1 Critical challenges for natural language processing

MADELEINE BATES, ROBERT J. BOBROW,
AND RALPH M. WEISCHEDEL

1.1 Introduction

Although natural language processing (NLP) has come very far in the last twenty years, the technology has not yet achieved a revolutionary impact on society. Is this because of some fundamental limitation that can never be overcome? Is it because there has not been enough time to refine and apply theoretical work that has already been done?

We believe it is neither. We believe that several critical issues have never been adequately addressed in either theoretical or applied work, and that, because of a number of recent advances that we will discuss, the time is due for great leaps forward in the generality and utility of NLP systems. This paper focuses on roadblocks that seem surmountable within the next ten years.

Rather than presenting new results, this paper identifies the problems that we believe must block widespread use of computational linguistics, and that can be solved within five to ten years. These are the problems that most need additional research and most deserve the talents and attention of Ph.D. students. We focus on the following areas, which will have maximum impact when combined in software systems:

1. *Knowledge acquisition* from natural language (NL) texts of various kinds, from interactions with human beings, and from other sources. Language processing requires lexical, grammatical, semantic, and pragmatic knowledge. Current knowledge acquisition techniques are too slow and too difficult to use on a wide scale or on large problems. Knowledge bases should be many times the size of current ones.
2. *Interaction with multiple underlying systems* to give NL systems the utility and flexibility demanded by people using them. Single application systems are limited in both usefulness and the language that is necessary to communicate with them.

Some of the work reported here was supported by the Advanced Research Projects Agency under Contracts No. N00014-85-C-0016 and N00014-89-C-008, monitored by the Office of Naval Research. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency of the United States Government.

3. *Partial understanding* gleaned from multi-sentence language, or from fragments of language. Approaches to language understanding that require perfect input or that try to produce perfect output seem doomed to failure because novel language, incomplete language, and errorful language are the norm, not the exception.

In the following sections, we first present a brief overview of the state of the art in the traditional areas of syntax, semantics, and pragmatics. Then each of the three critical challenge topics is discussed in a separate section, beginning with an explanation of the problem and its importance, and continuing through with an explanation of the techniques that we believe are now ready to be applied to those problems. Each section concludes with a summary of the challenges and the suggested new approaches.

1.2 State of the art

The most visible results in NLP in the last five years are several commercially available systems for database question-answering. These systems, the result of transferring technology developed in the 1970s and early 1980s, have been successfully used to improve productivity by replacing fourth-generation database query languages. The following case study (Bates, 1989) illustrates their capabilities: with 8-person-weeks, the Parlance™ system[1] was ported to a Navy relational database of 666 fields (from 75 relations) with a vocabulary of over 6,000 (root) words. Queries from a new user of one of these systems succeed 60–80% of the time; with use of the system, users naturally and automatically adjust to what data is in the database and to the limits of the language understood by the system, giving a success rate of 80–95% depending on the individual.

The success of these systems has depended on the fact that sufficient coverage of the language is possible with relatively simple semantic and discourse models. The semantics are bounded by the semantics of the relations used in databases and by the fact that words have a restricted number of meanings in one domain. The discourse model for a query is usually limited to the previous answer (usually numeric, simple strings, or a table) and the noun phrases mentioned in the last few queries.

The limitations of today's practical language processing technology have been summarized (Weischedel et al., 1990) as follows:

1. Domains must be narrow enough so that the constraints on the relevant semantic concepts and relations can be expressed using current knowledge representation techniques, i.e., primarily in terms of types and sorts. Processing may be viewed abstractly as the application of recur-

[1]Parlance is a trademark of BBN Systems and Technologies.

sive tree rewriting rules, including filtering out trees not matching a certain pattern.

2. Handcrafting is necessary, particularly in the grammatical components of systems (the component technology that exhibits least dependence on the application domain). Lexicons and axiomatizations of critical facts must be developed for each domain, and these remain time-consuming tasks.
3. The user must still adapt to the machine, but, as the products testify, the user can do so effectively.
4. Current systems have limited discourse capabilities that are almost exclusively handcrafted. Thus current systems are limited to viewing interaction, translation, and writing and reading text as processing a sequence of either isolated sentences or loosely related paragraphs. Consequently, the user must adapt to such limited discourse.

It is traditional to divide natural language phenomena (and components of systems designed to deal with them) into three classes:

1. Syntactic phenomena – those that pertain to the *structure* of a sentence and the order of words in the sentence, based on the grammatical classes of words rather than their meaning.
2. Semantic phenomena – those that pertain to the *meaning* of a sentence relatively independent of the context in which that language occurs.
3. Pragmatic phenomena – those that relate the meaning of a sentence to the *context* in which it occurs. This context can be linguistic (such as the previous text or dialogue), or nonlinguistic (such as knowledge about the person who produced the language, about the goals of the communication, about the objects in the current visual field, etc.).

1.2.1 Syntax

Syntax is without doubt the most mature field of study in both computational linguistics and the closely related field of linguistics. The most thorough computational accounts of natural language phenomena exist for syntax; and grammars with very large coverage of English have existed since the early 1980s. Formalisms for describing syntactic phenomena, mathematical analyses of the expressive power of those formalisms, and computational properties of processors for those formalisms have existed for more than twenty-five years, since the definition of the Chomsky hierarchy (finite state languages, context-free languages, context-sensitive languages, and the recursively enumerable languages).

During the 1970s and most of the 1980s, the dominant NLP formalism for writing grammars of natural language was the augmented transmission network (ATN) (Woods, 1970), a procedural language that allowed compact statements of

not only the context-free aspects of language but also the apparently context-sensitive aspects as well. In the late 1980s, a shift began from the ATN and its procedural aspects toward a declarative formalism. What the new dominant formalism will be is not yet clear, but a likely candidate is a class of grammar formalisms (Shieber, 1986) that combine context-free rules with unification as a way of compactly capturing both context-free and context-sensitive aspects of language. The declarative formalisms offer the promise of exploring alternative parsing algorithms and pose minimum constraints on parallel algorithms for parsing. This shift in the research community from procedural specifications of syntax, such as ATN grammars, to declarative specifications, such as unification grammars, parallels the similar shift in interest in programming language research away from procedural languages and toward newer functional programming languages and declarative representations.

Because syntax is by far the most mature area in natural language processing, it is difficult to foresee that further developments in syntax will have as great an impact on utility as would emphasis on research and development on other, less developed areas of technology.

1.2.2 Semantics

In semantics, much recent progress has been made by focusing on limited application domains. For database access, the semantics of the system can be confined to that of the individual entities, classes of entities, relationships among the entities, attributes of the entities, and the typical operations that are performed in database retrieval. This simplifies the problem of semantics in at least the following ways: first, the meaning of individual words and of the phrases they compose can be restricted to the domain-specific meanings actually modeled in the database. Instead of needing to come up with a very general semantics for each word, a very literal semantics providing the mapping from the words to the entities modeled in the database is all that is required, for the database could not provide additional information even if a more general semantics were available. Second, problems of semantic ambiguity regarding alternative senses for a word are reduced, for only those word senses corresponding to entities in the database will contribute to the search space of possible alternatives.

For the task of database updating from messages, a key simplifying condition is that the information sought can be characterized ahead of time. Suppose the goal is to update automatically a database regarding takeover bids. Suppose further that the information desired is the date of the bid, the bidder, the target, the percentage of stock sought, the value of the offer, and whether it is a friendly or hostile bid. A first approximation is that the remaining information in the article can be ignored. The assumption is that although other concepts may be mentioned in the message or news wire item, they normally do not impact the

values of the fields to be updated in the database. If that assumption applies in the proposed message processing application, then one can model the literal semantics of those words and phrases that have a correlate in the data being sought. For cases where the proposed update would be in error because the unanalyzed text does impact the update, human review of the proposed update can minimize erroneous entries into the database. Semi-automatic update with a human in the loop may be more cost effective and timely than fully manual update and may be more desirable than not having the data at all.

