IEEE Region 10 Conference on Computer and Communication Systems, September 1990, Hong Kong

Towards a speech oriented programming environment

Claudio Bettini
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
Via Moretto da Brescia, 9
Milano, ITALY

Steven Chin
IBM Corporation
Data System Division
Dept. 48B/428
Neighborhood Road
Kingston, NY 12401

Abstract: This paper presents first results from a research project designed to improve the man-machine interaction and in particular to utilize speech recognition and natural language interfaces in the programming environment of the LCAP/3090 parallel supercomputer. An outline of the whole project is given and two specific applications being developed are described: scientific text dictation and a speech understanding system for Fortran debugging. The first application is based on the Tangora speech recognition system, developed at the IBM T.J. Watson Research Center. Research in the debugging project is focused particularly on the natural language module.

1. Introduction

The LCAP/3090 system is the most recent member of the LCAP family of parallel supercomputers and is the result of research on parallel computing and scientific applications in the SEC department [4]. This computing system is designed to respond to the pressing needs for super-computing power for the simulation of complex physical systems; the scalar, vector and parallel paradigm is one solution to the inherently complex structure of problems involved in the global simulation approach.

The system can be considered as a prototype for future generation computing system and, to take advantage of its capabilities, the programming environment is under consideration to be enhanced with more friendly man-machine interfaces, new programming tools (debuggers, context-sensitive editors, interactive tools for help in vectorization and parallelization) and sophisticated graphics representation systems.

This paper presents results from a research project on man-machine interaction (see [1]) and in particular about speech recognition and natural language interfaces as necessary components in the programming environment of a future generation computing system.

The goal of the SOPE project (Speech Oriented Programming Environment) is to develop speech-driven interfaces between the users and the LCAP/3090 parallel supercomputer. The speech is intended to be used together with graphic interfaces to provide input from the user to common programming tools such as editors and debuggers as well as to application programs. Since the goal is to provide a speech system able to recognize not only commands, but also actual English sentences, some research efforts have been tailored towards natural language processing and large vocabulary speech systems. We decided to work first on scientific text dictation as an experimental application of the Tangora system. This involves customizing the speech system and possibly coming up with a powerful tool for preparing documentation [5].

2. The Tangora speech recognition system

The IBM speech recognition system is based on the use of Hidden Markov Models (HMM) and is described in various papers [6, 8, 9]. It uses artificially constructed, statistical word models as basic elements for speech recognition, which are trained with the help of a mathematical algorithm. The two main modules of the system are the acoustic processing module and the linguistic decoder. Figure 1 shows a simple block diagram of the system.

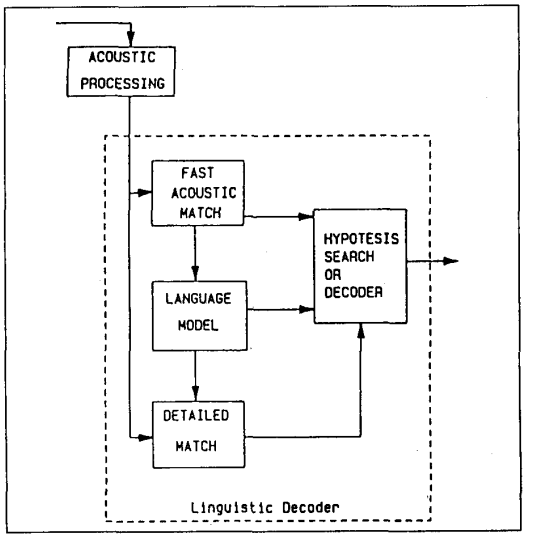

Figure 1. Block diagram of the IBM speech recognition system.

The function of the acoustic processing module can be in some way compared to the human ear: certain acoustic features are calculated from the speech signal and it is finally transformed into a sequence of labels, which can be processed by the linguistic decoder. These labels (or acoustic prototypes) are determined by clustering of the acoustic training data during the training session. The decoder has access to the statistical word models described by the transition and output probabilities of the Hidden Markov Models for the characteristic speech sounds. These probabilities are obtained from a training session in which the resulting label streams of several sentences of human speech are evaluated. The task of the linguistic decoder is to reconstruct the spoken word sequence from the label stream it receives from the acoustic processor. It selects possible word candidates with the help of the *Fast acoustic match* module, with the language model determining which of these words are likely to occur in the given context, and the final candidates are given by the *Detailed match* module; in this way it determines the

word sequence that has the highest probability of being produced by the corresponding acoustic label stream.

