The cat," "the city." (Indirect reference - "a cat" - does not seek coreference.)
- Verbs
- This, that - These are used for backreference and demonstrative reference.
- Other pronouns - It, him, her. This is called pronominal reference.

The outputs of running the coreference mechanism are `referent` statements. These are what the rest of the system cares about, either directly or because they are what the lifting process that builds the discourse interpretation DRS looks at. `candidateReferent` and `resolveReferences` statements are part of the process of generating referent statements, but they're not completed reference resolution. Because coreferent items are resolved to the same symbol, reference is effectively

transitive and commutative. Things that look for the subject of a sentence look for the actual head of the subject of the sentence. In "The dog's hat is fancy," "hat" is the subject, not "dog."

A.2 `filterByPreference` and `preferredBinding`

`filterByPreference` is a dynamic update predicate. It takes one argument, a statement. When `filterByPreference` is wrapped around a query, the query is handled specially. First, the system executes the query normally. If more than one binding is returned, however, it queries it again, this time wrapped in `preferredBinding`. Special rules handle queries wrapped in preferred binding, and select from among the possible bindings. The priorities generated by the binding operations described below are used by `preferredBinding` to select from among competing options.

A.3 Backreference

If the reference is a `Backreference-NLAttrType` (“this” or “that”), the term refers to the DRS of the previous sentence. (Priority = 0.1).

- "The valve closes. This causes the temperature to rise." "This" will refer to the sentence DRS of "The valve closes".

A.4 Possessed Reference

If an entity is possessed (anything that uses the `possessiveRelation` predicate), it will refer to anything it can match to that is possessed by the same thing. (Priority = 0.1). If it cannot refer to something that is possessed by the same thing, it will follow the rules as normal, and if it has no other options, it will refer to itself. It will not refer to something that's possessed by something else, though, or in a different container.

- "The capital of the country is small. A capital is nice. The capital of the country is cold." Third 'capital' refers to first, because they are possessed by the same thing.
- "A capital is nice. A capital of the state is small. The capital of the country is cold." Third 'capital' refers to first. It skips over the second one because it is possessed by something else.

A.5 Standard Pronominal Reference

Pronominal reference is triggered by pronouns. Pronominal reference only looks one sentence back. This is because it uses `sentenceAttachment`, which looks one sentence back. It does NOT look within the same sentence.

- "A man eats a sandwich. He smiles." "He" refers to "man".
- "A man eats a sandwich. The sky is gray. He smiles." "He" refers to nothing, because there is no valid referent in the previous sentence. (We know that "sky" cannot be "he".)

Gendered pronouns do not match to a different gender.

- "A man eats a sandwich. She smiles." "She" does not refer to anything.

Pronouns do not match to events.

Gendered pronouns prefer to map to subjects of sentences, rather than non-subjects. (Priority = 0.2).

- "A man gives a man a sandwich. He smiles." "He" matches to the first "man".
- "A cat gives a boy a dead rat. He smiles." "He" matches to "cat".

It still needs to be valid in terms of gender, however.

- "A woman gives a boy a sandwich. He smiles." "He" matches to "boy".
- "A woman gives a cat a treat. He smiles." "He" matches to "cat".

Number has to match too. The system does not support singular "they".

- "The cats are tired. They sleep." "They" matches to "cats".
- "A dog sees the cats. They run." "They" matches to "cats".
- "The cats are tired. He sleeps." "He" matches to nothing.

These distinctions are handled by `matchPronounFeatures`. We do not distinguish between things that could possibly have a specified gender (cat, president) and things that do have a specified gender (girl, boy). If a gender is possible, it is a valid referent. The system does not do any reasoning to determine anything's gender; it only looks to see if the collection it belongs to is a spec of `MaleOrganism` or `FemaleOrganism`. (For example, it knows that “girl” is a spec of FemaleOrganism.)

- "My cat is male. She eats bugs." "She" refers to "cat", because "cat" could be any gender, and we do not look for other evidence of gender. The coreference mechanism does not do the additional reasoning to determine that the cat is male, and thus would not be referred to as “she.”

A.6 Standard Definite Reference

Definite reference is triggered by nouns introduced with the definite article, “the.”

Definite reference prefers subjects to non-subjects.

- "My dog gave your dog a tennis ball. The dog smiled." "The dog" refers to my dog.

When there are two non-subjects to resolve to, it resolves to nothing.

- "The cat saw your dog and my dog. The dog ran." "The dog" resolves to nothing.

- "The cat saw a dog. The dog ran." "The dog" resolves to "a dog".

Definite reference is based on collection membership, not words.

- "A hound was hungry. The dog ate a boot." If `Dog` is the collection for both “dog” and “hound,” they will corefer.

Definite reference will go back up to five sentences. It starts looking one sentence back, meaning it will not resolve to things in the same sentence.

- "A dog sleeps. A cat eats. The cat runs. The cat jumps. The cat shakes. The dog wakes." "The dog" refers to "A dog".
- "The woman knew the man was watching the woman." "The woman" does not resolve to "the woman".

