McDermott's discussion of NLP

Language

How well do computers solve the problem of language use?

Before we can ask that question, we have to decide whether there *is* a problem of language use, and if so, what it might be. If the problem is to get computers to understand spoken commands, then it is often quite easy to define, at least in principle. Given the set of all possible commands, and a particular utterance, then the question is, which command was intended by that utterance? If there are just two possible commands, as might happen with an automated balloon guidance system that need only respond to *Up* and *Down*, then the problem is to figure out whether a certain yell sounds more like one than the other. If there is essentially an infinite number of commands (say, strings of coordinates given to a more elaborate navigation system), then, as we shall see below, the problem gets somewhat more complicated.

But suppose the problem is simply to "understand" an ordinary English utterance. Historically this has been the ultimate problem of the field of "natural language processing," or perhaps of AI in general. For example, consider these two sentences (due to Winograd 1972):

The city council refused the demonstrators a parade permit because they advocated violence.
The city council refused the demonstrators a parade permit because they feared violence.

In the first sentence, one sees instantly that "they" refers to the demonstrators. In the second, it is obvious that "they" refers to the city council members. If you saw either sentence without the other, it would probably never occur to you that there was an ambiguity, especially if the sentence occurred in the context of a coherent story about a political squabble. Whatever else you may say about sentence understanding, it's clear that you don't understand sentences such as these unless you can successfully infer what "they" and other pronouns refer to.

Unfortunately, although it is not hard to enumerate inferences that a language-understanding program ought to be able to make, such an enumeration does not really pin down what problem such a program is supposed to be solving. There is an infinite number of possible inferences; which ones need to be found in order to understand a sentence?

At one time it appeared that there was a natural answer to this question. Suppose there is an *internal notation* system in which thoughts are expressed. (Fodor 1975 calls it the "language of thought.") The brain keeps track of a proposition it believes by, in essence, writing down the expression for that proposition in the internal notation, and tagging the expression with a "B" for "believed." It infers new beliefs from old ones by processes akin to the manipulations of a formal deductive system. Just as a formal deductive system allows you to conclude Q from P and *If P then Q*, so the brain concludes "the butler did it" from "this is a mystery set in an English country house" and "no one else but the butler could have done it." If you don't quite see how to make the leap from the inference involving *If P then Q* to the inference involving the butler, you're not alone; but perhaps part of the difficulty is that we don't yet understand how the internal notation works.

The nice thing about the internal-notation proposal is that, if it could be carried out, it would tell us what inferences need to be drawn in understanding a sentence, namely, just those required to recover the proper internal representation of it. The problem of pronoun interpretation fits neatly here. The internal representation of a sentence should presumably not contain any pronouns. If there is a term X_1 that is the proper translation of "the city council," and another, X_2, that is the proper translation of "the demonstrators," then the internal representation of either of the example sentences given above contains either X_1 or X_2 in the expression

about the cause of X_1's decision regarding X_2's permit. Pronouns make sense as abbreviatory devices in speech, but expressions in the internal notation are going to be stored for a long time and used in many different contexts, so it's hard to see how such abbreviations could be useful. If someone tells me "Go see Ms. Smith in Accounting. She has something for you," what I want in the internal representation of "She has something for you" is some label for the bundle of facts I know about her. For instance, if I am engaged in looking for Ms. Smith, my visual system must have ready access to information about what she looks like. A pointer to this bundle of facts is very different from the pronoun "she."[10]

Assuming you are convinced by this argument, you will then agree that you can't translate the sentences about granting the demonstrators a parade permit unless you can make inferences about the motives and likely behavior of city councils and demonstrators. If we make similar arguments about other aspects of the sentences, we might eventually wind up with a formal representation of the second sentence that looks something like this:

```
group(d3, person) /* d3 is a group of persons */
city_council(c4, city8) /* c4 is the city council of some city */
hypothetical event(e66, parade(d3, city8)
                  & 0 < time_interval(e65, e66) < 1 month)
event(e65,
      request(d3, c4,
              permission(e66)))
/* e65 is the event of d3 requesting permission for a parade in the city
   in the near future */

event(e67,
      deny(c4, d3, e65))
/* e67 is the event of c4 denying the request referred to as e65 */

reason(e66, fear(c4, violent(e66)))
```

Note that "they" has vanished, replaced by c4, the term denoting the city council. Note also that I have made several other facts explicit that were only implicit in the sentence, including the fact that the violence the council was afraid of would be associated with the planned parade.