No uniform semantics representation language has emerged, although three general classes of semantic representations are employed: those that allow one to state anything that arises in a propositional logic, those that allow expressions equivalent to a first-order logic, and those that allow expressions not representable in a first-order logic. Most framed-base representations are equivalent to a propositional logic, since they do not allow expression of quantifiers. Most systems for database retrieval use a first-order logic. Many research systems employ extensions beyond first-order logic, such as the modal intensional logic defined by Montague (1970). Encoding the semantics of all the words and phrases for a particular application domain is one of the most significant costs in bringing up a natural language system in a new application domain. Knowledge acquisition procedures that would reduce this cost would therefore have great impact on the applicability of the technology.

1.2.3 Pragmatics

The modeling of context and using context in understanding language is the most difficult, and therefore the least well-understood, area of natural language processing. Unlike programming languages where one can define contextual influence in a limited and controlled way, context is all-pervasive and very powerful in natural language communication.

Context is fundamental to communicating substantial information with few words. For instance, if one says, *How about Evans?,* those three words may suggest a lot. If the context had been that the immediately previous request was *List the salary, highest degree, race, and marital status of Jones,* then *How about Evans?* means *List the salary, highest degree, race, and marital status of Evans.* If the context has been your boss saying *I need someone to go to Phoenix next week without jeopardizing meeting the XYZ deadline,* then *How about Evans?* means *Consider Evans for going to Phoenix next week without jeopardizing meeting the XYZ deadline.*

The single phenomenon that has received the most attention in pragmatics is pronominal or other referring expressions. Progress has been substantial enough that pronouns *it, they, those,* etc.) and definite reference (*those submarines, the first three men,* etc.) can be used rather freely in today's systems.

1.3 Knowledge acquisition for language processing

It goes without saying that any NLP system must know a fair amount about words, language, and some subject area before being able to understand language. Currently, virtually all NLP systems operate using fairly laboriously hand-built knowledge bases. The knowledge bases may include both linguistic knowledge (morphological, lexical, syntactic, semantic, and discourse) and nonlinguistic knowledge (semantic world knowledge, pragmatic, planning, inference), and the knowledge in them may be absolute or probabilistic. (Not all of these knowledge bases are necessary for every NLP system.)

1.3.1 Types of knowledge

Typically, porting a NLP system to a new domain requires acquiring knowledge for the domain-dependent modules, which often include:

> *Domain model.* The major classes of entities in the domain and the relations among them must be specified. In a Navy command and control domain, example concepts are Naval unit, vessel, surface vessel, submarine, carrier, combat readiness ratings, and equipment classes. Class–subclass relationships must be specified, e.g., every carrier is a surface vessel, and every surface vessel is a vessel and a Naval unit. Other important relationships among concepts must be specified. For instance, each vessel has a single overall combat readiness rating, and each Navy unit has an equipment loadout (a list of equipment classes).
> *Lexical syntax.* Syntactic information about each word of the domain includes its part of speech (e.g., noun, verb, adjective, adverb, proper noun), its related forms (e.g., the plural of *ship* is regular *ships,* but the plural of *sheep* and *child* are irregular *sheep* and *children*), and its grammatical properties (e.g., the verb *sleep* is intransitive).
> *Lexical semantics.* For each word, its semantics must be specified as a concept in the domain model, a relation in the domain model, or some formula made up of concepts and relations of the domain model.
> *Mappings to the target application.* Transformations specify how to map each concept or relation of the domain model into an appropriate piece of code for the underlying application system. For example, to find out whether a given vessel is equipped with helicopters, one might have to check whether there is a "Y" in the HELO field of the VES table of the database.

Currently, domain-independent knowledge is usually hand-built and is not reacquired when moving to a new domain, although it may be necessary to "tweek" rules and extend this knowledge, again, often by hand. It includes:

> *Grammar rules.* Most rules of English grammar are domain independent, but almost every domain encountered in practice either turns up instances

of general rules that had not been encountered in previous domains, or requires that some domain-specific additions be made to the grammar.
General semantic interpretation rules. Some semantic rules may be considered to be domain independent, such as the general entity/property relationship that is often expressed with the general verb "have" or the general preposition "of." To the extent that such general rules can be found and embedded in a system, they do not have to be redone for every new domain.

The success of all current NLP systems depends on what we call the Limited Domain Assumption, which may be stated as follows: one does not have to acquire domain-dependent information about words that do not denote some concept or relation in the domain. Another way of looking at this assumption is that it says understanding can be confined to a limited domain. The Limited Domain Assumption simplifies the problem of NLP in three ways: (1) formal modeling of the concepts and relationships of the domain is feasible, (2) enumeration of critical nonlinguistic knowledge is possible, and (3) both lexical and semantic ambiguity are limited. Reducing lexical ambiguity reduces the search space and improves effectiveness of most NL systems.

Those three facts have the combined effect of making it more tractable to determine what the user meant by a given input, among a welter of possibilities. But whether one tries to loosen the domain restrictions or is willing to live within them, it seems obvious (although we will examine this assumption later) that the more knowledge that is available to the system, the better its chances of understanding its input.

1.3.2 Types of knowledge acquisition

Just as there are many kinds of knowledge, there are a number of different ways of acquiring that knowledge:

Knowing by being pre-programmed – this includes such things as hand-built grammars and semantic interpretation rules.
Knowing by being told – this includes things that a human can "tell" the system using various user-interface tools, such as semantic interpretation rules that can be automatically built from examples, selectional restrictions, and various lexical and morphological features.
Knowing by looking it up – this means using references such as an online dictionary, where one can find exactly the information that is being sought.
Knowing by using source material – this means using references such as an encyclopedia or a corpus of domain-relevant material, from which one might be able to find or infer the information being sought; it may also mean using large volumes of material as the source of probabilistic

Knowledge Type	Pre-Program	Being Told	Look It Up	From Sources	Figure It Out	Combined
Lexical (morph, etc)	o	o	X	X	X	X
Grammar rules	o					
Select. restrict.	o	o		X		
Domain semantics	o	o	X	X	X	
Real world semantics	o	o	X	X		
Plans	o	o				
Inference	o					

Figure 1.1. How natural language systems acquire knowledge, today (o) and in the near future (X).

knowledge (e.g., "bank" is more likely to mean a financial institution than the side of a river).

Knowing by figuring it out – this means using heuristics and the input itself (such as the part of speech of words surrounding an unknown word).

Knowing by using a combination of the above techniques – this may or may not involve human intervention.

Figure 1.1 shows, for a typical NLP system of the 1980s, which knowledge bases are derived from which processes, and where we expect significant changes in the near future. Notice that the ways of learning do not necessarily correspond to the types of knowledge in any direct way. Certainly all of the types of knowledge can be pre-programmed into an NLP system; indeed that is how most of the current systems were created. It is a fairly simple step from that to learning by being told – usually all that is needed is a nice user interface for creating the same structures that can be pre-programmed. It is not until we reach the level of knowing by looking it up that it seems right to use the word "learning" to describe what is going on.

The two areas of particular interest here are learning from sources, and learning by figuring it out, or some combination of these with learning by being told by a human being reserved for situations that cannot be covered by the other means.

Learning by looking it up

It is hard to learn by looking it up or from sources, but it is going to get easier.

On-line dictionaries and other reference books have been available for many years, as have bodies of text such as news wires and technical abstracts, but they

have not enjoyed wide usage. Why not? It is not entirely a matter of cost, or speed of access. We believe there are four fundamental reasons why computational linguists have been avoiding these sources:

1. The required information is often not there.
2. Information is hard to extract from the sources.
3. Once extracted, the information is hard to use.
4. The information is often incomplete and/or incorrect.

Most domains use common English words with specialized meanings. For example, most dictionaries contain definitions of the words "virus" and "worm," but not with the meanings that are current in the computer industry. Even if a word is found with its appropriate meaning, the dictionary entry may lack information that is critical to the NL system (e.g., selectional restrictions). And if the word is found in a corpus of source material, how is the meaning to be inferred? As a larger volume of domain-specific material becomes available for many domains, this problem may be reduced, but it will always be with us.

Extracting detailed information about words or concepts from the kind of text found in dictionaries and encyclopedias is an enticing prospect, but it presents a chicken-and-egg problem. A system cannot read a dictionary or encyclopedia entry unless it knows all the words in the definition (and usually a great deal more). Since language of this type is often beyond the capabilities of NL systems (particularly those built on the premise that the input and output must be complete), NLP systems typically cannot read the reference material. One solution to this problem is to pre-process the reference material, as is being done by Mitch Marcus at the University of Pennsylvania in an effort to produce text roughly annotated with part of speech and syntactic structure.

