Natural Language Processing

ARAVIND K. JOSHI

Natural language processing (NLP) is the study of mathematical and computational modeling of various aspects of language and the development of a wide range of systems. These include spoken language systems that integrate speech and natural language; cooperative interfaces to databases and knowledge bases that model aspects of human-human interaction; multilingual interfaces; machine translation; and message-understanding systems, among others. Research in NLP is highly interdisciplinary, involving concepts in computer science, linguistics, logic, and psychology. NLP has a special role in computer science because many aspects of the field deal with linguistic features of computation and NLP seeks to model language computationally.

LANGUAGE (SPOKEN AND WRITTEN) IS CENTRAL TO ALL aspects of our communication. Therefore natural language processing systems (NLP), both current and future, are bound to play a crucial role in our communication with machines and even among ourselves. The importance of NLP to progress in telecommunications and computer science cannot be underestimated. NLP systems include systems for speech recognition, language understanding, and language generation. Spoken language systems are those that integrate speech and language systems. Such systems provide an interface to databases and knowledge bases (airline information and reservation systems, for example), expert systems for scheduling, planning, and maintenance, among others. Text processing and message understanding systems are useful for extracting information from texts and formating it in a variety of ways for further use. Language communication often occurs in two or more languages. Multilingual NLP has applications to a variety of multilingual interfaces ranging from providing aids for translating foreign language correspondence, translating equipment manuals, and speech-to-speech translation in limited domains, among others.

NLP is concerned with (i) the study of mathematical and computational models of the structure and function of language, its use, and its acquisition and (ii) the design, development, and implementation of a wide range of systems as mentioned above. On the theoretical side, the study involves mathematical and computational modeling of syntax, semantics, pragmatics (that is, certain aspects of the relationship of the speaker and the hearer, or user and the system in the case of an NLP system), and discourse aspects of language. These investigations are interdisciplinary and involve concepts in computer science including artificial intelligence, linguistics, logic, and psychology. NLP has a very special role because many branches of computer science deal with linguistic aspects of computation and NLP aims to model language computationally. This mutual relationship has roots in the history of logic and linguistics and has been especially visible since the mid-1950s. It is also important for the key role NLP plays in cognitive science.

It is impossible to cover the whole range of theoretical and practical issues in NLP in the limited space available. I have therefore selected three topics for detailed discussion: (i) grammars and parsing, an active theoretical area in NLP; (ii) statistical approaches to NLP, which entail the use of very large quantities of data in the development of the theories, a relatively new trend in NLP; and (iii) multilingual processing, a rich domain for testing current and new formalisms in all aspects of NLP and integrating a variety of techniques in NLP for a very important application such as machine translation.

This article is clearly not a survey of the entire field of NLP; it is not even a comprehensive survey of the three selected areas. Many major topics have been omitted, all of which are very important to NLP. I have not discussed speech recognition and synthesis at all, and in the language area, I have not discussed planning and discourse structure, which are crucial to natural language understanding and generation and their applications to cooperative interfaces (*1*).

Grammars and Parsers

Almost every NLP system has a grammar and an associated parser. A grammar is a finite specification of a potentially infinite number of sentences, and a parser for the grammar is an algorithm that analyzes a sentence and assigns one or more structural descriptions to the sentence according to the grammar, if the sentence can be characterized by the grammar. The structural descriptions are necessary for further processing, for example, for semantic interpretation. Chomsky's work on formal grammars in the late 1950s was the beginning of the investigations of mathematical and computational modeling of grammars (*2*). He introduced a hierarchy of grammars (finite state grammars, context-free grammars, context-sensitive grammars, and unrestricted rewriting systems) and investigated their linguistic adequacy.

Many NLP systems are based on context-free grammars (CFG). I will briefly describe CFGs. A CFG, G, consists of a finite set of nonterminals (for example, S: sentence; NP: noun phrase; VP: verb phrase; V: verb; ADV: adverb), a finite set of terminals (for example, *Harry, peanuts, likes, passionately*), and a finite set of rewrite rules of the form $A \rightarrow W$, where A is a nonterminal and W is a string of zero or more nonterminals and terminals. S is a special nonterminal called the start symbol. In Fig. 1 a simple example of a CFG is given. The rewrite rules in the left column are called syntactic rules and the rules in the right column are called lexical rules, as these rules rewrite a nonterminal into terminals or lexical items. A derivation in a grammar begins with S, the start symbol. S is rewritten as a string of nonterminals and terminals, with the use of a rewrite rule applicable to S. The new nonterminals are then rewritten according to the rewrite rules applicable to them, until no

The author is in the Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104.

Syntactic Rules	*Lexical Rules*
S → NP VP	NP → Harry
VP → VP ADV	NP → peanuts
VP → V NP	V → likes
	ADV → passionately

Structural description assigned to the sentence:
Harry likes peanuts passionately

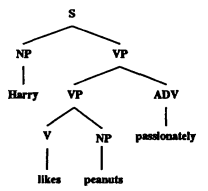

Fig. 1. A context-free grammar.

further rules can be applied. It is easy to see that the sentence Harry likes peanuts passionately can be generated by the grammar. In Fig. 1, the tree on the right shows the structural description assigned by the grammar to the sentence spelled out by the lexical items appearing at the frontier nodes of the tree. A finite-state grammar is like a CFG, except that the rewrite rules are of the form $A \rightarrow aB$ or $A \rightarrow a$, where A and B are nonterminals and a is a terminal symbol. A context-sensitive grammar is also like a CFG, except that the rewriting of a nonterminal is dependent on the context surrounding the nonterminal, unlike the rewrite rules in CFG where the rewriting is context-independent.

CFGs, as defined above, are inadequate for a variety of reasons and need to be augmented. The two main reasons are as follows: (i) The information associated with a phrase (a string of terminals) is not just the atomic symbols used as nonterminals. A complex bundle of information (sets of attribute-value pairs, called feature structures) has to be associated with strings, the syntactic category of the phrase being only one such feature, for example. Appropriate structures and operations for combining them are needed together with a CFG skeleton. (ii) The string combining operation in a CFG is concatenation, that is, if u and v are strings, v concatenated with u gives the string $w = uv$, that is, u followed by v. More complex string-combining as well as tree-combining operations are needed to describe various linguistic phenomena. I will illustrate these two kinds of augmentations by some simple examples.