Unfortunately, the internal-representation theory is not as healthy as it used to be. There are two problems: how to recover internal representations from sentences (and other inputs), and what to do with them once they have been recovered. The first problem may sound harder, but in fact the second is worse. Many of the solutions to the first problem don't work until you've solved the second. One such solution is to choose the internal representation that minimizes contradictions with what you already know; but this requires being able to tell when a set of beliefs in the internal notation is inconsistent.

One problem is that it appears there is no way to make inferences efficiently using the sorts of complex symbolic structures that the theory posits. If we restrict our attention to deductive systems, then most inference problems can be shown to be intractable in the worst case. "Intractability" is a technical property of a problem, and it is difficult to explain exactly what its consequences are. (I will explain it a bit further in chapter 5.) For our purposes, a problem is intractable if any algorithm that solves every instance of it will take a very long time on some instances, and in particular if the delay grows much faster than the size of instances of the problem. For every problem size larger than some low threshold there is an instance of that size that the algorithm will take more than a billion years to solve on any conceivable computer. Such problems are not out of bounds to AI; in fact, many problems of interest to AI are intractable. It may sound crazy to try to solve an unsolvable problem, but there are some loopholes: it may be useful to solve *some* problems of large size, even if not all can be solved; it may be useful to find a near-solution when no solution can be found; and, in some cases, even though a problem class is intractable there is a special subclass that can be solved efficiently.

The problem is that no one has proposed anything like a special subclass of the problem of making inferences from internal language representations that (a) is big enough to include the representations of all sentences people are likely to utter (or think about); and (b) supports efficient inference algorithms.

However, the absence of an efficient deductive inference algorithm is not the biggest obstacle to the success of the theory. The biggest obstacle is that most inferences we make are not deductive, and there is no general theory

Disambiguation Is NOT a process of minimizing contradictions, but one of conforming with familiar patterns. Consider, "He saw a monkey with yellow tail feathers" – we tend to understand this in a way that contradicts common knowledge, yet there's a "logical" interpretation …

Here he seems to make the common mistaken assumption that symbolic propositions only allow for *deductive* reasoning. On the next page he allows for nondeductive inferences, but still insists that logical symbolisms restrict us to "justifiable" inferences. And since no general such method is known, he concludes that we have no general "internal notation" and just use a bag of special computational tricks … I very much disagree.

of nondeductive inference (McDermott 1987). A deductive inference is an inference that must be true if its premises are. If you hear that your older brother got more jellybeans than your younger brother, and that your younger brother got more jellybeans than you, then you can be sure of the inference that your older brother got more than you, so long as you don't doubt either of the two premises you heard. This is an example of a deductive inference. They are hard to find outside of geometry class. If you vote for Jones because she is a Republican and therefore will support a balanced budget, you must be prepared for disappointment. That's an obvious example, but consider something much more straightforward: you go to class at 9:30 on Wednesday because the class meets Monday, Wednesday, and Friday at 9:30. It's very likely that the class will meet on this occasion, but it's not definite. There are many reasons why it might not. That's life.