The current language model considers as context just the candidate word and the previous two words. It uses a database which stores all the number of occurrences of trigrams (sequences of three adjacent words), bigrams (sequences of two adjacent words) and unigrams (the simple word) calculated from a very large sample text (tens of millions words).

$$P(w_3 \mid w_2, w_1) = \lambda_3 \, f(w_3 \mid w_2, w_1) + \quad (1)$$
$$+ \lambda_2 f(w_3 \mid w_2) + \lambda_1 f(w_3) + \lambda_0$$

For each candidate word the language model computes the conditional probability of word w3 assuming that the two previous words w_2 and w_1 are correctly recognized. In (1) λ_i are weights set during the construction of the language model and are dependent from the database, while the f( ) are respectively frequences of trigrams, bigrams and unigrams. The candidate words are ordered according to these probabilities and sent to the hypothesis search and to the detailed match.

The hypothesis search controls the decoding and maintains several possible sequences of words ordered in decreasing order of a function L(W) [10].

This function depends on the estimates given by the three modules described above for each word in the sequence. At each step only the sequences with L > D. (threshold) are expanded with as many branches for each sequence as the number of candidate words given by the fast acoustic match. The words and contexts are sent to the language model and then to the detailed match, in this way updating the values of L. The algorithm used is called fast sequential decoding and is described in [7].

We can classify Tangora as a large vocabulary, speaker dependant, isolated-utterance, real time speech recognition system.

Based on the Tangora system, several speech systems for different languages were built: one for Italian [11], one for German, and there is on-going work for French and Spanish.

3. Scientific text dictation

In order to customize the speech system for scientific text dictation the current Tangora vocabulary has to be updated to include all of the words specific to the scientific field of interest. These fields are, for the SEC department, chemistry, molecular dynamics, parallel processing, etc.. The original 20,000 word Tangora vocabulary and the language model were built based on tens of millions of words taken from IBM internal electronic mail. Our scientific language domain requires a smaller set of common English words, but several new technical words. Updating the vocabulary involves building new statistical word models for each new word. In addition the language model has to be completely reviewed because the vocabulary has changed and the context in which words appear is also different.

The first result of the project is the compilation of a new vocabulary through statistical analysis of a huge amount of text related to the specific fields. This required the collection and cleaning of over one million words of electronically stored, specialized text. In this context, cleaning means erasing formatting commands and replacing objects like formulas, graphics and tables with constant values representing object categories. For example, if the text defining an equation is removed, this text should be substituted with a symbol (e.g. <EQUATION>) to signify that an equation is present. It is important that the sentences still maintain their structure and meaning and that the context does not change. Punctuation and certain formatting commands are also important (for example to signify new sentences and paragraphs) and must be retained. This process has been accomplished by developing specific software tools.

Here we present same results obtained through the analysis of 750,000 words of cleaned text. These results indicate that a reasonable vocabulary coverage can be obtained by using a relatively small number of words in the vocabulary.

The auto-coverage test shows the coverage of the full text as a function of the most frequent words.

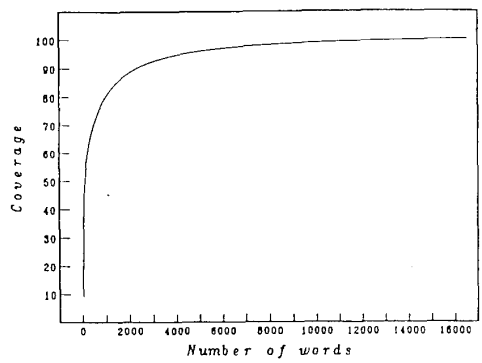

Figure 2. Auto-coverage for the 750.000 words text.

This function shows that the first 1,000 most frequent words cover up to 70% of the whole text, the first 3,000 cover 93% and that, to arrive to 98%, that is our minimum requirement, 7,000 words are needed. This result is typical of technical text.

Extracting a vocabulary entirely from the sample text guarantees a fixed coverage on that text, but what happens when trying to recognize a different text on the same subject? Cross-coverage tests generally answer to this kind of questions. Cross-coverage is performed by extracting a vocabulary from a sample text and then using it to analyze the coverage over a different text. In this way we can gain an understanding of the generality of the vocabulary within the subject. The first cross-coverage test was done using the original 20,000 words vocabulary against our sample text (750,000 words). Of course, the vocabulary did not have many of the scientific words and the cross-coverage was 89%. This value shows that the origianal vocabulary is not appropriate, but that value is still surprisingly high if we consider that the vocabulary was done for a different subject. More significant cross-coverage tests were taken using disjoint texts from our specific sample text. This was accomplished by extracting a vocabulary from one part of the sample text* and calculating the cross-coverage on the remainder. The results were fairly good, arriving to 96.5%. An example of a cross-coverage plot is shown in the figure 3; the solid line is the auto-coverage (over the first text), while the dashed line is the cross-coverage; for the cross-coverage function (dashed line), abscissa values are the size of the vocabulary.