CFG-Based Unification Grammars

A feature structure consists of a set of attribute-value pairs, where a value may be atomic or may be another feature structure. In Fig. 2, the feature structure X_1 consists of a feature *cat* (*category*) whose value is NP and a feature *head* whose value is another feature structure. This feature structure has only one attribute, *agreement*, whose value is another feature structure with attributes *number* and *person* with values *singular* and *third* respectively. X_1 is a feature structure that can be appropriately associated with the phrase *Fido*. Similarly X_2 is a feature structure than can be appropriately associated with the phrase *snores*. The context-free rewriting rule $X_0 \rightarrow X_1 X_2$ can be interpreted as an instruction for combining the strings *Fido* and *snores* to give the string *Fido snores* and building the feature structure X_0 to be associated with it, as shown in Fig. 2. This little example illustrates the main idea behind CFG-based unification grammars (*3*).

The main operation for combining feature structures is called unification. Given two feature structures A and B, we get a new feature structure C by unifying A and B, which has all the information in A and all the information in B and no more. Of course, if A and B have contradictory information, then A and B will fail to unify. A variety of grammars such as Generalized Phrase Structure Grammar (GPSG) (*4*), Head Driven Phrase Structure Grammar (HPSG) (*5*) and Lexical Functional Grammar (LFG) (*6*) are essentially based on CFG-based unification grammars. An introduction to unification-based grammars appears in (*3*). Unification is a very powerful operation and, unless restricted, CFG-based unification grammars are Turing-Machine equivalent (that is, their computing power equals the power of a general-purpose computing machine with unlimited working tape). From a linguistic point of view, these grammars have to be restricted so that their descriptive power is no more than necessary, and from a computational point of view, they have to be restricted in order to yield efficient parsing algorithms (*7*). Both these considerations form the basis for continued research in this area.

The logic-based approach to grammars also uses a restricted type of unification. In this approach, a grammar is viewed as a deductive system and derivations in a grammar can be viewed as deductions. Such grammars are typically embedded in a logic programming language such as Prolog (*8*). Recently, the Government and Binding theory of Chomsky has been implemented in a logic-based grammar (*9*).

Mildly Context-Sensitive Grammars

In any mathematical or computational grammar, a wide range of dependencies among the different elements in the grammar have to be described. Some examples of these dependencies are as follows: (i) Agreement features such as person, number, and gender. For example, in English, the verb agrees with the subject in person and number. (ii) Verb subcategorization, in which each verb specifies one (or more) subcategorization frames for their complements. For instance, *sleep* does not require any complement, (as in *Harry sleeps*), *like* requires one complement (as in *Harry likes peanuts*), *give* requires two complements (as in *Harry gives Susan a flower*), and so forth. (iii) Sometimes the dependent elements do not appear in their normal positions. In Who_i *did John invite* e_i where e_i is a stand-in for who_i, who_i the filler for the gap e_i. The filler and the gap need not be at a fixed distance. Thus in who_i *did Bill ask John to invite* e_i, the filler

$$X_1: \begin{bmatrix} \text{cat}: & \text{NP} \\ \text{head}: & \begin{bmatrix} \text{agreement}: & \begin{bmatrix} \text{number}: & \text{singular} \\ \text{person}: & \text{third} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$X_2: \begin{bmatrix} \text{cat}: & VP \\ \text{head}: & \begin{bmatrix} \text{form}: & \text{finite} \\ \text{subject}: & \begin{bmatrix} \text{agreement}: & \begin{bmatrix} \text{number}: & \text{singular} \\ \text{person}: & \text{third} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$X_0 \rightarrow X_1 \; X_2$$

$$X_0: \begin{bmatrix} \text{cat}: & \text{S} \\ \text{head}: & \text{(value is the same as the value of the head feature of } X_2\text{)} \end{bmatrix}$$

Fig. 2. CFC-based unification grammar.

and the gap are more distant than in the previous sentence. (iv) Sometimes the dependencies are nested. In German, for example, one could have *Hans$_i$ Peter$_j$ Marie$_k$ schwimmen$_k$ lassen$_j$ sah$_i$* (Hans saw Peter make Marie swim), where the nouns (arguments) and verbs are in nested order, as the subscripts indicate. (v) However, in Dutch, these dependencies are crossed, as for example, in *Jan$_i$ Piet$_j$ Marie$_k$ zag$_i$ laten$_j$ zwemmen$_k$ (Jan saw Piet make Marie swim)*. There are, of course, situations where the dependencies have more complex patterns.

Precise statements of such dependencies and the domains over which they operate constitute the major activity in the specification of a grammar. Mathematical and computational modeling of these dependencies is one of the key areas in natural language processing. Many of these dependencies (for example, the crossed dependencies discussed above) cannot be described by context-free grammars (*10–12*).

In the context-free grammar (CFG) in Fig. 1 the dependency between a verb (*likes*) and its two arguments [subject (NP) and object (NP)], is specified by means of two rules of the grammar. It is not possible to specify this dependency in a single rule without giving up the VP (verb phrase) node in the structure. That is, if we introduce a rule, S → NP V NP, then we express the dependency in one rule, but then we cannot have VP in our grammar. Hence, if we regard each rule of a CFG as specifying the domain of locality, then the domain of locality for a CFG cannot locally (that is, in one rule) encode the dependency between a verb and its arguments, and still keep the VP node in the grammar.

In the tree-adjoining grammar (TAG) in Fig. 3 (top), each word is associated with a structure (tree) (the word serves as an *anchor* for the tree) which encodes the dependencies between this word and its arguments (and therefore indirectly its dependency on other words which are anchors for structures that will fill up the slots of the arguments). Thus for *likes*, the associated tree encodes the arguments of *likes* (that is, the two NP nodes in the tree for *likes*) and also provides slots in the structure where they would fit. The trees for *Harry* and *peanuts* can be substituted, respectively, in the subject and object slots of the tree for *likes*. The tree for *passionately* can be inserted (adjoined) into the tree for *likes* at the VP node. In a TAG, the entire grammar consists of lexical items and their associated structures. There are universal operations, substitution, and adjoining which describe how structures can be combined (*13–15*).

