Working Paper Series
ISSN 1170-487X

Natural language processing in speech understanding systems

by Dr Geoffrey Holmes

Working Paper 92/6

September, 1992

Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

Natural Language Processing In Speech Understanding Systems

G. Holmes

Department of Computer Science, University of Waikato, New Zealand

Overview

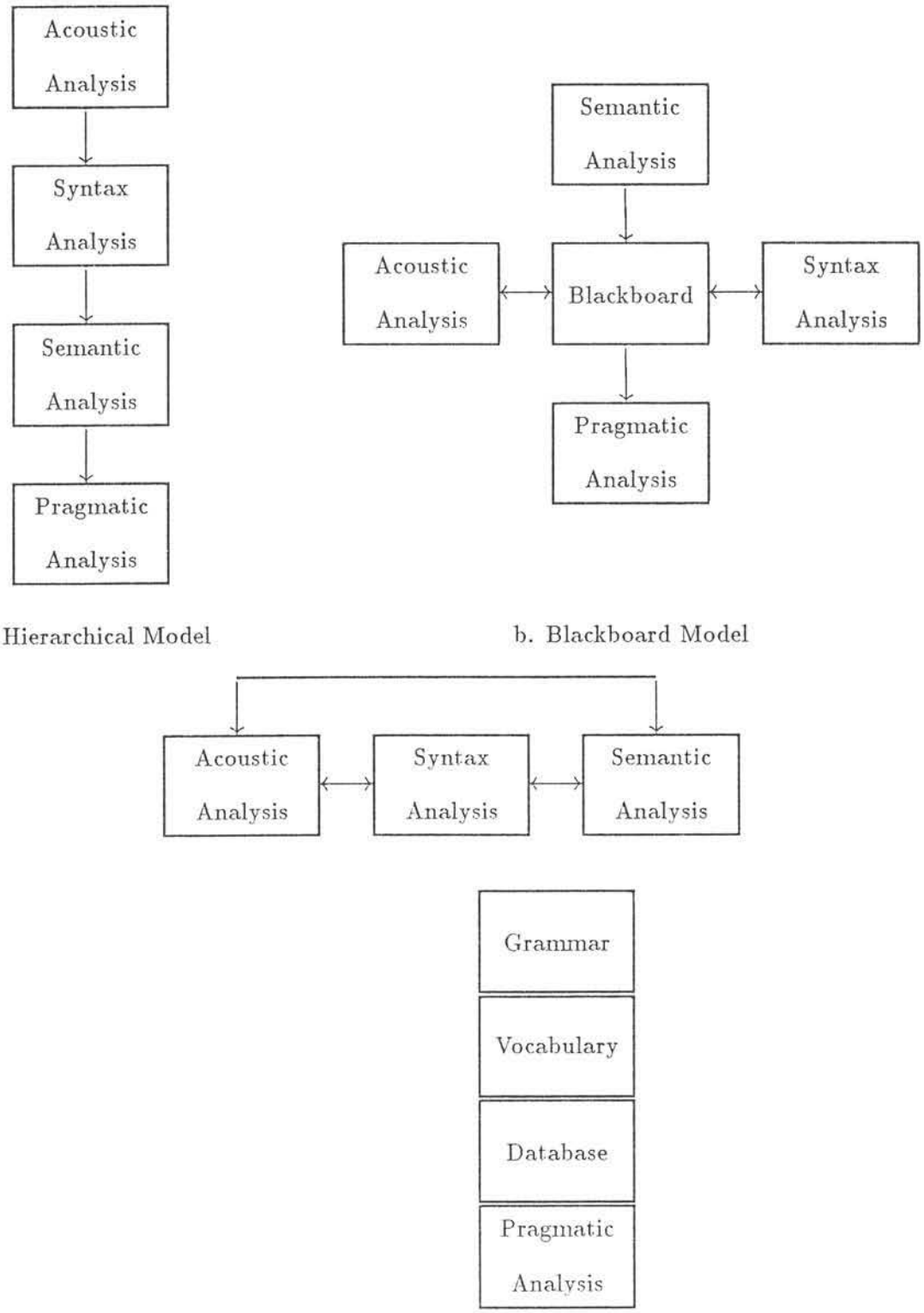
Speech understanding systems (SUS's) came of age in late 1971 as a result of a five year development programme instigated by the Information Processing Technology Office of the Advanced Research Projects Agency (ARPA) of the Department of Defense in the United States. The aim of the programme was to research and develop practical man-machine communication systems. It has been argued since, that the main contribution of this project was not in the development of speech science, but in the development of artificial intelligence. That debate is beyond the scope of this paper, though no one would question the fact that the field to benefit most within artificial intelligence as a result of this programme is natural language understanding. More recent projects of a similar nature, such as projects in the United Kingdom's ALVEY programme and Europe's ESPRIT programme have added further developments to this important field.

