Sepia: Semantic Parsing for Named Entities

by

Gregory A. Marton

Submitted to the Department of Electrical Engineering and Computer Science on August 29, 2003, in partial fulfillment of the requirements for the degree of Master of Science

Abstract

People's names, dates, locations, organizations, and various numeric expressions, collectively called Named Entities, are used to convey specific meanings to humans in the same way that identifiers and constants convey meaning to a computer language interpreter. Natural Language Question Answering can benefit from understanding the meaning of these expressions because answers in a text are often phrased differently from questions and from each other. For example, "9/11" might mean the same as "September 11th" and "Mayor Rudy Giuliani" might be the same person as "Rudolph Giuliani".

Sepia, the system presented here, uses a lexicon of lambda expressions and a mildly context-sensitive parser to create a data structure for each named entity. The parser and grammar design are inspired by Combinatory Categorial Grammar. The data structures are designed to capture semantic dependencies using common syntactic forms. Sepia differs from other natural language parsers in that it does not use a pipeline architecture. As yet there is no statistical component in the architecture.

To evaluate Sepia, I use examples to illustrate its qualitative differences from other named entity systems, I measure component performance on Automatic Content Extraction (ACE) competition held-out training data, and I assess end-to-end performance in the Infolab's TREC-12 Question Answering competition entry. Sepia will compete in the ACE Entity Detection and Tracking track at the end of September.

Thesis Supervisor: Boris Katz
Title: Principal Research Scientist
Acknowledgments

This thesis marks the beginning of a beautiful road, and I have many people to thank for what lies ahead.

- To Boris Katz for his guidance and patience over the past two years,
- To Jimmy Lin, Matthew Bilotti, Daniel Loreto, Stefanie Tellex, and Aaron Fernandes for their help in TREC evaluation, in vetting the system early on, and in inspiring great thoughts.
- To Sue Felshin and Jacob Eisenstein for their thorough reading and their help with writing this and other Sepia documents,
- To Deniz Yuret for giving me a problem I could sink my teeth into, and the opportunity to pursue it,
- To Philip Resnik, Bonnie Dorr, Norbert Hornstein, and Tapas Kanungo for many lessons in natural language, in the scientific process, and in life,
- To my mother and father for a life of science and for everything else,
- To all of my family for their support, encouragement, prayers, and confidence,
- To Michael J.T. O’Kelly and Janet Chuang for their warm friendship,

To each of you, and to all of those close to me, I extend my most heartfelt thanks. I could not have completed this thesis but for you.

Dedication

To my nephew Jonah; may the lad make the best use of his Primary Linguistic Data!
Contents

1 Introduction .......................................................... 10
  1.1 Named Entities and Question Answering .................... 10
  1.2 Motivation ....................................................... 12
  1.3 Approach .......................................................... 14
  1.4 Evaluation .......................................................... 15
  1.5 Contributions ....................................................... 15

2 Related Work .......................................................... 17
  2.1 The role of named entities in question answering ............ 17
  2.2 Named Entity Identification ...................................... 19
  2.3 Information Extraction Systems .................................. 22
  2.4 Semantic parsing ..................................................... 24
  2.5 Automatic Content Extraction ..................................... 27

3 Sepia's Design .......................................................... 29
  3.1 No Pipeline ......................................................... 30
  3.2 Working Example ................................................... 32
  3.3 Lexicon Design ...................................................... 35
    3.3.1 Separation of Language from Parsing ......................... 35
    3.3.2 Using JScheme for Semantics and as an Interface to the World 36
    3.3.3 Scheme Expressions vs. Feature Structures .................. 37
    3.3.4 Pruning Failed Partial Parses ............................... 37
    3.3.5 Multiple Entries with a Single Semantics ................... 38

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.6</td>
<td>String Entries vs. Headwords</td>
<td>38</td>
</tr>
<tr>
<td>3.3.7</td>
<td>Multiword Entries</td>
<td>39</td>
</tr>
<tr>
<td>3.3.8</td>
<td>Pattern Entries</td>
<td>39</td>
</tr>
<tr>
<td>3.3.9</td>
<td>Category Promotion</td>
<td>40</td>
</tr>
<tr>
<td>3.3.10</td>
<td>Wildcard Matching Category</td>
<td>40</td>
</tr>
<tr>
<td>3.3.11</td>
<td>Complex Logical Form Data Structures</td>
<td>42</td>
</tr>
<tr>
<td>3.3.12</td>
<td>Execution Environment</td>
<td>43</td>
</tr>
<tr>
<td>3.3.13</td>
<td>Stochastic Parsing</td>
<td>43</td>
</tr>
<tr>
<td>3.4</td>
<td>The Parsing Process</td>
<td>44</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Tokenization</td>
<td>44</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Multitoken Lexical Entries with Prefix Hashes</td>
<td>45</td>
</tr>
<tr>
<td>3.4.3</td>
<td>Initial Lexical Assignment</td>
<td>45</td>
</tr>
<tr>
<td>3.4.4</td>
<td>Parsing</td>
<td>45</td>
</tr>
<tr>
<td>3.4.5</td>
<td>Final Decision Among Partial Parses</td>
<td>46</td>
</tr>
<tr>
<td>3.5</td>
<td>Design for Error</td>
<td>46</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Underlying Data Type of Arguments is Invisible</td>
<td>46</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Category Promotion Can Cause Infinite Loop</td>
<td>47</td>
</tr>
<tr>
<td>3.5.3</td>
<td>Fault Tolerance</td>
<td>47</td>
</tr>
<tr>
<td>3.5.4</td>
<td>Testplans</td>
<td>47</td>
</tr>
<tr>
<td>3.5.5</td>
<td>Short Semantics</td>
<td>47</td>
</tr>
<tr>
<td>4</td>
<td>Illustrative Examples</td>
<td>49</td>
</tr>
<tr>
<td>4.1</td>
<td>Overview of the Lexicon</td>
<td>50</td>
</tr>
<tr>
<td>4.2</td>
<td>Mechanics</td>
<td>52</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Function Application</td>
<td>52</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Context Sensitivity</td>
<td>54</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Extent Ambiguity</td>
<td>56</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Nested Entities</td>
<td>58</td>
</tr>
<tr>
<td>4.2.5</td>
<td>Type Ambiguity</td>
<td>61</td>
</tr>
<tr>
<td>4.2.6</td>
<td>Coordination</td>
<td>62</td>
</tr>
</tbody>
</table>
4.3 Challenges ....................................... 64
  4.3.1 A Hundred and Ninety One Revisited ............ 64
  4.3.2 Overgeneration of Coordination .................... 65
  4.3.3 Coordinated Nominals .......................... 66
  4.3.4 Noncontiguous Constituents ........................ 67
  4.3.5 Mention Detection and Coreference ............... 67
  4.3.6 Real Ambiguity .................................. 68
  4.3.7 Domain Recognition and an Adaptive Lexicon .... 69

5 Evaluation on Question Answering ......................... 70
  5.1 Named Entities in Queries .......................... 71
    5.1.1 Experimental Protocol .......................... 71
    5.1.2 Results ....................................... 72
    5.1.3 Discussion .................................... 77
  5.2 Integration and End-to-End Evaluation ................. 78
    5.2.1 Query Expansion ................................ 79
    5.2.2 List Questions .................................. 80

6 Future Work ......................................... 82
  6.1 ACE ............................................. 82
  6.2 Formalism ........................................ 83
  6.3 Extending Sepia: Bigger Concepts and Smaller Tokens .. 84
  6.4 Integration with Vision, Common Sense ............... 84

7 Contributions .......................................... 85

A Test Cases ........................................... 87
  A.1 Person ............................................ 88
  A.2 Locations (including GPEs) ........................ 91
  A.3 Organizations .................................... 93
  A.4 Restrictive Clauses .............................. 95
  A.5 Numbers ......................................... 96
List of Figures

3-1 An example and a lexicon. ........................................ 33
3-2 The verb "slept" requires an animate subject. ............... 36
3-3 A category of the title King, as it might try to recognize "King of Spain". Note that arguments are left-associative. ............ 36

4-1 Particles that appear in mid-name. The eight separate rules take 11 lines of code, and result in four tokens specified. ............... 51
4-2 The lexicon required to parse "a hundred and ninety-one days" . . . 53
4-3 A lexicon to capture Title-case words between quotes ............. 55
4-4 A lexicon to capture nested parentheses ....................... 56
4-5 Categorial Grammar is context sensitive with composition. ..... 57
4-6 Lexicon for "I gave Mary Jane Frank's address" ................ 59
4-7 Entities can have multiple categories. If one type is used in a larger context, it will be preferred. ............................... 62
List of Tables

4.1 Distribution of fixed lists in the lexicon .................... 50
4.2 Breakdown of single-token, multi-word, and multi-subword lexical entry keys. ............................................. 51
Chapter 1

Introduction

People's names, dates, locations, organizations, and various numeric expressions, collectively called Named Entities, are used to convey specific meanings to humans in the same way that identifiers and constants convey meaning to a computer language interpreter. Natural Language Question Answering can also benefit from understanding the meaning of these expressions because answers in a text are often phrased differently from questions and from each other. For example, "9/11" might mean the same as "September 11th" and "Mayor Rudy Giuliani" might be the same person as "Rudolph Giuliani".

Sepia\textsuperscript{1}, the system presented here, uses a lexicon of lambda expressions and a mildly context-sensitive parser to create a data structure for each named entity. Each lexical item carries a Scheme lambda expression, and a corresponding signature giving its type and the types of its arguments (if any). These are stored in a lattice, and combined using simple function application. Thus Sepia is a full parser, but lexicon development has been geared toward the named entity understanding task.

1.1 Named Entities and Question Answering

There are several applications for better named entity understanding within natural language question answering, the Infolab's primary focus. Question answering is

\textsuperscript{1}SEmantic Processing and Interpretation Architecture
similar to information retrieval (IR), but where IR evokes images of Google, with lists of documents given in response to a set of keywords, question answering (QA) tries to give concise answers to naturally phrased questions.

Question answering research has focused on a number of question types including

**factoid questions** - simple factual questions with named entity answers:

Q: "When did Hawaii become a state?"
A: *August 21, 1959*

**definition questions** - questions with essentially one query term (often a named entity) to be defined:

Q: "What is Francis Scott Key famous for?"
A: *On Sept. 14, 1814, Francis Scott Key wrote “The Star-Spangled Banner” after witnessing the British bombardment of Fort McHenry in Maryland.*

**list questions** - questions asking for lists, often of named entities:

Q: "List musical compositions by Aaron Copland."
A: *Fanfare for the Common Man, Appalachian Spring, Billy the Kid and Rodeo*

possibly based on a sentence like:

American composer Aaron Copland (1900-1990) experienced several distinct musical phases during his fruitful career, but is best known for his compositions for Fanfare for the Common Man, Appalachian Spring, Billy the Kid and Rodeo
1.2 Motivation

Naturally people are most interested in things that have names, associated dates, and measures, so named entities are ubiquitous in people's questions and their answers, as shown in the examples above. The first step towards making use of these relatively closed-form language phenomena was simply finding bounds or extent (start and end) and one of a few gross categories for each identified named entity. For the Hawaii example above, once a question answering system knows to look for a date, it can search for entities categorized as dates in the text, just as if they were keywords. This is indeed a good start, but for several reasons is not enough.

One reason is that information must often be combined from several sources to answer a question. This is one of the stated goals of ARDA's AQUAINT Question Answering Program for the next two years. As an example, consider the question, "How many years passed between Lincoln's and Kennedy's assassinations?" It may be relatively easy to find the dates of Lincoln's and Kennedy's assassinations in a corpus, but there may be no simple sentence that tells the difference in years between the two. In this case we have to understand the points in time that the dates refer to, and be able to subtract them.

Another reason is that it is often important to know when two names refer to the same thing. Another stated goal of AQUAINT is to better "fuse, evaluate, and summarize" the information that answers the user's question. In the Hawaii example, it is better to present the one solid answer than to present "August of 1959", "1959", and "August 21, 1959" as three separate answers, each with nearly one-third confidence. To go beyond dates, many heuristics are useful for names as well, and it is possible to guess that "the Francis Scott Key Memorial Bridge", "Baltimore's Key Bridge", and "the F.S. Key Bridge" could very well refer to the same structure and that sentences about that bridge would not likely describe their namesake. It may be more difficult to decide heuristically that "Aaron Copeland" is very likely to be a misspelling of "Aaron Copland", but such knowledge would be useful in constructing a list of American composers!
Third, understanding a named entity's internal structure and its immediate context can help in finding its boundaries as well. Consider the sentence,

Like Gould, Pratt has become very influential among young musicians, says Oppens, the John Evans Distinguished Professor of Music at Northwestern University.²

The part "John Evans ... University" is a description of Oppens, and it would be a mistake to retrieve this sentence in search of information about Evans. Moreover, any further parsing would get confused if Evans were not known to modify Professor. The syntactic pattern "<person> Distinguished Professor of <subject> at <university>" is thus a valuable structure to teach the lexicon. To relate it to Oppens, we would also need to understand apposition, which is beyond the immediate scope of this thesis, but is a rich source of future work within the Sepia architecture.

Less common but less benign examples that whet the imagination include understanding that "two to three hundred miles" is a single measure, that "Ronald and Nancy Reagan" probably refers in part to Ronald Reagan, that "scores of a hundred and ninety-nine respectively" does not include the number 199, and that there are two separate people in the sentence, "I told Howard Dean would win."

Finally, the salient motivation for this work is to play with precisely the kind of difficult phrases discussed above. Since taking my first semantics class, I have wanted a system on which to experiment with natural language semantics as if they were computer programs, with a compiler to tell me what combinations worked and what didn't. Sepia is exactly such a compiler, and named entities are a rich, well explored, and somewhat restricted test domain. In generating a competitive broad coverage named entity understanding package that will be useful to a larger question answering system, I will thus also have the opportunity to explore lexical semantics from an implementer's point of view.

In Chapter 2, I review related work and show support for these motivations.

²AQUAINT document APW20000329.0104
1.3 Approach

Sepia consists of a mildly context-sensitive parser partially implementing Steedman's Combinatory Categorial Grammar [59] formalism, and a lexicon geared towards named entity understanding. A lexical entry consists of a string or regular expression, an associated Scheme lambda expression representing its meaning, and a function signature, called a category, to guide the parser in semantic application.

The Scheme expressions are evaluated as part of the parsing process, and can influence that process in a variety of ways. As they are combined using simple function application (Scheme apply), their primary goal is to compositionally build a data structure representing an understanding of the text they refer to. Secondly, if a function application fails or returns false, the resulting partial parse is pruned. Finally, the lambda expressions in the lexicon in JScheme have full access to the parser's Java runtime environment\(^3\), so they can change or add tokens, modify function signatures (categories), and temporarily change the lexicon. With this special access, the semantics of a particular lexical item can accomplish arbitrary, perhaps linguistically motivated structural changes in parsing, for example corrections or reanalysis.

Perhaps the most important decision in Sepia's implementation is to forgo a pipelined architecture. Rather than having separate modules (such as sentence breaking, tokenization, part-of-speech tagging, etc.) arranged in a pipeline with each module feeding its output to the next, Sepia starts with unannotated text, assigns interpretations where possible directly from the lexicon, and uses simple function application rules to combine those interpretations into larger structures. There is only one stage of processing, so it cannot commit to an incorrect interpretation "early on in the process." In the Dean example above, both "Howard Dean" as a unit and "Howard" and "Dean" separately would continue to be considered until the end of parsing. At that point the named entity constituents of the largest partial parses are reported.

In Chapter 3, I describe the system design and implementation decisions in more

---

\(^3\) a shared environment in the programming language sense—the JScheme language provides access to all the classes and methods available in Java. This is not an instance of safe programming: the data, in the form of lexical items, are specifically intended to execute as code.
1.4 Evaluation

The principal challenge in Sepia, as in any knowledge-based system, is in building a lexicon. This manual task is widely noted as difficult, time-consuming, and slow to adapt to a new language or domain. Yet knowledge engineering still has a large role in most natural language tasks. I hope to show throughout the evaluation that Sepia is an intuitive and effective tool for knowledge engineering.

In Chapter 4, I will walk through a set of illustrative examples to showcase strengths and weaknesses of the approach, and discuss some difficult problems in named entity semantics. The examples will show that the lexicon language is both simple and powerful.

In Chapter 5, I show an intrinsic evaluation on named entity understanding against a set of guidelines I first had in mind when creating Sepia, and also against the much more stringent ACE guidelines. I go on to describe two ways that Sepia was actually integrated into two components of the Infolab's TREC-12 / AQUAINT Question Answering competition entry, and the results of extrinsic evaluation of Sepia via the end-to-end system.

1.5 Contributions

In this thesis I present Sepia, a non-pipelined parser for compositional semantics in natural language, and a lexicon geared toward understanding named entities. This is a fully implemented parser that realizes the Combinatory Categorial Grammar formalism in a new way. The principal distinction from other parsers is that the meaning or semantics of each element in the lexicon is not a simple data structure, but a small program in JScheme, a functional programming language. The parser can thus be entirely language-independent and task-independent. This lexicon format is intuitive and powerful, allowing lexicon developers to write rules that compositionally...
assemble complex data structures to represent language.

I focus my lexicon building effort in named entity understanding because I believe it is a task in natural language that can greatly benefit from incorporating bottom-up compositional information with top-down constraints on the semantics of the entity that best fits into the larger sentence. Most named entity understanding systems preclude top-down information from benefiting named entity understanding because they look for named entities as a separate and prior step to parsing larger structures. In Sepia I show one way to avoid the pipelining problem by integrating all processing stages.

Finally, with Sepia, I make available to the Infolab a semantic named entity understanding system to be integrated into larger question answering systems. The system has been integrated with two question answering components, and I discuss lessons learned from those integrations.
Chapter 2

Related Work

In exploring the related literature, I will first relate Named Entity identification to Question Answering. Second, I will review the late 1990’s Message Understanding Conference (MUC) evaluations of named entity identification, and the related Information Extraction (IE) literature. Third, I will review semantic parsing strategies and systems, and motivate Sepia’s design decisions as a content extraction platform. Finally, I will describe a new hybrid between broad coverage named entity identification and information extraction, called Automatic Content Extraction (ACE), and summarize some goals that Sepia must address in order to do well in the ACE task.

2.1 The role of named entities in question answering

The Text REtrieval Conference (TREC) Question Answering (QA) Track [63, 62] provides a forum for evaluating and comparing question answering systems. TREC systems try to find answers to factoid and definition questions, and more recently to list questions, in a large corpus of English language newspaper articles. The crucial role that named entity identification can play has been recognized from the outset of the competition. Srihari [58] et al. of Cymfony Co., winners of the TREC-8 (1998) QA track, found that 80% of the 200 questions in the track asked for a named entity
response (a person, location, etc.).

More recently LCC, the winner of the TREC-2002 QA competition, performed a detailed error analysis [53] in which they estimated that without named entity identification, their system would lose 68% of its baseline performance, while their named entity identifier only caused 3.1% of their total error due to recall and 4.9% due to precision. Their largest source of error was in question analysis, where their system failed to identify the correct category of the required answer.

The conclusion that named entity identification technology is good enough is misleading. According to their analysis 67.5% of questions in the TREC-2002 QA question set required simple factual answers found within the corpus. 2002 was in fact the first year that not all questions were of that nature: an additional 27.9% of questions required simple reasoning. It is not possible to answer “Is Qatar bigger than Rhode Island” if we only know that “1045 square miles” and “11,437 sq. km.” are areas. The existing technology may be good enough for the simpler task, but not for upcoming tasks that require inference.

Both of these successful QA systems used some semantics in their named entity identification, and had many more identification classes than the MUC or ACE evaluations (see Sections 2.2 and 2.5) require. Cymfony’s finite-state Textract system went as far as building “a mini-CV” for person entities, recording name, subtype (religious person, military person, etc.), gender, age, affiliation, position, birth time and place, spouse, parents, children, origin, address, phone, fax, email, and a generic “descriptors” field. In the coarser-grained MUC-7 named entity task, Textract performed at 91.24% F-measure.

It is not clear that named entity identification is even always crucial to an application. Semagix, Inc., has put no small effort into creating a regular-expression-based semantic suite of tools for their content management application [32], for example. But when they compared their named entity identifier, which performed at 76% F-measure, to a manually generated perfect named entity identification for a particular set of content needs, the end-to-end performance did not change [34]!

Sepia had similar results in end-to-end evaluation in question answering, as pre-
sented in Section 5.2. and the lesson to be learned is that how one uses a named entity understanding system is as important as how well it performs.

2.2 Named Entity Identification

The Sixth and Seventh Message Understanding Conferences (MUC) [33] evaluated named entity [14] identification and information extraction [13] tasks in 1997 and 1998. Named entities to be marked included the categories: organization, person, location, date, time, currency, and percentage. It could be said that some semantics were required. Metonymy was the most significant embodiment of semantics: artifacts (buildings, desks, copies of a newspaper) were not to be marked, so that The White House should be marked as an organization when it announces something, but not when someone takes a tour in it; “The Wall Street Journal” was not to be marked if it referred to a copy of the newspaper, but was markable when referring to the organization. Groups (e.g., Republicans), and non-currency or percentage numbers were also not to be marked. These are indeed semantic distinctions, but there were so few instances of these distinctions that it is not clear that semantics were being meaningfully evaluated. Some systems which simply picked the most common case performed very well. Furthermore, the categories are very coarse, and of little use in inference. In results, the clear winners were statistical systems, performing near human levels, and scores got monotonically poorer as systems became more knowledge intensive.

In the previous section, I motivated the need for the harder, knowledge-based approach, namely that semantics are required for inference in question answering. The following describes both the knowledge-based and statistical systems in MUC and outlines lessons to be learned for Sepia.

On the purely statistical end are three maximum entropy models. Edinburgh LTG’s system [50], NYU’s MENE system [11] and Kent Ridge Labs’ system [67] performed at 93, 92, and 78 percent F-measure, respectively. These systems identified bounds and picked a type for each entity, but gave no further semantic information.
about the entity's contents. The states and transitions in a finite state machine can be interpreted as a set of rules much more easily than the maximum entropy models above can, but no effort was made to assign rule-like interpretations to Nymble, BBN's Hidden Markov Model (HMM) system[52]. That system performed very well across languages, with case and punctuation removed, and with only 50,000 words of training data [7, 8, 51].

The lesson to be learned is that statistical approaches are robust to many forms of noise and unseen entities, are easy to train, and should be used as (at least) a guiding force for any modern named entity identifier. I have eschewed statistical methods in Sepia partly because it is clear that they can only help (and so are part of future work!), and partly to make a scientific point about how well a purely knowledge-based system can be expected to work.