Another solution is to relax the constraints on input and output of NLP systems, and to develop partial understanders that can glean some information from sources and, using that information, can re-read the sources to increase their understanding by bootstrapping. Recent work by Will Crowther (1989) has taken this approach quite successfully.

Learning from sources

Does knowing more mean that understanding is easier, or harder?

Suppose we solve the problem of extracting information about words and other things from reference books. Will that automatically mean that our NLP systems will perform better? There is strong evidence that this is not the case – because the increased lexical, syntactic, and semantic alternatives that are introduced by knowing, for example, all the parts of speech and all the possible meanings of all the words in a sentence can easily swamp a NL processor with an explosion of possible alternatives to explore, and unresolvable ambiguities may arise when exploring even just a few!

The last major reason for avoiding source material is that such sources, massive as they are, are inevitably incomplete and incorrect. Nearly all NLP systems deal with specific limited domains, generally rather technical domains (weather forecasts, Navy messages, banking, etc.) in which ordinary English words are given special or restricted meanings. Thus general sources such as dictionaries give meanings that are misleading or actually wrong, but the NLP system has no way of knowing this. It would be far better for the sources to have no information than to have the wrong information, but that is not realistic or even remotely possible.

Our conclusion is that dictionary and other source information will not be useful unless we learn how to focus NL processing, order meanings and partially understood phrases, and interact with other knowledge sources (including humans) when necessary. Fortunately, there are several ways of achieving these goals, including:

1. Representing ambiguity at many levels of processing in a computationally tractable way.
2. Using statistical probabilities at many levels to order choices and cut off low likelihood paths.

Learning by being told

Some situations will always call for learning by being told. To illustrate this, consider the following sentence:

Sebastian compensated his Glock.

Do you know what that means? What can you figure out, and how? Presumably you know that Sebastian is a male's name, although if you did not know that, you might find it out by consulting a good dictionary with a list of names. You already know the verb "compensate" (or can look it up), with meanings roughly comparable to "pay" and "make up for"; the latter meaning is unlikely since it requires a for-clause. The word "Glock" is a stumper. You are unlikely to find it in any dictionary or encyclopedia you have handy. It seems to be a proper noun, judging from the capitalization. You might guess that it is a person's name, although the use of the possessive pronoun with a proper name is quite unusual, and would probably carry some special meaning that cannot be figured out from the sentence itself. Perhaps you have some other hypothesis about the word "Glock". The point is, without help from a human being knowledgeable about the subject area (or an extremely specialized dictionary), you are unlikely to figure out what that sentence means, even with considerable effort.

Adding context is not necessarily a help! Suppose the sentence had come to you as part of a message which said, in its entirety:

Henry and Sebastian were rivals, each preparing for the upcoming competition in his own way. In order to improve his chances, Henry practiced hard. Sebastian compensated his

Glock. Lyn didn't think this would help, and advocated more practice instead, but Sebastian pursued his plan single-mindedly.

There is quite a lot of information in that paragraph, but nothing that is very helpful in figuring out about compensating Glocks.

But if you are told that Glock is a firearms manufacturer (and therefore can be used to refer generically to any firearm they produce, as is the case with Colt), and that certain guns can have a device called a compensator installed to reduce the recoil when they are fired, then you can probably figure out that *Sebastian compensated his Glock* means that Sebastian had a compensator added to his Glock pistol. There is no good alternative to being told this information.

The hard part is not developing rules to infer the meaning of XXXed from XXXor; such rules have been known for a long time. The hard part is to know when to apply those rules, and how to keep hundreds of those rules from interacting to produce more fog than clarity.

1.3.3 New approaches to knowledge acquisition

A breakthrough in the effectiveness and applicability of knowledge acquisition procedures may be possible within the next five years. In this section the following two research approaches are identified:

1. Employing large, growing knowledge bases acquired from reference texts such as dictionaries. This contributes to robustness by facilitating acquisition of knowledge for semantic and pragmatic components.
2. Acquisition of syntactic, semantic, and discourse facts from annotated bodies of language. This contributes to robustness of syntactic, semantic, and discourse components and allows semi-automatic learning of syntactic and semantic knowledge.

Knowledge from text

Recently a handful of efforts have focused on creating large knowledge bases of common facts. The CYC project at MCC is employing ten to twenty programmers to handcraft a knowledge base based on a selection of encyclopedia articles (Lenat et al., 1986). At IBM Yorktown Heights (Chodorow, Ravin, and Sachar, 1988; Jensen and Binot, 1988; Neff and Boguraev, 1989) and Bell Communications Research, efforts are underway to derive automatically synonym sets, lexical syntax, and other information from on-line dictionaries. One effort underway at BBN is the Common Facts Data Base (CFDB) (Crowther, 1989), which has been used to derive common facts from dictionaries and is being applied experimentally to other reference material.

In this section we illustrate two of the many ways such automatically derivable databases can increase robustness compared to today's systems. A long-standing

problem is the interpretation of nominal compounds, sequences of nouns such as *carrier task force*. Heretofore one had to handcraft a definition for each example or small class of examples. Some informal definitions are provided directly in dictionaries for frequently occurring, well-known expressions, e.g., *fire engine*. These may be automatically derivable by analyzing a dictionary. Others follow regular patterns (Ayuso, 1985; Finin, 1980), such as part-whole relations, which require common facts in order to be interpreted. For example, if the NLP system encounters *helicopter rotor* for the first time, it could be understood if the knowledge base contains the information that a rotor is part of a helicopter.

Another long-standing problem is interpreting definite references. The use of syntactic information to constrain and rank what an anaphoric expression can refer to is rather well understood. References involving the same terminology are also rather well understood, e.g., using *those ships* as a short form after mentioning *all C1 surface ships in the Indian Ocean*. However, the class of references that illustrates non-robustness in current discourse components are those that require a "bridge" (Clark, 1975) between what is mentioned, e.g., a connection between the expression *the flight deck,* and the expression that implies its existence, e.g., *the carrier Midway*. One hypothesis is that bridges fall into one of potentially a few dozen patterns, in this case, referring to a part after mentioning the whole. The common fact needed is that aircraft carriers have a flight deck. Such bridges require large volumes of common, mundane facts, such as those that might be derived from a dictionary, glossary, or parts manual.

Both nominal compounds and discourse anaphora seem to fall into a few dozen semantic patterns, each of which assumes a large set of common facts. It has been easy to implement such semantic patterns for some time; what has been lacking is a way to derive automatically the large set of common facts assumed.

Linguistic analysis of large bodies of text

NLP research has been hampered by a lack of sufficient linguistic data to derive statistically significant patterns. Volumes of text are available on-line; the problem has been how to derive linguistic facts from unanalyzed text. However, through a DARPA-funded effort at the University of Pennsylvania, corpora of annotated text will be available to other research sites. The annotations will include parts of speech and phrasal structure, e.g., syntactic structure. This syntactically annotated corpus should make two new developments feasible:

1. Development of acquisition procedures to learn new grammar rules for expressions never seen before by the NLP.
2. Collection of statistics regarding constructions and their probability of occurrence in context.

Automatic acquisition will reduce the need for handcrafting of both grammars and lexicons (the formal model of dictionary information for an NLP).

Original text:

Collection of statistics regarding constructions and their probability of occurrence in context.

Part of Speech Tagging:

Collection **Noun** of **Prep** statistics **Noun** regarding **Prep** constructions **Noun** and **Conjunction** their **Pro** probability **Noun** of **Prep** occurrence **Noun** in **Prep** context **Noun**

Structure Tagging:

[Collection [of [statistics [regarding [[constructions]$_{NP}$ and [their probability [of [occurrence [in [context]$_{NP}$]$_{PP}$]$_{NP}$]$_{PP}$]$_{NP}$]$_{PP}$]$_{NP}$]$_{PP}$]$_{NP}$

Figure 1.2. Examples of two types of tagging.

Example annotations of the type of syntactic information that might be useful appear in Figure 1.2.

Any rules implicit in the annotation but not present in the current grammar are candidates to be automatically added to the grammar.