This paper presents a review of some of the natural language processing techniques used within speech understanding systems. In particular, techniques for handling syntactic, semantic and pragmatic information are discussed. They are integrated into SUS's as knowledge sources.

The most common application of these systems is to provide an interface to a database. The system has to perform a dialogue with a user who is generally unknown to the system. Typical examples are train and aeroplane timetable enquiry systems, travel management systems and document retrieval systems.

Introduction

Typical architectures for SUS's are given in figure 1.



a. Hierarchical Model

b. Blackboard Model

c. Hybrid Model

Figure 1. Architectures of speech understanding systems

Whatever the architecture a SUS comprises a number of components or knowledge sources which interact in some way. Some systems have components not included in the figure and others separate off components, for example in fig.1c where vocabulary, grammar and parser are separate entities (they are represented as one entity, namely syntax in fig.1a and fig.1b). The components in the diagram above have been chosen as they are of relevance to this paper. Fig.1a represents the simplest form of interaction where communication is only possible between adjacent components. This model is totally data driven and is straightforward to control. In fig.1b communication is performed via a blackboard. Each component may take items off and put items onto the blackboard. Information as to a possible interpretation of an utterance is built up in the form of a database of knowledge from the various sources surrounding the blackboard. The control of searching this accumulated knowledge base is a major problem with this particular architecture, but even so, it has been used successfully in a SUS [Erman and Lesser, 1980] The architecture of fig. 1c is a hybrid of these two and is more typical of SUS in general (it is a version of a SUS developed by AT&T [Levinson, 1985]).

Recognition Component

The SUS's discussed in this paper are all recognisers of continuous (connected) speech by persons known or unknown to the system. The continuous aspect of the speech forces the recognition component to perform a segmentation of the unknown utterance into smaller units (speech tokens) such as whole words, syllables or even smaller phonetically defined units (phonemes) prior to actual recognition [Vaissière, 1985]. The choice of speech token on which to base recognition has a large impact on the processing required within the system. If tokens smaller than whole words are chosen then a process is needed which encodes the way such tokens combine to form words. It is frequently the case that systems generate a 'phonetic lattice' which is passed on from the recognition component to a lexical analysis component (another knowledge source).

The recognition of speech tokens can be performed in a variety of ways. The most popular methods are template matching using dynamic programming (often referred to as Dynamic Time Warping (DTW) [Bridle and Brown, 1979]) and a statistical approach using Hidden Markov Models (HMM's) [Levinson *et al*, 1983]. Both methods operate in one of two modes, a training mode and a recognition mode. The speech tokens are 'created' by extensive training using speakers from a

wide variety of populations to achieve storage of speech patterns corresponding to as general a speaker as possible. The tokens are stored using a parameterisation of the speech which maximises the acoustic content and minimises the amount of data needed to represent the token. Once training is complete the recognition component is switched to recognition mode where it performs segmentation and pattern association on sentences uttered by a (usually) unknown speaker.

The choice of speech token will determine the vocabulary size of the system. The systems have to store their 'knowledge of sounds' somewhere in memory. Speech data is extremely expensive in terms of computer storage even after its essential features are extracted (the process of parameterisation). If whole words are chosen as tokens and the system is expected to respond in real time then the system must be limited in vocabulary size (1000 words is average). If phonemes are chosen then, in English for example, only about 70 tokens need to be stored. The necessary theories of how phonemes combine in different contexts and how they change with different stresses placed on them, can be added in cheaper (storage wise) software. The phonemes can then be linked to a phonetic dictionary containing possibly thousands of entries, enough to cover very general conversations.

Currently, however, recognition accuracy for whole words is considerably better than recognition accuracy for any sub-division of whole words [Lea, 1980]. Research has tended to focus on word sub-divisions for their obvious storage advantages and less restrictive applications. If the application is small enough, however, real time systems can be built to perform useful tasks employing whole word tokens. Most commercial recognisers fall into this category.