There is no problem in principle with a computer making deductive or nondeductive inferences. If a robot makes a plan for going to a destination based on a map inferred from sensory data, the plan might or might not work. The inference to the plan can be wrong for all sorts of reasons, even if the premises are true (i.e., the input data were accurately sensed). Even in the case of something as straightforward as the calculation done by the IRS to determine whether you get a refund or must pay more taxes, it is not always obvious if the inference is deductive. Suppose the IRS computer is figuring the taxes of a consultant, and sees three income items, one for $5467, one for $1076, and one for $1076.39. Is the inference that the total income = $7619.39 a deductive inference? A human accountant might wonder if the second two figures were the same amount reported twice, by two different channels. Is he doubting the premises, and if so what are they?

The point is one I have made above: computers don't deduce, they calculate. Whether the conclusions they draw are deductive or not is seldom an issue.. The problem is not with getting computers to draw nondeductive conclusions; they do it all the time. The problem is to get them to do it with an arbitrary formula in the internal notation. The IRS computer can represent facts about tax returns. The map-building robot can represent facts about the layout of buildings. What's missing is a general theory of inference that will tell us what we are justified in inferring from an arbitrary

collection of facts. In the first half of the twentieth century, philosophers like Carnap worked hard on finding such a theory, and instead found many reasons to doubt that such a theory exists (Putnam 1963).

It could turn out that there is some marvelous computational engine in the brain that can manipulate the sorts of expressions exemplified above, making nondeductive inferences rapidly, smoothly, and fairly accurately without turning a hair—but I doubt it. My guess is that life tends to present us with a series of stereotypical problems, for which our brains have specialized solution techniques. Information is not stored in a general-purpose notation, but in a set of notations specialized for the algorithms that will be used to solve them. For instance, there might be one representation system for maps, a different one for manipulable objects in the immediate vicinity, and another for faces of people we are acquainted with.

Language appears to be the big counterexample to this proposal, because we can apparently hear a sentence on any topic and immediately assimilate the information it contains. But this appearance might be misleading. It is now accepted that any normal person can perform a purely *syntactic* analysis of an arbitrary novel sentence with no conscious effort. Syntactic analysis---or *parsing*---segments a sentence so that the phrases it contains are properly grouped. For example, in a sentence like "The man Fred yelled at was more helpful," we know immediately that Fred yelled at the man and that the man was more helpful (and not, for instance, that the man yelled at Fred or Fred was more helpful(, The question is what happens to the word groups after such syntactic parsing. Consider a riddle such as this one: "If a plane crashes right on the border between the United States and Canada, where would they bury the survivors?" Or this one: "A train leaves New York headed for Albany at 80 miles per hour, and simultaneously another train leaves Albany headed for New York at 40 miles per hour. When they collide, which one is closer to New York?" A significant number of people perform such shallow analysis of these seemingly simple questions that they get the meanings wrong. What model of semantic processing would account for that?

To me, this illustrates "pattern-based" interpretation

Fortunately, we can learn a lot about language without solving the problem of what it means to understand an arbitrary sentence. For instance, consider the problem of information extraction, in which the computer's

job is to extract data from sources such as newspaper stories or commercial message traffic. Suppose we are interested in tracking the occurrence of terrorist incidents around the world. We could hire people to read newspapers and summarize all the stories about terrorism. But we can provide a more precise description of the job: scan every story; if it doesn't describe a terrorist incident, discard it. Otherwise, figure out who attacked whom, on what date, in what location, with what weapons, and what damage was done. In other words, the crucial data about each story can be fit into a simple table:

Date: _____
Location: _____
Terrorist: _____
Victim: _____
Weapon: _____

Furthermore, each blank can be filled in with something simple. The date and location can be in standard formats. The terrorist can be one of several anticipated organizations (the IRA, the Shining Path, the PLO—the usual suspects), and if we can't extract a familiar name, we can just include whatever phrase was used in the article.

The point is that by focusing on this task we can sidestep the question of true understanding and replace it by the question: can computers perform this task as well and as cheaply as people?

Another example is the problem of translating from one language to another, possibly in a restricted domain. Suppose I want to translate computer-software manuals from Japanese to English. I can hire a person to do it, or I can write a computer program. In the case of a person, I would assume that a prerequisite to doing the job is the ability to understand both Japanese and English. But that's not strictly part of the definition of the problem. There might be rules that enable me to select the right syntactic structures and word choices in the target language without understanding what the words mean.