Pattern matching was another popular paradigm, and the Infolab's Blitz system [42], developed around the same time, took a pattern matching approach. Pattern matching has the advantage that it is relatively easy to extract somewhat more semantics (salutation, first name, last name; value of a unit of measure, its units, and its normalized form). The best of these, NetOwl [44], achieved 91.6% F-measure on MUC-7. In MUC-6 (where results were slightly higher), the FASTUS [2, 38] system achieved 94% F-measure. In the FASTUS error analysis, however, Appelt et al. pointed out that they had no systematic approach to separating the various roles of named entities—whether they were artifacts or not. If failing to distinguish roles only accounted for part of a six percent loss in MUC-6 performance, then one must wonder how much the semantic capacities of any of the systems were really tested in either evaluation.

Both systems boasted rapid development, so perhaps the lesson from their good performance is that knowledge is the most useful when it is easy to enter, easy to understand, and easy to test.

Dekang Lin's approach [46] was unique and achieved the best result of the more semantic systems. He used his minimalist parser, MINIPAR [47], to find dependency parses of the text to be annotated. He then compiled "collocation" statistics—counts
of participation in relation triples for each word within the corpus being annotated—to make a decision with a naive Bayes classifier. This approach yielded 86.37% F-measure on named entity identification. In contrast, the LOLITA [28] system depended greatly on fully parsing sentences, building a logical representation for them, and doing inference. It proved to be brittle, with only 76.43% F-measure. The lesson, found elsewhere in the literature as well, is that being robust to partial parses leads to better performance than requiring full parses in many sentences.

The FACILE [9] system had, in many ways, a similar inspiration to Sepia, but fared badly, at only 82.5% F-measure. FACILE used a context-sensitive grammar with regular-expression-style operators to compositionally combine attribute vectors. The failure was most likely in the opacity of the notation. Aside from heavy use of abbreviation in the grammar rules, a list of attributes for a constituent proved insufficient (from their description) for coreference resolution and template building. The parser was expanded with hooks to let the features influence subsequent parsing, but that strategy separates the resolution action from the part of the grammar indicating resolution, and is thus harder to understand. As if to prove the point, the system was under development for 20 months before its first deployment in this setting.

The LaSIE-II [37] system attained 85.83% officially, but for the IE tasks, its authors added a small amount of knowledge, boosting named entity performance to 90%. LaSIE was built on the General Architecture for Text Engineering (GATE), a modular pipelined architecture that makes it easy to do large-scale software engineering right. In order, the modules included a tokenizer, gazetteer, sentence breaker, part-of-speech tagger, POS-based morphology analyzer, a ten-stage named entity CFG parser, in one step a phrase structure CFG parser and Quasi-Logical Form generator, a coreference analyzer, a discourse interpreter, and a domain-specific template writer. By modularizing the task in this way, the Sheffield team was able to use others' work, such as the Brill part of speech tagger, and efficiently divide resources.

The modularity blessing of GATE is also its curse. Note the position of the gazetteer before the sentence breaker or part of speech tagger. In the previous year's competition, there were sentence boundary and part-of-speech errors on words in the
gazetteer, but because the sentence had already been split, the gazetteer could not
recover. As reordered for the following year (as listed above), the sentence breaker
and part-of-speech tagger treat gazetteer matches as opaque tokens. The gazetteer
might thus recognize a name which should have been overridden by an actual sentence
boundary, but this is less common. Similarly, in the ten-stage named entity grammar,
it was impossible to correctly identify both “Julian Hill” as a person and “Pearl
Harbor” as a place at the same time. If known first names stage ran first, both would
be marked as person; if the location cue stage ran first, both would be locations. In
their words, they lacked a “controlled propagation of ambiguity”. That propagation is
unbounded in Sepia—both possibilities are allowed—and I will rely on the semantics
of the context to determine which entity is of which type.

The Message Understanding Conferences stopped partly because some systems
were achieving near-human performance at the task, and partly because the task was
so far from the much more useful task of understanding the meanings of the named
entities identified. Many participants chose to pursue the more semantic Information
Extraction task, described in the next section, and following that, the two have been
fused into a new named entity understanding task focused on meaning: the Automatic
Content Extraction conference.

2.3 Information Extraction Systems

Information Extraction (IE) is the task of using a natural language (most often news)
corpus to fill a predetermined template appropriate to the content of the document.
This is a domain specific task that was also addressed in the Message Understanding
Conference (MUC) competitions. The general consensus for IE is that knowledge
engineering approaches have significant advantages over pure machine learning ap-
proaches, but new learning approaches may be gaining ground.

In MUC-6, the CIRCUS [27] team described the shortcomings of their automatic
dictionary acquisition tool with respect to the Template Extraction task:

Unfortunately, CRYSTAL’s CN [Concept Node] definitions offer little help
with noun phrase analysis, since they operate at a relatively coarse level of granularity with respect to a complex noun phrase. The CNs that CRYSTAL induces are designed to locate relevant noun phrases for an extraction task, but they do not help us understand where to look inside a given noun phrase for the relevant information.

This indicates that sentence-level features can disambiguate the boundaries of the targets needed for information extraction, but they must still have access to the contents of those noun phrases for successful information extraction. Sepia aims to combine both the sentence level and named entity level understanding into a single system so the two levels can help disambiguate each other.

In integrating the two levels, one must keep in mind that the performance of separately evaluated systems may not reflect the performance of the system as a whole. In reviewing FASTUS and another system, TextPro, during an information extraction tutorial [3], Douglas Appelt and David Israel voiced two familiar morals, one advocating partial parsing, the other end-to-end integration of components. Full parsing, especially in new domains where training data is expensive, often breaks, so that an IE engine must take advantage of any partial analyses robustly. It matters not, they say, that full sentence analyses are broken or underspecified—that is not the greatest source of end-to-end error. The second moral emphasizes the point: improving the performance of any IE component, measured with tests of that component, does not necessarily translate into better end-to-end performance.

On the partial parsing angle, the LaSIE team also offered a lesson: in building the 150 named entity rules and 400 phrase-structure rules, they had initially bootstrapped from the Penn Treebank, automatically acquiring syntactic structures, then annotating them with semantics.

The resulting analyses were poor, and the effort of manually annotating the rules with features for semantic interpretation substantially reduced the benefits of the grab-and-run approach to grammar acquisition.

Instead, they ended up rebuilding the rules from scratch for this system, using an
iterative refinement approach on the MUC-7 training data. This sort of iterative refinement approach is what we are using in preparing Sepia for the Automatic Content Extraction competition next month.

Probabilistic methods were not competitive in information extraction when MUC-6 was held, but there is now one promising probabilistic IE system. The ALICE system [12] has shown promising results on the MUC-4 information extraction task, largely because it uses the Collins parser's grammatical relations as features. So at least on a particular domain, information extraction competitive with knowledge based methods might be usefully learned.

The lesson I take for Sepia from the initial success of the knowledge-based approaches and this new hope for a statistical approach is that the knowledge based approaches serve to locate the most salient features for a learning system to use, and the most useful data structure for it to build. My goal is to build Sepia with a flexible data structure that will serve as a new vehicle for machine learning algorithms to identify relevant semantics. In Section 3.3.13, I discuss how Sepia might be immediately augmented with statistics for an improved hybrid approach.

Yorick Wilks argues that the TREC Question Answering track is the one to watch in the debate between traditional AI approaches to Information Extraction and statistical language modelling approaches [65], in part because indexing by template might be a good way to answer questions about common events in the corpus. The Question Answering track may also be where the two ideologies can most easily work together to find the most likely meaningful answer to a question.

\section*{2.4 Semantic parsing}

In the majority of parsing systems, any attempt at semantics takes the form of unification over a vector of attribute-value pairs [55, 19, 5]. Unification in this case means bringing all attributes of both attribute vectors into a new encompassing attribute vector. Unification \textit{fails} if there are incompatible attribute values. Just what values are compatible is up to the linguistic theory they implement, and it takes work to
make the various unification-based grammars compatible so that they can be directly compared [6].

One instance of a unification-based robust parser is the prolog-implemented Gemini parser [23]. Like Sepia, Gemini uses the same formalism throughout its processing and interleaves its semantics into the logical forms as partial parses are built during processing, dropping partial parses that do not create a new well-formed logical form. The logical forms built in Gemini were originally used to recognize spoken language utterances in a flight corpus, but has been expanded with 50,000 new lexical items to become part of the QUARK multi-document multi-media question answering system [64].

A unification model can support a lambda calculus: one or more attributes can be assigned lambda expressions as values, and “compatibility” might be defined as whether the two expressions can combine. Combinatory Categorial Grammar (CCG) is ideal for supporting just this sort of processing [59], but Grok [5], a CCG parser developed with Steedman at Edinburgh, does not, in fact, do that. Its logical forms are devoid of lambda calculus, again using only feature vectors.

Instead of using unification over typed feature vectors as a basic semantic structure, Sepia uses the CCG formalism with Scheme expressions for lexical semantics. The simple combination rule is now function application and the data structures associated with a lexical item are small pieces of Scheme code, thus forming a lambda calculus.

In CCG, the lexicon contains triples: a word form, an associated category that tells the parser what categories the entry can combine with, and a lambda expression for the meaning of the word form. Parsing is done by function application, composition, coordination, and type raising, each of which translate into simple transformations on the lambda expressions involved. CCG is mildly context-sensitive because it lets each lexical item specify a finite number of non-terminals to expect. For someone familiar with context free grammars, this is akin to being able to specify not simply a string of terminals and nonterminals on the right hand side, but instead a structured tree of them. Even though CCG is mildly context-sensitive, Jason Eisner [25] showed that
there is a normal-form which allows efficient parsing with the Earley algorithm [24]. Statistical parsing is also possible, with full semantics, in this framework [36], and Hockenmaier was able to bootstrap a statistical grammar from the Penn Treebank in LaSIE style [35]. CCG is also very good at accounting for linguistic data and their interpretations, especially coordination and gapping structures. Extensions such as Set-CCG [4] and ordered Set-CCG [39] can account for some difficult cross-linguistic variation. Thus the formalism is a very attractive one for semantic processing in general. By implementing the named entity semantics in this formalism, I hope to be able to interface with the best practices of higher level semantics and full statistical parsing at a later stage.

The use of thematic roles as semantics is also common [30, 26, 20]. Acquisition of semantic lexicons for determining thematic roles has been explored by Cindi Thompson [61] and Bonnie Dorr [22] (whose lexicon for English is available [21]). Parsers which label thematic roles include MINIPAR [47], and Gildea and Jurafsky's system [30]. I find this work inspirational, and useful in methods to automatically acquire a semantic lexicon, but its application to named entity identification is more distant. It will be interesting to see any examples where a thematic parse will cause the system to perform well or poorly. The benefit of using the CCG system is that it is easy to put in a small bit of thematic parsing and show its effect on named entity understanding when the need arises.

Finally, Lenat's Cyc [45] system is slowly coming into its own, with work by O'Hara [54] to map Framenet and Penn Treebank roles to Cyc structures, and the newly Free variant OpenCyc [17]. Commonsense projects such as Cyc and Open-Mind [60] could provide much needed additional data and a source of semantics for a parser like Sepia. Perhaps Sepia lexicons should be developed as an open source commodity!
2.5 Automatic Content Extraction

Most recently, NIST has launched the Automatic Content Extraction (ACE) [57] evaluations on Entity Detection and Tracking (EDT), and Relation Detection and Categorization (RDC). The tasks are very similar to the MUC tasks, though with finer control over metonymy, and more natural classes. EDT is what I have been calling named entity understanding, as opposed to the relatively shallow task of named entity identification, which requires only bounds and type. RDC goes a step further and requires recognition of relations between named entities, such as one being located at, having a structural (e.g., part of), or a role relation (e.g., founder) to another entity. ACE also requires that simple nominals and pronouns be tagged, not only name mentions. Overall, this is a much more semantic set of tasks.

LaSIE has jumped the GATE [16]1, and the resulting information extraction system, ANNIE [15], is publicly available and currently being adapted for the upcoming ACE competition [31]. It would be interesting to compare the performance of the hand-crafted rules in its MUSE named entity semantics module to the hand-crafted rules generated for Sepia. I expect that many of the rules would be similar, and the primary differences would be less attention to ordering in Sepia, and fewer special cases to fix bad rule orderings.

The GATE MUSE system reports accuracies in the 80% range on ACE. Other systems from MITRE, BBN, SRI, IBM, and NYU report earlier results with accuracies around 75% on the ACE site. This fall’s evaluation will be the first not marked as a “pilot” evaluation and thus if I understand correctly, the first with citable publications.

One of the major challenges in the ACE competition will be dealing with lack of case and inappropriate punctuation, especially in the broadcast news texts. One article in the training collection opens,

   grief engulfed as tree, a one day after a fire in a cable car filled can skiers
   killed at least 155 people. it happened inside a tunnel high in the austrian

1Their joke, not mine. But it is cute.
alps.

By the end of the second sentence we as humans were able to decipher “as tree, a” to mean “Austria”, but I still cannot guess what might have produced “filled can skiers”. The Austrian Alps are easier to restore because they are unique words despite capitalization, but several organizations, such as “NOW”, present a greater challenge.

That challenge can be called “truecasing”, and Lita et al. [48] present a statistical approach to truecasing arbitrary text from which they report an increase in ACE EDT mention detections\(^2\) on Automatic Speech Recognized broadcast news (e.g., the excerpt above) from 5% F-measure to 46% after truecasing, and a 26% improvement in F-measure overall in named entity understanding. I would thus like to incorporate some truecasing measure into Sepia as well for broadcast news, if time permits before the actual ACE competition in four weeks, because it would help easily overcome some of the constraints of noisy data.

---

\(^2\) detection of coreferring entities
Chapter 3

Sepia’s Design

The Sepia system consists of two primary parts: the parser and the lexicon. The parser is a partial implementation of a CCG parser, and aims to be language-independent, pushing all language-specific knowledge to the lexicon. The lexicon is not a simple list of words, but a set of strings or regular expressions associated with semantics in a lambda calculus (specifically, in JScheme).

Every semantics, or logical form, in the lexicon must have an associated function signature, or category, that guides the parser in applying functions to matching arguments. This is used, for example, to specify subcategorization information of verbs for their arguments, and of adjectives for the nouns they can modify. If a function application is attempted and fails (for example because of knowledge outside the lexicon), then no new combined constituent is constructed.

In this chapter, I will detail the design and implementation decisions embodied in Sepia, and hope to give a clear understanding of the lexicon format and of how the parsing mechanism uses the lexicon, so that examples in the upcoming sections will be fully understandable.

1or of the arguments for the verbs they may have various roles with. Whichever does the specifying the other has to agree that it matches the specification, so the which is the head and which is the modifier seems akin to deciding whether the Earth revolves around the Sun or vice versa, which is to say a matter of interpretation.
3.1 No Pipeline

The most common natural language processing architecture is a pipeline of modules where each has access only to the final decisions of the ones before it [49, 16]. One module will identify sentence boundaries, the next will tokenize, and so on. A mistake in an early module is usually irreparable by a later one. This means that there is no way to incorporate top-down understanding into the lowest level decisions.

The following are examples where a state-of-the-art named entity identifier would cause a parsing failure in a later module:

- "They scored a hundred and ninety-one respectively."\(^2\)
- "I told Harold Dean would win."
- "Mr. and Mrs. Hodson and their children..."

In the first case the word *respectively* can have no meaning, because the coordination it requires has been analyzed as part of a number. By having access to the intermediate candidates *a hundred* and *ninety-one* separately, Sepia can decide to use *and* as a coordination rather than part of a larger numeric expression.

In the second case, *told* cannot find a good argument structure. If Sepia had lexical entries for *told* and other verbs, the intermediate representation with *Harold* and *Dean* as separate names would be available to them as a possibility. Adding broader language coverage including verbs is a goal for future work.

In the third case, a later coreference module would not be able to associate a pronoun "their" with the Hodson couple because only one person is recognized. Sepia (with the current lexicon) sees *Mr.* and *Mrs.* as functions that take a person as an argument, and add title information to that person’s record. One meaning of *and* combines adjacent meanings that have like categories into a new meaning that has the same category. It will create a new one-place function that will map the

\(^2\) Parsing numeric expressions seems unpopular: it is not discussed in MUC, and ANNIE does not make the attempt beyond marking "hundred", "ninety", and "one" separately. Numbers of this sort do fall under the common definition of a named entity, and ACE requires "five astronauts" to be marked, but does not expand on the requirement.
two underlying one-place functions over their argument. The result will be a data structure that contains two people with a name Hodson, one of which has a title Mr., and the other has a title, Mrs..

Controlled propagation of ambiguity starts at the lowest levels. Sepia does not expect any form of preprocessing before it sees a text. The parser makes no decisions about what interpretation of a partial parse is correct until it finishes parsing. Therefore it cannot make an incorrect decision due to lack of contextual information in a particular stage of processing. All decisions take into account both bottom-up and top-down information from the whole text.

Of course the system may fall prey to missing vocabulary, to a combinatorial explosion, or to an incorrect decision at the end of processing. The only problem unique to Sepia is the combinatorial explosion, on which the bet is that semantic pruning (see Section 3.3.4) will be sufficient.

It is possible to propagate ambiguity and semantics between separate modules in a pipeline explicitly, but there are drawbacks. Much of the practical usefulness of pipelining is that very different modules can be interchangably replaced and evaluated so long as they have the same input and output formats, but the more information one passes between modules in a pipe, the more difficult it is to make any component fit.

Furthermore, it's not clear what information to pass: top-down information is likely to be in the form of a constraint, and the probability ordering of the options given the constraint might be different than the order was without the constraint. Thus passing the top k options even with original probabilities is not enough, because the answer that matches constraints might be unlikely (and therefore surprising, interesting!). Even passing all of the options is not theoretically enough, because useful information about the structures and their likelihoods then needs to be duplicated in the later module in order to evaluate the possibilities under the new constraint. Thus

\[3\text{though truecasing and removal of other markup before processing would both very likely help, given that we have been writing lexicon for newspaper text. Of course I believe that these processes should also ideally become part of a system like Sepia, but there is only so far that one can push an agenda before hitting a deadline.}\]
to fully incorporate top-down constraints, one has to duplicate the prior module in the later one.

Sepia is still impoverished because it does not yet have an infrastructure to use probabilities for lexical items, but the thrust of this work is in showing a proof-of-concept architecture that can integrate the modules, and thus best reap the benefits when probabilities are later added.

Though it has no pipeline, Sepia can serve as part of a pipeline. Sepia is built with standoff annotation in mind, and there is a wrapper that adds Sepia-recognized entities as annotations to an XML standoff document. This interface was used to integrate Sepia with two modules in the Infolab’s TREC 12 / AQUAINT Question Answering competition entry.

3.2 Working Example

Throughout this chapter I will use examples to illustrate each idea. The example in Figure 3-1 will serve in multiple explanations, so I begin by summarizing it.

The default tokenization is shown in parentheses, but two items from the lexicon. 'O' and 's are also recognized as tokens. At the same time, any multi-word lexical entries will also be recognized, (e.g., a pre-existing lexical entry for “Margaret O’Hanlon”, were she famous). Lexicon-based tokenization is discussed in Section 3.4.2.

Hanlon is recognized not as a word in the lexicon, but by a regular expression pattern. All tokens are tested against each pattern in the lexicon. The special *i* notation allows the semantics to have access to the groups a match generates. In this example, there is no difference between *0* and *1* because the first set of parentheses in the pattern happens to capture the whole string. Regular Expression entries are discussed in Section 3.3.8.

O’is a function that requires a capitalized word and generates a common lastname. It applies rightward (a backslash in the category would indicate leftward application) to Hanlon to generate the string “O’Hanlon”. The new category common lastname
Example: Margaret O'Hanlon's Bar & Pub

Default tokenization: (Margaret) (((O)'(Hanlon)'(s))) ((Bar)(&)(Pub))

Lexicon:

(import "edu.mit.nlp.sepia.AtValSet")
(import "edu.mit.nlp.sepia.Coordination")

;; Margaret : common_female.firstname
*0*

;; 's : possessive
#t

;; /^[A-Z][a-z]+$/ : capitalized_word
*1*

;; O' : common.lastname / capitalized_word
(lambda (w) (string-append *0* w))

;; :common_female.firstname: : firstname
(AtValSet. 'name *O* 'gender 'female)

;; :common.lastname: : person \ firstname
(lambda (p) (.append p 'name *O*))

;; Bar : organization \ person \ possessive

;; Pub : organization \ person \ possessive
(lambda (possessive)
 (lambda (namesake)
  (AtValSet. 'type 'food-service
   'name (list namesake "'s *0*)))))

;; & : */ */ \ *
(lambda (back)
 (lambda (forw)
  (Coordination. 'and back forw))))

Figure 3-1: An example and a lexicon.
Disclaimer: entries are meant to be syntactically correct and illustrative, but not realistic.
and its new logical form, the string “O’Hanlon”, are assigned to the existing token *O’Hanlon*.

*Margaret* is first recognized as a common female firstname. That category can be promoted to *firstname* using a category promotion rule. Here the *0* still refers to the entirety of what has matched, but that is now a Scheme object (not guaranteed to be a string). In this case the new promoted logical form is a variant on a feature structure, a set of attributes associated with lists of values. Category promotion is discussed in Section 3.3.9. Complex logical forms including the Attribute-Value-Set are discussed in Section 3.3.11.

*O’Hanlon* is promoted to a function that applies to the *firstname* *Margaret*. The *firstname* *Margaret’s* underlying object is an AtValSet (see Section 3.3.11). This semantics appends the string “O’Hanlon” to the name field of that AtValSet and returns the resulting new structure. That new structure is housed in a new token *Margaret O’Hanlon* which is created on the fly with category *person*. The creation of new tokens and its impact on parsing strategy is discussed in Section 3.4.1.

The word & is a coordinate conjunction that combines two tokens to either side that have the same category (function signature), and creates a new token spanning both that has the same category as its arguments. The Coordination logical form data type automatically adds semantics for mapping applications over their arguments to create a new Coordination of organizations when “Bar&Pub” is applied. Coordinations and other complex logical forms are discussed in Section 3.3.11.

*Bar* alone will also combine with *Margaret O’Hanlon’s*, to generate a parse for only *Margaret O’Hanlon’s Bar*, but a final heuristic after parsing is finished will remove partial parses entirely subsumed by larger ones. This final heuristic is discussed in Section 3.4.5.

In the following sections, I will first discuss general design principles, and then address issues of implementation.
3.3 Lexicon Design

A lexical item consists of a word or regular expression and its “definition”. To a computer, a definition must be a function or data structure, and the language I have chosen to build definitions in is JScheme [1]. Each function and each data structure has an associated function signature, called a category, used to guide the parser in applying the right functions to the right data structures.