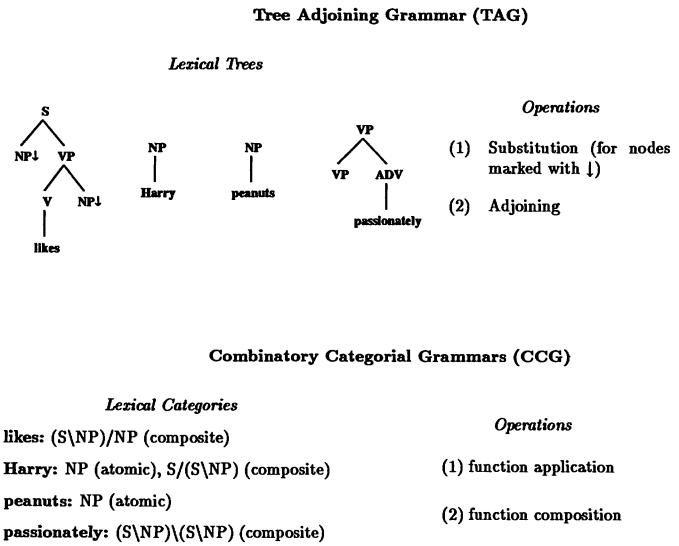

Fig. 3. Two grammar formalisms with domains of locality larger than the domain of locality for CFG. (**Top**) Tree-adjoining grammar. (**Bottom**) Combinatory categorical grammar.

In the combinatory categorical grammar (CCG) in Fig. 3 (bottom), each word is assigned a category, atomic or composite. The category for *Harry* and *peanuts* is NP, an atomic category. For *likes*, the category is (S\P)/NP. This expression encodes the information that *likes* has two arguments. The category can be interpreted as a function, which when applied to an argument NP (the object) on the right, returns (S\P), which is also a function. This function, when applied in turn to an argument NP (the subject) on the left, returns S (sentence). In this representation, (S\P) serves the same role as VP. In a CCG, the entire grammar consists of lexical items and their category assignments. There are two universal operations, function application and function composition, which describe how categories are combined. Note that *passionately* is combined with *likes peanuts* by function composition. CCG also allows type raising. For example, *Harry* has the category NP, but we can also assign another category to *Harry*, namely S/(S\P), that is, a function requiring a verb-phrase on the right and returning S. This category assignment is appropriate only if *Harry* is in the subject position (*16, 17*).

Both CCG and TAG have domains of locality that are larger than that for CFG, because in each case all the arguments of the verb *likes* are encoded in structures associated with the verb and yet, the node VP (= S\P in CCG) is available. The larger domain of locality allows TAG to completely factor out recursion from the domain of dependencies, thus localizing all dependencies in the elementary trees (*14*). For the linguistic significance of CCG and TAG, see (*16–20*).

TAG and CCG are very similar. In fact, they have been shown to be formally equivalent with respect to their weak generative capacity (that is, the sets of sentences they generate). They are more powerful than CFG and belong to a class of grammars that we call mildly context-sensitive grammars (MCSG) (*21*). This class preserves many of the essential properties of CFG and yet is able to provide enough power to capture a wide range of dependencies of language structure, such as the crossed dependencies we discussed earlier. Several other recent formalisms, for example, Linear Indexed Grammar and Head Grammar, have also been shown to be equivalent to TAGs (*21–23*). This equivalence of a number of linguistically motivated grammars based on quite distinct insights into the structure of language has led to the search for invariances across this class of grammars, these invariances being more important in some sense than the individual grammars (*21*).

We have been implicitly assuming that a grammar assigns a unique structure to a sentence (assuming that the sentence is unambiguous). Thus for example *Harry likes peanuts* will be bracketed as follows (ignoring the phrase labels and ignoring some brackets not essential for our present purpose):

(a) (Harry (likes peanuts))

It is possible in a CCG to assign multiple structures to unambiguous sentences (*16*). Thus CCG assigns the following two groupings to *Harry likes peanuts*:

(b) ((Harry (likes peanuts))

(c) (((Harry likes) peanuts)

The justification for such multiple structures is their use in coordinations (for example, with *and*) and in defining intonational phrases. Thus the bracketing (b) is necessary for (d) and the bracketing (c) for (e).

(d) ((Harry ((likes peanuts) and (hates cashews)))

(e) ((((Harry likes) and (Bill hates)) cashews)

Also, (b) corresponds to the intonational phrasing if the previous context is (f) and (c) if the previous context is (g).

(f) (Who like peanuts? (Harry (likes peanuts))

(g) (What does Harry like? ((Harry likes) peanuts)

The flexibility in the assignment of structure is achieved by giving up the notion of a canonical structure. Thus in Fig. 3 (bottom), if *Harry* is assigned the category S/(S\P), it can either combine with

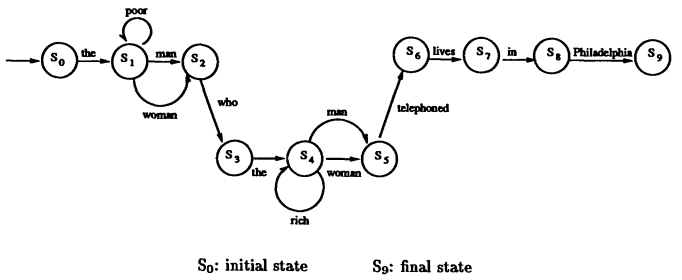

Fig. 4. A finite state machine generating sentences.

likes by function composition giving the structure in (c) above, or it can apply to the predicate *likes peanuts* to yield (b) above (*16*). However, it is not necessary to give up the notion of canonical structure. It is possible to maintain a fixed structure at a certain level (at the level of elementary trees in a TAG, for example) and still achieve the kind of flexibility needed for examples shown above (*24*). Similar results can be obtained in the Head-Driven Phrase Structure Grammar (HPSG) framework (*25*).

Parsing Complexity

A parser for a grammar is an algorithm that assigns to a sentence one or more structural descriptions according to the grammar, if the sentence is generable by the grammar. Parsing of sentences according to different grammars and the complexity of this process are important research areas in NLP. For a CFG a number of parsing algorithms are known and the time required to parse a sentence of length n is at most Kn^3 where K depends on the size of the grammar. This result extends to almost all CFG-based grammars used in NLP. The constant K can become very large however. In practice, of course, the worst case complexity is really not the important measure. Most parsers perform much better than the worst case on typical sentences. [A similar situation holds for morphological analyzers, that is, analyzers which break words into their parts (*26*).] There are no mathematical results, as yet, to characterize the behavior on typical sentences. Grammars that are more powerful than CFG are, of course, harder to parse, as far as the worst case is concerned. The grammars in the class of Mildly Context-Sensitive Grammars discussed earlier can all be parsed in polynomial time just as CFG, however, the exponent for n is 6 instead of 3. For some further results on complexity, especially in the context of the so-called principle-based parsing of the Government and Binding Theory Grammar, see (*27, 28*).

A crucial problem in parsing is not just to get all possible parses for a sentence but to rank the parses according to some criteria. If a grammar is combined with statistical information (see below), then that information can be used to provide this ranking. This is exactly what is done in many spoken language systems, that is, systems that integrate speech recognition and language processing (*29*).