The annotations also allow acquisition of lexical information; for words not in the system dictionary, the annotations state part of speech and the syntactic context in which they occur. Suppose *regarding* were not known to the system before it encountered the annotated example above; this word could be added to the system lexicon as a preposition through processing the annotated example. Thus, annotations provide data that can be used to create systems that adapt by acquiring grammar rules and information about new words.

Summary

The challenge: To develop ways of acquiring knowledge in such a way as to permit focusing parsing and semantic interpretation without combinatorial explosion.

The new approach: Glean probabilistic information from bodies of text. Develop ways to combine automatically acquired knowledge with "being told" by humans. Use disjunctive processing, merging alternatives at all levels, guided by probabilistics and using cutoffs to reduce the number of alternatives considered. (Also use the understanding-by-fragments approach outlined in Section 1.5 of this chapter.)

1.4 Interfacing to multiple underlying systems

Most current NL systems, whether accepting spoken or typed input, are designed to interface to a single homogeneous underlying system; they have a component

geared to producing code for that single class of application systems, such as a relational database (Stallard, 1987; *Parlance User Manual, Learner User Manual*). These systems take advantage of the simplicity of the semantics and the availability of a formal language (relational calculus and relational algebra) for the system's output.

The challenge is to recreate a systematic, tractable procedure to translate from the logical expression of the user's input to systems that are not fully relational, such as expert system functions, object-oriented and numerical simulation systems, calculation programs, and so on. Implicit in that challenge is the need to generate code for non-homogeneous software applications – those that have more than one application system.

The norm in the next generation of user environments will be distributed, networked applications. A seamless, multi-modal, NL interface will make use of a heterogeneous environment feasible for users and, if done well, transparent. Otherwise, the user will be limited by the complexity, idiosyncrasy, and diversity of the computing environment.

Such interfaces will be seamless in at least two senses:

1. The user can state information needs without specifying how to decompose those needs into a program calling the various underlying systems required to meet those needs. Therefore, no seams between the underlying systems will be visible.
2. The interface will use multiple input/output modalities (graphics, menus, tables, pointing, and natural language). Therefore, there should be no seams between input/output modalities.

In military uses, we expect that the need to access several heterogeneous application systems will arise as the norm in command and control, in logistics, and in contract management. Because of the need to include previously existing application software, each having its own assumptions regarding operating systems, heterogeneous software environments will arise. Because of the relative performance-cost trade-offs in workstations, mainframes, and parallel hardware, the hardware equipment will be heterogeneous as well.

For example, in DARPA's Fleet Command Center Battle Management Program (FCCBMP), several applications (call them underlying systems) are involved, including a relational database (IDB), two expert systems (CASES and FRESH), and a decision support system (OSGP). The hardware platforms include workstations, conventional time-sharing machines, and parallel mainframes. Suppose the user asks *Which of those submarines has the greatest probability of locating A within 10 hours?* Answering that question involves subproblems from several underlying applications: the display facility (to determine both what *those submarines* means and to display those which fulfill the user's request); FRESH to calculate how long it would take each submarine to get to the area *A;* CASES, for an intensive numerical calculation estimating the probabilities; and the display facility again, to present the response.

Although acoustic and linguistic processing can determine what the user wants, the problem of translating that desire into an effective program to achieve the user's objective is a challenging, but solvable problem.

In order to deal with multiple underlying systems, not only must our NL interface be able to represent the meaning of the user's request, but it must also be capable of organizing the various application programs at its disposal, choosing which combination of resources to use, and supervising the transfer of data among them. We call this the Multiple Underlying Systems (MUS) problem. BBN's approach and results on the MUS problem are part of the back end of the Janus natural language interface and are documented in Resnik (1989).

1.4.1 The scope of the problem

In our view of access to multiple underlying systems, the user's request, whatever its modality, is translated into an internal representation of the meaning of what the user needs. We initially explored a first-order logic for this purpose; however, in Janus (Weischedel, 1987) we have adopted an intensional logic (Hinrichs et al., 1987; Weischedel, 1989) to investigate whether intensional logic offers more appropriate representations for applications more complex than databases, e.g., simulations and other calculations in hypothetical situations. From the statement of what the user needs, we next derive a statement of how to fulfill that need – an executable plan composed of abstract commands. The executable plan is in essence an abstract data-flow program on a virtual machine that includes the capabilities of all of the application systems. At the level of that virtual machine, specific commands to specific underlying systems are dispatched, results from those application systems are composed, and decisions are made regarding the appropriate presentation of information to the user.

Thus, the Multiple Underlying Systems (MUS) problem is a mapping,

$$\text{MUS: Semantic representation} \rightarrow \text{Program}$$

that is, a mapping from what the user wants to a program to fulfill those needs, using the heterogeneous application programs' functionality.

Although the statement of the problem as phrased above may at first suggest an extremely difficult and long-range program of research in automatic programming (e.g., see Rich and Waters, 1988), there are several ways one can narrow the scope of the problem to make utility achievable. Substantially restricting the input language is certainly one way to narrow the problem to one that is tractable.

In contrast, we allow a richer input language (an intensional logic), but assume that the output is a restricted class of programs – acyclic data-flow graphs. We assume that all primitives of the logic have a defined transformation from the level of the statement of the user's needs to the level of the executable plan. That definition will have been elicited from the application system experts (e.g., expert system builders, database administrators, and systems programming staff of other application systems).

A way to paraphrase the effect of assuming acyclic data-flow graphs as the output of the component is that the programs generated will be assumed to include

> Functions available in the underlying applications systems,
> Routines pre-programmed by the application system staff, and
> Operators on those elements such as: functional composition, if-then-else, operators from the relational algebra, and MAPCAR.

Therefore, the system need not derive programs for terms that it does not already know. Contrast that with the general automatic programming problem. Suppose that someone says to the system *Find the square root of the sum of the squares of the residuals,* so that the input can be correctly translated into a logical form, but that the underlying applications do not provide a square-root function. Then the interface will not be expected to derive a square-root program from arithmetic functions. Rather, this system will be expected to respond *I don't know how to compute square root.* Furthermore, if all the quantifiers are assumed to be restricted to finite sets with a generator function, then the quantifiers can be converted to simple loops over the elements of sets, such as the mapping operators of Lisp, rather than having to undertake synthesis of arbitrary program loops.

Even with these simplifying assumptions, there are interesting problems remaining, and the work offers highly desirable utility. The utility arises from two dimensions:

1. It frees the user from having to identify for each term (word) pieces of program that would carry out their meaning, for the application system programmers would do that for some appropriate set of terms.
2. It provides good software engineering of the interface, so that table input/output functionality, for instance, is insulated from the details of the underlying application or applications as they evolve.

1.4.2 Approach

The problem of multiple systems may be decomposed into the following subproblems:

> *Representation.* It is necessary to represent underlying system capabilities in a uniform way, and to represent the user request in a form independent of any particular underlying system. The input/output constraints for each function of each underlying system must be specified, thus defining the services available.
> *Formulation.* One must choose a combination of underlying system services that satisfies the user request. Where more than one alternative

exists, it is preferable to select a solution with low execution costs and low passing of information between systems.

Execution. Actual calls to the underlying systems must be accomplished, information must be passed among the systems as required, and an appropriate response must be generated.

Representing the semantics of utterances

Since the meaning of an utterance in Janus is represented as an expression in WML (World Model Language [Hinrichs et al., 1987], an intensional logic, the input to the MUS component is in WML. The choice of WML was based on two grounds: first and foremost, although we found first-order representations adequate (and desirable) for NL interfaces to relational databases, we felt a richer semantic representation was important for future applications. The following classes of representation challenges motivated our choice: explicit representations of time and world, for instance, to support object-oriented simulation systems and expert systems involving hypothetical worlds; distributive/collective readings; generics, and mass terms; and propositional attitudes, such as statements of user preference and belief. Our second motivation for choosing intensional logic was our desire to capitalize on other advantages we perceived for applying intensional logic to natural language processing (NLP), such as the potential simplicity and compositionality of mapping from syntactic form to semantic representation and the many studies in linguistic semantics that assume some form of intensional logic.