3.3.1 Separation of Language from Parsing

Combinatory Categorial Grammar (CCG) is a formalism for semantic parsing, pioneered by Mark Steedman, that can capture many previously problematic phenomena across languages in a compositional structure [59]. The key idea is that the parser be very simple and language-independent, following pure function application rules, and that the lexicon contain all language-specific rules, word meanings, and associations. This separation is a central theme in Lexical Semantics, and is a central design guideline in Sepia. Moreover, keeping the parser as simple as possible makes it easier for a lexicon builder to predict what effect his changes will have, and so makes his job easier.

A CCG parser uses a category associated with every logical form to guide its processing, and this category is all that the parser knows about each semantics, or logical form. The category indicates whether each logical form is a function or a data structure. Functions have arguments, whereas data structures do not; if a category has slashes in it, it is a function, otherwise it is a data structure. A category with slashes—a category indicating a function—is a list of categories separated by slashes. The first category specifies the type of the function’s result. After that, a slash indicates the direction in which an argument is expected (a forward slash (/) indicates rightward in the string, and a backslash (\) indicates leftward), and the category which follows indicates the required type of the argument. The lexical entry for the verb “slept” (see Figure 3-2) might specify that it requires an animate argument to its left.
The arguments are filled from right to left\textsuperscript{4}, so that one may assume left-associativity in the category. Thus one logical form for “King” that was intended to recognize “King of Spain” might be specified as shown in Figure 3-3.

\begin{verbatim}
;: King : person / country / of
\end{verbatim}

Figure 3-3: A category of the title King, as it might try to recognize “King of Spain”. Note that arguments are left-associative.

Sepia’s CCG parser will attempt to apply a function to all constituents of a matching category in the correct relative direction.

### 3.3.2 Using JScheme for Semantics and as an Interface to the World

I chose JScheme as the language of Sepia semantics for several reasons. Lisps are acknowledged to be the most intuitive way to express functional programs such as the lambda expression semantics proposed by Steedman. I wanted to use Java for the parser to make Sepia most easily usable by others. I also wanted to enable the semantics to make use of others’ Java programs as resources for building meaningful semantics. For example one might imagine a particular function wanting to call out to Google to decide whether “Dana Farber” was male or female.\textsuperscript{5} JScheme is a Scheme with a strong Java interface both in being called from Java and in calling out to other

\textsuperscript{4}The rightmost argument corresponds to the outermost lambda in the logical form.

\textsuperscript{5}only to find, of course, that it’s an organization! For the curious, it is a cancer institute founded by industrialist Charles Dana and Dr. Sidney Farber.
Java modules. The strong type checking that Scheme provides has been helpful in developing initial lexicons.

### 3.3.3 Scheme Expressions vs. Feature Structures

Grok [5] and other existing semantic parsers use a vector of attribute-value pairs as their base semantics and often perform unification as part of parsing. I think unification on attributes can be very helpful in syntax checking, and won't rule it out of future versions of Sepia. However, using full Scheme expressions for the semantics offers several advantages. The first is that the parser becomes simpler and more transparent to the lexicographer. The second is that it becomes possible to communicate meaningfully with other expert systems to make decisions about a meaning.

### 3.3.4 Pruning Failed Partial Parses

One of the primary uses of unification is to prune parses which have conflicting features, interpreted as meaning. Sepia can also prune partial parses because of a meaning conflict, but the possible conflict is more generic: if any semantics returns `#null` or `#f` as its result, or exits via an exception, then that partial parse is dropped.

For example, the syntax need not specify that in "two one hundred dollar bills", there are two separate numbers, namely "two" and "one hundred", because the semantics can recognize that these cannot conjoin the same way that the ambiguous "thirty one hundred dollar bills" can. The Scheme expression that would combine "two" and "one hundred" simply needs to return failure.

The key difference between this pruning strategy and pruning in other parsers is that a parse in Sepia is not pruned because it is unlikely. Rather, it's pruned because it's semantically not possible or plausible.

Statistical pruning can complement such direct meaning-based pruning by assigning likelihood to individual logical forms. The most likely parses which are still available after processing may often be correct.
3.3.5 Multiple Entries with a Single Semantics

To avoid unnecessary redundancy in the lexicon, it is possible to specify multiple entry words and categories that all share a single semantics. The format is simply to specify all the lines of word and category before the block of semantics. For example one might use the following to specify multiple names:

```
;: Mary : common_female_firstname
;: Margaret : common_female_firstname
;: Michael : common_male_firstname
*0*
```

The three names have different strings, and different categories, but share the same simple semantics (*0* indicates the actual string matched—one of their names).

3.3.6 String Entries vs. Headwords

In a traditional lexicon, inflectional variants of a word (fly, flew, flown, flying, flies) are grouped under a headword (fly), and their definition given with the assumption that the inflectional transformations in meaning are known. Handling headwords would involve handling morphology in general, and this is not a problem I have considered as part of Sepia. That decision makes Sepia difficult to use for morphologically productive languages like German, Hungarian, or Turkish. Sepia would not essentially change if a morphological analyzer were put in place of the tokenizer, but that would have been more difficult to implement.

String entries may use the *0* variable to retrieve the actual string that triggered them (as in the example in Section 3.3.5) rather than repeating the string in the semantics. This usage is similar to (and inspired by) the use of the *0* variable for pattern entries, as discussed in Section 3.3.8.
3.3.7 Multiword Entries

Multiword lexical items are desirable because many noun compounds, e.g. "World Cup", verb-particle constructions, e.g. "took off" (as an airplane or idea), and other phrases have noncompositional or idiomatic meanings. There are also entries which span token boundaries as currently defined in Sepia, but are parts of words. For example 's as a possessive seems to encode a semantic unit. See Section 3.4.2 for implementation details of multi-token lexical entries.

3.3.8 Pattern Entries

Regular Expressions are a common way to recognize classes of strings that may have a particular closed form. Allowing regular expressions in the lexicon is an intuitive way to recognize, for example, the string "28" in the text being processed and convert it into the number 28. The lexical entry for converting a sequence of digits to a number might be:

```
(define number-format (java.text.NumberFormat.getInstance))

;: /"(\d+)$/ : digits
(.parse number-format *0*)
```

Similarly, currency or date recognition can take advantage of Java's library of date and currency parsers, and immediately transform the text into a form that has semantics to the program. Numbers, dates, and currencies can be readily compared with other objects of the same type, or converted to other formats.

The lexicographer need not even be careful to only try to parse legal numbers or legal values in other formats. If a DateFormatException (or any other exception) is thrown, the partial parse will simply be abandoned, as discussed in Section 3.3.4.

Special variables of the format *i*, where i is a number 0 or greater, are available to the semantics of pattern entries. These refer to the groups captured by the regular
expression. In the example above, *0* refers to the entire match (as group(0) does in Java, perhaps as $0 does in perl). *1* would be equivalent, as the parentheses capture the whole token.\(^6\)

**3.3.9 Category Promotion**

It is often useful to specify an is-a hierarchy for things in the world. A verb might require that its agent be animate, or a first name might want to combine with a last name. WordNet has an extensive is-a hierarchy which many find useful, and which will likely be useful in specifying semantics of a broader coverage lexicon.

For Sepia's parser, the only notion of type is the category system, and so it is useful to be able to say that any object of one type is also necessarily an object of another type. In the Margaret O'Hanlon's Bar&Pub example (Figure 3-1), a `common_female_firstname` is necessarily a `firstname`. Presumably a `common_male_firstname` would be as well. A `lastname` can then specify that it wants to combine with a `firstname`, whatever sort of `firstname` its argument might underlyingly be.

The notation for this category promotion is like any other lexical entry, except that in the place of a string to be recognized is the category to be promoted, inside a pair of colons.\(^7\) The semantics specify the transformation involved in promoting the type from the base category to the higher category, and have access to the actual object of the base category through the *0* variable. The use of the *0* variable to mean "whatever this rule refers to" is inspired by the use of *0* in pattern rules (Section 3.3.8) and used here in a metaphorical sense.

**3.3.10 Wildcard Matching Category**

Coordinations such as "and" and "or" can apply to any category, including complex categories that have arguments. Rather than handling coordination specially in the

---

\(^6\)Please refer to the [java.util.regex.Matcher javadoc](https://docs.oracle.com/javase/8/docs/api/java/util/regex/Matcher.html), specifically the `group(int)` function for implementation details.

\(^7\)A text can unfortunately fool the parser into thinking that a category is present by happening to have a category name between colons, but the semantics would likely not get too far. In any case, Sepia is not designed with security in mind.
parser, I chose to create a wildcard category label * that would:

- match any other category, and
- when applied to a real category, would cause the wildcards in the resulting category to be filled with the matched category.

Consider the lexical entry

\[
\text{;; and : * / * \ *}
\]

\[
\text{(lambda (back) (lambda (forw) (Coordination. 'and back forw)))}
\]

When applied leftward to a category a, the new logical form will have category a / a. Similarly, when applied to a category a / b, the new logical form will have category (a / b) / (a / b). Because categories are left-associative, that is just the same as a / b / (a / b).

In the "Bar&Pub" example, Bar and Pub both have category organization \ person \ possessive. When & is applied to Bar, it creates a new function with category organization \ person \ possessive / (organization \ person \ possessive). This new category can apply directly to Pub, and the result is a Coordination that has the same type each constituent started out with. The Coordination class takes care of the correct combination of map and apply to create a result of application that is a Coordination of the two objects that would have been created had each of Bar and Pub applied separately. This may not always be the desired semantics for coordination, and I will address that in the next chapter.

There is another type of wildcard character that also matches anything, but does not do anything special when matched, as * does. This wildcard is denoted with a period.

41
3.3.11 Complex Logical Form Data Structures

A semantics cannot live on strings and alists\(^8\) alone. Some formal semantics use a Typed Feature Structure, a form of association list, to represent meaning. Unification over a feature structure\(^9\) has been a useful tool in combinatory semantics. Sepia provides a variant on a feature structure called an AtValSet (Attribute-Value Set) as one possible type of logical form, provides a Coordination as another, and leaves open the possibility of other types.

AtValSets are the most common complex logical forms in the named entity lexicon. These are HashMaps keyed on Scheme Symbols whose values are LinkedLists of Objects. Append and prepend are defined to append or prepend objects or lists of objects to the list associated with a symbol. Append and prepend also support entire AtValSets, in which case they perform the appropriate list-appends key by key. AtValSets support deep copying so long as their contents do as well. They also support an asymmetric match operation that requires an argument AtValSet to have all but not only the same properties.

Coordinations contain a simple ordered list of Objects and a coordination type. They support a match operation as a set would (without regard to order) by calling the match operator on the element data types. They handle mapping over a single argument if their contents are functions, or mapping an applying function over multiple arguments otherwise. As a disclaimer, this is simply the most obvious interpretation of coordination, and I discuss problems with this form of coordination in Chapter 4.

Though I have not implemented them, I imagine that Lexical Conceptual Structures \([21]\) might be another useful type of complex logical form data structure to represent paths, causality, and event relations, for which simple sets of attributes seem impoverished.

Each data structure discussed above is language-independent, which is why I provide them as part of the Java framework. Complex logical forms like Person might

\(^8\)alist is a lispism for “association list”, akin to the Perl hash or the Java Map
\(^9\)Typed Feature Structure Unification is essentially the process of merging two Typed Feature Structures (sets of attributes and their values, where those values may also be Typed Feature Structures) while ensuring that there is no unresolvable conflict or disagreement between them.
3.3.12 Execution Environment

The JScheme semantics are evaluated in an execution environment which is initially empty. Even in an empty environment, the Scheme logical forms would have access to the Java environment through JScheme’s JavaDot notation, but a lexicon can provide additional definitions to its logical forms.

Any Scheme that appears in a lexicon file before the first lexical entry is evaluated at load time, and treated as part of every lexical entry’s environment. This was hinted at in the NumberFormat example in Section 3.3.8, but was not explicitly discussed. The `number-format` variable there would become available to any logical-form in any lexicon file. It is also common to see Java-like import statements here to give more readable access to classes like AtValSet and Coordination, as appear in the first two lines of the Pub example lexicon (Figure 3-1).

3.3.13 Stochastic Parsing

A major design decision in the current version of Sepia is conscious inattention to statistical methods in parsing. The motivation is to see how far purely knowledge-based methods can be pushed. I do not see a conflict between stochastic and knowledge based parsing, and in fact would like to attempt an integration of probabilities on logical forms associated with lexical items, and thus on partial parses. It is easy to see how stochastic parsing would be implemented in this framework: as a set of probabilities of a logical form being associated with its string, pattern, or category promotion. However, if the probabilities were made available to the semantics, and parsing decisions were made during processing based on the intermediate probabi-

10extend in the very concrete Java subclassing sense
ties, then it would be difficult to separate the effect of the statistics from the effect of the richer semantics in the final results.

When statistics are incorporated into Sepia, they will not be visible in the lexicon. Humans are not good at specifying probabilities or at understanding their impact. Probabilities should come from some form of training, and be stored separately from the human-readable semantic lexicon.

3.4 The Parsing Process

3.4.1 Tokenization

Every parser must have a smallest unit of meaning, and Sepia primarily uses word boundaries and boundaries between sequences of letters and numbers for its smallest division points. Specifically, strings are first split on /\ s/, then into substrings matching /((\d)+|[a-zA-Z]+[^a-zA-Z0-9])/. Tokens are stored in a lattice data structure that makes access to adjacent nodes efficient. The initial set of tokens also includes any multtok en sequences present in the lexicon: thus the tokenizer can be said to be lexically aware, even though it remains language independent. One can argue that it is not language-independent because it is tied to a particular character set. But adapting the tokenizer to another character set would be trivial, involving a change to the two regular expressions mentioned above. A more fundamental change would be to break on morphemic boundaries, as discussed in Section 3.3.6.

Tokens are “activated”, added to a parse queue, when they are first created, and when a new token is created adjacent to them. The parser evaluates tokens, attempting to combine them with adjacent ones, in the order that they are enqueued\(^{11}\). Logical forms can themselves generate new tokens, in particular they can create tokens representing linguistic empty categories, or they could use this feature to simulate

\(^{11}\)This is an area for further research. It might make sense in some cases to explicitly promote an ordering, as for a hyphenated sequence of words.
morphological analysis.\textsuperscript{12} No lexical entries make use of this flexibility at the moment, however.

\subsection*{3.4.2 Multitoken Lexical Entries with Prefix Hashes}

Sequences of tokens are recognized by precompiling from the lexicon a pair of prefix hashes, one for entries spanning word boundaries (the first regular expression above) and one hash for entries spanning within-word boundaries (the second regular expression). These contain, for every multi-token lexical entry, all token subsequences starting at the beginning of the entry.

For \textit{Margaret O’Hanlon}, were it a single lexical entry, the word prefix hash would contain \textit{Margaret}, and the subword prefix hash would contain \textit{O} and \textit{O’} (with the apostrophe).

During tokenization, which happens left to right, if a current sequence of tokens (including the base case of one token) is in the prefix hash, then it is kept in consideration as a possible multi-token entry. If any of these finds a match in the lexicon, a token is generated.

\subsection*{3.4.3 Initial Lexical Assignment}

Each token may be associated with several categories in the lexicon, and each category may have several logical form functions. All logical forms associated with all matches to a token are initially postulated as possibilities. Moreover if a category can be promoted, it will be, and the promoted logical forms also activated. Category promotion was discussed in Section 3.3.9.

\subsection*{3.4.4 Parsing}

As these logical forms combine through application, new tokens are generated with new categories and logical forms. When a new token is inserted into the lattice, all

\textsuperscript{12}A pattern might recognize a word ending and generate two tokens of the appropriate boundaries for the stem and affix.
adjacent tokens are activated, and have a chance to combine with the new token.

This strategy is almost equivalent to an $n^3$ chart parsing strategy, but allows extra flexibility by letting logical forms to influence the course of parsing, as discussed in Section 3.4.1.

When a logical form combines with another and yields a null or false result (including failures due to Scheme errors), the partial parse is not added to the lattice, and parsing continues. This sort of semantic pruning was discussed in Section 3.3.4.

3.4.5 Final Decision Among Partial Parses

When parsing is complete, some set of named entities must be chosen to report as discovered entities. Sepia finds all partial parses that are not entirely subsumed by strictly larger partial parses, and reports all the constituents that compositionally generated this partial parse which are also marked with a named entity type.

3.5 Design for Error

To Err is Human, and to err in writing computer programs is the rule. Sepia's design makes it brittle, not only in the sense that it cannot know about every lexical item, but also in the sense that it is easy to write logical forms badly. There are some common types of errors that Sepia lexicons are prone to, and various ways of mitigating their effects.

3.5.1 Underlying Data Type of Arguments is Invisible

All logical forms must take into account the types of their arguments, if any, and there is little standardization. Arguments may be base scheme types, arbitrary Java classes, or any of the complex logical forms discussed in Section 3.3.11, and a lexicographer's only clue as to the type of object a function will receive is the category it can apply to. It is very easy to make a type error of this sort. Use of Hungarian-like category naming conventions (e.g., categories names which are expected to be strings begin
3.5.2 Category Promotion Can Cause Infinite Loop

Category promotion is another source of error. One could accidentally create two rules that promote to the same category but have different underlying data types, which would make engineering functions to apply to that category difficult. One can also accidentally create a category promotion loop, which will cause the parser to enter an infinite loop.

3.5.3 Fault Tolerance

Sepia tolerates faults in the lexicon by limiting the damage they can cause. The infinite loop will eventually cause an OutOfMemoryException, which will cause the partial parse to be abandoned, and the rest of the execution will go on, and the memory reclaimed. A type error or any other exception will have similar effect.

3.5.4 Testplans

Sepia also provides a testing infrastructure that allows a lexicon developer to create testplans whose testcases specify a string, a part of that string to be recognized, the category, and a Scheme expression to build the logical form that the recognized object must match. Testcases may also be negative, where it is checked that the substring given does not have a logical form interpretation with the category specified and which matches the logical form given. As the test proceeds, errors and derivations for failed parses are shown. The testplan infrastructure helps lexicon developers quickly find and eradicate most bugs.

3.5.5 Short Semantics

Finally, a mitigating factor for bugs in the lexicon is that most semantics (so far) tend to be fewer than three lines long, and there is very little repetition of code. The
amount of context a lexicographer needs when writing a particular piece of semantics is small (the arguments, products, and sister functions that create similar products). These attributes make it relatively painless (I daresay even fun) to write lexicon for Sepia.
Chapter 4

Illustrative Examples

In the previous chapter, I showed how Sepia’s parser evaluates lexical items to form compositional data structures. In this chapter, I will detail the proof-of-concept lexicon I created, and show how it interprets some interesting examples. In creating this lexicon, I have tried to remain true to my intuitions about the meanings of the words, and will clearly mark those cases in which I was forced to depart from those intuitions.

The basic challenge throughout this chapter is to capture enough ambiguity that the correct answer (or the set of correct answers) is always available, while reducing the number of available incorrect or inplausible possibilities. The goal is to show that Sepia can capture important elements of meaning in a useful, intuitive, and compositional way.

I begin with an overview of the contents of the initial lexicon, then show solutions to common problems in named entity understanding, and finally review problems that Sepia cannot handle. Throughout, I will refer to the Automatic Content Extraction (ACE) evaluation in which we plan to compete in four weeks. Many of the examples are taken directly from ACE guidelines, and I will discuss how Sepia will change to handle them.
Table 4.1: Distribution of fixed lists in the lexicon

4.1 Overview of the Lexicon

The current lexicon was built in a relatively ad-hoc manner to fit examples. In the coming month, we will be preparing for the Automatic Content Extraction (ACE) evaluation, and will extend the lexicon in a data-driven process of iterative refinement.

The current lexicon contains 20195 total lines, including comments and whitespace as well as fixed lists and rules. There are only a few known verbs, and the coverage generally does not reflect what is required for the ACE task. The distribution of fixed lists is shown in Table 4.1.

For ACE, we will separate the fixed lists into a database, and we plan to add large lists of famous people, organizations, facilities, and locations, as well as a set of verb frames from PropBank [43] and Bonnie Dorr's publicly available LCS lexicon [21].

There are 12581 total unique keys in the final constructed lexicon. This does not mean that there were exactly 266 combinatory rules (including cue words), as one might guess by subtracting the number of fixed list entries. It simply means that there were at least 266 unique strings, patterns, or promoted categories that
Figure 4-1: Particles that appear in mid-name. The eight separate rules take 11 lines of code, and result in four tokens specified.

rules were associated with. Figure 4-1 shows how a single string may have multiple associated rules.¹

Of the 12581 keys in the lexicon, 58% were single words without punctuation. The breakdown by key type is shown in Table 4.2.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7248²</td>
<td>single-word</td>
<td></td>
</tr>
<tr>
<td>4789</td>
<td>multi-word</td>
<td></td>
</tr>
<tr>
<td>558</td>
<td>contain sub-word token sequences</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>are regular expression patterns</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2: Breakdown of single-token, multi-word, and multi-subword lexical entry keys.

¹note also that some of the fixed list entries themselves may have been duplicate strings with different meanings, and may have been the same as some cues, so the subtraction is meaningless.

²not 7214 because 34 multi-word tokens also contain a first or last word that can break into sub-word tokens, e.g., John Boyle O'Reilly
4.2 Mechanics

This section of examples is intended to illustrate the mechanics of the parser and lexicon, showing how a set of lexical items can combine to capture both surface and meaning phenomena in simple cases. The section will also introduce some of the tools the current lexicon uses for entity recognition.

4.2.1 Function Application

Function application and category promotion are Sepia’s bread and butter. Turning “a hundred and ninety-one days” into a duration of 191 day units is a straightforward process. We start with the lexicon fragment in Figure 4-2. The lexical assignment is as follows:

\[
\begin{align*}
a & \Rightarrow 1: \text{socard.ones} \\
hundred & \Rightarrow 100: \text{socard.hunds}, \\
\quad \{\text{type} => \text{cardinal}; \text{value} => 100\}: \text{quantity}, \\
\quad (\lambda \ a/\text{socard.ones} \ . \ (* \ a \ 100)) , \\
\quad (\lambda \ a/\text{socard.teens} \ . \ (* \ a \ 100)) \\
and & \Rightarrow (\lambda \ u/\text{socard.teens} \ . \\
\quad (\lambda \ h/\text{socard.hunds} \\
\quad \quad (\text{if} \ (\text{equal}? \ 0 \ (\% \ h \ 100)) \ (+ \ u \ h)))) \\
ninety & \Rightarrow 90: \text{socard.tens} \\
\quad (\lambda \ q/\text{socard.ones} \ (\text{if} \ (> \ *0* \ 10) \ (+ \ *0* \ q))) \\
\quad (\lambda \ h/\text{hyphen} \ (\lambda \ q/\text{socard.ones} \ (\text{if} \ (> \ *0* \ 10) \ (+ \ *0* \ q)))) \\
- & \Rightarrow '(): \text{hyphen} \\
one & \Rightarrow 1: \text{socard.ones} \\
days & \Rightarrow (\lambda \ q/\text{quantity} \ {\text{unit} => \text{day}; \text{value} => q})
\end{align*}
\]

While for a class in semantics, I would show several pages of derivation, here I will simply outline the steps the derivation takes:
Figure 4-2: The lexicon required to parse “a hundred and ninety-one days”
1. *hundred* can apply to a to create 100:*socard_hunds*. That is, a new token a *hundred* is generated, with category *socard_hunds* that has semantics 100.