Recognition need not be a stand alone process within a SUS. It is often assisted by other components. These influences are often tied to the natural language aspects of the system. In terms of natural language processing most work has centred on the syntactic, semantic and pragmatic components. These will be discussed in turn.

Syntax Component

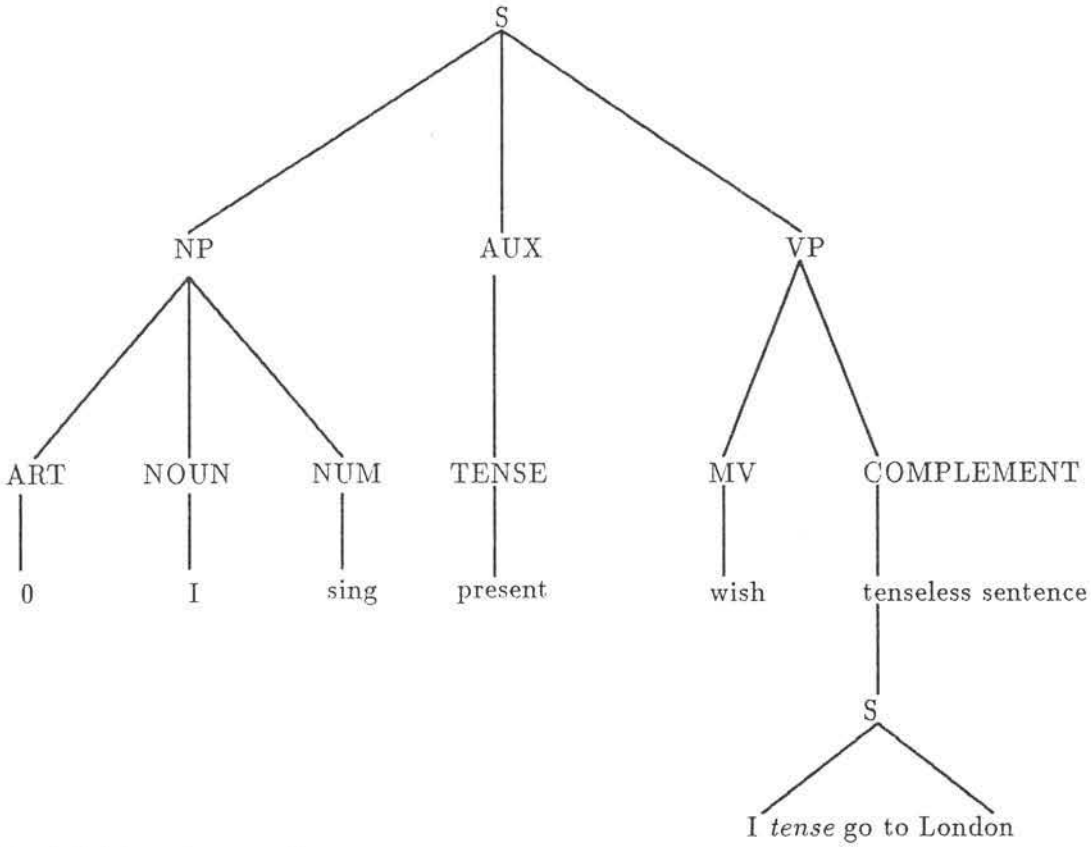
Given a problem domain for a SUS such as a train timetable inquiry system then the syntax component needs to work with a dictionary and a grammar for that domain. One approach to the problem of establishing a grammar and dictionary is to record (over the telephone) inquiries made to an information desk. The activity of the person at the desk is, after all, what the system is trying to model. The words and grammar used in the converstaions can then be transcribed

into syntax rules (lexical and grammar rules). Experiments of this nature have been conducted to include simulated inquiries where the inquirer believes they are speaking to a computer (a voice synthesiser is enough to fool most people). The results are encouraging for SUS's. When addressing computer simulations people tend to speak *more* grammatically than when addressing a fellow human.