Then there is the problem of translating spoken speech into written words. Like many skills we usually take for granted, it seems effortless but is really extremely difficult. If you have ever tried to understand a native speaker of a language you have been exposed to only in school,

you have an idea of how hard it is to extract words from what sounds like a rapid stream of meaningless babble. However, this problem does have the advantage of being well defined. A string of speech sounds does usually correspond to a single string of words, and all we have to do is extract it.

In all three of these cases, we can view the computer as a proxy. It is extracting information, or translating text, or capturing spoken words, so that eventually a human being can look at them. Hence, with a couple of reservations, we can postpone the task of getting the computer to understand the words.

One reservation is that you probably can't do any of the tasks I described *perfectly* without actually understanding the words being manipulated. In the 1960s, Yehoshua Bar-Hillel wrote a paper arguing that you can't translate properly without actually understanding what you're translating (Bar-Hillel 1960). In the two sentences:

The ink is in the pen
The pig is in the pen

it's impossible to translate the word "pen" without understanding what's being described, because in most languages the word for "writing instrument" and the word for "enclosure for animals" are not the same. He concluded that translation would be impossible without first solving the understanding problem. (Then he went on to conclude that translation was impossible because the understanding problem was impossible.) But this argument is not airtight. For one thing, it may be possible to get the right translation of "pen" from the mere presence of "ink" or "pig" in the vicinity. With enough statistical knowledge of the patterns of discourse in English one might be able to fake understanding. But even if there are examples that absolutely require understanding, we can still ask whether it is possible to automate the process so that the computer costing $\$X$/hour makes about as many errors as a person costing $\$X$/hour would (or fewer)? After all, people aren't perfect at information extraction, translation, speech transcription, or anything else. Can computers compete?

The other reservation is that in the long run we will have to say more about what it means to understand language. Fortunately, the problem

doesn't seem particularly urgent. If machines have no thoughts about city councils and parade permits, getting them to talk about those things is a sterile exercise. What I expect to happen is that as people and machines have an increasing need to communicate in areas they are collaborating on, the machines will begin speaking and understanding some very minimal subsets of natural language, and these will gradually grow. At the same time we may gain greater insight into what really happens when people speak and understand.

Since that day has not yet come, let's look in some detail at what computers can do with natural language now. We start with speech recognition. If a speech signal were presented as a stream of consonants and vowels, then the problem would be to figure out how to group them into words. But the speech signal is actually just a time-varying sound, and the problem for the human ear and the computer is to extract the consonants and vowels—or *phonemes*—before it can even look for words.

Sound waves consist of slight pressure disturbances traveling through the air. A transducer, such as the human ear, converts these vibrations into a form suitable for information processing. A key fact about waves is that they can be summed, or *superposed*. If you throw two rocks into a pool at slightly different places, the two wave patterns expand forward together, and the net effect at any given point on the pool is the sum of the effects of the two rocks. Similarly for sound waves. Suppose we transmit a sound consisting of just two pitches. Each pitch corresponds to a wave of a different frequency. The two combined yield a wave that is the sum of the individual waves (figure 2.10). To hear the two pitches (as we in fact can), the ear must take the summed wave and decompose it into the two waves that it comprises. This is called a *frequency analysis* or *Fourier analysis* of the wave.