Recency is prioritized over matching to a subject.

- "A dog sleeps. A cat eats. The cat runs. The cat jumps over a dog. The cat shakes. The dog wakes." "The dog" refers to the dog that was jumped over, not the dog that sleeps.

A.7 Verb Coreference

Verbs will corefer as long as the verbs are the same type action and there are no role relation conflicts, meaning that anything in the same role relation slot has to also corefer.

- "The water flows into a bucket. The water flows quickly." "Flows" are coreferent.
- "The lava flows into a bucket. The water flows quickly." No coreference, because `primaryObjectMoving` is different in each case.

A.8 Things that do not look for other referents

- Pnames map to themselves.
- Indefinite references ("A cat") map to themselves.
- References to the listener ("you") map to the LISTENER token.

B. Selected Corpus Texts

Chapter 16 of Sun Up to Sun Down	
Original Text	Simplified Text
Now that we've examined all the elements of a solar heating system, let's see how it operates during a typical winter day.	We examined the elements of a solar heating system.
	We will examine the solar heating system operating during a typical day.
Right now we'll consider only collecting and storing heat; later we'll see how we use the stored heat.	Now, we will consider collection of heat and storage of heat.
	After that, we will consider using the stored heat.
We'll also follow the operation of our rainwater collecting apparatus so comparisons can be made between the two at various stages.	We also compare the rainwater collection system and the solar heating system.
Let's assume that both collecting systems begin the day partly full.	At the start of each day, the solar heating system is semifull.
	At the start of each day, the rainwater collection system is semifull.
For example, in the rainwater collector, there might have been some rainwater left from previous rain.	The rainwater collection system could have some rainwater in it.
In the solar collecting system, we'll assume the heat storage tank is still warm from the previous day's collection.	The solar heating system's heat storage is warm because it collected heat during the previous day.
In the rainwater apparatus, the check valve has closed to keep the stored water from leaking out.	In the rainwater collection system, the valve closes to prevent the flow of the stored water.
No rain is falling, and any water in the pipe has long since leaked out.	The rain is not falling.

	The water in the pipe has leaked out.
As for the solar heater, the sun has not yet risen.	The sun has not yet risen on the solar heating system.
The collector, which has been exposed to the cold night air, is much colder than the heat storage tank.	The solar collector was exposed to the cold air at night.
	Because of this, the solar collector's temperature is lower than the heat storage tank's temperature.
The heat storage is warmer than the outdoors because it still contains heat.	The heat storage contains heat.
	Because of this, the heat storage's temperature is greater than the air's temperature.
The controller had sensed that the collector was colder than the heat storage tank and shut off the pump, preventing the storage tank from cooling.	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.
As the sun rises, the collector starts to get a little warmer.	The height of the sun rises.
	This increases the solar collector's temperature.
But since a south-oriented collector does not face the sun early in the morning, not much solar radiation is absorbed.	A solar collector faces south.
	In the morning, the solar collector is not facing the sun.
	Because of this, the amount of sunlight absorbed is low.
	The solar collector absorbs the sunlight.
The collector will get warmer, but not as warm as the heat storage.	The solar collector's temperature increases.
	The heat storage's temperature exceeds the solar collector's temperature.
Since the collector is still colder than the storage tank, the controller wo not turn on the pump.	Because the heat storage's temperature exceeds the solar collector's temperature, the control device does not turn on the pump.
Of course, the collector is a little warmer than the outdoors, so it loses heat through its glazing.	Because the solar collector's temperature exceeds the air's temperature, heat leaves the solar collector

	through the glaze.
In fact, it loses all the heat absorbed by the collector: since none is saved, all of the incoming heat is lost.	The heat in the solar collector leaves.
	No heat in the solar collector remains.
	The solar collector does not collect heat.
For the rainwater apparatus to be analogous to our solar collector, let's suppose that it starts to drizzle on the rainwater collecting tray.	The rainwater collection system is like the solar heating system.
	Rainwater falls onto the rainwater collection tray.
Since little rainwater is being collected, the water in the collection pipe does not get as deep as the storage depth.	The amount of captured rainwater is low.
	Because of this, the depth of the water in the storage tank exceeds the depth of the water in the pipe.
No rainwater flows into storage, and efficiency is zero-all the rain captured by the tray leaks away.	Rainwater does not flow into the storage tank.
	The rain captured by the tray leaks away.
Until the temperature of a collector gets hotter than the storage tank, no heat is collected.	The system does not capture heat, when the heat storage's temperature exceeds the solar collector's temperature.
Often on cloudy days, a solar collector does not collect any heat whatsoever, just as when the sun first rises in the morning not enough solar radiation falls on the collector to make its stagnation temperature hotter than the storage tank's.	During a cloudy day, a solar collector does not capture heat.
	Similarly, when the sun is low, the amount of solar radiation on the solar collector is low.
	Because of this, the heat storage's temperature exceeds the solar collector's stagnation temperature.
The heat leaks away from the collector as fast as it is absorbed.	The rate of heat flow from the solar collector equals the rate of absorption of heat.
The cold temperature outdoors aggravates the situation, since the incoming solar radiation must further raise the collector's temperature so that it equals the storage temperature.	The air is cold.