This empirical approach generates more explicit syntax rules than those used for general syntactic processing. For example, in the train timetable domain a popular request is :

```
User :  I want to go to London
```

This particular example (in a text understanding system, for example) might require the determination of the subject, auxiliary, main verb, object etc. of the sentence. Further the example involves the notion of an embedded sentence whose subject noun phrase is the same as the subject noun phrase of the overall sentence, i.e. 'I' does both the wanting and the going. To handle this properly, transformational rules, such as a rule for subject noun phrase deletion, and one for ensuring that the tense of the embedded sentence comes to the surface as an infinitive phrase, are required. Thus, the sentence must be analysed for both its deep and surface structure. The syntax tree produced by such analysis is indicated in figure 2.



by infinitive phrase rule: to

by subject-NP-deletion rule: 0

by flip-flop rule: wish - present

want

Figure 2 - Syntax tree for *I want to go to London*

In a SUS this type of sentence is more likely to be determined by using a context-free grammar such as :

```
query       = request togo where
request     = I want | I would like | I wish
togo        = to go
where       = toplace | fromplace
toplace     = to someplace
fromplace   = from someplace
someplace   = LONDON | CAMBRIDGE | MANCHESTER ...
to          = TO | TOO | TUH ...
...
```

These syntax rules are written in a Backus Normal Form where the symbol | indicates options for the rule on the left hand side. Notice that the terminal symbols (uppercase letters) occuring in the lexical rules may take a number of alternative forms. The example above assumes a whole word recognition component is being used. The different rewrite forms for the syntactic category *to* reflect the possible pronunciations that can be recognised by the system. These lexical rules define a *semantic template grammar*. The meaning of the word used as a recognition template is intrinsic. The templates are said to define meaningful categories and, therefore, require no further semantic analysis.

It is generally sufficient within a SUS to use a context free grammar. Context sensitivity issues can then be resolved using syntactic restrictions, for example checking for plural agreement and tense agreement for verbs. For efficiency reasons this method is preferred over using a context sensitive grammar from the onset. Moreover, these restrictions can be waved in favour of recognising the intention of an utterance rather than worrying about the correct use of tense and plurals.

As with the software life cycle of an expert system, syntax rules are subject to an ongoing test and modify cycle. If the rules are insufficient to capture the likely utterances at a particular stage in a dialogue then they are updated and new rules (and possibly new words to be recognised) are added. Much of this syntax analysis is hand crafted though techniques have been developed to assist the process [Goodman, 1976].

The role of a syntax component is to check that an utterance fits its syntax rules. If it does, then a suitable representation of that utterance (parse tree) is passed on for further analysis. If no complete sentence may be formed then a partial representation (sub-tree) containing the most promising words may be passed on.

Some systems use their syntax components for other purposes than generating parse trees. They can be used, for example, in close association with the recognition component to predict likely follower words [Wolf and Woods, 1980]. They can also be used to build up semantic representations while parsing using ATN's [Woods, 1985].

Semantic Component

The arguments over the roles of syntax and semantics in natural language processing are not as contentious in SUS's mainly due to their subordinate position with respect to the recognition component. In the ARPA project, the semantic components were either completely separate from the syntax components, or they were combined with the syntax rules into one large network. In both approaches each syntax rule was associated with one or more semantic rule(s). More recent systems [Young *et al*, 1989 and Levinson, 1985] (for an example of a typical SUS dialogue see [Levinson, 1985, pp 269]) perform semantic analysis through a dialogue system.

To a large extent the meaning of words and sentences within a SUS is defined *a priori*, certainly in the case of semantic template grammars. The intention behind many of the requests in a SUS do not require large scale analysis. Consequently, the meaning of individual words or groups of words are not normally modelled as they might be in a text understanding system. As with the syntax component, a major objective of a SUS is to obtain real time performance, and so efficiency has tended to dominate over forming semantic interpretations of syntax trees. Interleaving semantic processing with syntax analysis has, however, been achieved in a SUS [Wolf and Woods, 1980] with the system operating at not quite real time. The general approach is to separate the processes and perform minimal analysis.

Semantic processing is needed to reject syntactically valid yet meaningless sentences (since most systems generate a number of alternative words, phrases and sentences these are likely to occur), for example, "the 30th of February" and "to London from London". In the context of an ongoing dialogue, the semantic component is essential in deciding what meaning is intended in exchanges with the user. For example, the intention in the request for "all trains to Cambridge" is substantially different from that of "a train to Cambridge". The system has to respond appropriately, in the first case with a list of trains, in the second with a further question such as "In the morning or afternoon ?" It is worth noting that syntax analysis does not help the dialogue progress effectively

in any of these examples.