* This text is an heterogeneous collection of papers and books.

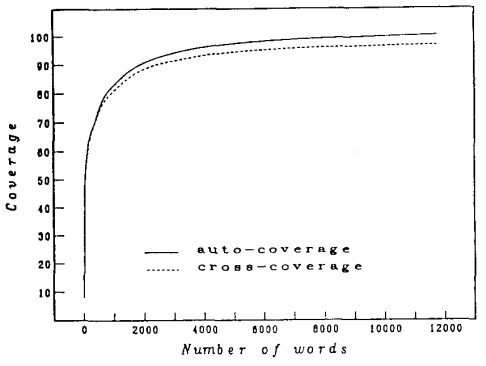

Figure 3.Cross-coverage between two disjoint collections of papers.

5,000 of the 7-8,000 words selected during these tests were already present in the 20,000 words vocabulary. The final vocabulary for our application has 2,500 new words replacing the same number of words from the original vocabulary. The words which have been removed are chosen among those with frequency 0 in our sample text and lowest frequency in large common English texts. This vocabulary guarantees 98-99% coverage over typical texts that the user will dictate.

To have this vocabulary working we still have to derive the statistical word models; this will be done recording 10 speakers uttering sentences containing the new words. Statistical models for each word are then computed from these data by existing software based on mathematical algorithms and phonetic rules .

Some work has also been done for the new language model, mainly categorisation and a first calculation of trigrams, bigrams and unigrams occurrences. There are still many incorrect and incomplete sentences in the text due to the automatic cleaning process and this will require further refinements on the sample text. Moreover storage optimization requires words and categories to be mapped into numbers (this process is called tokenization) and lambda values used in (1) have to be computed.

Based on experiments with the original Tangora system and considering the results of tests previously described, we estimate the average error rate of the scientific dictation system to be around 5% for first prototypes. Speakers of American English are assumed. The system is real time for most speakers in the sense that it takes less elapsed time to recognize what has been said than it does to say the words. Some speech driven editor commands are also provided.

A problem that stil needs to be resolved is how to insert mathematical expressions, tables and figures. It looks more reasonable, in this case, that different input tools other than speech (e.g. tablets to write formulas) should be used. There is on-going work in interfacing tablets prototypes with the speech system, but until these tools will be available the dictation has to be integrated with the typing of mathematical expressions and anything else that is not plain text.

4. Speech Understanding: an Interface for Debugging

In a speech understanding system the speech recognition module has to be integrated or interfaced with a comprehension module to translate the spoken sentence into an internal representation that uniquely identifies its meaning. Our approach is to interface different modules to create a debugging tool that will be driven by speech and will enable the user to formulate his requests in a very friendly way, speaking English sentences instead of typing syntactically complex commands. Figure 4 shows the functional structure of the system.

The speech recognition system recognizes the spoken sentence and its output, a printed sentence, is sent to the English Language Debugger Interface (ELDI). The ELDI is the actual comprehension module and it has to generate syntactically correct commands for the debugger to satisfy the user request. The debugger that we interface to, is the FORTIAD (VS Fortran Interactive Debugger) running on VM . The project is in its infancy; to this point most of the efforts have been concentrated on the ELDI interface. Some work has been done on speech recognition problems that arise in these kinds of applications and on the interfaces between the modules involved.

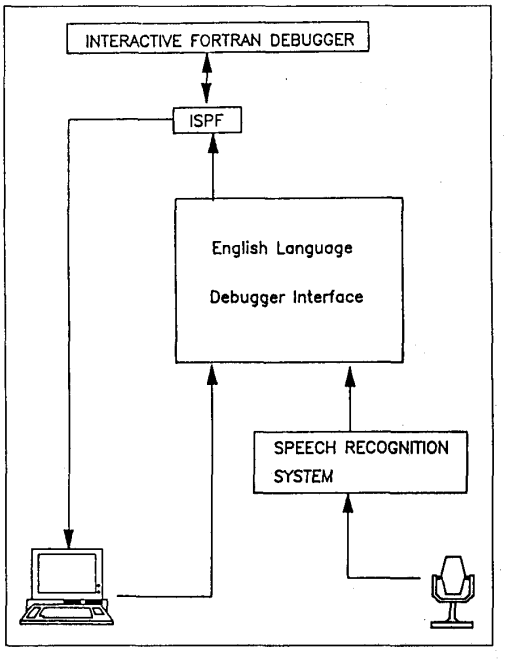

Figure 4. Block diagram for the speech driven debugging system.