For a sentence such as *Display the destroyers within 500 miles of Vinson,* the WML is as follows:

```
(bring-about
  ((intension
    (exists ?a display
      (object-of ?a
        (iota ?b (power destroyer)
          (exists ?c
            (lambda (?d) interval
              (& (starts-interval ?d VINSON)
                 (less-than
                    (iota ?e length-measure
                       (interval-length ?d ?e))
                    (iota ?f length-measure
                       (& (measure-unit ?f miles)
                          (measure-quantity ?f 500))))))
                 (ends-interval ?c ?b))))))
  TIME WORLD))
```

https://doi.org/10.1017/CBO9780511659478.003 Published online by Cambridge University Press

Representing the functions of the applications

To represent the functional capabilities of underlying systems, we define services and servers. A server is a functional module typically corresponding to an underlying system or a major part of an underlying system. Each server offers a number of services: objects describing a particular piece of functionality provided by a server. Specifying a service in MUS provides for the mapping from fragments of logical form to fragments of underlying system code. For instance, the following is a list of services in a naval application. Each service has associated with it the server it is part of, the input variables, the output variables, the conjuncts computed, and an estimate of the relative cost in applying it.

```
Land-avoidance-distance:
  owner: Expert System 1
  inputs: (x y)
  locals: (z w)
  pattern:
    ((in-class x vessel)
     (in-class y vessel)
     (in-class z interval)
     (in-class w length-measure)
     (starts-interval z x)
     (ends-interval z y)
     (interval-length z w))
  outputs: (w)
  method: ((route-distance (location-of x) (location-of y)))
  cost: 5

Great-circle-distance:
  owner: Expert System 1
  inputs: (x y)
  locals: (z w)
  pattern:
    ((in-class x vessel)
     (in-class y vessel)
     (in-class z interval)
     (in-class w length-measure)
     (starts-interval z x)
     (ends-interval z y)
     (interval-length z w))
  outputs: (w)
  method: ((gc-distance (location-of x) (location-of y)))
  cost: 1
```

In the example above, there are two competing services for computing distance between two ships: Great-circle-distance, which simply computes a great circle route between two points, and Land-avoidance-distance, which computes the distance of an actual path avoiding land and sticking to shipping lanes.

Clause lists

Usually, the applicability of a service is contingent on several facts, and therefore several propositions must all be true for the service to apply. To facilitate matching the requirements of a given service against the needs expressed in an utterance, we convert expressions in WML to a disjunction normal form (DNF), i.e., a disjunction of conjunctions where quantifiers and higher level operators have been removed. We chose DNF because:

> In the simplest case, an expression in disjunctive normal form is simply a conjunction of clauses, a particularly easy logical form with which to cope.
> Even when there are disjuncts, each can be individually handled as a conjunction of clauses, and the results then combined together via union, and
> In a disjunctive normal form, each disjunct effectively carries all the information necessary for a distinct subquery.

For details of the algorithm for converting an intensional expression to DNF, see Resnik, (1989). For the sentence *Display the destroyers within 500 miles of Vinson,* whose WML representation was represented earlier, the clause list is as follows:

```
((in-class ?a display)
 (object-of ?a ?b)
 (in-class ?b destroyer)
 (in-class ?c interval)
 (in-class ?d interval)
 (equal ?c ?d)
 (starts-interval ?d VINSON)
 (in-class ?e length-measure)
 (interval-length ?d ?e)
 (in-class ?f length-measure)
 (measure-unit ?f miles)
 (measure-quantity ?f 500)
 (less-than ?e ?f)
 (ends-interval ?c ?b))
```

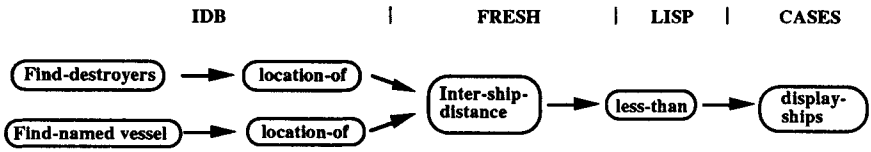

Figure 1.3. A data-flow graph.

Formulation

If one takes the input request to be a conjunction of requirements, finding the services to fulfill the request may be viewed as a form of covering problem: one seeks a plan of execution that satisfies all requirements at minimal cost.

A search is required both to find collections of services that fulfill the request, and to find a low cost solution. A beam search is used.

Inherent in the collection of services covering a DNF expression is the data flow that combines the services into a program to fulfill the DNF request. The next step in the formulation process is data-flow analysis to extract the data-flow graph corresponding to an abstract program fulfilling the request.

In Figure 1.3, the data-flow graph for *Display the destroyers within 500 miles of Vinson* is pictured.

Note that the integrated database (IDB) is called to identify the set of all destroyers, their locations, and the location of Vinson. An expert system FRESH is being called to calculate the distance between pairs of locations using land-avoidance routes. A LISP utility for comparing measures is called, followed by the display command in the CASES system.

Execution

The execution phase has two purposes:

1. Walk through the data-flow graph, calling operators in the underlying systems corresponding to the nodes of the graphs.
2. Supply functions for data combination not available in any of the underlying systems. In our example, a general function for comparing two measures, performing the appropriate unit conversions, was assumed.

Previous approaches to the multiple systems problem seem to have assumed that the data-flow model passes streams of values. This is not always adequate; in many cases, it is necessary to pass sets of tuples rather than sets of values, using a generalization of the join operation to combine data. The details of this are provided in Resnik (1989).

Most previous work dealt with simpler problems, e.g., access to a single

relational database. Two pioneering efforts at Honeywell and at USC/Information Sciences Institute dealt with multiple systems but under a highly restrictive assumption: the user request had to be expressible as a conjunction of simple relations, equivalent to the select/project/join operations of a relational algebra. That restriction is removed in Janus. The class of formal expressions handled includes those involving negation of elementary predicates, existential and universal quantification, cardinality, and some common disjunctions, as well as queries that are simply conjunctions of clauses. Wh-questions (who, what, when where, etc.), commands, and yes/no queries are handled.

Experience in applying the system

The MUS component has been applied in the domain of the Fleet Command Center Battle Management Program (FCCBMP), using an internal version of the Integrated Database (IDB) – a relational database – as one underlying resource, and a set of LISP functions as another. The system includes more than 800 services.

An earlier version of the system described here was also applied to provide natural language access to data in Intellicorp's KEE knowledge-base system, to objects representing hypothetical world-states in an object-oriented simulation system, and to LISP functions capable of manipulating this data.

We have begun integrating the MUS component with BBN's Spoken Language System HARC (Hear And Respond to Continuous speech).

1.4.3 MUS conclusions

We have found the general approach depicted in Figure 1.3 quite flexible. The approach was developed in work on natural language processing; however, it seems to be valuable for other types of I/O modalities. Some preliminary work has suggested its utility for table input and output in managing database update, database retrieval, and a directly manipulable image of tabular data. Our prototype module generates forms in the intensional logic; then the components originally developed for the natural language processor provide the translation mechanism to and from intensional logic and underlying systems that actually store the data.

1.4.4 Summary

The challenge: To develop ways of easily interfacing NL systems to multiple instances of various types of underlying application systems.

The new approach: Define functional representations of the capabilities of underlying systems. Produce mappings to underlying systems based on this func-

tionality. Represent the result of NL processing in this form. Use techniques from expert systems to formulate a process to satisfy the user's request or command.

1.5 Partial understanding of fragments, novel language, and errorful language

It is time to move away from dependence on the sentence as the fundamental unit of language. Historically, input to NL systems has often had to consist of complete, well-formed sentences. The systems would take those sentences one at a time and process them. But language does not always naturally occur in precise sentence-sized chunks. Multi-sentence input is the norm for many systems that must deal with newspaper articles or similar chunks of text. Subsentence fragments are often produced naturally in spoken language and may occur as the output of some text processing. Even when a sentence is complete, it may not be perfectly formed; errors of all kinds, and new words, occur with great frequency in all applications.

1.5.1 Multi-sentence input

Historically, computational linguistics has been conducted under the assumption that the input to a NL system is complete sentences (or, in the case of speech, full utterances) and that the output should be a complete representation of the meaning of the input. This means that NL systems have traditionally been unable to deal well with unknown words, natural speech, language containing noise or errors, very long sentences (say, over 100 words), and certain kinds of constructions such as complex conjunctions.