	This increases the distance between the temperature of the solar collector and the temperature of the heat storage.
	This increases the amount of necessary solar radiation.
As the sun gets higher in the sky, its radiation strikes the collector more and more head-on, and more solar radiation is absorbed by the collector plate.	As the height of the sun increases, the amount of solar radiation on the solar collector increases.
Eventually the collector plate gets hotter than the storage tank.	Eventually, the solar collector's temperature exceeds the heat storage's temperature.
When it does get hotter, the pump controller senses the temperature difference between the collector and the storage tank and turns on the pump.	The control device senses the temperature difference between the solar collector and the heat storage.
	This causes the control device to turn on the pump.
With pump water flowing, some of the heat absorbed by the collector is moved into storage: water flowing from storage is heated as it passes through the collector.	The water in the pump flows.
	Because of this, the solar collector's heat moves to the heat storage.
	Water flows from the storage tank to the solar collector.
	The water is heated in the solar collector.
Warm water from the collector further heats up the storage tank.	Warm water from the solar collector increases the temperature of the heat storage.
Now not all the collector's heat is lost, since some of it is going into storage.	Heat flows from the solar collector to the heat storage.
	Because of this, heat is saved.
In the rainwater collection system, let's suppose the rain increases from a drizzle to a light rain; this is analogous to more sunshine being absorbed by the solar collector.	The amount of rain that falls on the rainwater collection system increases.
	That increase is like an increase of the amount of solar radiation on the solar collector.
With more rainwater collected and flowing down the pipe, the pipe's water depth gets higher than	The amount of rainwater in the pipe increases.

the storage water depth.	
	Because of this, the depth of the water in the pipe exceeds the depth of the water in the storage tank.
The check valve opens and not all the rainwater is lost through the leak, since some is flowing into storage.	The valve opens.
	Because of this, the rainwater flows into the storage tank.
In the late morning the sun faces the collector nearly head-on; and much more solar radiation is absorbed by the collector plate.	During the late morning, the sun faces the solar collector directly.
	Because of this, the solar collector absorbs extra solar radiation.
With the pump on, warm water from storage is pumped through the collector, where it is further heated.	The pump is turned on.
	Because of this, warm water from the storage tank flows through the solar collector.
	The water's temperature increases in the solar collector.
It returns to the storage tank hotter and continues to heat the tank.	The water is hot when it returns to the heat storage.
	Because of this, the water increases the temperature of the heat storage.
The collector always stays a little hotter than the storage tank, so the controller always keeps the pump on.	The solar collector's temperature exceeds the heat storage's temperature.
	Because of this, the pump is turned on.
Efficiency is high in the late morning.	Efficiency is high in the late morning.
The collector receives water from storage, and before noon storage is not very hot yet.	Water flows to the solar collector from the heat storage.
	The heat storage is cool before noon.
When the collector is not hot, heat losses are low, resulting in high efficiency.	When the solar collector is cool, leakage of heat is low.
	This increases efficiency.
The rainwater equivalent to the sun hitting the collector nearly head-on is a heavy rain.	Heavy rain is like direct sunlight.

The incoming water fills up the pipe and flows through the check valve into the storage tank.	Water fills the pipe.
	The water flows through the valve into the storage tank.
Since the pipe's water depth is not too high yet, not much rainwater is lost and efficiency is high.	Because the depth of water in the pipe is low, rainwater is saved.
	Because the depth of the water in the pipe is low, efficiency is high.

Table 15 The facts that were included in a case built using the Sentence-Based Segmentation method. 28 facts were included in total, across 4 DRSs. Representations simplified for space and readability. Note that some facts, such as the ones that include `ObservenceDay` and `Surgery`, are incorrect interpretations of the sentence "We will examine the solar heating system operating during a typical day." Because the interpretations are generated completely automatically, they contain some word sense errors. The facts shown here were derived from that sentence and from "We examined the elements of a solar heating system" and "At the start of each day, the solar heating system is semifull."

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)
(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)
(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)
(isa examine04 Inspecting)
```

Table 16 . The simplified version of the second paragraph of one document in the corpus containing the example sentence from above.

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.

Table 17. Facts generated with Fact-based Segmentation on the example seed, running to a depth of 3.