The information received by the semantic process is unreliable. A users intended course of action can as a result be difficult to obtain, consider the following exchange :

```
User   :  I want to go to Lincoln
System :  So you want to go to London
User   :  No
```

No meaning can be obtained here due to mis-recognition of the place name. To resolve the problem it may be necessary to repeat the earlier destination request or to run through the list of alternative place names recognised by the system. The semantic process must indicate that a problem has occurred and must direct the conversation to resolve the problem.

Semantic processes can be implemented using production rules to check for inconsistencies, for example :

```
IF Rule is where THEN to_place <> from_place

IF X is a date AND a field of X is February THEN that field must be
a number in the range 1 to 29.

IF Request is morning AND Current Time > 12 THEN something is wrong
```

When "something is wrong" the system must take some action. If this fails to produce acceptable results then a graceful degradation is required. The system must report that the user's intention is not clear and must try again (using some pre-defined strategy) to resolve the misunderstanding.

Production rules are not the only way of implementing semantic analysis. Another method, adopted in [Young *et al*, 1989] and [Levinson, 1985] is to use a frame language. For example, in the experimental language UFL a where frame may be defined to request the source and destination information and then check its consistency :

```
FRAME place :  enum
(London, Manchester, Cambridge, ...)

FRAME where
(ako standard,
* goingto   :  place,
* goingfrom :  place,
   ( check :  required(goingto <> goingfrom),
   if_inconsistent :  proc
   (output('You cannot travel on the spot, please try again'),
   ^where.destroy
   ),
   )
if_needed ask
)
```

Exact details of this can be found in [Young and Proctor, 1986]. This example is not complete but serves as an illustration of the general principles. Broadly then, when this frame is instantiated the *if_needed* slot is executed and the user is asked where they want to travel to and from. A value frame will be returned, (in general this will be a parse tree from the syntax component) for example (London, London). The values of this frame will be written into the *goingto* and *goingfrom* slots, and the *check* function will be executed. If the Boolean value of the *required* frame is false (as in this case) the *if_inconsistent* slot will be executed. A message is output to the user, the instance of this *where* frame is destroyed and the *if_needed* procedure is called to try to solve the inconsistency. The strategy for doing this can be encoded in the *ask* frame of the system.

Frame languages provide facilities for controlling semantic processing and dialogue flow. Notice in the above example that since the frame system is controlling the flow of questions to the user, it assumes that the user's replies conform to the syntax rules of the system. By having control of the dialogue the immediate context of the conversation (the syntax rule *where* in the above example) is known to the frame system and may be passed to the recognition component to improve recognition accuracy. In the case of misunderstandings the frame system can decide to enter a 'yes or no' mode of operation, as in :

```
System :  Do you want to go from London (please answer yes or no)
User   :  Yes
System :  Where are you travelling to ?
User   :  Lincoln
```

The frame structure also provides an elegant framework for storing the general context of a dialogue. This is the subject of the next section.

Pragmatic Component

In a conversation between two individuals information is provided and stored by both parties. Replies to questions can contain assumptions of the problem domain, and may serve to restrict future conversation.

Assumptions made in a conversation reduce the need to repeat information. For example, in the train timetable domain the system must be able to cope with the following types of request :

```
User :  When is the next one?
User :  Does it stop at Cambridge?
```

This type of referencing using *it* and *next* is known as anaphoric reference. Any system which engages in a meaningful dialogue with a user must be able to understand anaphoric references. The context of the conversation must, therefore, rest somewhere in the system.

Forward references can constrain the context of future dialogue exchanges and database searches, examples of these are :

```
User :  I want to travel in the morning
User :  I want to get there quickly
```

Contextual information then, may be specific, back referencing or forward referencing. As noted above, specific context aids recognition performance and guides conversational flow. It is vital, for example, for a system to gain control of the conversation from the start:

```
System :  Good morning, where do you want to go to?
```

is better in this respect than:

```
System :  Good morning, can I help you?
```

The opening question serves two purposes. Firstly, the question invites an 'Eliza' type [Weizenbaum, 1966] response where the words in the question are used in the reply :

```
User :  I want to go to London
```

Secondly, the context of the expected reply can be assumed to be that of the rule 'query'. The recognition component can expand (into a network) all the terminal symbols associated with the right hand side of the query rule and await a suitable response.