One of the problems is that advocates of local processing have tended to ignore syntactic and other constraints, while advocates of top down processing have tended to ignore coherent fragments unless they fit properly in the overall scheme.

The solution, we believe, is to move away from thinking that language comes in sentences and that the goal of understanding is a complete representation of meaning. We must move toward processing bits and pieces of language, whether the input to our NL systems comes that way or not, and toward creating structures that, like the fractals found in nature, have a kind of coherency that can be viewed at many levels.

Some semantic distinctions have no selectional import (e.g., quantifiers, and some adjuncts), while others have considerable selectional import.

One of the ideas whose time has passed is the notion of prepositional phrase attachment. Although in many cases it is not harmful to think of a PP attaching to a particular constituent, sometimes it is more useful to think of a single PP attaching simultaneously at several different points (for example, "I kicked the shell on the beach"), or relating two different constituents in a sentence (for

example, "The average concentration of aluminum in breccias"). When fixed constituent structure pinches too much, language should not be forced into it.

What is the right representation for the meaning of multi-sentence language? The "right" representation for text may depend on the type of text and its purpose. For example, commands may be represented very differently from questions. It may also depend on the purpose of the user: for example, question answering versus controlling a process versus storing information for later retrieval.

Currently, most systems that attempt to understand multi-sentence text create a frame as output (or some other structure that is similar in function). Generally, the names of the slots of the frame consist of the type of information and relationships that were to be gleaned from the text, and the fillers describe the entities that were found. Thus it is difficult to represent unexpected information.

1.5.2 Errorful language, including new words

Handling novel, incomplete, or errorful forms is still an area of research. In current interactive systems, new words are often handled by simply asking the user to define them. However, novel phrases or novel syntactic/semantic constructions are also an area of research. Simple errors, such as spelling or typographical errors resulting in a form not in the dictionary, are handled in the state-of-the-art technology, but far more classes of errors require further research.

The state-of-the-art technology in message understanding systems is illustrative. It is impossible to build in all words and expressions ahead of time. As a consequence, approaches that try for full understanding appear brittle when encountering novel forms or errorful expressions.

The state of the art in spoken language understanding is similarly limited. New words, novel language, incomplete utterances, and errorful expressions are not generally handled. Including them poses a major roadblock, for they will decrease the constraint on the input set, increase the perplexity[2] of the language model, and therefore decrease reliability in speech recognition.

There is ample evidence that the ability to deal with novel, incomplete, or errorful forms is fundamental to improving the performance users can expect from NLP systems. Statistical studies for written database access (Eastman and McLean, 1981; Thompson, 1980) show that novel, errorful, or incomplete language comprises as much as 25–30% of type input; such phenomena (Fromkin, 1973) probably arise even more frequently in spoken language than in written language. In addition, we believe that interpreting incomplete input is particularly important for the following reasons:

1. Fragments occur frequently in military messages, such as Navy CASREPs, Navy OPREPs, Army SITREPs, and Army Operations Orders.

[2]Perplexity is a measure of the average number of words that may appear next at any point in the input.

2. Incomplete input arises in spoken language not only because we speak in fragments but also because acoustic processing at times can detect only fragments with high confidence.
3. Fragments result when processing an incomplete, novel, or errorful input, since a complete interpretation cannot be produced.

The problem

In current technology, almost all systems employ a search space of the possible ways of combining the meanings of words into meaningful phrases and a meaningful whole in context. In artificial intelligence terms, the search is a constraint satisfaction problem: find one or more interpretations such that no applicable constraint is violated. Formal models of grammar, semantics, and discourse state constraints on language in an all-or-nothing fashion, as if we always spoke and wrote in complete thoughts, saying exactly what we mean without vagueness, inaccuracy, error, or novelty in expression.

In constraint satisfaction problems, if a search fails to find a solution where all constraints are satisfied, many search alternatives will have been tried without leading to ultimate success. The problem is to come up with a partial solution (in the case of language processing, a partial interpretation), an explanation of why no solution is found (e.g., why no interpretation can be found), or a way to relax a constraint to produce with a complete solution (a complete interpretation). Which of the partial solutions, if any, is the most likely path to lead to success if a constraint is relaxed? Which partial path(s) in the search space is a good basis for explaining why no solution can be found?

All previous work suffers from this problem mentioned above, unless the application domain is very limited or the types of errorful/novel forms allowed are very few. This is because too many alternatives for what was meant are possible; an NLP system does not even have a foolproof way of knowing whether the user's input is errorful or whether the input represents a novel form. How to hypothesize the problem in an input and how to deal with it is understood for a large class of possible problems, e.g., see Carbonell and Hayes (1983); Jensen et al. (1983); Weischedel and Sondheimer (1983). What is not known is how to rank the many alternative interpretations that arise, as illustrated in the example above. The lack of a reliable scoring mechanism has been a technological roadblock.

Real language may be absolutely ill-formed (a native speaker would judge it to be something to be edited, an error, not what was intended, or otherwise "bad"), or relatively ill-formed (ill-formed with respect to a NL system's well-formedness constraints, even though a native speaker may judge it well-formed).

The following kinds of problems were enumerated by Weischedel (1987); others are readily available. Some examples of absolutely ill-formed language that are peculiar to written language are:

1. Typographical errors, e.g., *oter,* instead of *other.* Typos may also result in recognizable words, such as *an* instead of *and.*
2. Spelling errors, e.g., *Ralf* instead of *Ralph.*
3. Punctuation errors, e.g., inserting or omitting commas incorrectly, misplacement or omission of apostrophes in possessives, etc.
4. Homonym errors, e.g., *to* instead of *too,* or confusing *there, their,* and *they're.*

Similarly, there are classes of absolute ill-formedness peculiar to spoken language.

5. Mispronunciations, e.g., saying that word as if it were spelled *mispronounciations,* or stressing the wrong syllable. Fromkin (1973) has provided a taxonomy of human speech production errors that appear rule-based, as opposed to ungoverned or random occurrences.
6. Spoonerisms, e.g., saying *fauter waucet* instead of *water faucet.*

Each of the classes above are human performance errors, resulting in absolute ill-formedness. However, the overwhelming variety of ill-formedness problems arise in both the spoken and written modality; examples of absolute ill-formedness include:

1. Misreference, as in describing a purple object as *the blue one.*
2. Word order switching, as in saying *the terminal of the screen* when one meant *the screen of the terminal.* (Fromkin [1973] has recorded these errors.)
3. Negation errors, e.g., *All doors will not open* when the train conductor meant *Not all doors will open.*
4. Omitting words, as in *Send file printer* rather than the full form *Send the file to the printer.* (Although this may seem to occur only in typed language, we have heard such omissions in spoken language. Further, consider how many times, when struggling for the appropriate word, you start the utterance over, or someone supplies an appropriate word for you.)
5. Subject–verb disagreement, as in *A particularly important and challenging collection of problems are relatively ill-formed and arise in both spoken and written language* or in *One of the overwhelming number of troubles that befell them are . . .*
6. Resumptive pronouns and resumptive noun phrases, as in *The people that he told them about it,* where *them* is intended to be coreferential with *people.*
7. Run-together sentences, as if the person forgot how the sentence was started. An example is: *She couldn't expect to get a high standard salary and plus being so young.*

8. Restarted sentences, as in *Some people many try to improve society,* which was also collected in a written corpus.
9. Pronominal case errors, as in *between you and I.*
10. Word order errors, as non-native speakers can make, e.g., *I wonder where is the problem.*

Some particularly important and challenging problems are relatively ill-formed and arise in both spoken and written language.

1. Words unknown to the hearer or reader, but part of the language.
2. Novel or unknown word senses, although the word itself is known. For instance, Navy jargon includes phrases such as *What is Stark's readiness?* Although that sublanguage does not include *preparedness* as a synonym for readiness, it would be useful for a system to be able to infer what a user means by the input *What is Stark's preparedness?*
3. Novel (non-frozen) figures of speech, e.g., metaphor, metonymy, and synecdoche.
4. Novel nominal compounds, as in *window aisle seat,* which was used by a flight attendant on a wide-body jet.
5. Violated presuppositions, as in *Did John fail to go?* when John did not try to go.

