Compositional Semantics in Names and Numbers

Gregory Marton GREMIO@CSAIL.MIT.EDU
MIT Computer Science and Artificial Intelligence Laboratory, 200 Technology Square, Cambridge MA, 02139 USA

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, 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 data structure design is inspired by the guidelines for the recent Automatic Content Extraction pilot competition, and evaluation will be performed as part of this contest.

2. Related Work

Content Extraction

Named Entity Identification, the task of finding boundaries and categories for each named entity, was evaluated in the late 1990's (Marsh & Perzanowski, 1998), and statistical systems performed best on a wide range of data. This task does not capture any meaning, however, and is primarily suitable for treating the entities as opaque phrases.

In contrast, the Automatic Content Extraction (ACE) Workshop's task requires far more semantic understanding (Przybocki et al., 2003). The ACE task asks programs to distinguish between the White Houses in "visitors streamed through the White House", where White House is a facility, and "The White House vetoed the bill", where it is an organization. Programs should mark "New York City mayor Rudy Giuliani" as an antecedent for "The mayor" three sentences later. Programs should note that Giuliani is in the mayor relation to New York City.

The ACE competition is now in its third pilot phase, and testing will begin at the end of September.

```
Mr.  :  person / name
(lambda (name)
  (make-person 'name name
               'title "Mister"))
```

Figure 1. In Sepia's implementation of CCG, the word "Mr." is defined as a one-place function that takes a `name` to its right (a backslash would indicate leftward application) and results in a `person`. After the declaration, the function is defined in Scheme.

Applications

Content Extraction is directly applicable to answering questions like "Who is Rudolph Giuliani" or "Who is the mayor of New York". It can also help expand the initial document retrieval query so that a question about "September 11th" finds documents about "9/11". Equivalent answer candidates can be conflated instead of competing for score. The START natural language question answering system already incorporates an early content extractor (Katz et al., 1998), to be replaced by Sepia.

Combinatory Categorial Grammar

The most successful methods for named entity identification have treated it as a classification problem using linguistically shallow features. The content extraction task is better suited to a knowledge based approach because resolving relations and coreferences among entities requires a greater degree of language understanding.

One approach to knowledge engineering in language is to parse text with a compositional semantics, where the meaning assigned to each word helps to determine meanings of combinations of words. Combinatory Categorial Grammar (CCG) is a technique for doing this sort of semantic parsing. Its pioneer, Mark Steedman, has shown that CCG can capture many previously problematic phenomena across languages (Steedman, 2000). The parser follows pure function application rules, and the lexicon contains all language-specific rules, word meanings, and associations.

A lexical item consists of the word defined, a function signature called the *category* indicating how it can be applied (if at all), and a function definition in lambda calculus.

Consider for example the phrase “Mr. and Mrs. Hodson”. One lexical entry for “Mr.” is shown in Figure 1. A coordination like **and** takes two identical categories to either side and creates a new partial parse that has the same category and maps any arguments to both functions. The two single-place functions for **Mr.** and **Mrs.** combine into a new function of the same category that, when applied to the name **Hodson**, creates a set with meanings for “Mr. Hodson” and “Mrs. Hodson”.

Simple function application, function composition, coordination, and type raising allow CCG to capture compositional semantics for most of natural language. In Sepia I have implemented a CCG parser and manually created a lexicon for understanding English named entities.

3. System Overview - Sepia

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. Sepia has instead a single representation and propagates ambiguity throughout processing. In a pipeline, an early mistake is irreparable by later modules. In Sepia, the later combinations make a top-down choice from the early bottom-up possibilities.

A pipelined content extraction system would fail on the following cases:

- “Mr. and *Mrs. Hodson* and their children...”
- “They scored *a hundred and ninety-one* respectively.”
- “I told *Howard Dean* would win.”

In the first case, a later coreference module cannot associate “their” with the Hodson couple because only one person is recognized. Sepia’s solution was discussed above.

In the second case it is no longer possible to parse “respectively” because the required coordination has been consumed. 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 third case, “told” cannot find a good argument structure. If Sepia had lexical entries for “told” and other verbs, the intermediate representation with “Howard” 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.

3.2 The Lexicon

Sepia’s non-pipelined parsing algorithm guarantees that the right lexical items are selected, but only if they are available! As with any knowledge engineering task, manually constructing a lexicon is a time consuming and brittle process. The parser tempers brittleness by expecting failures and dropping failed partial parses. But the greatest current source of error is missing lexical items.

Mark&Margaret’s Apt. 3c ((Mark)(&)(Margaret)((’)(s))) ((Apt)(.)) ((3)(c))

Figure 2. **Tokenization at word boundaries in multiple levels.**

The current lexicon has lists of countries and cities, of common first and last names, and many cues for locations, organziations, and facilities. The lexicon needs many more entries capturing relations among entities, which would help select between ambiguous entity candidates.

3.3 Parsing Example

In the example in Figure 2, **’s** is recognized as a multi-token token because it is in the lexicon. That possessive can apply either to only **Margaret** or to the conjuction **Mark&Margaret**. What **’s** applies to then determines which of **Margaret** or **Mark&Margaret** is the possessor. But if **’s** applies only to **Margaret** instead of to the conjuction, then the conjuction tries to apply later and fails: **Margaret’s Apt. 3c** is not the same type (facility) as **Mark** (person). In this case, two partial parses are recognized instead of the one (preferred) complete parse.

4. Future Work

In the immediate future, the ACE competition begins September 29th! In the process I will extend the lexicon, and will add a coreference mechanism. In the longer term, I will attempt to automate learning of a broad-coverage lexicon (Hockenmaier et al., 2002).

References

Hockenmaier, J., Bierner, G., & Baldridge, J. (2002). Extending the coverage of a CCG system. *Proc. of the Assoc. for Computational Linguistics ’02*.

Katz, B., Yuret, D., Lin, J., Felshin, S., & *et al.*(1998). Blitz: A preprocessor for detecting context-independent linguistic structures. *Proc. of PRICAI ’98*.

Marsh, E., & Perzanowski, D. (1998). Muc-7 evaluation of ie technology: Overview of results. *Message Understanding Conference MUC-7 Proceedings*.

Przybocki, M., Harman, D., & Doddington, G. (2003). Automatic content extraction (ACE) (http://www.itl.nist.gov/iad/894.01/tests/ace/).

Steedman, M. (2000). *The syntactic process*. MIT Press.