In the discussion so far, I have been assuming that the parser only handles complete sentences and the parser either succeeds in finding the parse(s) for a sentence or it fails. In practice, we want the parser to be flexible—that is, it should be able to handle fragments of sentences—and it should fail gracefully—that is, it should provide as much analysis as possible for as many fragments of the sentence as possible, even if it cannot glue all the pieces together. A parser with such properties based on the idea of deterministic parsing (*30*) has been described in (*31*) and used in the construction of a large corpus of parsed text, a tree bank (*32*).

Finally, the actual grammars in major NLP systems are large, but even with this large size their coverage is not adequate. Building the grammar by hand soon reaches its limit and there is no guarantee that it will be increasingly better in coping with free text (say, text from a newspaper) by continuing to build it manually. Increasing attention is being paid now to automatically acquiring grammars from a large corpus (*32*). (See below for some further details.)

Statistical Approaches to Natural Language Processing

There is a long history of modeling language statistically. After all, some words occur more frequently than other words (for example, *the* occurs more frequently than *man*, which occurs more frequently than *aardvark*), some two-word sequences appear more frequently than some other two-word sequences (for example, *a man* occurs more frequently than *old man*, which occurs more frequently than *green man*), and so forth. Hence, it is reasonable to believe that language can be modeled statistically. A specific proposal along these lines was made by Shannon in 1948 (*33*). He viewed the generation process as modeled by stochastic processes, in particular, a Markov process. For the present purpose, I will characterize sentence generation by a finite state machine (Fig. 4). Given a state diagram, I generate a sentence by starting with the initial state and then traversing the diagram from state to state and emitting the word labeling the arc between a pair of states. The process ends when I reach the final state. A probability is assigned to each state transition together with the emitted symbol, that is, to a triple (S_i, a_j, S_k) representing the transition from state S_i to state S_k emitting the symbol a_j. Although such machines are clearly relevant to modeling language statistically, Chomsky (*34*) rejected the finite-state machine characterization as inappropriate for modeling grammars, for the following reason: In Fig. 4, *lives* is four words away from *man*, assuming that I did not follow the loop at S_4. Hence the dependency between these two words can be captured by the state sequence from S_2 to S_6. However, in the sentence *The man who the woman Harry met yesterday telephoned lives in Philadelphia*, (one that is a bit difficult to process but grammatical, and not generable by the machine in Fig. 4), *lives* is now seven words away from *man*. Because more clauses can be embedded and each clause can be lengthened by adding adjectives or adverbs, the distance between *lives* and *man* can be made arbitrarily large and thus the number of states required to model language cannot be bounded. Hence a finite-state machine is inadequate. Chomsky also rejected the possibility of associating the probability of a sentence with its grammaticality (the higher the probability, the higher the grammaticality of the sentence). This is because if I order the sequences of a given length (there will be W^n such sequences, if W is the number of words and n is the length of the sequences) according to the probabilities of the sequences then it will not be possible to sort out grammatical and ungrammatical sequences on the basis of this ranking (*34*). Chomsky then developed structural models, such as the phrase structure grammar and transformational grammar, which formed the basis for almost all of the work in mathematical and computational linguistics up until the present.

Although Chomsky rejected the statistical models, he commented (*34*, p. 17): "Given the grammar of language, one can study the use of the language statistically in various ways; and the development of probabilistic models for the use of language (as distinct from the syntactic structure of language) can be rewarding. . . . One might seek to develop a more elaborate relation between statistical and syntactic structure than the simple order of approximation model I have rejected. I would certainly not care to argue that any such relation is unthinkable, but I know of no suggestion to this effect that does not have obvious flaws. . . ."

Harris, around 1957, proposed a transformational theory (*35*) motivated by the considerations of normalizing sentence structures (*36*) so that the relevant co-occurrences among words can be stated in a local manner. Very roughly speaking, under this view, *The man who Harry met yesterday lives in Philadelphia*, is made up of S1: *The man lives in Philadelphia* and S2: *who Harry met* (which is a transformed version of S3: *Harry met the man*, with S1 and S3 sharing *the man*) and so on. There are clearly "meaningful" statistical dependencies between *lives* and the subject noun *man* and the object of *in*, namely, *Philadelphia*, and between *met* and *Harry*, the subject of *met*, and *man* the object of *met*, but not "meaningful" statistical dependencies between *lives* and *yesterday* or *met yesterday* (the one-word and two-word sequences before *lives*) and so on.

Although statistical approaches did not play a significant role in mathematical or computational linguistics, it is clear that the idea of somehow combining structural and statistical information was already suggested as early as the late 1950s. Now in the 1990s, there is a resurgence of these early ideas. There are two key reasons for this renewed interest. First there are now some formal frameworks which appear to be suitable for combining structural and statistical information in a principled manner and second, there is now the possibility of using very large corpora, annotated in various ways that can be used for reliably estimating the various statistics needed to deduce linguistic structure (*32*).

I will now give a few examples to show how structural and statistical information can be integrated. Context-free grammars (CFG) have been used extensively in modeling grammars. Each rule (production) in a CFG can be associated with a probability of its use. Thus, given a CFG with rules: (R1) S → NP VP (0.9) (R2) S → NP NP V (0.1), (R3) VP → V NP (0.7), (R4) VP → V (0.3), I have associated probabilities with each of the rules. The probabilities of all rules associated with a given nonterminal add up to 1. The probability of a sentence (more precisely the derivation of the sentence in the grammar) is simply the product of the probabilities of each rule in the derivation because the grammar is CFG and the application of a rule depends only on the nonterminal on the left-hand side of a rule and not on the context in which this nonterminal appears in a derivation. Probabilistic parsing methods and methods for estimating the probabilities of the rules from a training corpus are given (*37*). By making the probability associated with each rule somewhat context-dependent, for example, making it dependent on the preceding rule in the derivation, considerable improvement in the estimation of the probabilities and performance of the parser (in terms of getting correct parsers) can be achieved (*38*).

As I have mentioned, the really "meaningful" statistical dependencies are between words (lexical items) mediated most likely by grammatical relations. For example, there will be "meaningful" statistical dependencies between the verb *eats*, and the lexical items that can appear as subject and object of *eats*. CFGs and their generalizations are not directly based on lexical items, that is, they are not lexicalized, and in general, cannot be lexicalized (*15*). Lexicalized grammars, as described earlier, are more appropriate for integrating structural and statistical information in a uniform manner.