4.1. The ELDI system.

The ELDI prototype is based on *Augmented Transition Nets* (ATN) [3] [2] and is running on the IBM RT PC. The activities of parsing and generating commands are separated and the sentence is transformed by parsing in an intermediate internal representation.

Requests can be formulated in an English-like language with a particular grammar. This version of the system has quite a strict grammar.

Let's give a look to some example of requests.

Command requests:
-show me the variables i, j, k at instruction 56
-list array energy1 from component 30 to 100
-go in subroutine actfor

Questions:
-which breakpoints are on ?

Declarative requests:
-the variable x45 has to be between -1 and 1
-the instruction 25 must be executed at most 50 times

The main difference compared to using a formal language is flexibility: in the first command request, the user could also use a different expression (e.g. list values of i, j, k .....) obtaining the same desired behaviour from the system.

After each request is processed, the system gives an output on the screen. Up to now, the output from the system is just a string representing the stream of commands to be sent to the debugger in order to perform the action requested by the user.

The parser is able to recognize different structures (and therefore meanings) for the same sentence if there is ambiguity in the grammar (and this is usually the case for natural languages). Up to now we have a restricted *debugging vocabulary* (60-70 entries).

Although interfacing with the speech recognition system will be part of a next project, a feasibility study about the project of customizing Tangora to recognize the debugging language that we defined was made. In this case Tangora is not the most suitable speech system since it was specifically designed to work with a large vocabulary; moreover a language model would probably perform better if based on the ATNs syntax rules than on statistics.

The most difficult problem is in the heavy use of variable and subroutine names in the sentences. These are not words, and cannot be easily included in the vocabulary. The alternatives are spelling each name using the *Spellmode* facility or integrating the speech with some pointing device (tablet or mouse) to identify those names. The error rates for the use of Spellmode are usually higher than for the Normal mode. Actually, very short sounds are more difficult to recognize than long ones. For this reason is very difficult to recognize "i" from "y". On the other hand, spelling names of variables correctly is critical in programming or debugging. One solution is to use graphic and pointing devices coupled with speech.

5. Conclusions

The first results on the application of speech and natural language interfaces in a research and programming environment are described. A lot has been learned about specific problems of acoustic modelling, language modelling and interfacing; moreover, experience has been gained in customizing the Tangora speech recognition system for specific applications. Both the project of scientific text dictation and English language debugging still require work to be completed and to be actually available to the users. Most likely these tools will not be productive and general enough until the technology for speech recognition improves. However, it seems clear that the future man-machine interfaces will be based strongly on speech.

6. References

[1] **C.Bettini, S. Chin**, *"Towards a speech oriented programming environment"*, IBM Research Report KGN-192, June 21, 1989.

[2] **T.W.Finin, T.Christaller, J.Laubsch, K.Barth, K.Kochut**, *"The design of interpreters, compilers and editors for Augmented Transition Networks"*, edited by L.Bolc, Springer-Verlag, 1983.

[3] **E.Charniak and D. McDermott**, *"Introduction to artificial intelligence"*, Addison-Wesley publishing company, 1984.

[4] **E.Clementi, D.Logan, J.Saarinen**, *"ICAP/3090: Parallel processing for large-scale scientific and engineering problems"*, IBM Systems Journal, Vol.27, n.4, 1988.

[5] **J.D.Gould, J.Conti, T.Hovanyecz**, *"Composing Letters with a Simulated Listening Typewriter"*, Communications of the ACM, vol.26, n.4, April 1983.

[6] **IBM Speech Recognition Group**, *"Experiments with the Tangora 20,000 word speech recognizer"*, Proc. 1987 IEEE, ICASSP, Dallas, 1987.

[7] **F. Jelinek**, *"A fast sequential decoding algorithm using a stack"*, IBM Watson Research Development, Vol. 13, pp.675-685, November 1969.

[8] **F. Jelinek**, *"The Development of an Experimental Discrete Dictation Recognizer"*, Proc. IEEE, Vol.73, p.1616, 1985.

[9] **Averbuch, Bahl, Jelinek, et al.**, *"An IBM PC based large vocabulary isolated-utterance speech recognizer"*, Proc.1986 IEEE Int.Conf. on acoust. spe. and sign. proc. (ICASSP), pp.53-56, Tokyo, Japan, April 1986.

[10] **L.R. Bahl, F. Jelinek and R.L. Mercer**, *"A maximum likelihood approach to continuous speech recognition""*, IEEE Trans. Pattern Analysis and Machine Intelligence, 1983.

[11] **P.D'Orta et al.**, *"Large vocabulary speech recognition: A system for the Italian language"*, IBM Journal Research and Development, vol.32, n.2., pp.217-226, March 1988.