The Generic Information Extraction System

Jerry R. Hobbs
Artificial Intelligence Center
SRI International
Menlo Park, CA 94025

INTRODUCTION

An information extraction system is a cascade of transducers or modules that at each step add structure and often lose information, hopefully irrelevant, by applying rules that are acquired manually and/or automatically.

Thus, to describe an information extraction system is to answer the following questions:

- What are the transducers or modules?
- What are their input and output? Specifically,
  - What structure is added?
  - What information is lost?
- What is the form of the rules?
- How are the rules applied?
- How are the rules acquired?

As an example, consider the parsing module. The parser is the transducer. The input is the sequence of words or lexical items that constitute the sentence. The output is a parse tree of the sentence. This adds information about predicate-argument and modification relations. Generally, no information is lost. The rules might be in the form of a unification grammar and be applied by a chart parser. The rules are generally acquired manually.

Any system will be characterized by its own set of modules, but generally they will come from the following set, and most systems will perform the functions of these modules somewhere.

1. Text Zoner, which turns a text into a set of text segments.
2. Preprocessor, which turns a text or text segment into a sequence of sentences, each of which is a sequence of lexical items, where a lexical item is a word together with its lexical attributes.
3. Filter, which turns a set of sentences into a smaller set of sentences by filtering out the irrelevant ones.
4. Preparser, which takes a sequence of lexical items and tries to identify various reliably determinable, small-scale structures.
5. Parser, whose input is a sequence of lexical items and perhaps small-scale structures (phrases) and whose output is a set of parse tree fragments, possibly complete.

6. Fragment Combiner, which tries to turn a set of parse tree or logical form fragments into a parse tree or logical form for the whole sentence.

7. Semantic Interpreter, which generates a semantic structure or logical form from a parse tree or from parse tree fragments.

8. Lexical Disambiguation, which turns a semantic structure with general or ambiguous predicates into a semantic structure with specific, unambiguous predicates.

9. Coreference Resolution, or Discourse Processing, which turns a tree-like structure into a network-like structure by identifying different descriptions of the same entity in different parts of the text.

10. Template Generator, which derives the templates from the semantic structures.

I will elaborate on each of these modules in turn.

TEXT ZONING

This module "parses" the text into text segments. At a minimum it would separate the formatted from the unformatted regions. Some systems may go farther and segment the unformatted text into topic areas, either by looking for discourse particles like "meanwhile", or by statistical means. The header is the only formatted region in the Tipster texts. This module in MUC-5 systems will store the date and source information from the header for entry into the template, and the date will be used to interpret temporal deictics like "last month" during subsequent processing. Some header information is often thrown away as irrelevant.

Few if any systems have a systematic treatment of text zoning—only ad hoc code that is developed manually.

PREPROCESSOR

This module takes the text as a character sequence, locates the sentence boundaries, and produces for each sentence a sequence of lexical items. The lexical items are generally the words together with the lexical attributes for them that are contained in the lexicon. This module minimally determines the possible parts of speech for each word, and may choose a single part of speech. It makes the lexical attributes in the lexicon available to subsequent processing. It recognizes multiwords. It recognizes and normalizes certain basic types that occur in the genre, such as dates, times, personal and company names, locations, currency amounts, and so on. It handles unknown words, minimally by ignoring them, or more generally by trying to guess from their morphology or their immediate context as much information about them as possible. Spelling correction is done in this module as well.

The methods used here are lexical lookup, perhaps in conjunction with morphological analysis; perhaps statistical part-of-speech tagging; finite-state pattern-matching for recognizing and normalizing basic entities; standard spelling correction techniques; and a variety of heuristics for handling unknown words.

The lexicon might have been developed manually or borrowed from another site, but more and more they are adapted from already existing machine-readable dictionaries and augmented automatically by statistical techniques operating on the key templates and/or the corpus.

FILTER

This module uses superficial techniques to filter out the sentences that are likely to be irrelevant, thus turning the text into a shorter text that can be processed more quickly. There are two principal methods used in this module. In any particular application, subsequent modules will be looking for patterns of words that signal relevant events. If a sentence has none of these words, then there is no reason to process it further. This module may scan the sentence looking for these keywords. The set of keywords may be developed manually, or more rarely if ever, generated automatically from the patterns.

Alternatively, a statistical profile may be generated automatically of the words or n-grams that characterize relevant sentences. The current sentence is evaluated by this measure and processed only if it exceeds some threshhold.

PREPARSER

More and more systems recently do not attempt to parse a sentence directly from the string of words to a full parse tree. Certain small-scale structures are very common and can be recognized with high reliability. The Preparsing module recognizes these structures, thereby simplifying the task of the Sentence Parser. Some systems recognize noun groups, that is, noun phrases up through the head noun, at this level, as well as verb groups, or verbs together with their auxilliaries. Appositives can be attached to their head nouns with high reliability, as can genitives, "of" prepositional phrases, and perhaps some other prepositional phrases. "That" complements are often recognized here, and NP conjunction is sometimes done as a special process at this level.

Sometimes the information found at this level is merely encapsulated and sometimes it is discarded. Age appositives, for example, can be thrown out in many applications.

This module generally recognizes the small-scale structures or phrases by finite-state pattern-matching, sometimes conceptualized as ad hoc heuristics. They are acquired manually.

PARSER

This module takes a sequence of lexical items and perhaps phrases and normally tries to produce a parse tree for the entire sentence. Systems that do full-sentence parsing usually represent their rules either as a phrase structure grammar augmented with constraints on the application of the rules (Augmented Transition Networks, or ATNs), or as unification grammars in which the constraints are represented declaratively. The most frequent parsing algorithm is chart parsing. Sentence are parsed bottom-up, with top-down constraints being applied. As fragmentary parsing becomes more prevalent, the top-down constraints cannot be used as much. Similar structures that span the same string of words are merged in order to bring the processing down from exponential time to polynomial time.