2. *ninety* can apply first to the hyphen, then to one to create a new token *ninety-one* with category *socard_teens* that has semantics 91.

3. The special numerical non-coordination *and* can apply first to 91:*socard_teens*, then to 100:*socard_hunds* to create the new token *a hundred and ninety-one* with category *socard_hunds* that has semantics 191.

4. The category *socard_hunds* can be promoted, and during promotion, a new object is generated based on the old one, still assigned to token *a hundred and ninety one* but with category quantity and semantics `{type=>cardinal; value=>191}`.

5. *days* now has a quantity to its left, and so it can apply, and create the largest token, *a hundred and ninety-one days* with category duration, that has semantics `{unit=>day; value=>{type=>cardinal; value=>191}; quantity}`.

Note that the end result is a nested structure of Attribute Value Sets, and that the inner AtValSet retains its category (denoted after the colon). If either the category quantity or the category duration were markable in a task, they would both be available to the parser’s final selection algorithm.

### 4.2.2 Context Sensitivity

It is sometimes useful to find text between paired elements. A book title might often appear in title case (non-stopwords capitalized) between quotes.

Figure 4-3 shows a lexicon that would capture a series of title-case words between quotes. It would capture "*The Wizard of Oz*" by successively consuming *capwords*, *stopwords*, and *names* (not defined here) from left to right, appending them to a list, until it found a *title_end*, a capitalized word with an end quotation.³

³One may wish to allow non-capitalized words to also end titles, or quotations in general. The point here is that one can capture pairs, including pairs with certain properties, not the specific set of properties one may wish to capture in any domain.
Figure 4-3: A lexicon to capture Title-case words between quotes
Occasionally, it is useful to capture nested pairs, such as possibly nested parentheses. Figure 4-4 shows a lexicon to capture anything as nested lists from a set of nested parentheses.

Finally, though it's trivial to count cardinalities of lists in the semantics (Scheme is, after all, Turing-complete), it is worth noting that the category grammar itself is context sensitive, so long as standard function composition is available as an operator. Figure 4-5 shows a simple grammar to recognize the canonical non-context-free language $a^n b^n c^n$.

### 4.2.3 Extent Ambiguity

"John Smith" is a simple combination of a list-recognized first name and a list-recognized last name. "Sun He" on the other hand, would be very difficult to recognize as a name unless there were some solid cues, such as "Mr. Sun He met ...". The current lexicon, as I have written it, would not allow "He" as a last name even in this context, but this is a mistake of the lexicographer.

There are cases though, where names are ambiguous despite the most complete lexicon. Figure 4-6 shows a lexicon complete for parsing "I gave Mary Jane Frank's address", but two derivations are possible. "Mary Jane" may be the recipient, and
Two required rules:

base case: \( b : \text{done} \setminus a / c \)
generator: \( b : (\text{done} \setminus a \setminus a / c) / (\text{done} \setminus a) \)

Derivation for \( n = 4 \):

\[
\begin{align*}
\text{a a a a b b b c c c c c} & \quad \text{apply last b to first c} \\
\text{a a a a b b (done \setminus a / a) c c c} & \quad \text{apply generator from last b} \\
\text{a a a a b b (done \setminus a \setminus a / c) c c c} & \quad \text{apply to first c} \\
\text{a a a a b b (done \setminus a \setminus a / a) c c} & \quad \text{compose generator with the partial derivation} \\
\text{a a a a b (done \setminus a \setminus a / c \setminus a) c c} & \quad \text{compose generator with the partial derivation} \\
\text{a a a a a (done \setminus a \setminus a / c \setminus a / c \setminus a) c c} & \quad \text{apply to last a} \\
\text{a a a a a (done \setminus a \setminus a / c \setminus a / c) c c} & \quad \text{apply to last a} \\
\text{a a a a a (done \setminus a \setminus a / c \setminus a / c) c} & \quad \text{apply to first c} \\
\text{a a a a a (done \setminus a \setminus a) a} & \quad \text{apply to first c} \\
\text{a a a a a (done \setminus a \setminus a) a} & \quad \text{apply to last a} \\
\text{a a a a a (done \setminus a \setminus a)} & \quad \text{apply to last a} \\
\text{a (done \setminus a)} & \quad \text{apply to last a} \\
\text{done} & \quad \text{apply to last a}
\end{align*}
\]

Figure 4-5: Categorial Grammar is context sensitive with composition.
“Frank’s address” the gift, or “Mary” the recipient and “Jane Frank’s address” the gift.

It cannot be that “Mary Jane” is the recipient and “Jane Frank” is who the address pertains to. The reason is that when “Mary Jane” is consumed by “gave”, the new token becomes “gave Mary Jane”, and that can only apply to adjacent objects.

For “Jane” to have both roles, there would have to be two objects whose head was “address”. One would be “Jane Frank’s address” and the other would be “Frank’s address”. Only the latter is adjacent to “gave Mary Jane”, and so only the latter can be a candidate for application.

Similarly, “Mary Jane Frank” becomes a partial parse token of type person, but once “gave” consumes “Mary Jane Frank”, the adjacent element is a function of category NP / NP \ person, and “gave Mary Jane Frank” would require a simple NP.

At the end, the parser chooses the two largest parses, which are ambiguous, and marks all four subsumed named entities. It does not mark which pairs are consistent, so to an outside observer, or to a subsequent module, it looks as if “Jane” could be used twice. This is one kind of error to be careful of when passing more than one possible parse in a pipeline, and should be better addressed in Sepia’s future work, as part of Sepia’s goal to serve as a component in a larger question answering system.

4.2.4 Nested Entities

I showed in the first example that an AtValSet may hold another AtValSet as one of its values, and return it as a candidate if it turns out to be one of the largest spanning logical forms. This feature can be directly applied to the Automatic Content Extraction (ACE) task.

Two of the ACE guidelines require that:

- “Provo, Utah” evokes two entities: the city of Provo in Utah, which is the whole string, and the state of Utah itself. Thus ACE asks that the both nested annotations be present.
Figure 4-6: Lexicon for “I gave Mary Jane Frank’s address”
The adjective “Russian”, as in “Russian fighter plane” evokes the Geo-Political Entity (GPE) “Russia”; therefore, even though the fighter plane itself is not to be marked, “Russian” is, as a GPE.

Both of these can be readily implemented in Sepia. For Provo, the rule that combines a likely (or known) city name with its state must simply keep the state argument as a component of itself:

```
;; :city: : GPE / city / comma
(lambda (comma) (lambda (state) (.append *0* 'in state)))
```

“Russian” doesn’t have an explicit country name nearby, but it can create one:

```
;; :Russian : person_adj
(AtValSet. 'group #t 'from (.setCategory (AtValSet. 'type 'country 'name "Russia") (Category. "GPE")))
```

At the same time, ACE make two counterintuitive requirements:

- “Russian”, when used as a person (“seven Russians were among...”), is judged not to evoke a Geo-Political Entity.

- names such as “The New York Times” are atomic, and do not evoke a Geo-Political Entity such as “New York” in this case.

Whatever the reasoning, the distinction is easy to implement—we simply do not create the referent GPE. If a semantics later wants to look up the relevant GPE, it
can simply ask the lexicon.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Russian</td>
<td>person</td>
</tr>
<tr>
<td>(AtValSet. 'nationality &quot;Russia&quot;)</td>
<td></td>
</tr>
<tr>
<td>Russians</td>
<td>person</td>
</tr>
<tr>
<td>(AtValSet. 'group #t 'nationality &quot;Russia&quot;)</td>
<td></td>
</tr>
<tr>
<td>The New York Times</td>
<td>organization</td>
</tr>
<tr>
<td>(AtValSet. 'type 'news 'name <em>0</em>)</td>
<td></td>
</tr>
</tbody>
</table>

The psychological reality of including a single level of referent (an actual AtValSet within another) or only including references (a way to find the other AtValSet) is beyond the scope of this thesis, but either choice can be effectively modelled.

4.2.5 Type Ambiguity

Entities in the ACE task are often ambiguous as to their category depending on their role in the sentence context. The canonical example is that of the White House. It is considered an organization when it is said to take action, “The White House vetoed the bill”, but a facility when the physical structure is in question, “Visitors streamed through the White House.”

One difficulty is in differentiating the physical path from the metaphoric one: “Bills streamed through Congress and the White House in a marathon effort to pass new clean air regulations today.” One can argue that bills do pass through the White House facility on their way to becoming law, but there is no facility called “Congress”.

A possible solution is to allow an ambiguous entity to be truly ambiguous (specifying both types) in isolation. If one of the categories is used in a larger construction, the subsumed category that is not used will be pruned. In the White House veto example, the lexicon in Figure 4-7 allows both the facility and organization interpretations, but when “vetoed” applies, it creates a subsuming element using the organization. If

---

4A reader points out another possibility: “Bills streamed through the White House on the first annual Clinton Impersonation Day”
Entities can have multiple categories. If one type is used in a larger context, it will be preferred.

If it stores the organization's logical form within itself, then that logical form will be marked, and the subsumed facility will not.

That strategy does not help for entities that are ambiguous between syntactically similar types like person and organization. It is very difficult to tell how “Dana Farber” should be marked in many contexts without a priori knowledge that it is an organization. The disambiguating information may be in a coreferenced nominal or pronoun, but coreference is not yet in place in Sepia.

**4.2.6 Coordination**

Coordination under CCG combines two (or more) logical forms with like categories to form a single logical form that contains all the others, has the same category as each of its constituents, and specially handles application to properly map functions over a list of arguments, or to map a list of functions over a single argument, depending on whether the coordination is in a function or argument position.

The phrase “two or three hundred miles” can be processed by a lexicon similar to
the one presented in Section 4.2.1:

```lisp
;;; two : socard.ones
2
;;; three : socard.ones
3

;;; or : * / * \ *
(lambda (x)
  (lambda (y)
    (Coordination. 'or x y)))

;;; hundred : socard.hunds \ socard.ones
(lambda (a) (* a 100))

;;; :socard.hunds: : quantity
(AtValSet. 'type 'cardinal 'value *0*)

;;; :socard.ones: : quantity
(AtValSet. 'type 'cardinal 'value *0*)

;;; miles : length \ quantity
(lambda (q) (AtValSet. 'unit 'mile 'value q))
```

Because both socard.ones and socard.hunds are promoted to quantity, there are three possible ways for the coordination or to apply. It can apply to:

- `two:socard.ones` and `three:socard.ones`.
- `two:quantity` and `three:quantity`.
- `two:quantity` and `three hundred:quantity`.

One of those ambiguities is real, and one is an artifact of category promotion. The
model correctly predicts that *two* cannot coordinate with *three hundred miles* because the latter is not a bare quantity, but a length.

### 4.3 Challenges

In this section I describe problems with the current state of the lexicon, with Sepia’s architecture, and most fundamentally with CCG as a parsing formalism in terms of what it cannot capture.

#### 4.3.1 A Hundred and Ninety One Revisited

In the first chapter, I introduced an example, “Their scores were a hundred and ninety-one respectively.” In the first section of this chapter I showed how to process “a hundred and ninety-one” into 191. The challenge is to understand that there are two separate numbers, 100 and 91, in the sentence containing “respectively”.

The adverb “respectively” intuitively changes a verb’s default way to map conjoined subjects and objects from an all-pairs mapping to an ordered list mapping. “respectively” thereby causes the verb to require that both its subject and object be coordinations of the same cardinality.

Sepia’s lexicon does not yet have verbs, so I am forced to use a counter-linguistic solution for now, noting that unlike other systems, there is a solution! I make the simplifying assumption that “respectively” will appear after the verb (as opposed to “They respectively scored a hundred and ninety-one.”) and create two patterns which catch a conjunction:

```scheme
;;; respectively : * / *
;;; respectively : * \ *
(lambda (c) (if (instanceof? c Coordination)
   (.setRespectiveMapping c)))
```

This works because of two key features in Sepia: In the first place, if the argument
(which can be of any category) is not a Coordination logical form, then the semantics "fails", returning nothing, and the partial parse is dropped (Section 3.3.4). Thus the numeric interpretation of a hundred and ninety one cannot combine with respectively because it is underlyingly an AtValSet, not a Coordination.

Second, in the final parser decision, partial parses which wholly subsume other partial parses occlude the subsumed partial parse (Section 3.4.5). Because a hundred and ninety-one respectively strictly encloses a hundred and ninety-one, the latter will not be considered for combination. This applies transitively to any combination with a hundred and ninety-one: unless the combination finds another way to use respectively, it will be smaller than a combination which does use respectively, so the correct interpretation will win.

I hope to revisit this issue after developing a verb lexicon to explore a more linguistically satisfying solution. Even so, it is satisfying to have such problems in an implemented system.

### 4.3.2 Overgeneration of Coordination

One problem I came across in developing a lexicon for Sepia is that coordination overgenerates in some instances. In order to parse "Mr. and Mrs. Hill", I specify "Mr." and "Mrs." as functions over a name.

```lispy
(; Mr. : person / name
(; Mrs. : person / name
(lambda (name) (.append name 'title *0*))
```

Mr. then conjoins with Mrs. to create a new function that takes a single name and generates a coordination of people: Mr. Hill and Mrs. Hill.

"Jimmy and Rosalyn Carter" can be conjoined two ways. Common first names look for a last name to apply to, so they are functions that can conjoin. A common last name looks for a first name to apply to, and finds the conjoined firstnames as
arguments.\footnote{...and generates another version of the same coordination. As an aside, because the two coordinations are \texttt{.equal()}, the \texttt{HashSet} that stores logical forms will silently drop one.}

But the same mechanism causes problems:

- The meeting between Mr. Hill and DeGeneres yesterday...

- Prince Charles and Fergie

Clearly there is no relevant Mr. DeGeneres, nor a Prince Fergie.

In fact, English has special forms to indicate that a title may apply to multiple names: “Drs. Brown and Whitman”

This problem was a surprising finding for me, but is apparently well known in the literature. Colin Phillips gives one comprehensive account of such overgeneration phenomena, and suggests that parsing in linear order (strictly left to right) is a solution \cite{56}.

I was very happy to find this literature after the fact, because it indicates that Sepia was useful in finding one theoretical problem with CCG theory, and will hopefully be useful in establishing the solution and finding other linguistic problems. I look forward to implementing Colin Phillips’ solution as future work.

### 4.3.3 Coordinated Nominals

One goal in the ACE task is to understand nominals, including coordinated nominal expressions. A nominal must generate as many entities as it has heads.

- two heads $\Rightarrow$ two entities: “Two accountants and three linguists”

- one head $\Rightarrow$ one entity: “several descriptive and generative linguists”

The first case will generate two entities via the mechanism previously described.

Assuming that \textit{descriptive} and \textit{generative} are functions on \textit{linguists}, the second phrase would create a \texttt{Coordination} equivalent to “several descriptive linguists and several generative linguists”. This is clearly not what is intended in the ACE guidelines, and arguably not what is meant by the speaker. Again, the coordination would
have to have a different mapping (apply both functions serially to create a single new object) than usual. It is not clear how one would know which mapping to specify for any given function, or what the mechanism for specifying it in the lexicon might be. Perhaps * is too general a category, and each type of entity should specify the particular flavors of coordinations that may apply to it. Then adjectives like "French" and "German" could specify that they unlike other adjectives should apply in parallel: "the French and German soccer teams", "the old and young inhabitants", "the first and second year students".

4.3.4 Noncontiguous Constituents

As mentioned in Section 4.3.1, there is not currently a satisfactory treatment of verbs and adverbs. This can get in the way of recognizing nominal expressions that happen to have an adverb in the middle, e.g., "I met some people yesterday who love chess".

Ideally we would recognize "people who love chess" as a nominal expression. This is impossible in the current framework because elements can only combine with immediately adjacent ones, so "people who love chess" simply cannot form a constituent. One solution that would be supported would be an "ignore-me" marker. The semantics for an adverb could include adding links in the lattice around it, thereby making it possible for the normal rules to apply as if the element weren't there. Another might be an explicit movement, as the lattice also allows remove and insert operations, while preserving adjacent link structures. As our team moves to cover more of language, including pronouns and quantifiers, more sophisticated structural transformations may become necessary.

4.3.5 Mention Detection and Coreference

Sepia should find named, nominal, and pronominal mentions of a named entity within a text using the semantic features of the entities in question. The semantic match operator, now primarily used to ensure that parsed logical forms match the logical forms specified in their test cases, is one possible way of determining which entities
should be coreferenced. The match operator defines a form of entailment. It is asymmetric and transitive. For an AtValSet, \( a \) matches \( b \) if \( b \) has every key in \( a \), though it may have more, and for every key has at least the values in \( a \) in the same order. For a Coordination, \( a \) matches \( b \) if the coordinations have the same type and a matching set of elements in any order. As a first approximation, entities which match either way should be marked as possibly coindexed. But crucially, if two entities match a third but do not match each other, then they must not be coindexed, and the index of the entity they both match will be in question.

### 4.3.6 Real Ambiguity

Consider the sentence, "On August twenty first, two thousand protesters assembled at Hawaii’s state capitol building." A transcribed speech corpus, for example, might have numbers spelled out in this way. It is not clear whether there was an unspecified number of protesters there in the year 2000, or whether the current year was assumed, and the protesters numbered 2000. Whether a speech recognizer was able to recognize enough of a pause to generate a comma after “two thousand” or not is an unreliable way to decide between interpretations.

Similarly, for the ACE task, "...the State Department spokesman Uno Little..." must be marked as two entities (as if it were an apposition), rather than one entity for, "...told State Department spokesman Uno Little that...". They argue that there might be a missing comma after “spokesman” in the first example. It is not clear that the two are coindexed at all: consider, “the State Department spokesman Uno Little spoke with yesterday”.

In the first case, if Sepia recognized the possible date including year, and it recognized the phrase “two thousand protesters” as a group of people, it would still not output “August twenty first” as a possible date because it is entirely subsumed by the larger date. Only if it understood the whole sentence (incorporating both date and protesters into a single top-level partial parse) would the smaller date become available to a later stage in processing (if any).

The second case is similar: the only way “Uno Little” would be considered a name
mention without the title would be if a larger constituent than “State department spokesman Uno Little” included both. In this case though, it’s a feature rather than a bug.

4.3.7 Domain Recognition and an Adaptive Lexicon

Creating domain-specific lexicons and recognizing the domain of a particular text segment has been a very successful approach to increasing performance. This is difficult to do as part of Sepia, though Sepia could certainly be called with separate lexicons in different instances. This approach is particularly helpful in recognizing the metonymy in a sports-text instance like “Boston defeated Miami 21 to 15”. I hope to develop some alternative in Sepia that will accomplish similar behavior without a strong separation of lexicons.
Chapter 5

Evaluation on Question Answering

In the previous chapter, I showed examples of how Sepia handles individual test cases. In this chapter, I will show how Sepia has performed in intrinsic and extrinsic evaluations related to query expansion, and how I approached the new TREC list question set.

In a test of performance on the manually annotated question set from the 9th, 10th, and 11th TREC competitions, Sepia annotated named entities with an 85% F-measure against a MUC-like annotation standard\(^1\). A detailed error analysis is reported below; the character of the errors suggests that lexicon development time is the constraining factor.

As we prepare for the upcoming Automatic Content Extraction common evaluation, the lexicon remains under active development. The same set of questions as used above, annotated according to much more stringent ACE guidelines, and using a lexicon three weeks newer, today yielded 25% recall, up from 14% yesterday.

In end-to-end evaluation within a question answering system, I integrated Sepia into two question answering components. In query expansion a few simple strategies on identified names marginally improved document ranking. In list question answering, Sepia was used to identify named entities in retrieved passages, but there were too few relevant questions to make a statistical analysis. In both end-to-end anal-

\(^1\)This cannot be directly compared against the MUC-6 results, where other knowledge-based systems scored 76% to 86% F-Measure, because this is a different corpus, without official judgements.
yses, as predicted in Related Work, the common problem was that the task could not be greatly improved by even the best Sepia results, partly because named entity understanding is not the limiting factor, and partly because only very shallow understanding was actually used.

Sepia's results are promising, though not yet competitive with more established systems. I hope to improve Sepia's performance significantly in preparation for the ACE competition, and will integrate Sepia into our larger question answering systems in ways that will hopefully have greater impact on end-to-end performance.

5.1 Named Entities in Queries

The first numerical evaluation of Sepia was an intrinsic evaluation, designed to test how well Sepia performs in the task for which it was created. Having created a lexicon for Sepia based on AQUAINT documents, I chose to evaluate that lexicon on a different but related corpus: the set of TREC questions. By evaluating absolute performance on the same corpus that I later used for the query expansion test, I hoped to shed light on specific error types in both.

5.1.1 Experimental Protocol

The annotated corpus was the set of 1321 questions from the 9th, 10th, and 11th TREC competitions. I was developing the lexicon on actual documents from the AQUAINT corpus, but had not looked at the questions as part of development prior to August 8th.\(^2\)

Until August 8th, I had been developing Sepia lexicon based on the AQUAINT corpus, and aimed to correctly understand only name-level expressions\(^3\). Expressions were categorized in a manner more consistent with the Message Understanding

\(^2\)I did see these questions during last year's TREC competition, but not in the intervening 11 months, and did not remember more than a handful of famous examples.

\(^3\)an ACE term indicating that the name of the entity should appear in the named entity mention (Presidential hopeful Howard Dean"), as opposed to a nominal phrase with the same referent ("the Presidential hopeful") or a coreferring pronominal reference ("he")
Conference (MUC) guidelines than with the newer Automatic Content Extraction guidelines. In particular, markable categories included locations, geo-political entities (though these were labelled as locations, and country adjectives were not marked), persons (only singular, no groups of people represented as person), and organizations, but not facilities.

The Sepia evaluation run used Sepia (parser and lexicon) in the version that was used for this year’s TREC competition, dated August 8th under revision control. The full question list, Sepia’s responses, and annotator judgements are shown in Appendix B. Where an entity had incorrect bounds, the error was counted only once rather than as one missed entity and one spurious entity.