Real sounds tend to have contributions at all the frequencies over a range. When your mouth forms the sound "s," it is generating a large variety of frequencies, but only for the duration of that sound. In the word "so," the frequency bundle, or *spectrum*, of "s" is immediately followed by a different but equally complex spectrum for "o." Specifying a frequency bundle requires more than just specifying its component frequencies. We must also specify their strengths, that is, *how much* of each frequency to mix in. It turns out that as the mouth is shaped to produce

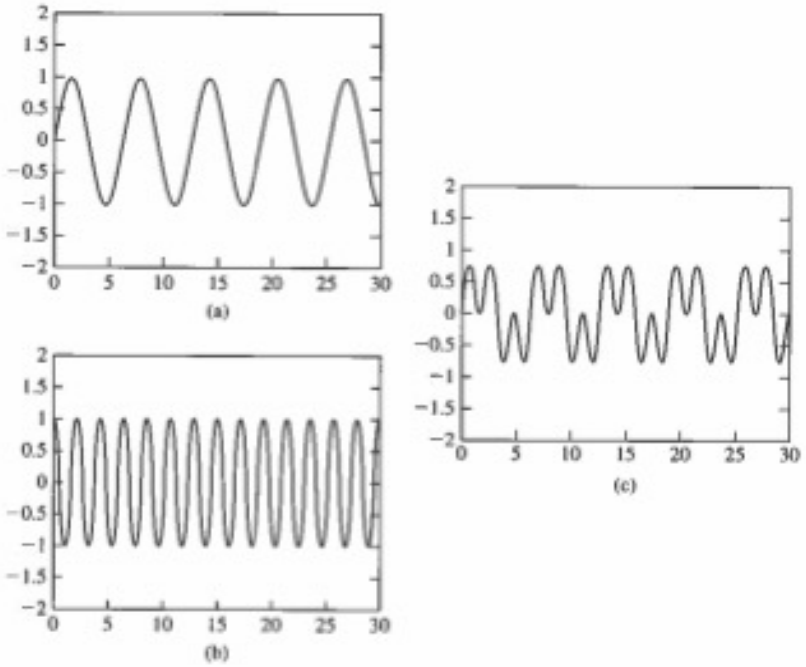

Figure 2.10
(a) Low-frequency signal. (b) High-frequency signal. (c) Sum of (a) and (b)

different vowels, it takes on different resonant frequencies. A *resonance* is a natural mode of vibration of an object. If an object is stimulated by vibrations at different frequencies, it will respond most vigorously to vibrations at its resonant frequency. The air inside the vocal passage has several resonances at any given moment. When stimulated by the sound from the vocal cords, it tends to pass the resonant-frequency components through and mute the others. The resonant frequencies associated with a particular vowel sound are called the *formant frequencies* of that vowel (Denes and Pinson 1973). We are, of course, completely unaware of this level of analysis. When we hear a vowel in our native language, we hear it as a single distinctive sound without any components. Vowels in foreign languages sound like weird versions of the vowels we are familiar with. Figure 2.11 shows the sound pattern of the sentence "Kids