```
(holdsIn (DrsCaseFn DRS-3630536315-157727) member element2249 group-of-element2249))
   (holdsIn (DrsCaseFn DRS-3630536315-157728) (isa element2249
ElementStuffTypeByNumberOfProtons))
   (holdsIn (DrsCaseFn DRS-3630536315-157728) (isa solar-heating-system2379
SolarHeatingSystem))
   (holdsIn (DrsCaseFn DRS-3630536315-157728) (possessiveRelation solar-heating-system2379
element2249))
   (holdsIn (DrsCaseFn DRS-3630536319-157731) (conceptuallyRelated day3151 Normal-Usual))
   (holdsIn (DrsCaseFn DRS-3630536319-157731) (evaluee-Direct examine2184 solar-heating-
system2379))
   (holdsIn (DrsCaseFn DRS-3630536319-157731) (isa solar-heating-system2379
SolarHeatingSystem))
   (holdsIn (DrsCaseFn DRS-3630536319-157731) (performedBy examine2184 we2662))
   (holdsIn (DrsCaseFn DRS-3630536319-157731) (temporallySubsumes day3151 operate2951))
   (holdsIn (DrsCaseFn DRS-3630536319-157731) (isa day3151 ObservanceDay))
   (holdsIn (DrsCaseFn DRS-3630536319-157731) (isa examine2184 Inspecting))
   (holdsIn (DrsCaseFn DRS-3630536319-157731) (isa operate2951 Surgery))
   (holdsIn (DrsCaseFn DRS-3630536369-157746) (doneBy collect12189 solar-heating-
system12161))
   (holdsIn (DrsCaseFn DRS-3630536369-157746)
     (followsInProgression previous-event13115 day9430 series13116))
   (holdsIn (DrsCaseFn DRS-3630536369-157746) (groupGathered collect12189 heat12490))
   (holdsIn (DrsCaseFn DRS-3630536369-157746) (isa collect12189 Collecting))
   (holdsIn (DrsCaseFn DRS-3630536369-157746) (isa heat12490 ThermalEnergy))
   (holdsIn (DrsCaseFn DRS-3630536369-157746)
     (isa solar-heating-system12161 SolarHeatingSystem))
   (holdsIn (DrsCaseFn DRS-3630536369-157746) (primaryObjectMoving collect12189 heat12490))
   (holdsIn (DrsCaseFn DRS-3630536369-157746)
     (relationInstanceMember primaryObjectMoving collect12189
       heat12490))
   (holdsIn (DrsCaseFn DRS-3630536369-157746) (temporallySubsumes day9430 collect12189))
   (holdsIn (DrsCaseFn DRS-3630536369-157746) (isa day9430 ObservanceDay))
   (holdsIn (DrsCaseFn DRS-3630536369-157747) (conceptuallyRelated heat-storage12014 Warm))
   (holdsIn (DrsCaseFn DRS-3630536369-157747) (isa heat-storage12014 HeatStorage))
   (holdsIn (DrsCaseFn DRS-3630536369-157747)
     (isa solar-heating-system11952 SolarHeatingSystem))
   (holdsIn (DrsCaseFn DRS-3630536369-157747)
     (possessiveRelation solar-heating-system11952 heat-storage12014))
   (holdsIn (DrsCaseFn DRS-3630536389-157750) (isa close15268 ClosingAContainerArtifact))
   (holdsIn (DrsCaseFn DRS-3630536389-157750)
     (isa rainwater-collection-system5407 RainwaterCollectionSystem))
   (holdsIn (DrsCaseFn DRS-3630536389-157750) (objectOfStateChange close15268 valve15262))
   (holdsIn (DrsCaseFn DRS-3630536389-157750)
     (occursDuring close15268 rainwater-collection-system5407))
   (holdsIn (DrsCaseFn DRS-3630536389-157750) (isa valve15262 Valve))
   (holdsIn (DrsCaseFn DRS-3630536412-157756) (conceptuallyRelated rise16227
       (IntervalAfterFn Now AnIndefiniteAmountOfTime)))
   (holdsIn (DrsCaseFn DRS-3630536412-157756) (isa rise16227 AscendingEvent))
   (holdsIn (DrsCaseFn DRS-3630536412-157756) (isa solar-heating-system18243
SolarHeatingSystem))
```