5.1.2 Results

There are 1321 official questions from the previous three years of the TREC competition (numbered 200–1890, with gaps). There were 776 ground-truth entities and 711 recognized entities, of which 631 were correct.

\[
\text{Precision: } \frac{628}{711} = 88.33\% \\
\text{Recall: } \frac{628}{777} = 80.82\% \\
\text{F-Measure: } \frac{p+r}{2} = 84.57\% 
\]

The 83 incorrectly marked entities can be broken down as follows:

Miscategorization

There was no case where Sepia marked something where there was no entity, but it sometimes marked the wrong entity type or marked an entity that was not markable (such as a facility). There were in fact several facility keywords (such as gate, gates, stadium, and house) which were cues for organization and should not have been. There was a cue word “Star” for newspapers of that common name.

None of these took into account any context other than what is marked.

person: Where is [Santa Lucia]?
person: Who owns the [St. Louis Rams]?  
person: Where is [Ocho Rios]?  
person: When was the battle of [Shiloh]?  
person: What city does the [Tour de France] end in?  
person: ...member of the [Pink Floyd] band?  

location: ...[Bill Gates]...  
location: ...[New Coke]...  
location: ...‘Brave [New World]’’?  
location: ...sing at the [Lincoln] Memorial?  
location: American patriot Paul [Revere]?  
location: What are the two houses of the [Legislative branch]?  
location: [Which mountain] range in ...  
location: [Which river] runs through...  

organization: Where did the [U.S. Civil War] begin?  
organization: When did [Yankee Stadium] first open?  
organization: Where was the movie, ‘‘Somewhere in [Time]’’ filmed?  
organization: What was the original name before ‘‘[The Star] Spangled Banner’’?  
organization: What is another name for the [North Star]?  
organization: What is the street address of the [White House]?  
organization: When was the [White House] built?  
organization: ...first female [United States Representative]?  
organization: Where is the [Shawnee National Forest]?  

Missing rule to combine correctly recognized parts  

Sepia would often correctly identify a city and country adjacent to one another with a comma between, but I had forgotten to add a rule that combined names with countries, as I combined names with states. Thus examples like the following were
marked incorrect:

[loc: Madrid], [loc: Spain]
[loc: London], [loc: England]
[loc: Bombay], [loc: India]
[loc: Rome], [loc: Italy]

Windsor, [loc: Ontario]
Parsippany, NJ?

[loc: Kentucky] [loc: Horse Park]

There was one instance of a name not having a hyphen-combination rule:

[person: Rosanne Rosanna]-[person:Dana]

Name heuristic made incorrect decision

The only statistical code in the system was explicitly specified by the lexicon: a unigram count from the AQUAINT corpus for how often a word appeared capitalized versus uncapitalized serves as a naive heuristic for whether something is a good name. If Sepia had had more relations, it would have made sense to simply posit all reasonable names and let the grammar decide whether two words should be a name or not. By resorting to this simple heuristic, Sepia made predictable mistakes.

In each line below, Sepia tagged the bracketed portion as a name, but decided that the other portion appeared in lower case often enough that it was unlikely to be part of a name, but might instead be a headline word. This heuristic is not reasonable in the question analysis genre, where capitalized words rarely appear except as part of a proper name, but I did not make any attempt to adapt to the new genre.

Babe [Ruth]
[Henry] Clay
[Aaron] Burr

A burr is a dental instrument.
[Jude] Law
[Nicholas] Cage
[Billy] the Kid
[Winnie] the Pooh
[Sharon] Stone
[Prince Andrew] and Fergie (missed Fergie)
Teddy [Roosevelt]
[Marco] Polo
[Robert] Frost
[Tom] Cruise

In these cases the location appears only because the name did not subsume both tokens, again because the word was otherwise often lower case.

[Judy] [location: Garland]
[Paul] [location: Revere]

In one case, the heuristic overgenerated:

''What university was [person: Woodrow Wilson President] of?

Punctuation bug

A bug I had not noticed before this experiment had to do with recognizing entries that consisted of multiple sub-tokens when those sub-tokens did not make up all of a larger token.

In this case the recognizer for initials did not recognize them because there was no space between “C.” and “S.”.

C.S. [Lewis]
[Scarlett O]’Hara’s
U.S.?
Washington D.C.?
Parsippany, NJ?

[Mount St]. Helens? (Helens not recognized as a capitalized-word!)

However when I fixed this bug, a new incorrect person appeared:
Who is the evil [H.R. Director] in "Dilbert"?

Missing Cue Words

There were many location and organization keywords missing which would have aided recognition. After this evaluation, we looked at lists of such cue words compiled statistically from a large corpus, and we would have done well to incorporate such lists.

agency and company were missing organization cues. Less obvious missing organization cues would be United x or the whole set of sports terms Series, Cup, Crown, Bowl, Derby, etc.

Location cue words, such as Alley, Mountains, and galaxy would be useful, but even more useful might be encompassing rules like battle of <location> or directly for questions Where is <location>

I had not noticed a person's relations in looking at the training corpus, so <person>'s son, wife, mother, etc. were not in the lexicon.

Mother was not an available title, so “Mother Angelica” was missed. “Vasco da Gama” was missed because I did not have a linking “da”, despite having “de”, “du”, “di”, and many others as possible name linking cues.

How did <person> die? occurred twice in the questions, and my name heuristic had passed over both people. A variant on this was common in the training corpus, but as a negative example in headlines, I had to avoid <title> Dies as a person.

Similarly, Sepia could incorporate both internal information about sports team name structures and explicit external cues like “sports team”, from examples like,

What kind of a sports team is the [loc: Wisconsin] Badgers?
What kind of sports team is the Buffalo Sabres?

Missing List Items

Gazetteers of people, fictional characters, important artifacts, and other items
difficult to identify heuristically would have helped. There is little other way to know that a “Declaration of Independence” is not an unknown declaration from a place called “Independence”, or that “Winnie the Pooh” is a fictional character, and not a person, while “Billy the Kid” is a person.

The North and South poles are probably best entered as list items, but then it becomes impossible to recognize “North and South Poles”. Instead, there should be very specific rules that indicate what sorts of geographic poles one might talk about.

Additional list items would likely be useful in the following missed cases as well:

Who is Darth Vader’s son?
When is Snoop Dog’s birthday?
How did Mahatma Gandhi die?
How long did Rip Van Winkle sleep?
When was the Triangle Shirtwaist fire?
What is the Islamic counterpart to the Red Cross?
How tall is the [Sears] Building?
Who invented the instant Polaroid camera?
Where was the first McDonalds built?
How much are tickets to Disney World?
Where did Kublai Khan live?
What year was the first [Macy’s] Thanksgiving Day Parade held?
What is Buzz Aldrin’s real first name?
What award did [Sterling North]’s book "Rascal" win in 1963?
What TV series did Pierce Brosnan play in?

5.1.3 Discussion

There were two primary types of mistakes in the results above: missing lexical items, and incorrect statistical name decisions. The name heuristic was meant to be the most trivial possible guess, and remained in the lexicon longer than intended because it worked surprisingly well for the trivial amount of effort it took to write. In replacing it, I would like to use a more principled approach.
The missing lexical items are easy to add, but perhaps harder to find. A process of iterative refinement works well for finding cue words in the training corpus, but it might be well complemented by a statistical tool that found words similar to the cue words in a large corpus of the same genre, in an unsupervised way. The lists of candidate cue words, their relative frequencies, and their concordance data from the corpus might serve as an effective way to discover the most relevant cue words and surrounding syntactic alternations.

Building larger lists of cue words will increase recall, but will also increase overgeneration. For example, “The Star” may be a wonderful name for a newspaper, but is inappropriate in the quoted string, ““The Star Spangled Banner””. In Sepia, the most effective way to curb overgeneration is to have rules to understand the larger context of the candidate. In this example, both having the song on file and recognizing quoted strings might help. For other examples, a knowledge of verb selections might be key: consider whether the White House as organization or facility is more likely to be built, or to have a street address, or to veto a bill, or to make an announcement. Therefore, adding verb frames from a source like PropBank or an LCS lexicon will be a high priority in creating the competition system.

In preparing for the competition system, we have annotated the same set of questions according to ACE annotation guidelines, and currently have 24.56% recall of the new annotations. While others report 70% performance, we have been creating lexicon for this task for only a matter of days.

We did not rerun this evaluation, of course, because that would be mixing training data with test data, but for a more positive outlook on cases we handle well, please see Appendix A. It contains our regression test suite, including examples of semantics we correctly parse.

5.2 Integration and End-to-End Evaluation

We integrated Sepia into two components of our TREC 12 / AQUAINT Question Answering competition entry: Query Expansion and List Questions. Sepia became
ready for integration too late to be used as part of our primary factoid question answering system. In the query expansion module, Sepia served as one of several annotators to mark phrases in the question that should be treated as unbreakable units. In the list question module, Sepia’s category distinctions were linked to a question’s analyzed focus, and annotated retrieved passages to find candidate list items.

5.2.1 Query Expansion

We thought that Named Entity understanding would be able to improve question answering via query expansion by identifying named entities in the query and specially treating them. With normal keywords, our boolean query system required all keywords to be present in a document, but allowed inflected and derivational morphology variants in place of a query term. When a string was identified as a named entity, these expansions were no longer allowed. When a multiword string was identified as a named entity, then the query required either the exact string of the multiword entity, or its component words in proximity of two words from each other. When a named entity was a person, the query expander allowed all first names to be dropped, but not last names.

Sepia was not the only phrase-identification component in our query expansion system. A gazetteer, Omnibase, identified many fixed strings, there was a simple part-of-speech-based heuristic noun phrase tagger, many verb-particle constructions were also tagged, and quoted strings were identified and treated similarly.

Our first experiment was to compare the query expander with no phrase identifiers with the query expander using only Sepia. Indeed, Sepia made a relative improvement over baseline. Note that Mean Reciprocal Rank (MRR) is worse when it is higher,
and Sepia in this case made the MRR worse.

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>Baseline + Sepia</th>
<th>Relative Improvement</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>2.1%</td>
<td>2.5%</td>
<td>16.6%</td>
<td>0.00095</td>
</tr>
<tr>
<td>Recall</td>
<td>79.8%</td>
<td>80.9%</td>
<td>1.5%</td>
<td>0.10</td>
</tr>
<tr>
<td>MRR</td>
<td>0.188</td>
<td>0.204</td>
<td>8.16%</td>
<td>0.0045</td>
</tr>
</tbody>
</table>

We then measured the performance of all components against the performance of all except Sepia. Here, Sepia seems to have made a marginal improvement in both recall and MRR.

<table>
<thead>
<tr>
<th></th>
<th>All - Sepia</th>
<th>All</th>
<th>Relative Improvement</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>4.25%</td>
<td>4.37%</td>
<td>2.81%</td>
<td>0.19</td>
</tr>
<tr>
<td>Recall</td>
<td>80.66%</td>
<td>81.41%</td>
<td>0.93%</td>
<td>0.0021</td>
</tr>
<tr>
<td>MRR</td>
<td>.204</td>
<td>.200</td>
<td>2.05%</td>
<td>0.0082</td>
</tr>
</tbody>
</table>

However, of the 711 Sepia answers given, the query expansion system only used 447 (others were overridden by other methods), and only 326 of the entities were multi-word. In comparing performance on only Sepia-marked questions, and comparing performance on only multi-word annotated questions there were no significant differences.

The worse MRR measure of Sepia against baseline can be best explained by noting that when Sepia annotates a phrase, the phrase’s inflectional variants are no longer counted toward the document score. Thus correct documents with multiple inflected mentions would no longer be scored higher than documents without multiple mention forms, and so would lose place in rank.

The only part of Sepia’s semantics that this measure used, however, was a distinction between first and last names. For future evaluations, it would be important to find more semantic ways to expand queries.

### 5.2.2 List Questions

Sepia was integrated into list questions by annotating candidate passages retrieved by the document retrieval engine. The list answer component, knowing about some
of Sepia’s finer grained classes like for example corporate-organization, government-agency, educational-organization, news-organization, or religious-person, would use these subcategories if they appeared as the focus of a question. Unfortunately there were only three questions out of the 50 where a Sepia class was the focus of the question. Where the Infolab’s Omnibase gazetteer didn’t already mark entities to cover the focus (as with countries and cities). These three questions were:

31: List companies that manufacture tractors

38: What are institutions that have conferred an honorary degree on Nelson Mandela?

45: Who are authors who have written books about near death experiences?

In all of these cases, a simpler part-of-speech-based noun phrase chunker found the same entities as Sepia did, so the performance did not change due to Sepia.

Results are promising in that Sepia understood many of the correct responses, and a major step forward in this task would be to integrate Sepia into the question analysis portion as well as the candidate generation portion of list question answering, so that the mapping between question and answer semantics is more easily usable.
Chapter 6

Future Work

I have mentioned opportunities for future work throughout this thesis, but will collect them here, from the most urgent short range extensions of the existing system, to longer range research goals based on Sepia's central theme: compositional semantics without a pipeline.

6.1 ACE

The ACE competition will begin on September 29th, 2003, and each competitor will have five days to run their systems on the provided test data and return the results for testing.

There are three major extensions I will undertake for this competition: adding large gazetteer, a typographic case repair system, and a coreference system.

The first is motivated by a need to add large lists of known named entities including famous people book titles, locations, and facilities. The Infolab already has such large lists, and a well established database service for them, called Omnibase [40]. My first priority will be to add JDBC as an access method for the lexicon, and thus define large classes of entities without memory overhead. In addition, it would be useful to obtain data from the Alexandria Digital Library Project [18] with respect to known organizations and facilities.

Because poorly cased text plays such a large role in ACE, a second extension
would be an automatic case repair mechanism. We could rewrite parts of the lexicon to be less sensitive to case, or Sepia could use a preprocessing step such as Lita et al.'s truecaser [48]. The first would fit the theme better but the second might be more prudent, if it is possible, given the time remaining.

Finally, I hope to implement a minimal coreference system that would use the existing match operator to find compatible names and nominals (See Section 4.3.5), and to add a heuristic pronominal resolver.

6.2 Formalism

In the longer term, there are several fundamental changes I would like to make to Sepia's CCG parser.

The most obvious extension is addition of probability values to each lexical item, not in the visible lexicon for human consumption, but as an attribute that could be trained on a corpus. The more difficult part is to find a corpus to train the parser on, but Hockenmaier and Steedman have shown promising results in this area [36, 35].

Another extension to the formalism would be to add unification or linguistic feature-checking to a set of syntactic features that could be stored with every logical form. This would give a way independent of semantics to ensure consistency of person, number, gender, and other syntactic features. Also, by removing these from the semantics into a separate mechanism, it might become easier to "back off" from the purely syntactic constraints, as humans apparently do routinely, when they conflict with the semantic interpretations.

The third set of improvements would address some of the issues of overgeneration, psychological plausibility as a language model, and efficiency issues related to allowing more ambiguity than pipelined systems. I would like to explore and implement formalism changes such as Jason Eisner's normal form for CCG [25], Jason Baldridge's CCG formalism extension, Set-CCG [4], and parsing only in linear order [56]. I would also like to learn more about and model other psycholinguistic predictions, including for example Dependency Locality Theory [29].
6.3 Extending Sepia: Bigger Concepts and Smaller Tokens

In pursuing compositional semantics without a pipeline, it makes sense to attempt to build larger propositional structures, such as Lexical Conceptual Structures, starting from a phonemic or morphemic level. The advantage of building larger structures is simply better understanding and better pruning of incorrect hypotheses by their semantic context. The advantages of starting with smaller units of meaning than words include better portability to other languages, better robustness to noisy input such as broadcast news or even speech, and a framework in which to explore the compositional semantics of subword elements.

6.4 Integration with Vision, Common Sense

A final exciting future direction would be to use Sepia's Java interface capabilities to communicate with vision and motion understanding. In one scenario, Sepia could act as a query front end to a large video archive, translating user questions into the CLiViR formalism [41].

In another scenario, Sepia could have constant bidirectional communication with a vision and imagination system, as described in Winston's Bridge Project [10, 66]. Sepia would both accept visual-semantic constraints and generate language-semantic constraints in a common infrastructure designed to let the combined language and visual system understand things which would be meaningless or ambiguous to either component alone.

Integration with common sense applications would also be interesting. I would like to explore using the OpenCyc [17] and OpenMind [60] projects as sources of semantic plausibility. Moreover, Sepia could be used in some cases to help generate candidate generalizations like "The Secretary of State works for the Department of State" and "The Department of State is the same as the State Department" for users of a system like OpenMind to manually verify.
Chapter 7

Contributions

The goal of the Infolab is computer understanding of natural language, focusing on natural language question answering as a development application. This thesis contributes to those goals in several ways. I have:

- Framed the content-extraction task as a problem in compositional semantics.
- Designed and implemented Sepia, a pilot framework for research in compositional semantics.
- Suggested named entity understanding as a rich new source of test cases for computational semantics.
- Implemented a named entity understanding component usable in question answering for query understanding, query expansion, and identification of candidate answers.
- Integrated Sepia with our question answering framework for participation in AQUAINT/TREC 12 Question Answering evaluation.
- Identified key areas of progress necessary to do well in the upcoming Automatic Content Extraction evaluation.

I hope in the coming weeks to achieve state-of-the-art performance in named entity understanding, as measured by the ACE entity detection and tracking task, and thus
to provide a foundation for simple semantic inference in future question answering studies. The preliminary results of evaluation are promising, though not yet competitive with established systems. Integration with question answering components yielded some valuable lessons.
Appendix A

Test Cases

The following are the test cases currently in use for regression testing Sepia. They serve to illustrate in detail the current capabilities of Sepia. The format of these test cases is a four column tab separated file. In the first column is a string to be processed. In the second column is a category of entity that must have been recognized, preceded by a '!' character if the testcase is negative to ensure that the specified semantics were not recognized. The third column is the portion of the original string that must be recognized. The final column indicates the semantics that the recognized entity must match in order to be counted correct. The match operation for AtValSets requires that all of the attributes and contents in the test plan also be present in the data structure, but not vice versa. Text after a comment mark ('#') is ignored.
A.1 Person

TYPE person

<< some common overgeneration bugs
Panel OKs Plan for Reagan Monument By CARL ARTMAN WASHINGTON (AP) person Reagan Monument (AtValSet.)
U.S.A. Reagan License Plates Approved LOS ANGELES (AP) person Reagan License Plates (AtValSet.)
but former President Ronald Reagan's spirit infuses Campaign 2000 person President Ronald Reagan (AtValSet. 'name "Reagan"

on May 24 we called a General Assembly for several reasons person May (AtValSet.)
on May 24 we called a General Assembly for several reasons person General Assembly (AtValSet.)

>>

<< standard name tests
Dr. Michael Collins person Dr. Michael Collins (AtValSet. 'name "Michael" 'name "Collins" 'title "Doctor" 'gender 'male)
Abraham Lincoln person Abraham Lincoln (AtValSet. 'name "Abraham" 'name "Honest Abe" 'name "Lincoln" 'occupation 'us-president 'gender 'male)
H. Ross Perot person H. Ross Perot (AtValSet. 'name "H." 'name "Ross" 'name "Perot")
Louis de Luce person Louis de Luce (AtValSet. 'name "Louis" 'name "de" 'name "Luce")
Louis deluca person Louis deluca (AtValSet. 'name "Louis" 'name "deluca")
Moan Chonksy, Ph.D. MD. DDS person Moan Chonksy, Ph.D. MD. DDS (AtValSet. 'name "Moan" 'name "Chonksy" 'degree "Ph.D." 'degree "M.D." 'degree "D.D.S.")
Edward "Blackbeard" Teach person Edward "Blackbeard" Teach (AtValSet. 'name "Edward" 'name "Blackbeard" 'name "Teach")
W. E. B. Dubois person W. E. B. Dubois (AtValSet. 'name "W." 'name "E." 'name "B." 'name "Dubois")
David Fox-Grodzinsky person David Fox-Grodzinsky (AtValSet. 'name "David" 'name "Fox-Grodzinsky" 'gender 'male)
Mary Anne McDonald person Mary Anne McDonald (AtValSet. 'name "Mary" 'name "Anne" 'name "McDonald")
George Herbert Walker Bush III. person George Herbert Walker Bush III. (AtValSet. 'name "George" 'name "Herbert" 'name "Walker" 'name "Bush" 'generation 3)
Martin Luther King, Jr. person Martin Luther King, Jr. (AtValSet. 'name "Martin" 'name "Luther" 'name "King")

# these fail at the moment. (Rod Brooks I'd) seems to be a perfectly good name like O’kelly. :-)
# update: I fixed the capitalized_word pattern not to do the O’kelly thing, but until I get context,
# I don’t see how I’ll fix the l. I could hack it to check to it’s right, but then
# “Sure, [Pope Gregory I]’d do that…” but then ‘first’d sounds pretty bad and looks worse,
# so I might say “would” here, and save ’d for “Gregory I’d do that”.
# I told Rod Brooks I’d graduate person Rod Brooks (AtValSet. 'name "Rod" 'name "Brooks")
# I told Rod Brooks I’d graduate person Rod Brooks (AtValSet. 'name "Rod" 'name "Brooks")

# Interim Dean of the School of Electrical Engineering and Computer Science Howard Dean.
# Howard E. Dean Professor of Physiology Michael McGeechie
# Mr. and Mrs. Howard Dean
# the Chamberlains
# the Deans (?!?)