Two dependent words in a sentence can be an arbitrary distance apart, as discussed earlier. Hence, this dependency cannot be captured by one-word, two-word, three-word and *n*-word frequencies, for some fixed *n* (that is, uni-gram, bi-gram, tri-gram and *n*-gram statistics). However, in many situations these statistics work surprisingly well in determining some aspects of language structure. Tri-gram frequencies (of parts of speech—that is, syntactic categories—and not words directly) have been used very successfully for discovering an optimum assignment of parts of speech to words (*39, 40*). Almost all words are lexically ambiguous, that is, they belong to more than one category. For example, *table* is either a noun (N) or a verb (V); *pale* is either an adjective (ADJ) or an adverb (ADV); *see* can be a verb (V), an interjection (UM), or a noun (with capital S); *round* can be an adjective (ADJ), noun (N), verb (V), or an adverb (ADV), and so forth. The program in (*39*) uses a linear-time dynamic-programming algorithm to find an assignment of parts of speech optimizing the product of: (i) probability of observing a part of speech i, given the word j, and (ii) probability of observing part of speech i, given two previous parts of speech. Probability estimates are obtained by training on a tagged corpus [such as the well-known Tagged Brown Corpus (*41*)]. Error rates of only 3% to 4% have been reported (*39*), which compare very well with the error rate of human annotators. Similar techniques have been used to locate simple noun phrases with high accuracy (*39*).

Statistical techniques in conjunction with large corpora (raw texts or annotated in various ways) have also been used to automatically acquire other linguistic information such as morphological information (that is, parts of words such as prefixes and suffixes and inflected forms), subcategorization information (see the earlier section on grammars and parsers for subcategorization information), semantic classes (such as classification of nouns, based on what predicates they go with; compound nouns such as *jet engines, stock market prices*; classification of verbs, for example, *to know* describes a state of the world, while *to look* describes events, and so on), and, of course, grammatical structure itself as I have already mentioned (*38, 42–46*). Such results have opened up a new direction of research in NLP, which is often described as corpus-based NLP.

It should be clear from the previous discussion that, for the development of corpus-based NLP, very large quantities of data are required (the Brown Corpus from the 1960s is about 1 million words). Researchers estimate that about 100 million words will be required for some tasks. The technologies that will benefit from corpus-based NLP include speech recognition and synthesis, machine translation, full-text information retrieval, and message understanding, among others. The need for establishing very large text and speech databases, annotated in various ways is now well understood. It is recognized that no single organization can afford to create enough linguistic data even for its own research and development, let alone for the needs of the research community at large. This need, together with the size of the database and the need for sharing it, has been the key motivation for the plans for setting up a Linguistic Data Consortium (LDC) by DARPA (*47*). Initial plans of the LDC call for the collection of raw text (naturally occurring text from a wide range of sources, 5 to 10 billion words) annotated text (syntactic and semantic labeling of some parts of raw text, upwards of 20 million words), raw speech (spontaneous speech from a variety of interactive tasks, 400 hours, 2000 speakers), read speech (1,000 hours, 10,000 speakers), annotated speech (phonetic and prosodic labeling, 20 hours), a lexicon (a computational dictionary of 200,000 entries plus a term bank containing, for example, geographical, individual, and organizational names, 200 to 300 thousand entries), and a broad coverage computational grammar. The LDC will also develop a variety of shareable tools. Some examples in the area of speech are: programs for segmentation of speech, alignment of speech and text, prediction of pronunciation options from orthographic transcription. Some examples from text area are: a program for breaking text into sentences, a statistical parts-of-speech tagger, an efficient program for computing *n*-gram statistics and a variety of other statistics over very large corpora (*47*).

Multilingual Natural Language Processing

Almost all linguistic theories and mathematical and computational models are formulated so that they are in principle applicable to

natural languages at large, and not just one or more specific languages. In fact, these theories and models attempt to capture aspects of language structure that are judged to be universal. Of course, only empirical and experimental work (across diverse languages) can assess the validity of these theories or models. In this sense, all mathematical or computational work in natural language processing is multilingual. However, this is not what is meant by multilingual processing. This term instead refers to the computational models and the systems based on those models that deal with more than one language, of which a special and important case is machine-translation (MT). Multilingual processing, however, is not limited to MT, in fact, MT itself is not viewed currently as consisting only of systems that completely map a source language into the target language, it may include a translation component together with other multilingual tools that make the entire system useful.

The goal here is not to review the history of MT (quite rocky certainly in the United States). An excellent discussion of the status of MT can be found in (*48–51*) including a discussion of some of the well-known systems such as SYSTRAN, LOGOS, Mu, EUROTRA, among others. The current trend of focusing on multilingual processing and treating MT as a part of this activity is expected to create a more stable environment for both basic research and the development of useful systems in NLP.

A discussion of grammars and parsers is clearly relevant to MT, as almost all MT systems have them as components. However, in the context of an entire MT system, the choice of a particular grammar and parser (and, of course, a generator, a topic I have not discussed in this paper) at this stage of development is somewhat arbitrary (*52*). The grammar has to interface to many different components of an MT system, and this interface is not always smooth, and often overwhelms the considerations in the choice of the grammar. However, work on the MT problem is encouraging researchers to investigate properties of grammars from the point of view of their suitability for MT (*53–56*).

MT systems are usually classified as either direct, transfer-based, or interlingua-based. In the direct approach, there are no intermediate representations between the source language and the target language. The source language text is processed "directly" in order to transform it into the target text, essentially a word-to-word translation with some adjustments. This approach is not followed by any MT system at present on account of its obvious weakness attributable to eschewing all aspects of the internal structure of sentences. There is one exception however, the statistical system based on parallel texts briefly described at the end of this section can be viewed, in a sense, as a "direct" system.

In the transfer-based approach, information from the various stages of analysis from the source text is transferred to the corresponding stages of the generation of the target text, for example, transfer is achieved by setting up correspondences at the lexical level, at the grammar level, or at the level of the structures built by the grammar, and so forth. The transfer module obviously depends on a particular pair of languages. The source and target language representations on which the transfer is defined may also depend on the language pair but this need not be the case. In fact, some recent work on transfer-based approach attempts to show how one can work with language-independent representations (see below).