```
(holdsIn (DrsCaseFn DRS-3630536412-157756) (objectMoving rise16227 sun16153))
(holdsIn (DrsCaseFn DRS-3630536412-157756)
  (on-UnderspecifiedSurface sun16153 solar-heating-system18243))
(holdsIn Discourse-DRS-01 (doneBy prevent15366 (DrsCaseFn DRS-3630536389-157750)))
(holdsIn Discourse-DRS-01 (evaluee-Direct compare5061 rainwater-collection-system5407))
(holdsIn Discourse-DRS-01 (evaluee-Direct compare5061 solar-heating-system2379))
(holdsIn Discourse-DRS-01 (evaluee-Direct examine2184 element2249))
(holdsIn Discourse-DRS-01 (fullnessOfContainer solar-heating-system2379 PartiallyFull))
(holdsIn Discourse-DRS-01
  (implies (DrsCaseFn DRS-3630536369-157746) (DrsCaseFn DRS-3630536369-157747)))
(holdsIn Discourse-DRS-01
  (implies-DrsDrs (DrsCaseFn DRS-3630536315-157727) (DrsCaseFn DRS-3630536315-157728)))
(holdsIn Discourse-DRS-01 (isa close15268 ClosingAContainerArtifact))
(holdsIn Discourse-DRS-01 (isa compare5061 Comparing))
(holdsIn Discourse-DRS-01 (isa day9430 ObservanceDay))
(holdsIn Discourse-DRS-01 (isa examine2184 Inspecting))
(holdsIn Discourse-DRS-01 (isa flow15431 FluidFlow-Translation))
(holdsIn Discourse-DRS-01 (isa group-of-element2249 Set-Mathematical))
(holdsIn Discourse-DRS-01 (isa prevent15366 (PreventingFn flow15431)))
(holdsIn Discourse-DRS-01 (isa prevent15366 PreventingFromHappening))
(holdsIn Discourse-DRS-01 (isa rainwater-collection-system5407 RainwaterCollectionSystem))
(holdsIn Discourse-DRS-01 (isa solar-heating-system2379 SolarHeatingSystem))
(holdsIn Discourse-DRS-01 (isa sun16153 Sunlight-Direct))
(holdsIn Discourse-DRS-01 (isa valve15262 Valve))
(holdsIn Discourse-DRS-01 (isa water15524 Water))
(holdsIn Discourse-DRS-01 (not (DrsCaseFn DRS-3630536412-157756)))
(holdsIn Discourse-DRS-01 (performedBy compare5061 we5052))
(holdsIn Discourse-DRS-01 (performedBy examine2184 we2178))
(holdsIn Discourse-DRS-01 (possessiveRelation water15524 flow15431))
(holdsIn Discourse-DRS-01 (prevents-SitSit (DrsCaseFn DRS-3630536389-157750) flow15431))
(holdsIn Discourse-DRS-01 (primaryObjectMoving flow15431 water15524))
(holdsIn Discourse-DRS-01 (willBe (DrsCaseFn DRS-3630536319-157731)))
```

C. Full AWSD Stars Corpus

A star is a massive ball of plasma that gravity holds.

A star begins as a collapsing cloud of material.

The star is made primarily of hydrogen and helium and small amounts of heavier elements.

The star's radiation pressure prevents the collapse of the star.

The star then evolves into a degenerate form, and the star sends a portion of its matter into space.

The star forms a new group of stars with a higher proportion of heavy elements.

The star that is near to the Earth is the Sun.

When the Sun goes down, we can see another star in the night sky.

The sun is an example of a star like that.

A medium star that explodes transforms into a neutron star.

She turned into a successful star as a young adult.

The broken ankle kept the star player away from the field for months.

The United States was the land where I could turn into a star and get rich.

He emerged as a league star and his scoring entertained crowds.

A stage name is a name that a star uses in the place of their real name.

He was a star goalie for the team.

He was a left wing and he was the star player.

He was the biggest music star in the decade.

His musical skill and his stage personality transformed him into a huge star.

Rock Hudson was famous for his appearance and he is remembered as a romantic star during the middle of the century.

People think that a star is made in a nebula.

When a very big star dies, the star explodes.

A red giant is a very big star.

Stellar evolution is the study of a star changing over time.

It contains diverse information about star movement and eclipses.

It is formed, when a star like the sun sends out atoms.

Any star in the physical universe can be involved in a star impact.

This can happen, when fusion is active in the star.

This can happen, when the star is dead and fusion stops.

The star is not for correcting for the rotation of the Earth.

The star is a style icon that is famous for her beautiful hair.

He was a star player at this time.

Then he arrived and turned into a huge star in the league.

He was the team's star goalie.

She was in her first television series and she turned into a star in this role.

There a staff member saw that she was a potential star.

His criterion is a star's name's value for getting people to movie theaters.

A female star is valuable, when she is hired with a male star.

He said that obtaining a valuable star is difficult, because a star like that is rare.

He developed a method to measure the value of a star in a film.

The star is used to correct for waves in the Earth's atmosphere.

A compact star that is not a black hole is a degenerate star.

This means that the star can be viewed from the Earth's surface.

A giant star's size exceeds an ordinary star's size.

A star turns into a giant, when the hydrogen for fusion at its center disappears.

An exotic star is a compact star.

The star is in a late period in the evolution of some stars.

The star forms, when a red giant star loses its outer hydrogen layers.

Vega is the brightest star in Lyra.

It is the second brightest star in the northern celestial hemisphere.

When an actor is the main actor in a movie, he is the star of the movie.

He was a regular star of movies and television.

He is remembered by later audiences as the star of the television series.

He was the star of The Untouchables, and he served as the host of a mystery show.

He played a law enforcement officer in Florida in a boat that was the star of the show.

She made her Broadway debut as the star of the musical.

The white barber is the star of the shop, and he feels that he deserves the star treatment.

They learned to accept him, and he is the star barber at the shop.

She is the star of her own film where she moves to Atlanta and her daughter attends a private music school.

He is known as the star of the ABC sitcom and as a host of It's Showtime at the Apollo.

With Arcturus and Sirius, the star is one of the brightest stars in the Sun's neighborhood.

Vega is the second star that was photographed.

It was the first star that had its waves recorded.

It was the first star that had its distance estimated with diagonal measurements.

In a supernova, a star depletes its nuclear fuel and explodes.

If a star is large, then it transforms into a black hole.

If the star is small, then it will form a white dwarf.

A white dwarf is a star.

The color of a star indicates the heat of the star.

A white star is like the sun.

A blue star's temperature exceeds the sun's temperature.

She was the star of an American reality television series.

This serial was about the life of a Russian spy portrayed by the lead Russian star.

She played a small but significant role in a Russian hit film with another great Russian star.

An actress received the attention of the star of the theater.

He is an American professional skateboarder and was the star of an MTV reality show.

People may think the star of the show is the character, but the key element in telling the story is the vehicle.

He is known primarily as the star of the FOX television family sitcom.

He acted on the stage and television in Mexico, and he turned into a modest star.

The performing stopped the show, and Ricky Martin received unexpected applause, and the star was introduced.

He was ready to shatter the record of Wayne Gretzky, and he was the star in the previous year's game and would be the star in next year's game.

A red star's temperature is lower than the sun's temperature.

This heat is from the heat that was produced during the star's collapse.

This stays after the star's heat and light have disappeared.

It is the brightest star in the southern Centaurus constellation.

A red giant is a very big star, and its weight is like the sun's weight.

A star changes hydrogen into helium with nuclear fusion.

The star's outer layer expands.

The star will get brighter.

Because the outside of the star will get bigger, the energy will be spread over a much larger area.

He thought that the star was moving from the earth.

He thought that the movement caused the star's color changes.

A planet is a large object that orbits a star.

He was the star of William Shakespeare's theater company.

She was the star of a sketch comedy series that her friend created.

He was the star of the zoo in Berlin, Germany.

She was the first killer whale that survived in a tank, and she was the star of a very popular killer whale show at SeaWorld in San Diego.

He was the star and he was the only regular actor of this series.

She was the star of the ABC television show.

He wrote and directed film and television projects, and he was the star of the television series.

When he graduated, he was the star of Ohio's football team.

In the first game between Texas and Oklahoma, He was the throwing star in a win for Texas.

D. Full Example FBS Case