<< some of my friends with Polish and other weird names
Michael J. T. O'Kelly person Michael J. T. O'Kelly (AtValSet. 'name "Michael" 'name "J.T." 'name "O'Kelly")
Mehmet Can Yuran person Mehmet Can Yuran (AtValSet. 'name "Mehmet" 'name "Can" 'name "Yuran")
Elle. Elke Hoden person Elle. Elke Hoden (AtValSet. 'name "Elle" 'name "Hoden")
Anna Wniekiewski person Anna Wniekiewski (AtValSet. 'name "Anna" 'name "Wniekiewski")
Age Stokowski person Age Stokowski (AtValSet. 'name "Age" 'name "Stokowski")
Stanislav Lem person Stanislav Lem (AtValSet. 'name "Stanislav" 'name "Lem")
Krzysztof Cegos person Krzysztof Cegos (AtValSet. 'name "Krzysztof" 'name "Cegos")
Pearl Harbor person Pearl Harbor (AtValSet.)
Pearl Hill person Pearl Hill (AtValSet. 'name "Pearl" 'name "Hill")
MIa Heavener person MiA Heavener (AtValSet. 'name "Mia" "Heavener")

a bunch of ministers

- former security affairs Minister Gen. Wiranto
- Palestinian International Cooperation Minister Nabil Shaath
- Defense Minister Paul Salzar
- South Korean Defense Minister Cho Sung-tae
- Education Minister Kweyli Kremen
- Environmental Minister Delia Itzik
- Venezuelan Environmental Minister Jesus Perez said
- Serbian Environment Minister Branislav Blazic
- former Finance Minister Yaakov Neeman
- Mozambican Foreign Minister Leonardo Simao
- With Syrian Foreign Minister Farouk al-Sharaa recovering
- Syrian Foreign Minister Farouk al-Sharaa
- Foreign Minister Farouk al-Shares of Syria

the Syrian foreign minister, Farouk al-Sharaa
the Syrian foreign minister, Farouk al-Shares
Syria's Foreign Minister Farouk al-Sharaa

when Iran's Foreign Minister announced

Pakistan's Foreign Minister Abdul Sattar
Foreign Minister Mate Granic
But Deputy Foreign Minister Mirs Handoon
Then-Foreign Minister Ariel Sharon
Vice Foreign Minister Kim Gye Guan
against former Health Minister Wassana Zuma in November
Health Minister Manto Tshabalala-Msimang is
But Immigration Minister Philip Ruddock said
Interior Minister Yuriy Kravchanko
Interior Minister Moizuddin Haider
from Interior Minister Prince Mayef

Flores, the vice-Interior Minister

Information Minister Goran Ratic
former Intelligence Minister Ali Fallahian
Industry Minister Suvat Liptapallop
Justice Minister Amanda Vanstone
Israel Justice Minister Yossi Beilin
Australian Justice Minister Amanda Vanstone
Federal Justice Minister Amanda Vanstone
Oil Minister Amer Mohamed Hashid
Prime Minister Vladimir Putin
Prime Minister Tony Blair
Prime Minister Jean Chretien

person security affairs Minister Gen. Wiranto (AtValSet. 'name "Wiranto" 'title
person Palestinian International Cooperation Minister Nabil Shaath
person Defense Minister Paul Salzar
person South Korean Defense Minister Cho Sung-tae
person Education Minister Kweyli Kremen
person Environmental Minister Delia Itzik
person Venezuelan Environmental Minister Jesus Perez said
person Serbian Environment Minister Branislav Blazic
person former Finance Minister Yaakov Neeman
person Mozambican Foreign Minister Leonardo Simao
person Syrian Foreign Minister Farouk al-Sharaa
person Palestinian International Cooperation Minister Nabil Shaath
person the Syrian foreign minister
person Farouk al-Sharaa
person Syria's Foreign Minister Farouk al-Sharaa
person Iran's Foreign Minister
person Pakistan's Foreign Minister Abdul Sattar
person Foreign Minister Mate Granic
person Deputy Foreign Minister Mirs Handoon
person Then-Foreign Minister Ariel Sharon
person Vice Foreign Minister Kim Gye Guan
person former Health Minister Wassana Zuma
person Health Minister Manto Tshabalala-Msimang
person Immigration Minister Philip Ruddock
person Interior Minister Yuriy Kravchanko
person Interior Minister Moizuddin Haider
person the Syrian foreign minister
person Farouk al-Sharaa
person Syria's Foreign Minister Farouk al-Sharaa
person Iran's Foreign Minister
person former Health Minister Wassana Zuma
person Health Minister Manto Tshabalala-Msimang
person former Intelligence Minister Ali Fallahian
person Industry Minister Suvat Liptapallop
person Justice Minister Amanda Vanstone
person Israeli Justice Minister Yossi Beilin
person Australian Justice Minister Amanda Vanstone
person Federal Justice Minister Amanda Vanstone
person Oil Minister Amer Mohamed Hashid
person Prime Minister Vladimir Putin
person Prime Minister Tony Blair
person Prime Minister Jean Chretien
Prime Minister Atal Bihari Vajpayee
Prime Minister Sirimavo Bandaranaike
Prime Minister Hun Sen
Russian Prime Minister Vladimir Putin
Belarusian Prime Minister Grigoriy Miasnikov
Israeli Prime Minister Ehud Barak
Indian Prime Minister Rajiv Gandhi
India's Prime Minister Atal Bihari Vajpayee
Tuvalu's Prime Minister Ionatana Ionatana
former Indian Prime Minister Rajiv Gandhi
South Korean Prime Minister Kim Jong-pil
Thai Deputy Prime Minister Supachai Panitchpakdi
former Prime Minister Viktor Chernomyrdin
former Prime Minister Shimon Peres

warned former Prime Minister and Nobel Peace Laureate Shimon Peres
Former Prime Minister Margaret Thatcher
Former Deputy Prime Minister Serhiy Tyhypko
Thursday, Prime Minister Lionel Jospin said
then-Prime Minister Shimon Peres
Production Minister Juan Falconi
On Wednesday, State Property Minister Farit Gazizullin
Egyptian Water Minister Mahmoud Abu Zeid
by Papua New Guinea's Minister for Bougainville Affairs Michael Hill

HEADLINES: yeah, these are hard. I don't really expect to get them.
in the Prime Minister's office
Iran Minister Chided for UK Visit
Iranian Minister Visits Britain
Natural Resources Minister To Meet With Striking Miners
S. Korea Approves Prime Minister SEOUL, South Korea
a pair of occupations this guy had:
Minister Cop Charged in Death
Ex Japan Education Minister Dies TOKYO
Wife of IOC President Dies at 67

"Deputy Minister" is the name of a horse: (ouch)
The top yearling to sell on Tuesday was hip number 599, a dark bay or brown colt by Deputy Minister out of the Seattle Slew mare Slew Boyers

# combine with occupations and nationalities
# astronaut person astronaut (AtValSet. 'occupation 'astronaut)
Russian astronaut person Russian astronaut (AtValSet. 'occupation 'astronaut)
astronaut John Glenn person astronaut John Glenn (AtValSet. 'occupation 'astronaut)
American astronaut John Glenn person American astronaut John Glenn (AtValSet. 'occupation 'astronaut 'from "The United States of America")

# combine with subject-extracted RC's
The first student to do a presentation was Alex. person The first student to do a presentation (AtValSet. 'occupation 'student 'definite #)
The first student doing a presentation was Alex. person The first student doing a presentation (AtValSet. 'occupation 'student 'definite #)
The first student who is doing a presentation will win a prize. person The first student who is doing a presentation (AtValSet. 'occupation 'student 'definite #)
The first student that is doing a presentation will win a prize. person The first student that is doing a presentation (AtValSet. 'occupation 'student 'definite #)
The first student who does a presentation will win a prize. person The first student who does a presentation (AtValSet. 'occupation 'student 'definite #)
The first student who did a presentation will have to present again. person The first student who did a presentation (AtValSet. 'occupation 'student 'definite #)
A.2 Locations (including GPEs)

**TYPE location**

New Mexico location New Mexico (AtValSet. 'name "New Mexico" 'type 'us-state)
New York location New York (AtValSet. 'name "New York")
NY location NY (AtValSet. 'name "New York")
Tex. location Tex. (AtValSet. 'name "Texas")
P.R. location P.R. (AtValSet. 'name "Puerto Rico")
Ukraine location Ukraine (AtValSet. 'name "Ukraine")
US location US (AtValSet. 'name "The United States of America")
USA location USA (AtValSet. 'name "The United States of America")
U.S. location U.S. (AtValSet. 'name "The United States of America")
U.S.A. location U.S.A. (AtValSet. 'name "The United States of America")
Chongqing location Chongqing (AtValSet. 'name "Chongqing" 'location "China" 'population 30000000 'type 'foreign-city)
Zliha city "zliha"
Chernihiv city "chernihiv"

Sun location Sun (AtValSet. 'name "Sun" 'type 'star)
Earth location Earth (AtValSet. 'name "Earth" 'type 'planet 'rank 3)

Atlantic Ocean location Atlantic Ocean (AtValSet. 'name "Atlantic" 'type 'water 'subtype 'ocean)

Mediterranean Sea location Mediterranean Sea (AtValSet. 'name "Mediterranean Sea" 'type 'water 'subtype 'sea)

Zambezi river "zambezi"
Alabama-Coosa river "alabama-coosa"

905 Main Aly street-address 905 Main Aly (905 (("Main") alley) . street)
905 Main Aly. street-address 905 Main Aly. (905 (("Main") alley) . street)

Cambridge, MA city-address Cambridge, MA ("Cambridge" . city) ("Massachusetts" . state)

Genesee River location_vater ("Genesee")
The Genesee River location_vater ("Genesee")

Lake Ontario location Lake Ontario (AtValSet. 'name (list "Lake" "Ontario") 'type 'water 'subtype 'lake)

Conesious Lake location Conesious Lake (AtValSet. 'name ("Conesious" "Lake") 'type 'geological 'subtype 'lake)

Mount Fishy location Mount Fishy (AtValSet. 'name ("Mount" "Fishy") 'type 'geological 'subtype 'mount)
Fishy Mountain location Fishy Mountain (AtValSet. 'name ("Fishy" "Mountain") 'type 'geological 'subtype 'mountain)

Cape Canaveral location Cape Canaveral (AtValSet. 'name ("Cape" "Canaveral") 'type 'geological 'subtype 'cape)

**TYPE gpe**

Ukraine gpe Ukraine (AtValSet. 'name "Ukraine" 'type 'country)
Johnston Atoll gpe Johnston Atoll (AtValSet. 'name ("Johnston" "Atoll") 'type 'us-territory)
MO gpe MO (AtValSet. 'name "Missouri" 'type 'us-state)
Alb. gpe Alb. (AtValSet. 'name "Alberta" 'type 'province)
American gpe American (AtValSet. 'name "The United States of America" 'type 'country)
Chuukese gpe Chuukese (AtValSet. 'name "Micronesia, Federated States of" 'type 'country)

91
A.3 Organizations

TYPE organization

Acron Beacon Journal organization Acron Beacon Journal (AtValSet. 'type 'news)
Asia Times organisation Asia Times (AtValSet. 'type 'news)
Baltimore Sun organization Baltimore Sun (AtValSet. 'type 'news)
Boston Globe organization Boston Globe (AtValSet. 'type 'news)
Cable News Network organization Cable News Network (AtValSet. 'type 'news)
Calgary Sun organization Calgary Sun (AtValSet. 'type 'news)
Channel News Asia organization Channel News Asia (AtValSet. 'type 'news)
Charleston Post Courier organization Charleston Post Courier (AtValSet. 'type 'news)
Charlotte Observer organization Charlotte Observer (AtValSet. 'type 'news)
Chicago Sun Times organization Chicago Sun Times (AtValSet. 'type 'news)
Christian Science Monitor organization Christian Science Monitor (AtValSet. 'type 'news)
Christianity Today Magazine organization Christianity Today Magazine (AtValSet. 'type 'news)
Cincinnati Enquirer organization Cincinnati Enquirer (AtValSet. 'type 'news)
Concord Monitor organization Concord Monitor (AtValSet. 'type 'news)
Daily Illini organization Daily Illini (AtValSet. 'type 'news)
Daily Telegraph organization Daily Telegraph (AtValSet. 'type 'news)
Detroit Free Press organization Detroit Free Press (AtValSet. 'type 'news)
Financial Times organization Financial Times (AtValSet. 'type 'news)
Greenville News organization Greenville News (AtValSet. 'type 'news)
Hindustan Times organization Hindustan Times (AtValSet. 'type 'news)
International Herald Tribune organization International Herald Tribune (AtValSet. 'type 'news)
Irish Examiner organization Irish Examiner (AtValSet. 'type 'news)
Irish Independent organization Irish Independent (AtValSet. 'type 'news)
Irish Times organization Irish Times (AtValSet. 'type 'news)
Jamaica Observer organization Jamaica Observer (AtValSet. 'type 'news)
Japan Today organization Japan Today (AtValSet. 'type 'news)
Kansas City Star organization Kansas City Star (AtValSet. 'type 'news)
Melbourne Herald Sun organization Melbourne Herald Sun (AtValSet. 'type 'news)
Middlesborough Evening Gazette organization Middlesborough Evening Gazette (AtValSet. 'type 'news)
Mobile Register organization Mobile Register (AtValSet. 'type 'news)
National Business Review organization National Business Review (AtValSet. 'type 'news)
National Post organization National Post (AtValSet. 'type 'news)
New Jersey Journal organization New Jersey Journal (AtValSet. 'type 'news)
New York Daily News organization New York Daily News (AtValSet. 'type 'news)
New York Times organization New York Times (AtValSet. 'type 'news)
Oakland Tribune organization Oakland Tribune (AtValSet. 'type 'news)
Orlando Sentinel organization Orlando Sentinel (AtValSet. 'type 'news)
Pacific Business News organization Pacific Business News (AtValSet. 'type 'news)
San Diego Union Tribune organization San Diego Union Tribune (AtValSet. 'type 'news)
San Francisco Chronicle organization San Francisco Chronicle (AtValSet. 'type 'news)
San Jose Mercury News organization San Jose Mercury News (AtValSet. 'type 'news)
Sky News organization Sky News (AtValSet. 'type 'news)
Strait Times organization Straits Times (AtValSet. 'type 'news)
Sydney Morning Herald organization Sydney Morning Herald (AtValSet. 'type 'news)
Taipei Times organization Taipei Times (AtValSet. 'type 'news)
The Register organization The Register (AtValSet. 'type 'news)
The Times organization The Times (AtValSet. 'type 'news)
Topeka Capital Journal organization Topeka Capital Journal (AtValSet. 'type 'news)
Toronto Star organization Toronto Star (AtValSet. 'type 'news)
U.S.A. Today organization U.S.A. Today (AtValSet. 'type 'news)
USA Today organization USA Today (AtValSet. 'type 'news)
Wall Street Journal organization Wall Street Journal (AtValSet. 'type 'news)
Washington Post organization Washington Post (AtValSet. 'type 'news)
Winnipeg Sun organization Winnipeg Sun (AtValSet. 'type 'news)
at St. Joseph's Catholic Church on East 87th Street in Manhattan and said organization St. Joseph's Catholic Church (AtValSet.)
the Feast of Pentecost, St. Patrick's Cathedral offers the Sacrament of Confirmation to organization St. Patrick's Cathedral (AtValSet.)
the archdiocese's Office for Persons with Disabilities organization archdiocese's Office for Persons with Disabilities (AtValSet.)

As the Taxi and Limousine Commission's chairwoman, Diane McGrath-McKechnie organization Taxi and Limousine Commission (AtValSet.)
and graduated from Columbia's School of General Studies in 1960 organization Columbia's School of General Studies (AtValSet.)
alled the Life Project, for the U.S. Department of Labor in the early 1970s organization Life Project (AtValSet.)
alled the Life Project, for the U.S. Department of Labor in the early 1970s organization U.S. Department of Labor (AtValSet.)
at Columbia University's Bureau of Applied Social Research, the organization Columbia University's Bureau of Applied Social Research (AtValSet.)
Research, and the Vera Institute of Justice in New York organization Vera Institute of Justice in New York (AtValSet.)
Bureau of Social Science Research in Washington organization Bureau of Social Science Research in Washington (AtValSet.)
an associate professor of sociology at John Jay College of Criminal Justice organization John Jay College of Criminal Justice (AtValSet.)

On hearing of the reopening of the Butterfly Zone at the Bronx Zoo organization Bronx Zoo (AtValSet.)
a leader of the Little Town Forum for the Historic Preservation of Sloatsburg, a group formed to res organization Little Town Forum for the Historic Preservation (AtValSet.)

The comma after Calif. prevents this version from working: (There is no Calif without the period, and there are not all subtokens)
#the City Council of San Leandro, Calif., reject organization City Council of San Leandro, Calif. (AtValSet.)
A.4 Restrictive Clauses

TYPE person

The first student to do a presentation was Alex. person The first student to do a presentation (AtValSet. 'occupation 'student 'definite #t)
The first student doing a presentation was Alex. person The first student doing a presentation (AtValSet. 'occupation 'student 'definite #t)
The first student who is doing a presentation will win a prize. person The first student who is doing a presentation (AtValSet. 'occupation 'student 'definite #t)
The first student that is doing a presentation will win a prize. person The first student that is doing a presentation (AtValSet. 'occupation 'student 'definite #t)
The first student who does a presentation will win a prize. person The first student who does a presentation (AtValSet. 'occupation 'student 'definite #t)
The first student who did a presentation will have to present again. person The first student who did a presentation (AtValSet. 'occupation 'student 'definite #t)
### A.5 Numbers

**TYPE** quantity

- one quantity one (AtValSet. 'value 1)
- two quantity two (AtValSet. 'value 2)
- three quantity three (AtValSet. 'value 3)
- four quantity four (AtValSet. 'value 4)
- five quantity five (AtValSet. 'value 5)
- six quantity six (AtValSet. 'value 6)
- seven quantity seven (AtValSet. 'value 7)
- eight quantity eight (AtValSet. 'value 8)
- nine quantity nine (AtValSet. 'value 9)
- ten quantity ten (AtValSet. 'value 10)
- eleven quantity eleven (AtValSet. 'value 11)
- twelve quantity twelve (AtValSet. 'value 12)
- thirteen quantity thirteen (AtValSet. 'value 13)
- fourteen quantity fourteen (AtValSet. 'value 14)
- fifteen quantity fifteen (AtValSet. 'value 15)
- sixteen quantity sixteen (AtValSet. 'value 16)
- seventeen quantity seventeen (AtValSet. 'value 17)
- eighteen quantity eighteen (AtValSet. 'value 18)
- nineteen quantity nineteen (AtValSet. 'value 19)
- twenty quantity twenty (AtValSet. 'value 20)
- thirty quantity thirty (AtValSet. 'value 30)
- forty quantity forty (AtValSet. 'value 40)
- fifty quantity fifty (AtValSet. 'value 50)
- sixty quantity sixty (AtValSet. 'value 60)
- seventy quantity seventy (AtValSet. 'value 70)
- eighty quantity eighty (AtValSet. 'value 80)
- ninety quantity ninety (AtValSet. 'value 90)
- hundred quantity hundred (AtValSet. 'value 100)
- thousand quantity thousand (AtValSet. 'value 1000)
- million quantity million (AtValSet. 'value 1000000)
- billion quantity billion (AtValSet. 'value 1000000000)
- trillion quantity trillion (AtValSet. 'value 1000000000000)

# additional test cases

- one shot quantity one (AtValSet. 'value 1)
- I shot one quantity one (AtValSet. 'value 1)
- I shot one! quantity one (AtValSet. 'value 1)
- One shot me! quantity One (AtValSet. 'value 1)
- I shot one quantity one (AtValSet. 'value 1)
- On one, shot, quantity one (AtValSet. 'value 1)

- sixty eight quantity sixty eight (AtValSet. 'value 68)
- a hundred quantity a hundred (AtValSet. 'value 100)
seven hundred quantity seven hundred (AtValSet. 'value 700)
fifteen hundred quantity fifteen hundred (AtValSet. 'value 1500)
twenty three hundred quantity twenty three hundred (AtValSet. 'value 2300)
five hundred three quantity five hundred three (AtValSet. 'value 503)
nineteen hundred four quantity nineteen hundred four (AtValSet. 'value 1904)
twenty one hundred six quantity twenty one hundred six (AtValSet. 'value 2106)
three hundred fifteen quantity three hundred fifteen (AtValSet. 'value 315)
fourteen hundred thirteen quantity fourteen hundred thirteen (AtValSet. 'value 1413)
eighty three hundred eleven quantity eighty three hundred eleven (AtValSet. 'value 8311)
six hundred twenty four quantity six hundred twenty four (AtValSet. 'value 624)
eleven hundred seventy seven quantity eleven hundred seventy seven (AtValSet. 'value 1177)
ninety nine hundred ninety nine quantity ninety nine hundred ninety nine (AtValSet. 'value 9999)
seven hundred one quantity seven hundred one (AtValSet. 'value 701)
eight hundred and twelve quantity eight hundred and twelve (AtValSet. 'value 812)
nine hundred and forty one quantity nine hundred and forty one (AtValSet. 'value 941)
eleven hundred and sixty four quantity eleven hundred and sixty four (AtValSet. 'value 1164)
thirteen hundred and fifty five quantity thirteen hundred and fifty five (AtValSet. 'value 1355)
seventeen hundred and three hundred quantity seventeen hundred and three hundred (AtValSet. 'value 1703)
seventy one quantity seventy one (AtValSet. 'value 71)
eighty six quantity eighty six (AtValSet. 'value 86)
ninety nine quantity ninety nine (AtValSet. 'value 99)
twenty two quantity twenty two (AtValSet. 'value 22)
three hundred quantity three hundred (AtValSet. 'value 300)
four hundred quantity four hundred (AtValSet. 'value 400)
five hundred quantity five hundred (AtValSet. 'value 500)
six hundred quantity six hundred (AtValSet. 'value 600)
six hundred sixty four quantity six hundred sixty four (AtValSet. 'value 6664)
six hundred sixty four quantity six hundred sixty four (AtValSet. 'value 6664)

# sets

# TYPE set
#
# ones set ones 1
# twos set twos 2
# tens set tens 10
# twenties set twenties 20
# seventeens set seventeens 170