The interlingua-based approach depends on the claim that a suitable intermediate representation can be defined such that the source text can be mapped into the intermediate representation which can then be mapped into the target text. In principle, this approach is clearly attractive because, unlike the transfer-based approach, it is not necessary to build a separate transfer module for each pair of languages. However, it is not clear whether a truly language-independent intermediate representation can be devised. Current interlingua-based systems are much less ambitious about their claims to the universality of the intermediate representation. For a high-quality translation, it is often necessary to have access to some particular aspects of the source and target languages. It is not clear how the interlingua-based approach will handle these aspects in general without implicitly encoding these aspects of the source language and making this information available during generation from the intermediate representation to the target language.

In the transfer-based approach, there have been some recent advances. In the development of mathematical and computational models of grammars there is increasing emphasis on locating syntactic as well as semantic information directly with the lexical items by associating structures with the lexical items and defining operations for composing these objects (see the section on grammars and parsers). From this perspective, all the information particular to a language is encapsulated in the lexical items and the structures associated with them. Different languages will be distinguished at this level, but not with respect to the operations for composing these structures, which are the same for all languages, on this approach. The idea then, is to define all bilingual correspondences at this level. Some recent attempts along these lines are described in (*54–56*) and it remains to be seen if this approach can be carried out across a variety of languages.

Conventional MT systems try to produce one translation for the source sentence, but, of course, in order to achieve this, as is well known, all kinds of other information is usually needed, such as discourse context (the previous text), the context of the situation (particularly, if it is a dialog), and a variety of domain-knowledge, exactly the kind of information that a natural language understanding and generation system needs. There has been some preliminary work on the so-called knowledge-based systems (*57*). So, in general, one might be tempted to say (and quite correctly in principle) that high-quality MT cannot be achieved without first solving the problem of natural language understanding and generation. The solution of this general problem is clearly not around the corner! However, in practice, there are a number of situations where this knowledge is available (albeit in highly constrained domains), can be acquired easily, or is indirectly there because the user(s) of the system is (are) in the loop if the MT system is interactive (*58, 59*). Some examples of such situations are:

Highly constrained texts. Generation of multilingual texts from a source text which is prepared in a highly constrained, unambiguous, highly stylized language. Such a system is useful for preparing manuals in different languages. Here the system is really not translating a manual written in one natural language into a set of other natural languages, but rather is generating multilingual texts from a highly constrained text, thus avoiding many problems in conventional MT.

The so-called sub-language–based MT systems also fall into this category. For example, in weather reports the sentences are usually short, highly stylized, and mostly employ a standard phraseology. A very well-known system (TAUM METEO) for translating weather reports from English to French has been in operation in Canada for some years (*49*).

Interactive situations. High-quality translation of business correspondence using pre-translated fragments with slots to be filled in by interaction. The user collaborates with the system. The advantage of such a system is that "[it] only translates what it 'knows' it can translate accurately, with the result that the systems shows what MT can do, rather than what it cannot as in traditional MT" (*60*, p. 8). Such a system will automatically compose foreign language texts.

A system to serve as the bilingual intermediary for the user in a dialogue with a conference office, desiring to obtain information

about a forthcoming conference. This system is a "dialogue MT system in the sense that is enters into a dialogue with the user about the translation and in that the object of the translation is the user's contribution to the dialogue" (*60*, p. 8).

So far, I have been assuming that an MT system will use a grammar and a parser of some sort. There are recent attempts to use purely statistical techniques along with parallel texts. There are parallel texts available in a pair of languages which are translations from one language to another, carried out by human translators. A well-known example is the Canadian *Hansard*, which contain the transcripts of the proceedings of the Canadian Parliament both in English and French. Such texts with several million words are available now. With the use of 3 million aligned sentences from the *Hansard* bilingual corpora and using only statistical techniques (the aligning itself is done statistically also), an MT system has been developed (*61*). There is considerable potential for such systems if they are suitably combined with some structural information, perhaps also obtained statistically. Aligned sentences from bilingual corpora have been used recently for constructing bilingual concordances and some multilingual tools, for example, providing translations of content words of a message to help the user to translate the message, assuming the user has some knowledge of the source language (*62*).

In summary, MT and multilingual natural language processing, in general, is clearly a major application of NLP. It is providing a rich domain for testing current or new formalisms in all aspects of NLP. Large-scale useful MT systems however, are, and will continue to be, based on a variety of techniques and tools in NLP.

Future Prospects for NLP

The trend of integrating speech and language will continue and we can expect much more robust spoken language systems, which also incorporate some aspects of discourse structure. More advanced and useful commercial systems for multilingual interfaces, machine translation, and message understanding systems will appear in the near future. Speech-to-speech translation systems will also appear in limited domains. Language systems will become even more integrated with other modalities such as graphics (including pointing) and will be useful for many applications such as providing instructions for assembly or maintenance of complex equipment. NLP systems will also play a major role in educational systems and prosthetic systems.

On the theoretical side, the mathematical and computational work has given us deep insights into the working of language. However, language is an enormously complex system. Therefore, in a sense, the computational understanding of the structure and function of language is very primitive still. Further mathematical and computational work will provide us more unifying accounts of syntax, semantics, and pragmatic aspects of language. The use of language corpora and statistical techniques in NLP will continue to grow and we will see an integration of structural and statistical techniques in NLP leading to more robust systems.

Finally, the mathematical, computational, and statistical work in NLP will contribute to psycholinguistic research which studies the human processing of language, a very important aspect about which I have not said anything in this paper.