```
(holdsIn (DrsCaseFn DRS-3630536389-157750)
  (isa close15268 ClosingAContainerArtifact))
(holdsIn (DrsCaseFn DRS-3630536389-157750)
  (isa rainwater-collection-system5407 RainwaterCollectionSystem))
(holdsIn (DrsCaseFn DRS-3630536389-157750)
```

```
  (objectOfStateChange close15268 valve15262))
(holdsIn (DrsCaseFn DRS-3630536389-157750)
  (occursDuring close15268 rainwater-collection-system5407))
(holdsIn (DrsCaseFn DRS-3630536389-157750) (isa valve15262 Valve))
(holdsIn (DrsCaseFn DRS-3630536443-157760)
  (containedObject contain23805 heat23844))
(holdsIn (DrsCaseFn DRS-3630536443-157760)
  (containingObject contain23805 heat-storage23789))
(holdsIn (DrsCaseFn DRS-3630536443-157760)
  (isa contain23805 ContainingSomething))
(holdsIn (DrsCaseFn DRS-3630536443-157760)
  (isa heat-storage23789 HeatStorage))
(holdsIn (DrsCaseFn DRS-3630536443-157760)
  (isa heat23844 ThermalEnergy))
(holdsIn (DrsCaseFn DRS-3630536459-157764)
  (comparativeRelation be24024
(Kappa (?larger ?smaller)
   (largerThan ?larger ?smaller hasEvaluativeQuantity))))
(holdsIn (DrsCaseFn DRS-3630536459-157764)
  (comparee be24024 temperature25970))
(holdsIn (DrsCaseFn DRS-3630536459-157764)
  (comparer be24024 temperature25265))
(holdsIn (DrsCaseFn DRS-3630536459-157764)
  (isa be24024 ComparisonEvent))
(holdsIn (DrsCaseFn DRS-3630536459-157764)
```

```
  (isa heat-storage23789 HeatStorage))
(holdsIn (DrsCaseFn DRS-3630536459-157764)
  (isa solar-collector23314 SolarPanel))
(holdsIn (DrsCaseFn DRS-3630536459-157764)
  (isa temperature25265 Temperature))
(holdsIn (DrsCaseFn DRS-3630536459-157764)
  (isa temperature25970 Temperature))
(holdsIn (DrsCaseFn DRS-3630536459-157764)
  (possessiveRelation heat-storage23789 temperature25265))
(holdsIn (DrsCaseFn DRS-3630536459-157764)
  (possessiveRelation solar-collector23314 temperature25970))
(holdsIn (DrsCaseFn DRS-3630536459-157764)
  (temperatureOfObject heat-storage23789 temperature25265))
(holdsIn (DrsCaseFn DRS-3630536459-157764)
  (temperatureOfObject solar-collector23314 temperature25970))
(holdsIn (DrsCaseFn DRS-3630536459-157765)
  (isa control-device24203 ControlDevice))
(holdsIn (DrsCaseFn DRS-3630536459-157765)
  (isa sense24220 Perceiving))
(holdsIn (DrsCaseFn DRS-3630536469-157768)
  (member pump27378 group-of-pump27378))
(holdsIn (DrsCaseFn DRS-3630536469-157769)
  (isa pump27378 Pump-TheShoe))
(holdsIn (DrsCaseFn DRS-3630536469-157770)
  (implies (DrsCaseFn DRS-3630536459-157764)
```