# ordinals

TYPE quantity

first quantity first (AtValSet. 'type 'ordinal 'value 1)
second quantity second (AtValSet. 'type 'ordinal 'value 2)
third quantity third (AtValSet. 'type 'ordinal 'value 3)
fourth quantity fourth (AtValSet. 'type 'ordinal 'value 4)
fifth quantity fifth (AtValSet. 'type 'ordinal 'value 5)
sixth quantity sixth (AtValSet. 'type 'ordinal 'value 6)
seventh quantity seventh (AtValSet. 'type 'ordinal 'value 7)
eighth quantity eighth (AtValSet. 'type 'ordinal 'value 8)
ninth quantity ninth (AtValSet. 'type 'ordinal 'value 9)
tenth quantity tenth (AtValSet. 'type 'ordinal 'value 10)
eleventh quantity eleventh (AtValSet. 'type 'ordinal 'value 11)
twelfth quantity twelfth (AtValSet. 'type 'ordinal 'value 12)
thirteenth quantity thirteenth (AtValSet. 'type 'ordinal 'value 13)
fourteenth quantity fourteenth (AtValSet. 'type 'ordinal 'value 14)
fifteenth quantity fifteenth (AtValSet. 'type 'ordinal 'value 15)
sixteenth quantity sixteenth (AtValSet. 'type 'ordinal 'value 16)
seventeenth quantity seventeenth (AtValSet. 'type 'ordinal 'value 17)
eighteenth quantity eighteenth (AtValSet. 'type 'ordinal 'value 18)
nineteenth quantity nineteenth (AtValSet. 'type 'ordinal 'value 19)
twentieth quantity twentieth (AtValSet. 'type 'ordinal 'value 20)
thirtieth quantity thirtieth (AtValSet. 'type 'ordinal 'value 30)
fortieth quantity fortieth (AtValSet. 'type 'ordinal 'value 40)
fiftieth quantity fiftieth (AtValSet. 'type 'ordinal 'value 50)
sixtieth quantity sixtieth (AtValSet. 'type 'ordinal 'value 60)
seventieth quantity seventieth (AtValSet. 'type 'ordinal 'value 70)
eightieth quantity eightieth (AtValSet. 'type 'ordinal 'value 80)
ninetieth quantity ninetieth (AtValSet. 'type 'ordinal 'value 90)
# using compound.quantity made from 'hundredth' and some ones
hundredth quantity hundredth (AtValSet. 'type 'ordinal 'value 100)
# using compound.quantity made from 'hundredth' and some tens
one hundredth quantity one hundredth (AtValSet. 'type 'ordinal 'value 100)
two hundredth quantity two hundredth (AtValSet. 'type 'ordinal 'value 200)
three hundredth quantity three hundredth (AtValSet. 'type 'ordinal 'value 300)
four hundredth quantity four hundredth (AtValSet. 'type 'ordinal 'value 400)
five hundredth quantity five hundredth (AtValSet. 'type 'ordinal 'value 500)
six hundredth quantity six hundredth (AtValSet. 'type 'ordinal 'value 600)
seven hundredth quantity seven hundredth (AtValSet. 'type 'ordinal 'value 700)
eight hundredth quantity eight hundredth (AtValSet. 'type 'ordinal 'value 800)
nine hundredth quantity nine hundredth (AtValSet. 'type 'ordinal 'value 900)
# using compound.quantity made from 'hundredth' and some hundreds
eleven hundredth quantity eleven hundredth (AtValSet. 'type 'ordinal 'value 1100)
twelve hundredth quantity twelve hundredth (AtValSet. 'type 'ordinal 'value 1200)
seventeen hundredth quantity seventeen hundredth (AtValSet. 'type 'ordinal 'value 1700)
# using compound.quantity made from ones.quantity and some tens
fifty eighth quantity fifty eighth (AtValSet. 'type 'ordinal 'value 58)
ninety sixth quantity ninety sixth (AtValSet. 'type 'ordinal 'value 96)
# using compound.quantity made from ones.quantity and some hundreds
one hundred seventh quantity one hundred seventh (AtValSet. 'type 'ordinal 'value 107)
thirteen hundred ninth quantity thirteen hundred ninth (AtValSet. 'type 'ordinal 'value 1309)
# using compound.quantity made from tens.quantity and some hundreds
one hundred thirteenth quantity one hundred thirteenth (AtValSet. 'type 'ordinal 'value 1313)
twenty seven hundred twelfth quantity twenty seven hundred twelfth (AtValSet. 'type 'ordinal 'value 2712)
fourteen hundred nineteenth quantity fourteen hundred nineteenth (AtValSet. 'type 'ordinal 'value 1419)
# using compound.quantity made from tens.quantity and some thousands
nine hundred twenty eighth quantity nine hundred twenty eighth (AtValSet. 'type 'ordinal 'value 920)
eighty six hundred seventieth quantity eighty six hundred seventieth (AtValSet. 'type 'ordinal 'value 8670)
# more advanced cases
one second helping quantity second (AtValSet. 'type 'ordinal 'value 2)
eleventh quantity eleventh (AtValSet. 'type 'ordinal 'value 11)
twenty-second quantity twenty-second (AtValSet. 'type 'ordinal 'value 22)
twenty seventh quantity twenty seventh (AtValSet. 'type 'ordinal 'value 27)
seven hundred and seventy seven thousand seven hundred seventy seventh quantity seven hundred and seventy seven thousand seven hundred seventy seventh (AtValSet. 'type 'ordinal 'value 777777)
# I fail on this because I decided not to have hundred-omitting in the cardinals
seventeen hundred ninetieth quantity seventeen hundred ninetieth (AtValSet. 'type 'ordinal 'value 1790)
eighty six hundred seventieth quantity eighty six hundred seventieth (AtValSet. 'type 'ordinal 'value 8670)

# spelled out fractions testplan

TYPE quantity

# singular ones fractions
one half quantity one half (AtValSet. 'type 'expression 'value '(/ 1.0 2))
two half quantity two half (AtValSet. 'type 'expression 'value '(/ 2.0 2))
one quarter quantity one quarter (AtValSet. 'type 'expression 'value '(/ 1.0 4))
two quarter quantity two quarter (AtValSet. 'type 'expression 'value '(/ 2.0 4))
one fourth quantity one fourth (AtValSet. 'type 'expression 'value '(/ 1.0 4))
four quarter quantity four quarter (AtValSet. 'type 'expression 'value '(/ 4.0 4))
one fifth quantity one fifth (AtValSet. 'type 'expression 'value '(/ 1.0 5))
five fifth (type expression 'value '/ 5.0 5)
one sixth quantity one sixth (type expression 'value '/ 1.0 6)
six sixth (type expression 'value '/ 6.0 6)
one seventh quantity one seventh (type expression 'value '/ 1.0 7)
seven seventh quantity seven seventh (type expression 'value '/ 7.0 7)
one eighth quantity one eighth (type expression 'value '/ 1.0 8)
eight eighth quantity eight eighth (type expression 'value '/ 8.0 8)
one ninth quantity one ninth (type expression 'value '/ 1.0 9)
nine ninth (type expression 'value '/ 9.0 9)

# singular tens fractions
one tenth (type expression 'value '/ 1.0 10)
three tenth (type expression 'value '/ 3.0 10)
one twentieth quantity one twentieth (type expression 'value '/ 1.0 20)
three twentieth (type expression 'value '/ 3.0 20)
one thirtieth quantity one thirtieth (type expression 'value '/ 1.0 30)
three thirtieth (type expression 'value '/ 3.0 30)
one fortieth quantity one fortieth (type expression 'value '/ 1.0 40)
three fortieth (type expression 'value '/ 3.0 40)
one fiftieth quantity one fiftieth (type expression 'value '/ 1.0 50)
three fiftieth (type expression 'value '/ 3.0 50)
one sixtieth quantity one sixtieth (type expression 'value '/ 1.0 60)
three sixtieth (type expression 'value '/ 3.0 60)
one seventieth quantity one seventieth (type expression 'value '/ 1.0 70)
three seventieth (type expression 'value '/ 3.0 70)
one eightieth quantity one eightieth (type expression 'value '/ 1.0 80)
three eightieth (type expression 'value '/ 3.0 80)

# singular teens fractions
one twelfth quantity one twelfth (type expression 'value '/ 1.0 12)
two fourteenth (type expression 'value '/ 1.0 14)
one eighteenth quantity one eighteenth (type expression 'value '/ 1.0 18)
nine sixteenth quantity nine sixteenth (type expression 'value '/ 9.0 16)

# singular hundred fractions
one hundredth quantity one hundredth (type expression 'value '/ 1.0 100)
three hundredth (type expression 'value '/ 3.0 100)

# plural ones fractions
ninety halves quantity ninety halves (type expression 'value '/ 90.0 2)
twenty two halves quantity twenty two halves (type expression 'value '/ 22.0 2)
three thirds quantity three thirds (type expression 'value '/ 3.0 3)
ninety thirds quantity ninety thirds (type expression 'value '/ 90.0 3)
two fourths quantity two fourths (type expression 'value '/ 2.0 4)
eighty fourths quantity eighty fourths (type expression 'value '/ 80.0 4)
twenty four fourths quantity twenty four fourths (type expression 'value '/ 24.0 4)
five fifths quantity five fifths (type expression 'value '/ 5.0 5)
five hundred five fifths quantity five hundred five fifths (type expression 'value '/ 505.0 5)
one hundred twenty five thousand seven hundred and fifty five quantity one hundred twenty five thousand seven hundred and fifty fifth (type expression 'value '/ 12500750.0 5)
six sixths quantity six sixths (type expression 'value '/ 6.0 6)
six hundred fifty four sixth quantity six hundred fifty four sixths (type expression 'value '/ 654.0 6)
twelve thousand three hundred and six sixths quantity twelve thousand three hundred and six sixths (type expression 'value '/ 1230066.0 6)
seventy seven sevenths quantity seventy seven sevenths (type expression 'value '/ 70.0 7)
two hundred ten sevenths quantity two hundred ten sevenths (type expression 'value '/ 210.0 7)
four eighteens quantity four eighteens (type expression 'value '/ 4.0 8)
twenty four eighteens quantity twenty four eighteens (type expression 'value '/ 24.0 8)
nine hundred ninety nine ninths quantity nine hundred ninety nine ninths (type expression 'value '/ 999.0 9)

# plural tens fractions
nine tenths quantity nine tenths (type expression 'value '/ 9.0 10)
nineteen twentieths quantity nineteen twentieths (AtValSet. 'type 'expression 'value '(/ 19.0 20))
twenty thirtieths quantity twenty thirtieths (AtValSet. 'type 'expression 'value '(/ 20.0 30))
twenty seven fortieths quantity twenty seven fortieths (AtValSet. 'type 'expression 'value '(/ 27.0 40))
one hundred eight fiftieths quantity one hundred eight fiftieths (AtValSet. 'type 'expression 'value '(/ 108.0 50))
one hundred twelve sixtieths quantity one hundred twelve sixtieths (AtValSet. 'type 'expression 'value '(/ 112.0 60))
one hundred ninety seventieths quantity one hundred ninety seventieths (AtValSet. 'type 'expression 'value '(/ 190.0 70))
one hundred seventy five eightieths quantity one hundred seventy five eightieths (AtValSet. 'type 'expression 'value '(/ 175.0 80))
one thousand four hundred six ninetieths quantity one thousand four hundred six ninetieths (AtValSet. 'type 'expression 'value '(/ 1406.0 90))

# plural teens fractions
five elevenths quantity five elevenths (AtValSet. 'type 'expression 'value '(/ 5.0 11))
fifteen twelfths quantity fifteen twelfths (AtValSet. 'type 'expression 'value '(/ 15.0 12))

# plural hundreds fractions
twenty four hundredths quantity twenty four hundredths (AtValSet. 'type 'expression 'value '(/ 24.0 100))

# compound fractions
three twenty firsts quantity three twenty firsts (AtValSet. 'type 'expression 'value '(/ 3.0 21))
sixteen thirty seconds quantity sixteen thirty seconds (AtValSet. 'type 'expression 'value '(/ 16.0 32))

# hyphen usage
three-fourths quantity three-fourths (AtValSet. 'type 'expression 'value '(/ 3.0 4))
A.6 Units

Note, the lexicon format for units is a prior version, and there is no third field to specify a substring in context. The whole string in the first field must be matched.

```plaintext
TYPE money
TYPE socard

a dollar money 1
two dollars money 2
two dollar money 2
2 dollar money 2
2-dollar money 2
2 dollars money 2
$2 money 2
2-dollars money 2

a hundred dollar bill money 100

# this one is different due to np-internal structure that needs to know
# about bills, and needs to assume that bill is a noun and that it's
# the head, or that hundred dollar can be an adjective but not a noun
# phrase or some such. ugh.
a hundred dollar bills money 100

twenty two hundred dollar bills money 2200
twenty two hundred dollar bills money 200
twenty two hundred dollar bills socard 20

a two hundred dollar bill money 200
ten hundred dollar bills money 1000
a twenty-one hundred dollar bill money 2100
21 hundred-dollar bills money 2100
twenty-one hundred dollar bills money 2100
```
A.7 Dates

Note, the lexicon format for units is a prior version, and there is no third field to specify a substring in context. The whole string in the first field must be matched.

TYPE date
TYPE rte
TYPE time
TYPE year
TYPE month
TYPE day-in-month
TYPE day-of-month
TYPE month-of-year
TYPE period
TYPE quarter
TYPE half
TYPE third

1992 year 1992
nineteen ninety-two year 1992
third quarter quarter 3
the third quarter quarter 3
third quarter of 1991 period 1
the fourth quarter ended Sept. 30 quarter 4
the fourth quarter ended Sept. 30 day-of-month ((9 month) (30 day-in-month))
the three months ended Sept. 30 period 3
the three months ended Sept. 30 day-of-month ((9 month) (30 day-in-month))
first-half profit half 1
fiscal 1989's fourth quarter period 1
4th period period 4
1978 World Series year 1978
February 12th day-of-month ((2 month) (12 day-in-month))
Tuesday, February 12th date ((2 day) ((2 month) (12 day-in-month)))
February 12, 1997 date (((2 month) (12 day-in-month)) 1997)
February 12, 8 A.M. day-of-month ((2 month) (12 day-in-month))
February 12, 8 A.M. time 8
shortly after the 4th of May date 1
noon time "12:00"
twelve o'clock noon time "12:00"

TYPE rte-time
last night rte-time 1
yesterday evening rte-time 1

5 p.m. EST time 5
5 pm EST time 5
5pm EST time 5

the first half of fiscal 1990 period 1
Monday through Friday period ((1 . day) (5 . day))
April through June period ((4 . month) (6 . month))
April - June period ((4 . month) (6 . month))

03/13 day-of-month 1303
03-13 day-of-month 1303
03/11/1996 date 19961103
03-11-1996 date 19961103
03-11-96 date 961103
03/11/96 date 961103

summer period "Summer"
last summer rte ("Summer" . period)
last summer's party rte ("Summer" . period)
A.8 Questions

These are 346 of the TREC questions, annotated according to ACE guidelines, on which we have begun to develop for ACE. This is the only testplan for which we do not pass all uncommented test cases. In particular, we currently pass 86 of them.

# We are having trouble deciding whether "many people" or "many casinos" or "many counties" are markable as generics in the context of "how many".

```
TYPE people
TYPE quantity
TYPE person
TYPE location
TYPE organization
TYPE facility
TYPE gpe
TYPE event
TYPE title
```