The above lists are not intended to be exhaustive. More thorough taxonomies of ill-formedness exist. Statistical studies of frequency of occurrence for various classes of ill-formedness have been conducted for written database access; those studies suggest that as much as 25–30% of typed input may be absolutely or relatively ill-formed.

From the definitions and examples, it is clear that

1. Ill-formed input need not be ungrammatical; there may be no interpretation due to semantic or pragmatic problems.
2. The NL system will probably not know whether the input contains an error or whether its models are too limited to process the input.
3. Since there is no interpretation for the input, then one or more of the constraints of the NL system are violated; understanding ill-formed input therefore is a constraint satisfaction problem.
4. Since one or more of the constraints are violated, relaxing constraints in order to find an interpretation will mean opening up the search space for an interpretation substantially.

A suggestion

One new approach is to use probabilistic language models based on statistics derived from a chosen corpus, and utilizing those statistics together with the

knowledge bases acquired from the corpus. The probabilistic model will rank partial interpretations for incomplete, errorful, or novel expressions. This will enable ranking of alternative interpretations when the input is complete, incomplete, or errorful.

The large annotated corpora described in the previous section will offer significant data to estimate such probabilities. For instance, the frequency of occurrence of types of phrases (e.g., NP and PP in the earlier annotated example) and statistics on relative frequency of grammar rules can be computed. Such statistics can be used to find the most predictive statistical language models for NLP systems.

The probabilistic language models in speech recognition are probably not directly applicable. Typically probabilities of two- or three-word sequences are computed from a corpus of utterances and are used in assigning weights to each alternative rendering of the speech wave into sequences of words. The limitation in those models is that only local information is used, whereas it is well known in linguistics that there are long distance dependencies well beyond three-word sequences.

Scoring techniques based on large annotated corpora may provide the missing link for progress in understanding fragmentary language, in processing errorful language, in determining what was meant in novel expressions, and in processing incomplete forms.

In the last ten years, it has often been suggested that ignoring constraints, or bottom-up parsing, or a semantics-first strategy might be used to deal with ill-formed input, but in each case, although particular examples could be made to work, the approach generated too many possibilities to be used in a truly general way. However, there seems to be a clear distinction between those classes of problems for which reasonably good syntactic and semantic strategies exist, and classes of ill-formedness that seem particularly intractable without a strong model of pragmatic knowledge for proper understanding. Examples of the latter include asias errors (spelling/typographical errors that result in a known word), run-together sentences, pragmatic overshoot, contextual ellipsis requiring considerable reasoning to resolve, and inferring the meaning of unknown words.

1.5.3 Summary

The challenge. To develop appropriate representations of fragmented, extended, errorful language, partially understood.

The new approach. Use local structure-finding processes that work primarily bottom-up, inserting local information into a global framework (not a standard parse tree or logical expression), and switching strategies to top-down when possible.

1.6 Additional research opportunities

We suggest the following as areas of opportunity for near-term research to make significant breakthroughs that will move NLP through the 1900s and beyond:

Acquisition of corpora, grammars, and lexicons. The development of useful systems requires observation of the behavior of potential users of interactive systems under realistic circumstances, and the collection of corpora of typical data for text analysis and machine translation systems. Although we believe it is unlikely that full grammars and lexicons can be induced completely automatically in the near future, useful results may be obtained soon from induction and acquisition techniques based on annotated corpora and machine-readable dictionaries. It is also likely that statistical measures useful for biasing algorithms can be extracted from a handcrafted grammar and a corpus. Approaches that appear promising are (1) the learning of grammatical structures where the input has already been annotated by part of speech and/or phrase structure, and (2) the learning of lexical syntax/semantics from examples and/or queries to the user given some pre-coded domain knowledge.

Increasing expressive power of semantic representation languages. Moving beyond database query systems will require increasing the expressive power of the languages used to express meaning, to include at least modal and higher-order constructs. Reasoning tools for modal logics and for second-order logics already exist, but appear intractable for language processing tasks. Approaches that seem promising include encoding modal constructs in first-order logic, hybrid approaches to representation and reasoning, and approaches to resource-limited and/or shallow reasoning, such as adding weights to formulae and subformulae.

Reasoning about plans. Recent work on plan recognition – the inference of the beliefs and intentions of agents in context – has provided formal definitions of the problem and some new algorithms. These have not yet been used as part of a discourse component to help resolve reference, quantification, and modification ambiguities or to formulate an appropriate response. The interaction between plans, discourse structure, and focus of attention must also be investigated. Promising approaches include incorporation of beliefs of the discourse participants, integrating existing models into discourse processing under simplifying conditions, and exploring prosodic/linguistic cues to dialogue.

Combination of partial information. The standard control structure by which various sources of information are combined in language interpretation seems to limit what NL systems can do. Several proposals for more flexible control structures have been made recently, each covering a subset of the knowledge sources available. More comprehensive schemes need to be developed. Two

promising approaches are generalization of unification to NL architectures, and use of global, weighted control strategies, such as in evidential reasoning.

Improving robustness. Published studies suggest that as much as 25–30% of typed input contains errors, is incomplete, uses novel language, or otherwise involves challenging phenomena that are not well handled theoretically. The frequency of occurrence for these classes is even higher in spoken language than in written language. The text of some messages, such as Navy RAINFORM and CASREP messages and bank telexes, is highly telegraphic. It should be possible to develop a domain-independent theory that allows at least partial understanding of some of these novel and errorful uses, and test it in narrowly defined domains. Promising approaches are to employ unification strategies, plan recognition, and/or weighted control strategies to determine the most likely interpretation and the most appropriate response/action.

Relating interpretation and action. The problem of how to relate interpretations expressed in a meaning representation language and calls to application systems (databases, summarizing algorithms, etc.) has not been fully resolved, nor in fact precisely stated. This is crucial to the systematic separation of the natural language part of the system from the application part. Any approach should deal with applications beyond databases (beyond the semantics of tables) and should avoid the challenges of automatic programming.

Finding the relationship between prosody, syntactic ambiguity, and discourse structure. Syntactic and discourse boundaries are one of the main sources of interpretation ambiguity. Recently discovered evidence shows that prosodic information is a good indicator of these boundaries. Automatic extraction of prosodic information would revolutionize the interpretation of spoken language. Further, generation systems could add prosodic information to signal syntactic structure and discourse structure.

Measuring progress. The means of measuring progress is still an active area of discussion among NL scientists, as evidenced by workshops on Natural Language Evaluation held in December 1988 and June 1991 (Neal and Walter, 1991). Measures of correctness can be relatively simply stated for database query systems without dialogue capabilities (e.g., without sequence-related queries or clarifications), or for text analysis systems for database entry. They are much more difficult to state when stylistic matters need to be considered (as in machine translation systems) or when system responses affect subsequent user utterances. They probably cannot be usefully stated in a domain- or task-independent way. Measures of task difficulty, or of ambiguity of the language model, analogous to speech recognition's perplexity, are much more difficult to state. The recent DARPA program in spoken language understanding is developing formalisms for evaluating spoken language systems (Boisen et al., 1989; Bates et al., 1991).

Measurement of NL systems requires three distinct types of comparisons:

1. Longitudinal: It is critical to be able to measure the performance of a system over time, so that progress can be tracked.
2. Cross-System: It should be possible to compare the overall performance of two systems in explicit terms. This focus on whole-system performance will help localize the strengths and weaknesses of complete systems and will identify topics for research and development efforts.
3. Component: It should be possible to evaluate and compare parts of systems and evaluate coverage of unknown phenomena. This focus on components will help point out areas of relative strength in different systems and will provide priorities and goals for specific research.

Both the longitudinal and cross-system measures should include not merely the percentage of inputs banded correctly but also estimates of productivity improvements for the end user.

1.7 Conclusion

The list of hard, interesting problems on which to make progress in computational linguistics could go on and on. However, we feel that knowledge acquisition, interaction with multiple underlying systems, and techniques for partial understanding are the three solvable problems that will have the most impact on the utility of natural language processing. We encourage students to embark on these rewarding research areas, and look forward with eagerness to see what advances the next decade will bring.

References

Ayuso, D. (1985). *The Logical Interpretation of Noun Compounds*. Master's thesis, Massachusetts Institute of Technology.

Ayuso, D. M., Shaked, V., and Weischedel, R. M. (1987). "An Environment for Acquiring Semantic Information." In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pp. 32–40. ACL.