```
(DrsCaseFn DRS-3630536459-157765)))
(holdsIn (DrsCaseFn DRS-3630536483-157773)
  (causes-ThingSit (DrsCaseFn DRS-3630536469-157770) shut-off26323))
(holdsIn (DrsCaseFn DRS-3630536483-157773)
  (implies-DrsDrs (DrsCaseFn DRS-3630536469-157768)
(DrsCaseFn DRS-3630536469-157769)))
(holdsIn (DrsCaseFn DRS-3630536483-157773)
  (isa control-device24203 ControlDevice))
(holdsIn (DrsCaseFn DRS-3630536483-157773)
  (isa group-of-pump27378 Set-Mathematical))
(holdsIn (DrsCaseFn DRS-3630536483-157773)
  (isa shut-off26323 ShuttingOffSomething))
(holdsIn
  (FactsFromReadingMtFn Reading-of-SUSD-Simplified-Jan2015-02)
  (causes-ThingSit (DrsCaseFn DRS-3630536443-157760) be24024))
(holdsIn
  (FactsFromReadingMtFn Reading-of-SUSD-Simplified-Jan2015-02)
  (causes-ThingSit (DrsCaseFn DRS-3630536469-157770) shut-off26323))
(holdsIn
  (FactsFromReadingMtFn Reading-of-SUSD-Simplified-Jan2015-02)
  (doneBy prevent15366 (DrsCaseFn DRS-3630536389-157750)))
(holdsIn
  (FactsFromReadingMtFn Reading-of-SUSD-Simplified-Jan2015-02)
  (doneBy prevent29988 (DrsCaseFn DRS-3630536483-157773)))
(holdsIn
```

```
  (FactsFromReadingMtFn Reading-of-SUSD-Simplified-Jan2015-02)
  (implies (DrsCaseFn DRS-3630536459-157764)
(DrsCaseFn DRS-3630536459-157765)))
(holdsIn
  (FactsFromReadingMtFn Reading-of-SUSD-Simplified-Jan2015-02)
  (implies-DrsDrs (DrsCaseFn DRS-3630536469-157768)
(DrsCaseFn DRS-3630536469-157769)))
(holdsIn
  (FactsFromReadingMtFn Reading-of-SUSD-Simplified-Jan2015-02)
  (in-UnderspecifiedContainer water15524 pipe15803))
(holdsIn
  (FactsFromReadingMtFn Reading-of-SUSD-Simplified-Jan2015-02)
  (isa be24024 ComparisonEvent))
(holdsIn
  (FactsFromReadingMtFn Reading-of-SUSD-Simplified-Jan2015-02)
  (isa close15268 ClosingAContainerArtifact))
(holdsIn
  (FactsFromReadingMtFn Reading-of-SUSD-Simplified-Jan2015-02)
  (isa contain23805 ContainingSomething))
(holdsIn
  (FactsFromReadingMtFn Reading-of-SUSD-Simplified-Jan2015-02)
  (isa control-device24203 ControlDevice))
(holdsIn
  (FactsFromReadingMtFn Reading-of-SUSD-Simplified-Jan2015-02)
  (isa cooling-process30191 CoolingProcess))
```

```
(holdsIn
  (FactsFromReadingMtFn Reading-of-SUSD-Simplified-Jan2015-02)
  (isa flow15431 FluidFlow-Translation))
(holdsIn
  (FactsFromReadingMtFn Reading-of-SUSD-Simplified-Jan2015-02)
  (isa group-of-pump27378 Set-Mathematical))
(holdsIn
  (FactsFromReadingMtFn Reading-of-SUSD-Simplified-Jan2015-02)
  (isa heat-storage23789 HeatStorage))
(holdsIn
  (FactsFromReadingMtFn Reading-of-SUSD-Simplified-Jan2015-02)
  (isa heat23844 ThermalEnergy))
(holdsIn
  (FactsFromReadingMtFn Reading-of-SUSD-Simplified-Jan2015-02)
  (isa leak15872 ExitingAContainer))
(holdsIn
  (FactsFromReadingMtFn Reading-of-SUSD-Simplified-Jan2015-02)
  (isa pipe15803 Pipe-SmokingDevice))
(holdsIn
  (FactsFromReadingMtFn Reading-of-SUSD-Simplified-Jan2015-02)
  (isa prevent15366 (PreventingFn flow15431)))
(holdsIn
  (FactsFromReadingMtFn Reading-of-SUSD-Simplified-Jan2015-02)
  (isa prevent15366 PreventingFromHappening))
(holdsIn
```