Frame based approaches to dialogues involve the concept of an inheritance hierarchy (in the UFL example of the previous section this is achieved through the *ako* (a kind of) slot). Frames can contain other (sub)frames, and they may contain an instance of their own frame. This structure forms an hierarchy into which values may be placed in an ongoing dialogue. The hierarchy is an environment for the global context of a conversation, and may be viewed as a large data structure. Operations can be applied to this data structure to resolve anaphoric reference. The development of the hierarchy can be constrained by exchanges which have a limiting effect on future discourse, as in the example above.

Pragmatic information may be implemented in production rules, finite state networks (including ATN's) and databases as well as frame languages. Each approach attempts to build a model of

what the user understands, believes to be true and assumes. In a production rule system, the model is constructed from rules of the form :

```
IF Reply => speaker holds P true AND it is known
that P is false THEN something is wrong with Reply

IF Reply = No AND Previous_Reply = Yes THEN abort Question
```

Databases are used to store asserted and inferred knowledge gained from the speaker in the course of the dialogue. This approach closely resembles that used in the development of user interfaces for knowledge based systems.

In a network approach, the states of the network represent possible stages of a dialogue [Hayes-Roth *et al*, 1977]. The finite-state network which models the dialogue is updated after successive exchanges with the user.

Conclusion

Natural language processing in SUS's has been largely subject to the 'empirical' approaches which have been the hallmark of many knowledge based and expert systems. The focus of research in natural language processing in this area has been in the support (by syntactic, semantic and pragmatic components) of the recognition component of the system. The roles played by such components in human speech comprehension is still unknown. Research in fundamental concepts of speech production and perception [Marslen-Wilson, 1985] (experimental psycholinguistics) continues to provide a major focus for future work in this area.

One of the main contributions made by the ARPA, ALVEY and ESPRIT projects has been the 'bringing together' of researches from various disciplines (phoneticians, linguists, engineers, computer scientists, etc.) to work alongside each other developing systems. A multi-disciplinary approach is the only way to provide a firm basis for developing more reliable systems for the future.

References

BRIDLE, J.S. & BROWN, M.D. (1979) Connected word recognition using whole word templates. In *Proc. Inst. Acoust. Autumn Conf., Windermere, U.K.*, pp 25-28.

ERMAN, L.D. & LESSER, V.R. (1980) The Hearsay-II speech understanding system: A tutorial. In *Trends In Speech Recognition* (ed W.A. Lea), Prentice-Hall Signal Processing Series.

GOODMAN, G. (1976) Analysis of languages for man-machine voice communication. *Tech. report, Dept. Computer Science, Carnegie-Mellon University, Pittsburgh, PA.*

HAYES-ROTH, F. GILL, G. & MOSTOW, D.J. (1977) Discourse analysis and task performance in the Hearsay-II speech understanding system. *Tech. report, Dept. Computer Science, Carnegie-Mellon University, Pittsburgh, PA.*

LEA, W. A. (1980) Speech recognition: Past, present, and future. In *Trends In Speech Recognition* (ed W.A. Lea), Prentice-Hall Signal Processing Series.

LEVINSON, S.E., RABINER, L.R. & SONDHI, M.M. (1983) An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition. *Bell Systems Technical Journal*, **62**, pp 1035-74.

LEVINSON, S.E. (1985) A unified theory of composite pattern analysis. In *Computer Speech Processing* (eds F. Fallside and W. A. Woods), Prentice-Hall.

MARSLEN-WILSON, W.D. (1985) Aspects of human speech understanding. In *Computer Speech Processing* (eds F. Fallside and W. A. Woods), Prentice-Hall.

VAISSIÈRE, J. (1985) Speech Recognition: A tutorial. In *Computer Speech Processing* (eds F. Fallside and W. A. Woods), Prentice-Hall.

WEIZENBAUM, J. (1966) ELIZA - a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, **9**, pp 36-45.

WOLF, J.J. & WOODS, W.A. (1980) The HWIM (Hear What I Mean) speech understanding system In *Trends In Speech Recognition* (ed W.A. Lea), Prentice-Hall Signal Processing Series.

WOODS, W. A. (1985) Language processing for speech understanding. In *Computer Speech Processing* (eds F. Fallside and W. A. Woods), Prentice-Hall.

YOUNG, S.J. *et al* (1989) VODIS (Voice Operated Database Inquiry System). A speech understanding system developed at Cambridge University, U.K. To appear in ALVEY directorate reports.

YOUNG, S.J. & PROCTOR, C. (1986) UFL: An experimental frame language based on abstract data types, *Computer Journal*, **29**, pp 340-347.