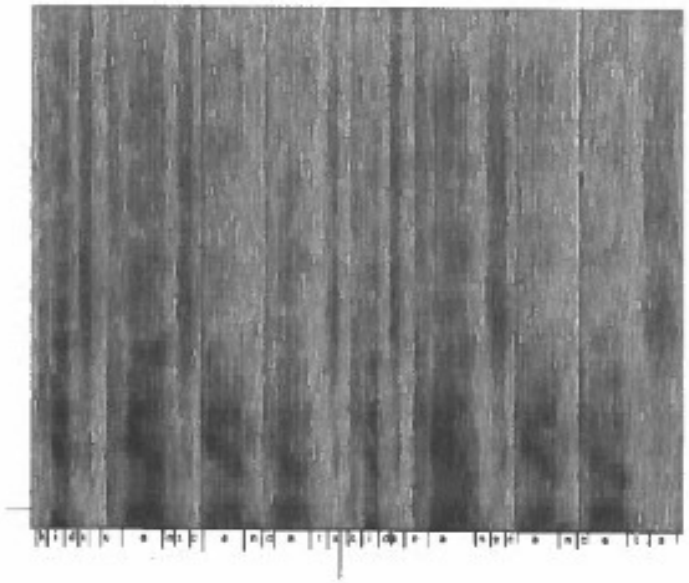

Figure 2.11
Spectrogram of sentence "Kids *can* scan cats" uttered twice

can scan cats" as spoken twice by a native of the American midwest. (The word "can" is emphasized so that it contains the same vowel as "scan.") The sentence is repeated to show that the variation between words is greater than the variation between different utterances of the same word. The vertical axis shows the amount of energy present at different frequencies as different sounds are produced. A darker patch indicates more energy at the corresponding frequency. Vowel sounds ("i" and "a" in this example) are longer than consonants, and have a distinctive pattern of dark, irregularly horizontal bands; the bands occur at the formant frequencies.

Consonants are more difficult to define than vowels. Although "s" has a distinctive spectrum, a typical consonant like "k" or "t" is defined as a transition in one of the formant frequencies. The transition is very fast and hard to spot in speech diagrams. Furthermore, what sounds like the same consonant in different words is actually a varying sound pattern

that depends on the context. A "k" before an "a" is a different pattern than a "k" before an "i" (figure 2.11).

So it seems that what we have to do in order to recognize speech is (a) break the sound stream into small segments and do a frequency analysis of each segment; (b) scan a catalog of phonemes to find the one that matches most closely. Unfortunately, it is hard to create such a catalog, because a given phoneme will correspond to many different frequency patterns, depending on the phonemes on either side of it, the exact shape of the speaker's mouth, the amount of background noise, and so on.

At this point we must resort to a probabilistic analysis of the speech data, using statistical techniques to find the most likely interpretation of a stream of sounds, the same techniques that are so useful in map learning and in other areas. What we are interested in is finding a word string W, such that

$$P(\textit{speaker said } W \mid \textit{heard sound stream } S)$$

is as large as possible. This quantity is the conditional probability that the speaker said W given that the hearer heard sounds S. Just as in map learning, we are interested in the hypothesis (W) that will maximize this probability, once we have gathered the evidence (S).

For instance, suppose the sound stream were something like, "Write before the game was tidy through an intersection." Of course, by using English words to describe this sound stream I am being (deliberately) misleading. What I should do instead is indicate it purely phonetically, perhaps this way: "R ie t b ee f or dh uh g ai m w uh z t ie d ee th r ue a n i n t ur s e k sh uh n," using one- or two-letter combinations to indicate sounds. (The sound "dh" is a voiced "th"; the word "this" starts with "dh," where as the word "thespian" starts with "th." You may never have noticed the difference, which just illustrates the point that it's easier to hear the senses than the sounds of your own language.) Call this sound stream S_1. Now consider these two possible word strings:

W_1 = Right before the game was tied he threw an interception

W_2 = Write before the game was tidy through an intersection

There are, of course, many other candidates (such as, "Write bee four the game ..." etc.). Most speakers of American English, at least those familiar with American football, would rank $P(W_1 \mid S_1)$ higher than

$P(W_2 \mid S_1)$. Although this hypothesis requires the assumption that the fourth-from-last sound was actually a "p" and not a "k," it is hard to imagine anyone saying W_2. We might assign probabilities as follows: $P(W_1 \mid S_1) = .98$; $P(W_2 \mid S_1) = .005$, ... The sum of the probabilities for all the candidates has to be 1, so the remaining .015 is divided among the other candidates. In principle, there could be a lot of them, with very tiny probabilities, but perhaps we can neglect most of those, as long as we always find the most likely candidates.

Now comes the hard part: getting a computer to compute the probabilities correctly. Over the past twenty years, there has been steady progress. The most successful programs are based on the idea of hidden Markov models of speech signals. A *hidden Markov model*, or HMM, is a way of characterizing all the different ways a word (or sentence) can be uttered, and assigning a probability to each pronunciation. Such a model is a network of *states* whose links are joined by *links* labeled with probabilities and outputs. A word is generated by starting in the special *initial state* of the HMM, then moving to a state along a link, then to another state, and so on, generating outputs as the links specify. If a link is labeled with p, this means that the link is followed with probability p, and when it is followed the output symbol s is generated. Output symbols are patterns of spectral energy, of short enough duration that every speech sound can be characterized as a sequence of such patterns.