```
  (FactsFromReadingMtFn Reading-of-SUSD-Simplified-Jan2015-02)
  (isa prevent29988 (PreventingFn cooling-process30191)))
(holdsIn
  (FactsFromReadingMtFn Reading-of-SUSD-Simplified-Jan2015-02)
  (isa prevent29988 PreventingFromHappening))
(holdsIn
  DiscourseInterpretation-01
  (isa rainwater-collection-system5407 RainwaterCollectionSystem))
(holdsIn
  DiscourseInterpretation-01
  (isa shut-off26323 ShuttingOffSomething))
(holdsIn
  DiscourseInterpretation-01
  (isa solar-collector23314 SolarPanel))
(holdsIn
  DiscourseInterpretation-01
  (isa store15498 StoringSomething))
(holdsIn
  DiscourseInterpretation-01
  (isa valve15262 Valve))
(holdsIn
  DiscourseInterpretation-01
  (isa water15524 Water))
(holdsIn
  DiscourseInterpretation-01
```

```
  (objectActedOn store15498 water15524))
(holdsIn
  DiscourseInterpretation-01
  (objectMoving leak15872 water15524))
(holdsIn
  DiscourseInterpretation-01
  (objectOfStateChange close15268 valve15262))
(holdsIn
  DiscourseInterpretation-01
  (occursDuring close15268 rainwater-collection-system5407))
(holdsIn
  DiscourseInterpretation-01
  (possessiveRelation heat-storage23789 cooling-process30191))
(holdsIn
  DiscourseInterpretation-01
  (possessiveRelation water15524 flow15431))
(holdsIn
  DiscourseInterpretation-01
  (prevents-SitSit (DrsCaseFn DRS-3630536389-157750) flow15431))
(holdsIn
  DiscourseInterpretation-01
  (prevents-SitSit (DrsCaseFn DRS-3630536483-157773)
cooling-process30191))
(holdsIn
  DiscourseInterpretation-01
```

```
(primaryObjectMoving flow15431 water15524))
```

E. ADA Evaluation Questions

Passage/Q		Neither	Just Analogy	Just Background	Both	Question
Sound - 01	Ball-Sound-Original	0	0	0	1	When a sound hits a pillow, does it bounce back?
Sound - 02	Ball-Sound-Original	0	0	0	0	When a sound hits a sidewalk, does it bounce back?
SUSD01 - 01	Gold-Mine-Solar-Original	1	1	1	1	What happens to the solar heating system if the value of electricity falls?
SUSD01 - 02	Gold-Mine-Solar-Original	0	0	0	1	What happens to the man with the solar heating system if the value of electricity falls?
SUSD01 - 03	Gold-Mine-Solar-Original	0	1	0	1	Is acquiring solar energy beneficial if solar collectors cost much?
SUSD02 - 01	Bucket-Brick-Original	0	0	0	1	The disappearance of what causes the heat to stop exiting the brick?
SUSD0601 - 01	Water-Storage-Heat-Storage-Original	0	0	0	1	Decreasing what causes a brick to be able to hold less heat?
SUSD0602 - 01	Phase-Change-Materials-	0	0	0	0	At the wax's melting point what happens to it's temperature?

	Original					
SUSD07 - 01	Faucet-Furnace-Original	0	0	0	1	What marks the target temperature of the house?
SUSD08 - 01	Leak-Insulation-Original	0	1	0	1	What reduces the loss of heat from the house?
SUSD10 - 01	Rain-Sunlight-Original	0	0	0	0	If the sunlight falls straight down how does the solar collector lie?
SUSD10 - 02	Rain-Sunlight-Original	0	0	0	1	What part of the solar collector points towards the sunlight?
SUSD11-01 - 01	Stagnation-Stagnation-Original	0	0	0	1	What fills to cause the heat to leak quickly in a solar collector?
SUSD11-01 - 02	Stagnation-Stagnation-Original	0	1	0	1	What part of a solar panel is like the tray in the rain collection system?
SUSD11-02 - 01	Intensity-Rain-Sunlight-Original	0	1	0	1	What situation will cause a low stagnation temperature to balance the intake of solar radiation?
gbw08 - 01		0	0	0	1	What traps the thermal energy of the planet?
gbw08 - 02		0	0	0	0	What covers the planet?

Atmosphere3031 - 01	Atmosphere-Greenhouse-Original	0	1	0	1	What part of the atmosphere is like the glass of a greenhouse?
Atmosphere3031 - 02	Atmosphere-Greenhouse-Original	0	0	0	1	What is the temperature outside the atmosphere like?
Atmosphere3031 - 03	Atmosphere-Greenhouse-Original	0	1	0	1	In the atmosphere, what is like the vegetation of the greenhouse?
Cell - 01		0	1	0	1	Food in a cell is like what in a power plant?
Cell - 02		0	0	0	1	What is used to power the cell?
Total		1	8	1	18	
Score		0.045	0.364	0.045	0.818	