REFERENCES AND NOTES

1. For a good introduction to work in discourse interpretation, language and action and intention, see *Readings in Natural Language Processing*, B. J. Grosz, K. Sparck Jones, B. L. Webber, Eds. (Morgan Kauffman, Los Gatos, CA, 1986), chaps. III and IV. For some recent work in these areas, see *Intentions in Communication*, P. Cohen, J. Morgan, M. E. Pollack, Eds. (MIT Press, Cambridge, MA, 1990).
2. N. Chomsky, *Information and Control* **5**, 137 (1959).
3. S. Shieber, *An Introduction to Unification-Based Grammars, Lecture Notes, no. 4* (Center for Studies in Language and Information, Stanford University, Stanford, CA, 1986). For some earlier work on Unification Grammars, see M. Kay, *Unification Grammars. Technical Report* (Xerox Palo Alto Research Center, Palo Alto, 1983).
4. G. Gazdar, E. Klein, G. K. Pullum, I. A. Sag, *Generalized Phrase Structure Grammars* (Harvard Univ. Press, Cambridge, MA, 1985).
5. C. Pollard and I. A. Sag, *Information-Based Syntax and Semantics* (Center for the Study of Language and Information, Stanford, CA, 1986), vol. 1.
6. R. M. Kaplan and J. Bresnan, in *The Mental Representation of Grammatical Relations*, J. Bresnan, Ed. (MIT Press, Cambridge, MA, 1983), pp. 173–281.
7. S. Shieber, in *Proceedings of the Association for Computational Linguistics Conference* (Association for Computational Linguistics, Morristown, NJ, July 1985), pp. 145–152.
8. F. C. N. Pereira and S. M. Shieber, *Prolog and Natural Language Analysis* (Center for the Study of Language and Information, Stanford, CA, 1987).
9. E. P. Stabler, *Relaxation Techniques for Relaxation Principles for Principle-Based Parsing, Technical Report 90-1* (UCLA Center for Cognitive Science, Los Angeles, 1991).
10. J. W. Bresnan, R. M. Kaplan, P. S. Peters, A. Zaenen, *Linguistic Inquiry* **13**, 613 (1982).
11. J. Higginbotham, *ibid.* **15**, 225 (1984).
12. S. Shieber, *Linguistics and Philosophy* **8**, 333 (1985); see also C. Culy, *ibid.*, p. 333.
13. A. K. Joshi, in *Natural Language Processing: Theoretical, Computational and Psychological Perspectives*, D. Dowty, L. Karttunen, A. Zwicky, Eds. (Cambridge Univ. Press, New York, 1985), pp. 206–250.
14. A. K. Joshi, in *Mathematics of Language*, A. Manaster-Ramer, Ed. (John Benjamin, Amsterdam, 1987), pp. 87–114.
15. A. K. Joshi and Y. Schabes, in *Definability and Recognizability of Sets of Trees*, M. Nivat and M. Podelski, Eds. (Elsevier, Amsterdam, in press).
16. M. Steedman, in *Categorial Grammars and Natural Language Structures*, R. T. Oehrle, E. Bach, D. Wheeler, Eds. (Foris, Dordrecht, 1986), pp. 417–442. See this volume for several other related papers on categorial grammars.
17. M. Steedman, *Natural Language and Linguistic Theory* **5**, 403 (1987). See also M. Steedman, *Language* **67**, 260 (1991).
18. A. S. Kroch, in *Mathematics of Language*, A. Manaster-Ramer, Ed. (John Benjamin, Amsterdam, 1987), pp. 143–172.
19. _____, in *New Conceptions of Phrase Structure*, M. Baltin and A. S. Kroch, Eds. (Univ. of Chicago Press, Chicago, IL, 1989), pp. 66–98.
20. _____ and B. Santorini, in *Principles and Parameters in Comparative Grammar*, R. Freidin, Ed. (MIT Press, Cambridge, MA, 1991), pp. 269–338.
21. A. K. Joshi, K. Vijay-Shanker, D. Weir, in *Processing of Linguistic Structure*, S. Shieber and T. Wasow, Eds. (MIT Press, Cambridge, MA, in press).
22. G. Gazdar, *Applicability of Indexed Grammars to Natural Languages. Technical Report CSLI 85-34* (Center for the Study of Language and Information, Stanford University, Stanford, CA, 1985).
23. There are, of course, other formalisms that do not quite fit into this class. Head Driven Phrase Structure Grammar (HPSG) shares many aspects of CCG and TAG, in particular, it uses a stack mechanism to encode the arguments, the post-verbal complements from left to right and then preverbal subject [C. Pollard, *Lecture Notes on Head-Driven Phrase Structure Grammars* (Center for the Study of Language and Information, Stanford University, Stanford, 1985)]. The Lexical Functional Grammar (LFG) uses two kinds of structures, constituent structures and functional structures (*6*). It is outside the class MCSG.
24. A. K. Joshi and Y. Schabes in *Proceedings of DARPA Workshop on Spoken Language Systems* (Morgan Kauffman, Palo Alto, Asilomar, February 1991), pp. 195–199.
25. I. Sag, personal communication.
26. K. W. Church and K. Koskenniemi, in *Proceedings of the International Conference on Computational Linguistics* (John von Neumann Society for Computing Sciences, Budapest, August 1988), pp. 335–340.
27. E. R. Barton, R. Berwick, E. Ristad, *Computational Complexity and Natural Language* (MIT Press, Cambridge, MA, 1987).
28. S. Fong, in *Principle-based Parsing: Computation and Psycholinguisitcs*, R. Berwick, S. Abney, C. Tenny, Eds. (Kluwer, Boston, MA, 1991).
29. See *Proceedings of the DARPA Workshops on Spoken Language Systems* (Morgan Kauffman, Palo Alto, 1989, 1990, 1991).
30. M. Marcus, *A Theory of Syntactic Recognition for Natural Language* (MIT Press, Cambridge, MA, 1980).
31. D. Hindle, *User Manual for Fidditch. Technical Memorandum #7590-142* (Naval Research Laboratory, Washington, DC, 1983).
32. E. Brill, D. Magerman, M. Marcus, B. Santorini, in *Proceedings of the DARPA Workshop on Spoken Language Systems* (Morgan Kauffman, Palo Alto, 1990), pp. 275–282.
33. C. E. Shannon, *Bell Systems Technical Journal* **27**, 379 (1948).
34. N. Chomsky, *Syntactic Structures* (Mouton, The Hague, 1957).
35. Z. S. Harris, *Language* **3**, 283 (1957).
36. Harris's motivation for normalization was related to his program of extending the distributional methods beyond sentences to texts.
37. F. Jelinek, J. D. Lafferty, R. L. Mercer, *Basic Methods of Probabilistic Grammars, Technical Report* (IBM, Yorktown Heights, NY, 1990). This report is a very good tutorial reference. The original reference for estimating the probabilities of the rules is J. K. Baker, in *Proceedings of the Spring Conference of the Acoustical Society of America* (MIT Press, Cambridge, MA, June 1979), pp. 547–550.
38. D. Magerman and M. Marcus, in *Proceedings of the Fifth Conference of the European Association for Computational Linguistics (EACL)* (Association for Computational