For concreteness suppose we have a single HMM for each word. If we run it repeatedly, we will get a variety of symbol sequences out, each corresponding to a different pronunciation of the word. The more likely pronunications will be generated more often. So the model characterizes fairly directly the following conditional probability:

$P(\textit{sound stream } S \mid \textit{word } W)$.

But what we want is the opposite, $P(W \mid S)$. Fortunately, we can use Bayes's Theorem again to estimate this quantity given a set of HMMs representing words. We can combine statistical models of the way sounds make up words with statistical models of the way words occur in sentences to produce models for recognizing words in context. In case you're wondering where all these models come from, the answer is that they can be inferred automatically from samples of speech. Consult Jelinek (1997)

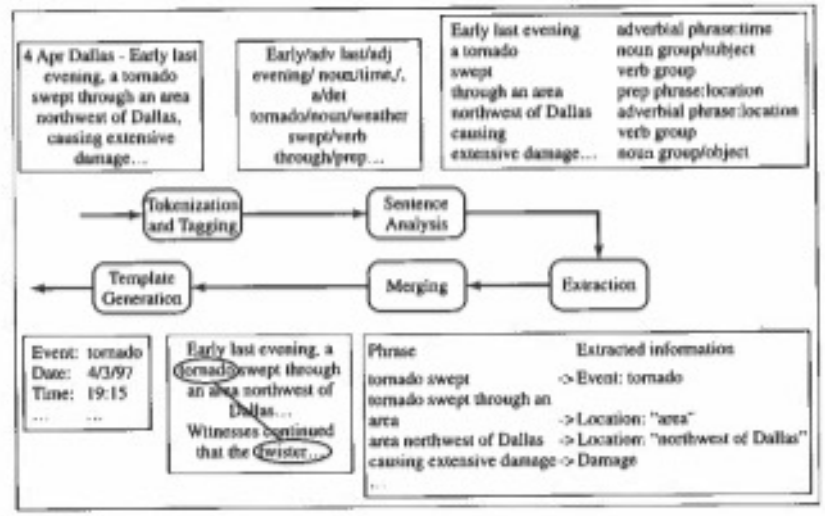

Figure 2.12
Architecture of an information-extraction system (from Cardie 1997, fig. 2)

or Rabiner and Juang (1993) for details, but be prepared to get through some fairly high-powered mathematics.

Today speech-recognition systems are becoming commercially available. They are reliable for isolated words, somewhat less so for recognition of continuous speech. One company claims 95% accuracy in recognition of continuous speech at 160 words per minute. Typically, to achieve this kind of performance the system must be getting sound input only from the speaker, without any loud distracting noises in the background. Still, these are very impressive figures.

Let's turn our attention now to *information extraction*, in which texts are scanned to find fragments that fill slots in an output template. Figure 2.12 shows the architecture of a typical information-extraction system. In the first two phases the sentence is analyzed syntactically. (These systems usually start with printed words, so no speech recognition is necessary.) As I implied above, this parsing process requires assigning the correct syntactic structure to the whole sentence (Jackendoff 1987), but information extractors usually don't try to do that. One reason is that sentences often have several different possible syntactic analyses, which

we are normally unaware of. Another is that some inputs use bad grammar, and so have no complete syntactic analysis, or no correct one. To sidestep these problems, information extractors do "partial parsing," in which only phrases whose analysis is fairly certain are found. For example, in a sentence like "The Liberation Front blew up the Minister with his bodyguards," it is obvious only after semantic analysis that "with his bodyguards" does not modify "blew up." (Contrast "The Front blew up the Minister with grenades.") But we can extract the phrases "The Liberation Front," "blew up," "the Minister," and "with his bodyguards" with fair confidence that they are at least constituents of the correct analysis.