Bates, M. (1989). "Rapid Porting of the Parlance Natural Language Interface." In *Proceedings of the Speech and Natural Language Workshop*, pp. 83–88. Morgan Kaufmann Publishers Inc., San Mateo, CA.

Bates, M., Bobrow, R., Boisen, S., Ingria, R., and Stallard, D. (1991). "BBN ATIS System Progress Report – June 1990," *DARPA Speech and Natural Language Workshop*, Hidden Valley, PA, Morgan Kaufmann Publishers, pp. 125–126.

Bates, M., Boisen, S., and Makhoul, J. (1991). "Developing an Evaluation Methodology for Spoken Language Systems," *DARPA Speech and Natural Language Workshop*, Hidden Valley, PA, Morgan Kaufmann Publishers, pp. 102–108.

Bobrow, R., Ingria, R., and Stallard, D. (1991). "Syntactic and Semantic Knowledge in the DELPHI Unification Grammar," *DARPA Speech and Natural Language Workshop*, Hidden Valley, PA, Morgan Kaufmann Publishers, pp. 230–236.

Bobrow, R., and Ramshaw, L. (1991). "On Deftly Introducing Procedural Elements into

Unification Parsing," *DARPA Speech and Natural Language Workshop,* Hidden Valley, PA, Morgan Kaufmann Publishers, pp. 237–240.

Boisen, S., Ramshaw, L., Ayuso, D., and Bates, M. (1989). "A Proposal for SLS Evaluation," *DARPA Speech and Natural Language Workshop,* Cape Cod, MA, Morgan Kaufmann Publishers, pp. 135–146.

Brachman, R. J., and Schmolze, J. G. (1985). "An Overview of the KL-ONE Knowledge Representation System." *Cognitive Science,* 9(2).

Carbonell, J. G., and Hayes, P. J. (1983). "Recovery Strategies for Parsing Extragrammatical Language." *American Journal of Computational Linguistics,* 9(3–4):123–146.

Chodorow, M. S., Ravin, Y., and Sachar, H. E. (1988). "A Tool for Investigating the Synonymy Relation in a Sense Disambiguated Thesaurus." In *Proceedings of the Second Conference on Applied Natural Language Processing,* pp. 144–152. ACL.

Clark, H. H. (1975). "Bridging." In *Theoretical Issues in Natural Language Processing,* pp. 169–174. ACL.

Crowther, W. (1989). "A Common Facts Data Base." In *Speech and Natural Language,* pp. 89–93. Morgan Kaufmann Publishers Inc., San Mateo, CA.

Eastman, C. M., and McLean, D. S. (1981). "On the Need for Parsing Ill-Formed Input." *American Journal of Computational Linguistics,* 7(4):257.

Finin, T. W. (1980). "The Semantic Interpretation of Nominal Compounds." In *Proceedings of the First Annual National Conference on Artificial Intelligence,* pp. 310–312. The American Association for Artificial Intelligence.

Fromkin, V. A. (1973). *Janua Linguarum, Series maior 77: Speech Errors as Linguistic Evidence.* Mouton, The Hague.

Grosz, B., Appelt, D. E., Martin, P., and Pereira, F. (1985). "TEAM: An Experiment in the Design of Transportable Natural Language Interfaces." *Artificial Intelligence.*

Hinrichs, E. W., Ayuso, D. M., and Scha, R. (1987). "The Syntax and Semantics of the JANUS Semantic Interpretation Language." In *Research and Development in Natural Language Understanding as Part of the Strategic Computing Program,* Annual Technical Report December 1985–December 1986, pp. 27–31. BBN Laboratories, Report No. 6522.

Jensen, K., and Binot, J.-L. (1988). "Dictionary Text Entries as a Source of Knowledge for Syntactic and Other Disambiguations." In *Proceedings of the Second Conference on Applied Natural Language Processing,* pp. 152–159. ACL.

Jensen, K., Heidorn, G. E., Miller, L. A., and Ravin, Y. (1983). "Parse Filling and Prose Fixing: Getting a Hold on Ill-Formedness." *American Journal of Computational Linguistics,* 9(3–4):147–160.

Kaemmerer, W., and Larson, J. (1986). "A graph-oriented knowledge representation and unification technique for automatically selecting and invoking software functions." In *Proceedings AAAI-86 Fifth National Conference on Artificial Intelligence,* pp. 825–830. American Association for Artificial Intelligence, Morgan Kaufmann Publishers, Inc.

Learner User Manual, BBN Systems and Technologies.

Lenat, D., Prakash, M., and Shepherd, M. (1986). "CYC: Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks." *AI Magazine,* 6(4):65–85.

Montague, R. (1970). "Pragmatics and Intensional Logic." *Synthese,* 22:68–94.

Montgomery, C. A., and Glover, B. A. (1986). "A Sublanguage for Reporting and Analysis of Space Events." *Analyzing Language in Restricted Domains: Sublanguage Description and Processing,* pp. 129–162. Lawrence Erlbaum Associates.

Neal, J., and Walter, S. (editors). (1991). *Natural Language Processing Systems Evaluation Workshop,* Rome Laboratory.

Neff, M. S., and Boguraev, B. K. (1989). "Dictionaries, Dictionary Grammars, and Dictionary Parsing." In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics,* pp. 91–101. ACL.

Parlance User Manual, BBN Systems and Technologies.

Pavlin, J. and Bates, R. (1988). *SIMS: Single Interface to Multiple Systems.* Technical Report ISI/RR-88-200, University of Southern California Information Sciences Institute.

Resnik, P. (1989). *Access to Multiple Underlying Systems in Janus.* BBN Report 7142, Bolt, Beranek and Newman Inc.

Rich, C., and Waters, R. C. (1988). *Automatic Programming: Myths and Prospects.* MIT Press.

Schank, R., and Abelson, R. (1977). *Scripts, Plans, Goals, and Understanding.* Lawrence Erlbaum Associates.

Shieber, S. M. (1986). *An Introduction to Unification-Based Approaches to Grammars.* Center for the Study of Language and Information, Stanford, CA.

Stallard, David. (1987). "Answering Questions Posed in an Intensional Logic: A Multilevel Semantics Approach." In R. Weischedel, D. Ayuso, A. Haas, E. Hinrichs, R. Scha, V. Shaked, D. Stallard (editors), *Research and Development in Natural Language Understanding as Part of the Strategic Computing Program,* pp. 35–47. BBN Laboratories, Cambridge, Mass. Report No. 6522.

Thompson, B. H. (1980). "Linguistic Analysis of Natural Language Communication with Computers." In *Proceedings of the Eighth International Conference on Computational Linguistics,* pp. 190–201. International Committee on Computational Linguistics.

Weischedel, R. M., and Sondheimer, N. K. (1983). "Meta-rules as a Basis for Processing Ill-Formed Input." *American Journal of Computational Linguistics,* 9(3–4):161–177.

Weischedel, R., Ayuso, D., Haas, A., Hinrichs, E., Scha, R., Shaked, V., Stallard, D., (editors). (1987). *Research and Development in Natural Language Understanding as Part of the Strategic Computing Program.* Technical Report, BBN Laboratories, Cambridge, Mass. Report No. 6522.

Weischedel, R. M. (1987). "A View of Ill-Formed Input Processing." In *Research and Development in Natural Language Understanding as Part of the Strategic Computing Program.* Technical Report, BBN Laboratories, Cambridge, MA. Report No. 6463.

Weischedel, R. M., Bobrow, R., Ayuso, D. M., and Ramshaw, L. (1989). "Portability in the Janus Natural Language Interface." In *Speech and Natural Language,* pp. 112–117. Morgan Kaufmann Publishers Inc., San Mateo, CA.

Weischedel, R. M. (1989). "A Hybrid Approach to Representation in the Janus Natural Language Processor." In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics,* pp. 193–202. ACL.

Weischedel, R. M., Carbonell, J., Grosz, B., Marcus, M., Perrault, R., and Wilensky, R. (1990). *Natural Language Processing,* Annual Review of Computer Science, Vol. 4, pp. 435–452.

Woods, W. A. (1970). "Transition Network Grammars for Natural Language Analysis." *Communications of the Association for Computing Machinery,* 13(10):591–606.