Linguistics, Morristown, NJ, April 1991), pp. 15–20.
39. K. W. Church, in *Proceedings of the 2nd Conference on Applied Natural Language Processing* (Association for Computational Linguistics, Morristown, NJ, February 1988), pp. 136–143. For an application of this algorithm, see R. Weischedel, M. Meteer, R. Schwartz, J. Palmucci, *Applying Probabilistic Models to Knowledge-Based Algorithms in Natural Language Processing, Technical Report* (Bolt, Beranek, and Newman, Cambridge, MA, 1991).
40. S. DeRose, *Computational Linguistics* **14**, 31 (1988).
41. W. Francis and H. Kucera, *Frequency Analysis of English Usage: Lexicon and Grammar* (Houghton Mifflin, Boston, MA, 1982).
42. E. Brill, personal communication.
43. M. R. Brent and R. Berwick, in *Proceedings of the DARPA Workshop on Spoken Language Systems* (Morgan Kauffman, Palo Alto, CA, February 1991), pp. 342–345.
44. M. R. Brent, in *Proceedings of the Fifth Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Berlin (Association for Computational Linguistics, Morristown, NJ, April 1991), pp. 222–226.
45. D. Hindle, in *Proceedings of the Association for Computational Linguistics Conference*, Pittsburgh (Association for Computation Linguistics, Morristown, NJ, June 1990), pp. 268–275.
46. F. Smadja and K. McKeown, in *Proceedings of the Association for Computational Linguistics Conference*, Pittsburgh (Association for Computation Linguistics, Morristown, NJ, June 1990), pp. 252–259.
47. M. Liberman, *Guidelines for the Linguistic Data Consortium*, Report prepared for the Defense Advanced Research Projects Agency (DARPA), May 1991. See also the announcement by DARPA in the Commerce Business Daily Announcement (Department of Commerce, Washington, DC, 23 July 1991).
48. H. J. Hutchins, *Machine Translation: Past, Present and Future* (Ellis Horwood, Chichester, 1986).
49. M. Nagao, *Machine Translation: How Far Can It Go?* (Oxford Univ. Press, Oxford, 1989). [Translation by N. D. Cook of *kikei hon'yaku wa doko made kano ka* (Iwanami Shoten, Tokyo, 1986).]
50. S. Nirenberg, Ed., *Machine Translation: Theoretical and Methodological Issues* (Cambridge Univ. Press, Cambridge, 1987).
51. J. Slocum, Ed., *A Survey of Machine Translation: Its History, Current Status and Future* (Cambridge Univ. Press, Cambridge, 1985).
52. Y. Wilks, in *Proceedings of the International Forum for Translation Technology*, IFTT (IFTT, Oiso, Japan, April 1989), pp. 56–62.
53. R. M. Kaplan, K. Nutter, J. Wedekind, A. Zaenen, in *Proceedings of the European Association for Computational Linguistics (EACL) Conference* (Association for Computational Linguistics, Morristown, NJ, April 1989), pp. 272–281.
54. S. M. Shieber and Y. Schabes, in *Proceedings of the International Conference on Computational Linguistics (COLING-90)* (University of Helsinki, Helsinki, August 1990), vol. 3, pp. 253–258.
55. A. Abeille, Y. Schabes, A. K. Joshi, in *Proceedings of the International Conference on Computational Linguistics (COLING-90)* (University of Helsinki, Helsinki, April 1990), vol. 3, pp. 1–6.
56. J. Tsujii and K. Fujita, in *Proceedings of the European Association for Computational Linguistics (EACL) Conference* (Association for Computational Linguistics, Morristown, NJ, April 1991), pp. 275–280.
57. J. G. Carbonell, in *Machine Translation*, S. Nirenberg, Ed. (Cambridge Univ. Press, Cambridge, 1987), pp. 68–89.
58. H. L. Sommers, J. Tsuji, D. Jones, in *Proceedings of the International Conference on Computational Linguistics (COLING 90)* (University of Helsinki, Helsinki, August 1990), pp. 271–276.
59. J. Tsujii and M. Nagao, in *Proceedings of the International Conference on Computational Linguistics (COLING-88)* (John von Neumann, Society for Computing Sciences, Budapest, August 1988), pp. 688–693.
60. H. L. Sommers, in *Proceedings of a Workshop on Machine Translation* (University of Texas, Austin, TX, June 1990).
61. P. F. Brown *et al.*, in *Proceedings of the Second International Conference on Theoretical and Methodological Issues in Machine Translation* (Carnegie Mellon University, Pittsburgh, PA, June 1988).
62. W. A. Gale and K. W. Church, in *Proceedings of the Association for Computational Linguistics (ACL) Conference* (Association for Computational Linguistics, Morristown, NJ, June 1991), pp. 177–184.
63. This work was partially supported by ARO grant DAAL03- 89-0031, DARPA grant N00014-90-J-1863, and NSF STC grant DIR-8920230. I want to thank E. Brill, B. Cheikes, R. Frank, D. Griesbach, S. Heyner, A. Kroch, M. Liberman, M. Marcus, R. Pito, P. Resnik, O. Rambow, Y. Schabes, M. Steedman, M. Walker, B. Webber, and R. Weischedel for their valuable help in the preparation of this article.

Computer Vision

YIANNIS ALOIMONOS AND AZRIEL ROSENFELD

The field of computer vision is devoted to discovering algorithms, data representations, and computer architectures that embody the principles underlying visual capabilities. This article describes how the field of computer (and robot) vision has evolved, particularly over the past 20 years, and introduces its central methodological paradigms.

VISION IS THE MOST POWERFUL SENSE FOR MANY LIVING organisms, including humans. We take it so much for granted, because it is ordinarily so effortless, that we often fail to seriously consider how it works. Students of visual perception work in diverse fields, including neuroanatomy and physiology, psychology, computational and robot vision, and engineering. But researchers in different fields ask different questions about vision. Some ask empirical questions: How are existing biological visual systems actually designed? On the other hand, scientists and engineers try to answer theoretical and normative questions. The theoretical question in vision is, What is the range of possible mechanisms underlying perceptual capabilities in vision systems? The normative question is, How should a particular class of vision systems (or robots) be designed so that it can efficiently perform a set of specific visual tasks? The three types of basic questions do not in general have the same answers.

A very large part of the human brain is devoted to visual perception (*1*) (Fig. 1). Computational algorithms are implemented in this massive network of neurons; they obtain their inputs from the retina, and produce as output an "understanding" of the scene in view. But what does it mean to "understand" the scene? What algorithms and data representations are used by the brain? Analogously, given a set of images acquired by a TV camera, what computer architectures, data structures and algorithms should we use to create a machine that can "see" as we do? (*2–4*) (Fig. 2).

Many organisms possess visual capabilities, and their visual systems are not structured in the same way; moreover, they live in different environments and use vision for different purposes. But although a given visual capability, say for obstacle avoidance, is not necessarily implemented in the same way in the fly, the rat, and the human, the principles underlying this ability may be the same. It is these principles that are the subject of research in computer vision. As our understanding of visual principles advances, we can build robots that perform various tasks through the use of vision.

The authors are at the Center for Automation Research, University of Maryland, College Park, MD 20742–3411.