In the next phase, the phrases found are used to generate pieces of information that may ultimately be part of the answer. This phase relies on domain-specific relations between phrases. If "blew up" is followed by a noun phrase, then in the terrorism domain it is probably giving us two slot fillers: the victim of a terrorist act (the noun phrase) and the weapon used (explosives). Any phrase that doesn't fit an extraction pattern is ignored, the hope being that it is not relevant to the target domain.

However, the program can't just throw the information so gathered into the output template. There may be more than one such template, so it is important to figure out how the fragments fit together. This calculation is done during the merge phase of information extraction. The key step is to realize when different noun phrases and pronouns probably refer to the same entity. This is a generalization of the pronoun-reference problem I discussed at the beginning of this section. In this situation the program has to infer that, for instance, "the tornado" and "the twister" refer to the same entity. It does so by merging any two expressions that could be synonyms and have no contradictory properties.

The performance of an information-extraction program is measured using two quantities, recall and precision. *Recall* is the fraction of relevant facts in a text that the program actually finds. *Precision* is the fraction of facts the program finds that are correct. Suppose a program for extracting information about terrorist incidents fills its template as follows:

Date: July 15, 1998
Location: ??
Terrorist: the French Press

Victim: General Francois Mercredi
Weapon: ??

The original story might have been:

> The French government revealed yesterday that General Francois Mercredi was the victim of a terrorist incident last month. Gen. Mercredi was last seen on July 15. His body was found just yesterday, with two bullet holes to the head, and a note in the pocket from the Aquarian Liberation Front. He had had a rocky career, having been repeatedly attacked by the French Press, but seemed to be vindicated after being made Commander of the Fifty-Fifth brigade. He was last seen leaving his villa on the way to work.

In this case there are four pertinent facts specified in the article (the victim, the date, the terrorist group, and the weapon). The program has found two of them and also surmised that the terrorist was the "French Press." Hence the recall is $2/4 = 50\%$. Of the three "facts" the program found, two are correct, so the precision is $2/3 = 67\%$.

Several information-extraction programs have been written, notably those produced by the contestants in the Message Understanding Competitions sponsored by the Defense Advanced Research Projects Agency in the early 1990s. Since then programs have been written to extract information from other kinds of text, such as medical records (Soderland et al. 1995). Getting a program to work in a given domain requires careful analysis of the concepts, words, and phrases in that domain. In a typical case one can achieve recall rates of about 50% and precision rates of about 70% (Cardie 1997). These rates may not sound very impressive, but the rates for people are not that close to 100% either, and, moreover, people take longer. For many applications, such as doing a fast scan through thousands of articles for relevant information, it might be better to employ a computer than a person.

It is not clear how much programs like this have to say about the way people process language. Some of the assumptions they make appear to be obviously wrong. For example, it has been known since the 1950s (Chomsky 1957) that Markov models are not powerful enough to represent the grammar of a real human language, so it may seem crazy to rely on them so heavily in speech recognition. However, the role they play in speech recognition is to account for variations in data, not for grammar as such. Besides, there are ways of incorporating similar ideas into

more powerful grammatical mechnanisms, yielding probabilistic phrase-structure grammars (Charniak 1993).

Even if such technical objections can be dealt with, it seems as if little progress has been made on actually "understanding" natural language. The information-extraction model seems unable to account for humans' ability to hear something unexpected, something that would fall outside the range of the templates it is trying to fill in. And the theory is silent on the question of why and how sentences are generated in the first place.

These criticisms are reasonable but not conclusive. It is an open question how novel a sentence can be and still be understood by the average person the first time he or she hears it. Conversations often involve fairly formulaic topics. Perhaps after syntactic analysis people troll through a sentence looking for material relevant to the current topic, extract enough morsels of information to advance the conversation a bit, and discard the rest. In any case, the alternative—that an arbitrary content can be generated, placed in one's mental model, and used without further ado—seems very dubious for reasons outlined above.