201: What was the name of the first Russian astronaut to do a spacewalk? person the first Russian astronaut to do a spacewalk (AtValSet. 'occupation 'astronaut 'from "Russia")
201: What was the name of the first Russian astronaut to do a spacewalk? person Russian astronaut (AtValSet. 'occupation 'astronaut 'from "Russia")
201: What was the name of the first Russian astronaut to do a spacewalk? gpe Russian (AtValSet. 'type 'country 'name "Russia")
202: Where is Belize located? gpe Belize (AtValSet. 'type 'country)
204: What type of bridge is the Golden Gate Bridge? facility bridge (AtValSet. 'type 'bridge 'generic #1)
204: What type of bridge is the Golden Gate Bridge? facility the Golden Gate Bridge (AtValSet. 'type 'bridge)
205: What is the population of the Bahamas? gpe the Bahamas (AtValSet. 'type 'country)
206: How far away is the moon? location the moon (AtValSet. 'type 'astronomy)
208: What state has the most Indians? person Indians (AtValSet. 'group St)
210: How many dogs pull a sled in the Iditarod? organization the Iditarod (AtValSet. 'type 'competition)
211: Who is the leader of India? person the leader of India (AtValSet. 'occupation 'government)
211: Who is the leader of India? person India (AtValSet. 'type 'country)
216: What is the primary language of the Philippines? gpe the Philippines (AtValSet. 'type 'country)
219: What is the population of Japan? gpe Japan (AtValSet. 'type 'country)
220: Who is the prime minister of Australia? person the prime minister of Australia (AtValSet. 'occupation 'government)
220: Who is the prime minister of Australia? person Australia (AtValSet. 'type 'country)
221: Who killed Martin Luther King? person Martin Luther King (AtValSet. )
222: Where's Montenegro? gpe Montenegro (AtValSet. 'type 'country)
225: Who is the Greek God of the Sea? person the Greek God of the Sea (AtValSet. 'type 'deity)
225: Who is the Greek God of the Sea? person Greece (AtValSet. 'type 'country)
226: Where is the Danube? location the Danube (AtValSet. 'type 'river)
231: Who was the president of Vichy France? person the president of Vichy France (AtValSet. 'occupation 'government)
231: Who was the president of Vichy France? person Vichy France (AtValSet. 'type 'country)
237: Name one of the major gods of Hinduism? person the major gods of Hinduism (AtValSet. 'type 'deity 'group #1)
237: Name one of the major gods of Hinduism? person Hinduism (AtValSet. 'type 'deity)
238: What does the abbreviation OAS stand for? organization OAS (AtValSet. )
240: How many years ago did the ship Titanic sink? facility Titanic (AtValSet. 'type 'vehicle)
242: What was the name of the famous battle in 1836 between Texas and Mexico? gpe Texas (AtValSet. 'type 'state)
242: What was the name of the famous battle in 1836 between Texas and Mexico? gpe Mexico (AtValSet. 'type 'country)
245: Where can you find the Venus flytrap? location Venus (AtValSet. 'type 'astronomy)
246: What did Vasco da Gama discover? person Vasco da Gama (AtValSet. )
247: Who won the Battle of Gettysburg? gpe Gettysburg (AtValSet. 'type 'city)
247: Who won the Battle of Gettysburg? event Battle of Gettysburg (AtValSet. 'type 'battle)
249: Where is the Valley of the Kings? location the Valley of the Kings (AtValSet. )
250: Where did the Maya people live? person the Maya people (AtValSet. 'group #1)
251: How many people live in Chile? gpe Chile (AtValSet. 'type 'country)
254: What is California's state bird? gpe California (AtValSet. 'type 'state)
256: Who is buried in the great pyramid of Giza? facility the great pyramid of Giza (AtValSet. 'type 'monument)
256: Who is buried in the great pyramid of Giza? facility Giza (AtValSet. 'type 'city)
260: What does NAFTA stand for? organization NAFTA (AtValSet. )
262: What is the name of the longest ruling dynasty of Japan? person Japan (AtValSet. 'type 'country)
262: What is the name of the longest ruling dynasty of Japan? person Japan (AtValSet. 'type 'government)
263: Where was Babe Ruth born? person Babe Ruth (AtValSet. 'occupation 'sports)
```
264: Who wrote the Farmer's Almanac? (organization Farmer's Almanac)
266: Where was Pythagoras born? (person Pythagoras)
268: Who killed Caesar? (person Caesar)
270: Where is the Orinoco? (location Orinoco)
272: Who was the first U.S. president ever to resign? (person the first U.S. president ever to resign)
273: Who was the first U.S. president ever to resign? (gpe U.S.)
274: Who invented the game Scrabble? (we are not sure about this case)
275: About how many soldiers died in World War II? (person many soldiers)
277: How much money does the Sultan of Brunei have? (person the Sultan of Brunei)
278: How large is Missouri's population? (gpe Missouri)
279: What was the death toll at the eruption of Mount Pinatubo? (location Mount Pinatubo)
280: What's the tallest building in New York City? (facility the tallest building in New York City)
281: When did Geraldine Ferraro run for vice president? (person Geraldine Ferraro)
282: Where is Ayer's rock? (location Ayer's rock)
283: When was the first railroad from the east coast to the west coast completed? (location the first railroad from the east coast to the west coast completed)
284: Who found Hawaii? (person Hawaii)
285: How many films did Ingmar Bergman make? (person Ingmar Bergman)
286: What is California's state tree? (gpe California)
287: Who was the architect of Central Park? (location Central Park)
288: What is Alice Cooper's real name? (person Alice Cooper)
289: Why can't ostriches fly? (why can't ostriches fly?)
290: Where was Rome located? (location Rome)
291: When did the Carolingian period begin? (time the Carolingian period)
292: How big is Australia? (gpe Australia)
293: Who is the prophet of the religion of Islam? (person the prophet of the religion of Islam)
294: How many home runs did Babe Ruth hit in his lifetime? (person Babe Ruth)
295: What is the size of Argentina? (gpe Argentina)
296: Where is Tornado Alley? (location Tornado Alley)
297: Who was the first king of England? (person the first king of England)
298: Where is the bridge over the river Kwai? (river the river Kwai)
299: What's the tallest building in New York City? (facility the tallest building in New York City)
300: What is California's state tree? (gpe California)
301: Who was the first coach of the Cleveland Browns? (person the first coach of the Cleveland Browns)
302: Where is Romania located? (gpe Romania)
303: When was the D Beers company founded? (organization the D Beers company)
304: When did Geraldine Ferraro run for vice president? (person Geraldine Ferraro)
305: Where is the bridge over the river Kwai? (location the bridge over the river Kwai)
306: Where was Bozeman born? (location Bozeman)
307: Who was the architect of Central Park? (location Central Park)
308: Who was the first king of England? (person the first king of England)
309: What is California's state tree? (gpe California)
310: Who was the architect of Central Park? (location Central Park)
311: How many Superbowls have the 49ers won? (organization the 49ers)
312: Who was the architect of Central Park? (location Central Park)
313: Who was the architect of Central Park? (person Alice Cooper)
314: Who was the architect of Central Park? (gpe Central Park)
315: When did Hurricane Hugo? (time Hurricane Hugo)
316: Who was the architect of Central Park? (person Alice Cooper)
317: Where is Guam? (gpe Guam)
318: Who was the architect of Central Park? (person Alice Cooper)
319: Who was the architect of Central Park? (organization the 49ers)
320: Where was Rome located? (location Rome)
321: Who was the architect of Central Park? (organization the 49ers)
322: Who was the architect of Central Park? (organization the 49ers)
323: Who was the architect of Central Park? (organization the 49ers)
324: What is Alice Cooper's real name? (person Alice Cooper)
325: What is California's state tree? (gpe California)
536: What is the population of the United States? gpe the United States (AtValSet. 'type 'country)
534: Where is Windsor Castle? facility Windsor Castle (AtValSet. 'type 'building)
533: Who won the rugby world cup in 1987? organization the rugby world cup (AtValSet. 'type 'competition)
530: What other name were the "Little Rascals" known as? title "Little Rascals" (AtValSet. )
527: When was the Six Day War fought? event Six Day War (AtValSet. 'type 'war)
524: Where did Wicca first develop? organization Wicca (AtValSet. 'type 'religious)
1523: What is Chiricahua the name of? location Chiricahua (AtValSet. )
522: What does EKG stand for? 
519: What is the zip code for Parsippany, NJ? gpe Parsippany, NJ (AtValSet. 'type 'city)
518: In what area of the world was the Six Day War fought? event Six Day War (AtValSet. 'type 'war)
517: What continent is Bolivia on? gpe Bolivia (AtValSet. 'type 'country)
516: Where did Woodstock take place? event Woodstock (AtValSet. 'type 'concert)
515: For what disease is the drug Sinemet used as a treatment? 
513: When did the royal wedding of Prince Andrew and Fergie take place? person Fergie (AtValSet. )
512: What movie did Madilyn Kahn star in with Gene Wilder? person Gene Wilder (AtValSet. )
511: Where is Tufts University? organization Tufts University (AtValSet. 'type 'educational)
510: What's the name of the tiger that advertises for Frosted Flakes cereal? 
509: When was Beethoven born? person Beethoven (AtValSet. )
508: Where is Las Vegas? gpe Las Vegas (AtValSet. 'type 'city)
507: When did the California lottery begin? organization the California lottery (AtValSet. 'type 'governmental)
505: What city is Massachusetts General Hospital located in? facility Massachusetts General Hospital (AtValSet. 'type 'hospital)
502: What is one of the cities that the University of Minnesota is located in? gpe the cities that the University of Minnesota is located in (AtValSet. 'type 'city 'group t)
501: What is the population of Ohio? gpe Ohio (AtValSet. 'type 'state)
500: What city in Florida is Sea World in? gpe Florida (AtValSet. 'type 'state)
499: Where is Venezuela? gpe Venezuela (AtValSet. 'type 'country)
498: Who portrayed Fatman in the television show, "Jake and the Fatman"? person Fatman (AtValSet. )
497: Who portrayed Jake in the television show, "Jake and the Fatman"? person Jake (AtValSet. )
496: Who wrote the book, "Song of Solomon"? title "Song of Solomon" (AtValSet. 'type 'book)
495: When did Aldous Huxley write, "Brave New World"? person Aldous Huxley (AtValSet. 'type 'author)
493: What is Betsy Ross famous for? person Betsy Ross (AtValSet. )
492: What is the name of the movie that starred Sharon Stone and Arnold Schwarzenegger? person Sharon Stone (AtValSet. )
491: Where did Woodstock take place? event Woodstock (AtValSet. 'type 'concert)
490: Who is the monarch of the United Kingdom? gpe the United Kingdom (AtValSet. 'type 'country)
489: Who shot Billy the Kid? person Billy the Kid (AtValSet. )
488: What continent is Bolivia on? gpe Bolivia (AtValSet. 'type 'country)
487: What was the name of the movie that starred Sharon Stone and Arnold Schwarzenegger? person Arnold Schwarzenegger (AtValSet. )
486: What is the zip code for Fremont, CA? gpe Fremont, CA (AtValSet. 'type 'city)
485: Where is Amsterdam? gpe Amsterdam (AtValSet. 'type 'city)
484: What is the name of the Lion King's son in the movie, "The Lion King"? person Lion King (AtValSet. 'type 'movie)
483: What is the name of the Lion King's son in the movie, "The Lion King"? person Lion King (AtValSet. 'type 'movie)
482: Who is the monarch of the United Kingdom? gpe the United Kingdom (AtValSet. 'type 'country)
481: Who shot Billy the Kid? person Billy the Kid (AtValSet. )
480: What is the zip code for Fremont, CA? gpe Fremont, CA (AtValSet. 'type 'city)
479: Who provides telephone service in Orange County, California? gpe Orange County, California (AtValSet. 'type 'county)}
1017: Who discovered America? location America (AtValSet. 'type 'continent)
1020: How wide is the Milky Way galaxy? location the Milky Way galaxy (AtValSet. 'type 'astronomy)
1025: Who was the abolitionist who led the raid on Harper's Ferry in 1859? person the abolitionist who led the raid on Harper's Ferry in 1859 (AtValSet. 'occupation 'abolitionist)
1025: Who was the abolitionist who led the raid on Harper's Ferry in 1859? event the raid on Harper’s Ferry in 1859 (AtValSet. )
1025: Who was the abolitionist who led the raid on Harper’s Ferry in 1859? facility Harper’s Ferry (AtValSet. )
1029: What is the major fault line near Kentucky? location the major fault line near Kentucky (AtValSet. )
1029: What is the major fault line near Kentucky? gpe Kentucky (AtValSet. 'type 'state)
1030: Where is the Holland Tunnel? facility the Holland Tunnel (AtValSet. 'type 'tunnel)
1031: Who wrote the hymn “Amazing Grace”? title “Amazing Grace” (AtValSet. 'type 'song)
1032: What position did Willie Davis play in baseball? person Willie Davis (AtValSet. 'occupation 'sports)
1035: What is the name of Roy Roger’s dog? person Roy Roger (AtValSet. )
1036: Where is the Shawnee National Forest? location the Shawnee National Forest (AtValSet. 'type 'forest)
1037: Where is the volcano Mauna Loa? location the volcano Mauna Loa (AtValSet. 'type 'mountain)
1039: What is another name for vitamin B1?
1040: What U.S. state’s motto is “Live free or Die”? gpe the United States (AtValSet. )
1043: What peninsula is Spain part of? gpe Spain (AtValSet. 'type 'country)
1046: Who founded American Red Cross? organization American Red Cross (AtValSet. )
1048: What year did the Milwaukee Braves become the Atlanta Braves? organization the Milwaukee Braves (AtValSet. 'type 'sports)
1048: What year did the Milwaukee Braves become the Atlanta Braves? organization the Atlanta Braves (AtValSet. 'type 'sports)
1050: What is the capital of Mongolia? gpe Mongolia (AtValSet. 'type 'country)
1050: What is the capital of Mongolia? gpe the capital of Mongolia (AtValSet. 'type 'city)
1053: Where are the National Archives? facility the National Archives (AtValSet. )
1054: What is another name for vitamin B1?
1055: Where did Howard Hughes die? person Howard Hughes (AtValSet. )
1057: Where is the volcano Mauna Loa? location the volcano Mauna Loa (AtValSet. 'type 'mountain)
1058: What is another astronomical term for the Northern Lights?
1065: Who was the abolitionist who led the raid on Harper’s Ferry in 1859? person the abolitionist who led the raid on Harper’s Ferry in 1859 (AtValSet. 'occupation 'abolitionist)
1066: What is another astronomical term for the Northern Lights?
1068: Where is the volcano Mauna Loa? location the volcano Mauna Loa (AtValSet. 'type 'mountain)
1070: Where is the Louvre? facility the Louvre (AtValSet. 'type 'museum)
1073: What is another astronomical term for the Northern Lights?
1074: What is another astronomical term for the Northern Lights?
1075: What is another astronomical term for the Northern Lights?
1076: What is another astronomical term for the Northern Lights?
1077: What is another astronomical term for the Northern Lights?
1078: Who is a German philosopher? person a German philosopher? (AtValSet. 'occupation 'philosopher 'from "Germany" 'generic t)
1079: Who is the Prime Minister of Canada? gpe Canada (AtValSet. 'type 'country)
1079: Who is the Prime Minister of Canada? gpe the capital of Canada (AtValSet. 'type 'city)
1080: What year did the Milwaukee Braves become the Atlanta Braves? organization the Milwaukee Braves (AtValSet. 'type 'sports)
1080: What year did the Milwaukee Braves become the Atlanta Braves? organization the Atlanta Braves (AtValSet. 'type 'sports)
1081: What is the birthstone for June?
1082: What are Christopher Columbus’ three ships? facility Christopher Columbus’ three ships (AtValSet. 'type 'vehicle 'group t)
1083: What is the birthstone for June?
1084: What is the nickname for Michigan?
1085: What is the distance in miles from the earth to the sun? location the earth (AtValSet. 'type 'astronomy)
1085: What is the distance in miles from the earth to the sun? location the sun (AtValSet. 'type 'astronomy)
1086: What is the body of water that contains the Canary Islands? location the Canary Islands (AtValSet. 'type 'island 'group t)
1087: Where is the capital of Egypt? gpe Egypt (AtValSet. 'type 'country)
1088: Where is the capital of Egypt? gpe the capital of Egypt (AtValSet. 'type 'city)
1089: What is the range of Pluto from the sun? location the sun (AtValSet. 'type 'astronomy)
1090: What is the distance in miles from the earth to the sun? location the sun (AtValSet. 'type 'astronomy)
1090: What is the distance in miles from the earth to the sun? location the sun (AtValSet. 'type 'astronomy)
1090: What is the distance in miles from the earth to the sun? location the sun (AtValSet. 'type 'astronomy)
1090: What is the distance in miles from the earth to the sun? location the sun (AtValSet. 'type 'astronomy)
1090: What is the distance in miles from the earth to the sun? location the sun (AtValSet. 'type 'astronomy)
1091: What is the range of Pluto from the sun? location the sun (AtValSet. 'type 'astronomy)
1092: Where did Howard Hughes die? person Howard Hughes (AtValSet. )
1093: Where did Howard Hughes die? person Howard Hughes (AtValSet. )
1094: What is another astronomical term for the Northern Lights?
1096: What is another astronomical term for the Northern Lights?
1097: What is another astronomical term for the Northern Lights?
1098: What is another astronomical term for the Northern Lights?
1099: What is another astronomical term for the Northern Lights?
1100: What is another astronomical term for the Northern Lights?
1101: What is another astronomical term for the Northern Lights?
1102: What is another astronomical term for the Northern Lights?
1103: What is another astronomical term for the Northern Lights?
1104: What is another astronomical term for the Northern Lights?
1105: What is another astronomical term for the Northern Lights?
1106: What is another astronomical term for the Northern Lights?
1117: What is the electrical output in Madrid, Spain? gpe Spain (AtValSet. 'type 'country)
1118: Which mountain range in North America stretches from Maine to Georgia? location North America (AtValSet. 'type 'continent)
1118: Which mountain range in North America stretches from Maine to Georgia? gpe Maine (AtValSet. 'type 'state)
1118: Which mountain range in North America stretches from Maine to Georgia? gpe Georgia (AtValSet. 'type 'state)
1120: What is the population of Nigeria? gpe Nigeria (AtValSet. 'type 'country)
1122: Where is the Grand Canyon? location the Grand Canyon (AtValSet. )
1124: What year did the U.S. buy Alaska? gpe the U.S. (AtValSet. 'type 'country)
1124: What year did the U.S. buy Alaska? gpe Alaska (AtValSet. 'type 'state)
1125: What is the name of the leader of Ireland? person the leader of Ireland (AtValSet. 'occupation 'government)
1125: What is the name of the leader of Ireland? gpe Ireland (AtValSet. 'type 'country)
1126: What are the two houses of the Legislative branch? organization the two houses of the Legislative branch (AtValSet. 'type 'governmental 'group #t)
1126: What are the two houses of the Legislative branch? organization the Legislative branch (AtValSet. 'type 'governmental)
1130: In Poland, where do most people live? gpe Poland (AtValSet. 'type 'country)
1132: What is the location of the Sea of Tranquility? location the Sea of Tranquility (AtValSet. 'type 'sea)
1134: What French province is cognac produced in?
1139: What is the longest suspension bridge in the U.S.? facility the longest suspension bridge in the U.S. (AtValSet. 'type 'bridge)
1139: What is the longest suspension bridge in the U.S.? gpe the U.S. (AtValSet. 'type 'country)
1146: What did Edward Binney and Howard Smith invent in 1903? person Edward Binney (AtValSet. )
1146: What did Edward Binney and Howard Smith invent in 1903? person Howard Smith (AtValSet. )
1150: What is the depth of the Nile river? location the Nile river (AtValSet. 'type 'river)
Appendix B

Query Judgements

This appendix shows the full judgements for the named entity evaluation. This was done in two segments. one segment where the annotations differed between sepia only and all phrase annotators, and one segment where no other annotator marked phrases in that question. therefore they did not differ.
same 202
is] located?
same 204
sale: Golden Gate Bridge]?
same 205
ale: Bahamas]?
same 208
e: India]?
same 215
location: India]?
same 216
ale: Philippines]?
same 218
population: Japan]?
same 223
location: Montenegro]?
same 242
ale: famous battle in 1836 between [sepia-location: Texas] and [sepia-location: Mexico]?
same 245
venue] flytrap?
same 250
ale: Maya] people live?
same 251
ale: Chile]?
same 254
ale: California]’s state bird?
same 262
ale: longest ruling dynasty of [sepia-location: Japan]?
same 268
ale: Queen]?
same 269
ale: Caesar]?
same 273
ale: president ever to resign?
same 276
ale: Sult] have?
same 277
ale: Pennsylvania]?
same 286
ale: Australia]?
same 293
ale: Hawaii]?
same 298
ale: California]’s state tree?
same 301
ale: Civil War] have?
same 302
ale: U.S.] per year?
same 310
ale: river Kwai]?
same 311
ale: 49ers] won?
same 314
ale: Cooper]’s real name?
same 317
ale: Romania] located?
same 320
ale: England]?
same 322
ale: capital]?
same 325
ale: Argentina]?
same 329
ale: Mexico]?
same 332
ale: Earth] to [sepia-location: Mars]?
same 345
ale: Kansas]?
same 359
ale: Melbourne]?
same 363
ale: Haiti]?
same 364
ale: Nebraska] in the mid 1980s?
same 365
ale: Mozambique]?
same 368
ale: Trinidad]?
same 377
ale: Earth] revolve around the sun?
same 378
ale: Japan]?
same 380
ale: Brazil]?
same 384
ale: Brazil] become a state?
same 388
ale: Columbus] built?
same 390
ale: John Adams] born?
same 393
ale: John Adams] born?
same 397
ale: Brandenburg Gate] built?
same 404
ale: Mississippi]?
same 407
ale: Tokyo] most famous for?
same 413
ale: Procter and Gamble]?

Where is [sepia-location: Belize] located?

What type of bridge is the [sepia-location: Golden Gate Bridge]?

What is the population of [sepia-location: Bahamas]?

What state has the most [sepia-organization: Indians]?

Who is the leader of [sepia-location: India]?

What is the primary language of [sepia-location: Philippines]?

What is the population of [sepia-location: Japan]?

Where's [sepia-location: Montenegro]?

What was the name of the famous battle in 1836 between [sepia-location: Texas] and [sepia-location: Mexico]?

Where can you find the [sepia-location: Venice] flytrap?

Where did the [sepia-person: Maya] people live?

How many people live in [sepia-location: Chile]?

What is [sepia-location: California]’s state bird?

What is the name of the longest ruling dynasty of [sepia-location: Japan]?

Who killed [sepia-person: Queen]?

Who was the first [sepia-location: U.S.] president ever to resign?

How much money does the Sultan of [sepia-location: Brunei] have?

How large is [sepia-location: Missouri]’s population?

What is the nickname of [sepia-location: Pennsylvania]?

How big is [sepia-location: Australia]?

Who found [sepia-location: Hawaii]?

What is [sepia-location: California]’s state tree?

Who was the first coach of the [sepia-organization: Cleveland Browns]?

How many people die from snakebite poisoning in the [sepia-location: U.S.] per year?

Where is the bridge over the [sepia-location: river Kwai]?

How many Superbowls have the [sepia-organization: 49ers] won?

What is [sepia-person: Alice Cooper]’s real name?

Where is [sepia-location: Guam]?

Where is [sepia-location: Romania] located?

Who was the first king of [sepia-location: England]?

What is [sepia-location: California]’s capital?

What is the size of [sepia-location: Argentina]?

What is the population of [sepia-location: Mexico]?

How long would it take to get from [sepia-location: Earth] to [sepia-location: Mars]?

What is the population of [sepia-location: Kansas]?

Where is [sepia-location: Melbourne]?

What is the capital of [sepia-location: Haiti]?

How many people lived in [sepia-location: Nebraska] in the mid 1980s?

What is the population of [sepia-location: Mozambique]?

Where is [sepia-location: Trinidad]?

At what speed does the [sepia-location: Earth] revolve around the sun?

Who is the emperor of [sepia-location: Japan]?

What language is mostly spoken in [sepia-location: Brazil]?

What year did [sepia-location: Montana] become a state?

What were the names of the three ships used by [sepia-location: Columbus]?

Where was [sepia-person: John Adams] born?

Who painted [sepia-location: Olympic]?

When was the [sepia-location: Brandenburg Gate] in [sepia-location: Berlin] built?

What is the state nickname of [sepia-location: Mississippi]?

What is [sepia-location: Black Hills, South Dakota] most famous for?

Where are the [sepia-location: U.S.] headquarters for [sepia-organization: Procter and Gamble]?

What does [sepia-organization: CNN] stand for?
What is the primary language spoken in the Netherlands? What is the primary language spoken in Iceland?

How did Janice Joplin die? How did Janice Joplin die?

What instrument did Glenn Miller play? What instrument did Glenn Miller play in the U.S. Navy?

How many Admirals are there in the U.S. Navy? How many Admirals are there in the U.S. Navy?

What is the depth of the Nile River? What is the depth of the Mississippi River?

What are the two houses of the Legislative branch? What are the two houses of the Legislative branch?

What is the name of the leader of Ireland? What is the name of the leader of Ireland?

What year did the U.S. buy Alaska? What year did the U.S. buy Alaska?

What is the population of Nigeria? What is the population of Nigeria?

Which mountain range in North America stretches from Maine to California? Which mountain range in North America stretches from Maine to California?

Where is Perth? Where is Madrid?

What is the electrical output in Spain? What is the electrical output in Spain?

Who is a famous philosopher? Who is a famous philosopher?

What province is Montreal in? What province is Montreal in?

Where did Howard Hughes die? Where did Howard Hughes die?

How far is Pluto from the sun? How far is Pluto from the sun?

When was Lyndon B. Johnson born? When was Lyndon B. Johnson born?

What continent is Egypt on? What continent is Egypt on?

Who is a famous musician? Who is a famous musician?

What is Alaska's state flower? What is Alaska's state flower?

What is the length of the coastline of Hawaii? What is the length of the coastline of Alaska?

What year was the Mona Lisa painted? What year was the Mona Lisa painted?

What is the population of China? What is the population of China?

Which mountain range in North America stretches from Maine to California? Which mountain range in North America stretches from Maine to California?

What is Texas's state flower? What is Washington's state flower?

When did Idaho become a state? When did Hawaii become a state?

Where is the capital of Oregon? Where is the capital of Oregon?

What is the capital of Washington? What is the capital of Washington?

What is the oldest university in the U.S.? What is the oldest university in the U.S.?

Who was the first governor of Alaska? Who was the first governor of Alaska?

What is Ohio's state bird? What is Wisconsin's state bird?

What river flows between Fargo, North Dakota and Moorhead, Minnesota? What river flows between the Philippines and Papua New Guinea?

What is the capital of the state of Michigan? What is the capital of the state of Florida?

What is the capital of California? What is the capital of California?

What is the capital of Arizona? What is the capital of Arizona?

What is the capital of Colorado? What is the capital of Colorado?

What is the capital of Oregon? What is the capital of Oregon?

What is the capital of Washington? What is the capital of Washington?

What is the capital of Idaho? What is the capital of Idaho?

What is the capital of Nevada? What is the capital of Nevada?

What is the capital of Nevada? What is the capital of Nevada?

What is the capital of Nevada? What is the capital of Nevada?

What is the capital of Nevada? What is the capital of Nevada?
same 1431 

Who is the governor of [sepia-location: Colorado]?

same 1434 

What was the length of the [sepia-person: Wright] brothers' first flight?

same 1436 

y's founded?

same 1438 

: Rosenberg trial?

same 1439 

i: Alaska} become a state?

same 1440 

When did [sepia-person: Mike Tyson] bite [sepia-location: Holyfield]'s ear?

same 1442 

location: Philippines]?

same 1444 

Who won the [sepia-person: Caesar] for best actor in 1970?

same 1445 

What is the population of [sepia-location: Maryland]?

same 1446 

Who is the governor of [sepia-location: Tennessee]?

same 1448 

What was [sepia-person: Andrew Jackson]’s wife’s name?

same 1449 

ew Jackson]’s wife’s name?

same 1450 

What female leader succeeded [sepia-person: Ferdinand Marcos] as president of the [sepia-location: Philippines]?

same 1455 

The [sepia-location: Hindenburg] disaster took place in 1937 in which [sepia-location: New Jersey town]?

same 1457 

Ferdinand Marcos]?

same 1461 

rson: Bentley] cars?

same 1470 

on B. Johnson] born?

same 1474 

m: sea]?

same 1480 

[sepia-location: Algeria]?

same 1481 

[sepia-location: Algeria]?

same 1482 

on: Wilmington, Delaware] in?

same 1488 

er Williams], pianist, grow up?

same 1490 

ation: India] on?

same 1491 

What was the name of [sepia-person: Sherlock Holmes]’ brother?

same 1493 

olan Ryan] when he retired?

same 1495 

Shannon]’s mother die?

same 1496 

ion: Berlin] in?

same 1499 

n: Emmitt Smith] go to?

same 1501 

How much of [sepia-location: U.S.] power is from nuclear energy?

same 1502 

President Kennedy] killed?

same 1507 

[sepia-location: China]?

same 1509 

[sepia-person: Penney] store opened?

same 1513 

What is the current population in [sepia-location: Bombay], [sepia-location: India]?

same 1514 

[sepia-location: Canada]’s most populous city?

same 1517 

What is the state bird of [sepia-location: Alaska]?

same 1524 

What is the name of the ballpark that the [sepia-organization: Milwaukee Brewers] play at?

same 1526 

What school did [sepia-person: Thomas Jefferson] found?

person: Thomas Jefferson] found?

same 1530 

What is the capital city of [sepia-location: New Zealand]?

same 1532 

[sepia-location: Cuba]?

same 1533 

: Lake Washington] by?

same 1535 

How many electoral college votes in [sepia-location: Tennessee]?

same 1540 

[sepia-location: America]?

same 1550 

What is the deepest lake in [sepia-location: America]?

same 1555 

What is the southwestern-most tip of [sepia-location: England]?

same 1557 

n: Vietnam]?

same 1562 

How much does it cost to register a car in [sepia-location: New Hampshire]?

same 1563 

[sepia-location: New Hampshire]?

same 1564 

When did Led Zeppelin appear on [sepia-organization: BBC]?

same 1568 

Malone]’s nickname?

same 1570 

What is the legal age to vote in [sepia-location: Argentina]?

same 1572 

For whom was the state of [sepia-location: Pennsylvania] named?
What year did [sepia-person: General Montgomery] lead the Allies to a victory over the Axis troops in [sepia-location: North Africa]?
same 1944:
When did [sepia-person: Willis Haviland Carrier] make the air conditioner rise?
same 1923:
Who was elected President of [sepia-location: South Africa] in 1994?
same 1805:
Who was [sepia-person: Michael Jordan]’s arch enemy?
same 1851:
Which country colonized [sepia-location: Hong Kong]?
same 1849:
Who was [sepia-person: Marilyn Monroe]’s mother?
same 1828:
What number did [sepia-person: Mike Tyson] wear?
same 1840:
What is the state song of [sepia-location: Kansas]?
same 1864:
What was the name of the plane that dropped the Atomic Bomb on [sepia-location: Hiroshima]?
same 1848:
What is [sepia-person: Jacob]’s mother?
same 1875:
What county is [sepia-location: St. Paul, Minnesota] in?
same 1823:
What is Haviland Carrier’s real name?
same 1800:
What is the motto for [sepia-location: California]?
same 1859:
What is [sepia-location: China]’s original name?
same 1863:
What is [sepia-location: Italy]’s original name?
same 1820:
Who committed suicide in [sepia-location: Salzburg]?
same 1803:
Who was [sepia-person: Willis Haviland Carrier] make the air conditioner rise?
same 1804:
Which river runs through [sepia-location: Dublin]?

[Red marks: 38]
Bibliography


[17] Cycorp and OpenCyc.org. Opencyc: the open source version of the cyc technology, the world’s largest and most complete general knowledge base and commonsense reasoning engine.


126