Recently more and more systems are abandoning full-sentence parsing in information extraction applications. Some of these systems recognize only fragments because although they are using the standard methods for full-sentence parsing, their grammar has very limited coverage. In other systems the parser applies domain-dependent, finite-state pattern-matching techniques rather than more complex processing, trying only to locate within the sentence various patterns that are of interest in the application.

Grammars for the parsing module are either developed manually over a long period of time or borrowed from another site. There has been some work on the statistical inference of grammar rules in some areas of the grammar.

FRAGMENT COMBINATION

For complex, real world sentences of the sort that are found in newspapers, no parser in existence can find full parses for more than 75% or so of the sentences. Therefore, these systems need ways of combining the parse tree fragments that they obtain. This module may be applied to the parse tree fragments themselves. Alternatively, each fragment is translated into a logical form fragment, and this module tries to combine the logical form fragments. One method of combination is simply to take the logical form of the whole sentence to be the conjunction of the logical form fragments. A more informative technique is to attempt to fit some of the fragments into unfilled roles in other fragments.

The methods that have been employed so far for this operation are ad hoc. There is no real theory of it. The methods are developed manually.

SEMANTIC INTERPRETATION

This module translates the parse tree or parse tree fragments into a semantic structure or logical form or event frame. All of these are basically explicit representations of predicate-argument and modification relations that are implicit in the sentence. Often lexical disambiguation takes place at this level as well. Some systems have two levels of logical form, one a general, task-independent logical form intended to encode all the information that is in the sentence, and the other a more specifically task-dependent representation that often omits any information that is not relevant to the application. A process of logical-form simplification translates from one to the other.

The method for semantic interpretation is function application or an equivalent process that matches predicates with their arguments. The rules are acquired manually.

There are a number of variations in how the processing is spread across Modules 4-7. It may be as I have outlined here. The system may group words into phrases, and then phrases into parsed sentences, and then translate the parsed sentences into a logical form. The more traditional approach is to skip the first of these steps and go directly from the words to the parsed sentences and then to the logical forms. Recently, many systems do not attempt full-sentence parsing. They group words into phrases and translate the phrases into logical forms, and from then on it is all discourse processing. In a categorial grammar framework, one goes directly from words to logical forms.

LEXICAL DISAMBIGUATION

This "module", if it is such, translates a semantic structure with general or ambiguous predicates into a semantic structure with specific, unambiguous predicates. In fact, lexical disambiguation often occurs at other levels, and sometimes entirely so. For example, the ambiguity of "types" in "He types." and "The types ..." may be resolved during syntactic processing or during part-of-speech tagging. The ambiguity of "... rob a bank ..." or "... form a joint venture with a bank ..." may be resolved when a domain-dependent pattern is found. The fact that such a pattern occurs resolves the ambiguity.

More generally, lexical disambiguation usually happens by constraining the interpretation by the context in which the ambiguous word occurs, perhaps together with the a priori probabilities of each of the word senses.

These rules are in many cases developed manually, although this is the area where statistical methods have perhaps contributed the most to computational linguistics, especially in part-of-speech tagging.

COREFERENCE RESOLUTION

This module turns a tree-like semantic structure, in which there may be separate nodes for a single entity, into a network-like structure in which these nodes are merged. This module resolves coreference for basic entities such as pronouns, definite noun phrases, and "one" anaphora. It also resolves the reference for more complex entities like events. That is, an event that is partially described in the text may be identified with an event that was found previously; or it may be a consequence of a previously found event, as a death is of an attack; or it may fill a role in a previous event, as an activity in a joint venture.

Three principal criteria are used in determining whether two entities can be merged. First, semantic consistency, usually as specified by a sort hierarchy. Thus, "the Japanese automaker" can be merged with "Toyota Motor Corp." For pronouns, semantic consistency consists of agreement on number and gender, and perhaps on whatever properties can be determined from the pronoun's context; for example, in "its sales", "it" probably refers to a company.

Second, and more generally, there are various measures of compatibility between entities; for example, the merging of two events may be conditioned on the extent of overlap between their sets of known arguments, as well as on the compatibility of their types.

The third criterion is nearness, as determined by some metric. For example, we may want to merge two events only if they occur within n sentences of each other (unless they are in The Financial Times). The metric of nearness may be something other than simply the number of words or sentences between the items in the text. For example, in resolving pronouns, we should favor the Subject over the Object in the previous sentence; this is simply measuring nearness along a different path.

These rules have to be developed manually (and by "manually" I mean "cerebrally"). The sort hierarchy used in consistency checking is usually developed manually, although it would be interesting to know if researchers have begun to use WordNet or other thesauri for sort hierarchy development, or have attempted to use statistical means to infer a sort hierarchy.

The term "discourse processing" as used by MUC sites almost always means simply coreference resolution of application-relevant entities and events. There have been no serious attempts to recognize or use the structure of the text, beyond simple segmenting on the basis of superficial discourse particles for use in nearness metrics in coreference resolution.

TEMPLATE GENERATION

This module takes the semantic structures generated by the natural language processing modules and produces the templates in the official form required by the rules of the evalution. Events that do not pass the threshhold of interest defined in the rules are tossed out. Labels are printed, commas are removed from company names, percentages are rounded off, product-service codes are pulled out of a hat, and so on. And on and on.

There are no automatic methods for developing the rules in this module. The only method available is long, hard work.

A FINAL WORD

In this overview of the generic information extraction system, I have described what seemed to be the principal methods used in the MUC-4 systems. The reader may find that the MUC-5 systems exhibit interesting innovations over and above what I have described.