STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025 · U.S.A.

August 1976

EXPERIMENTS IN SPEECH UNDERSTANDING SYSTEM CONTROL

William H. Paxton

Artificial Intelligence Center
Technical Note 134

SRI Project 4762

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and monitored by the U.S. Army Research Office under Contract No. DAAG29-76-C-0011. The system used in the experiments was developed jointly by Stanford Research Institute (SRI) and System Development Corporation (SDC).

August 1976

EXPERIMENTS IN SPEECH UNDERSTANDING SYSTEM CONTROL

William H. Paxton

Artificial Intelligence Center
Technical Note 134

SRI Project 4762

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and monitored by the U.S. Army Research Office under Contract No. DAAG29-76-C-0011. The system used in the experiments was developed jointly by Stanford Research Institute (SRI) and System Development Corporation (SDC).

EXPERIMENTS IN SPEECH UNDERSTANDING SYSTEM CONTROL

William H. Paxton

Artificial Intelligence Center
Stanford Research Institute
Menlo Park, California 94025

ABSTRACT

A series of experiments was performed concerning control strategies for a speech understanding system. The main experiment tested the effects on performance of four major choices: focus attention by inhibition or use an unbiased best-first method, "island-drive" or process left to right, use context checks in priority setting or do not, and map words all at once or map only as called for. Each combination of choices was tested with 60 simulated utterances of lengths varying from 0.8 to 2.3 seconds. The results include analysis of the effects and interactions of the design choices with respect to aspects of system performance such as overall sentence accuracy, processing time, and storage. Other experiments include tests of acoustic processing performance and a study of the effects of increased vocabulary and improved acoustic accuracy.

INTRODUCTION

This paper reports a series of experiments concerning control strategies for a speech understanding system.* The basic goal of the system was to perform a data-base management task using input in the form of continuous speech rather than isolated words, and simple English rather than an artificial command language.

One of the major problems in developing the system was to design a framework both to integrate the components and to provide an overall control strategy. For a speech system, the choice of a control strategy is particularly important because "false alarms," words incorrectly accepted as occurring in the input when they are not really there, present many opportunities to make mistakes. Rather than picking a particular control strategy, we designed the system framework so that it was possible to test the

* This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and monitored by the U.S. Army Research Office under Contract No. DAAG29-76-C-0011. The system used in the experiments was developed jointly by Stanford Research Institute (SRI) and System Development Corporation (SDC).

effects of several major design choices. The results of these tests are reported below. Details regarding the system framework are presented elsewhere (Paxton 1976 and forthcoming).

Several experiments were performed. Information regarding the acoustic processing was gathered in the first experiment. As well as being of interest in its own right, this information was used in simulating the acoustic processing for the other experiments. The second experiment dealt with the "fanout," both for the language alone and in combination with the acoustics. Fanout provides a quantitative measure of the difficulty of the task related to the average number of alternatives confronting the system. The third experiment measured the performance of two special test systems with extremely simple designs. In the fourth experiment, the main experiment of the series, the standard speech system was measured on a set of 60 test sentences for all combinations of 4 control strategy design choices. The performance of the best configuration from Experiment 4 was tested in the fifth experiment, allowing different sizes of gaps and overlaps between words in the simulated acoustic processing. The sixth and last experiment studied the performance of the most promising system configurations from Experiment 4, while varying vocabulary size and acoustic processing accuracy.

This paper will not discuss all of the experiments in detail. We will sketch the experiments and the main results with emphasis on the fourth experiment, concerning control-strategy design alternatives. A detailed discussion of the entire series of experiments will be given in Paxton (forthcoming).

EXPERIMENT 1--MAPPER PERFORMANCE

The first experiment deals with the performance of the system component called the "mapper" (described in Ritea, 1975). The mapper carries out acoustic tests. Given a predicted word and location in the input, the mapper either rejects the word, or accepts it and reports its beginning and ending boundaries rounded to the nearest 0.05 second. If the word is accepted, the mapper also gives it a score between 0 and 100, indicating how well it matches the input (100 indicates a perfect match). Words accepted by the mapper are either "hits," words really in the input sentence, or false alarms, words accepted although not in the input.

The mapper was tested by calling it for all of the words in the vocabulary at the start of an utterance and then at each position where a previously accepted word ended. This procedure resulted in testing the entire vocabulary at an average of about 16 out of 20 positions per second of speech (recall that word boundaries are rounded to multiples of .05 seconds, so there are 20 possible ending positions per second).* Overall, tests were made at 180 positions in 11 test utterances. For the 305-word vocabulary used in the following experiments, the mapper had 48 hits and 1564 distinct false alarms. The false alarms were distributed throughout the vocabulary (229 of the 305 words (75%) were falsely accepted at least once), with small words like "a" and "the" each accounting for more than 30 false alarms.

The false alarm rate for the mapper was determined by counting the number of false alarms that fell within a section of the input. For the 305 word vocabulary, the average rate was 114 false alarms per second of speech. Since there were about 3 hits per second of speech, this rate indicates that the mapper produced an average of almost 40 false alarms for each hit. As partial compensation for this result, there were no "misses" (cases in which the mapper failed to accept a correct word) and the mean hit score was higher than the mean false alarm score (73.5 versus 59.4), although both score distributions spread over the entire range, from near 100 down to the threshold of 45.

The remaining experiments use a simulation of the `mapper based on the data gathered in the first experiment. To simulate the performance of the mapper on a particular sentence, the words of the sentence were first assigned lengths in seconds of speech. Each word was then assigned a score picked at random from the hit scores actually produced by the mapper. The words were concatenated to determine the length of the utterance, and the length was multiplied by the false alarm rate (114 false alarms per second of speech) to give the total number of false alarms to be simulated. The false alarms were

----

* This experiment was originally designed to record the results of all mapper calls that might be made in a left to right parse. The intention was to use this information in place of the mapper in tests of the entire system. However, technical difficulties made it impossible to gather enough information to satisfy the original goal. If the original goal had been to provide data for a simulation of the mapper, the mapper would have simply been tested on the entire vocabulary across each utterance at .05 second intervals. The change in goals may have resulted in slightly underestimating the false alarm rate for the mapper because of the untested positions where no word ended.

selected randomly from the 1564 false alarms produced by the mapper, and then positioned randomly in the sentence. As a check on our simulation, we calculated the correlations between the observed score distributions and the score distributions in the simulated utterances used in the following experiments. The hit score distributions had a .97 correlation, and the false alarm distributions had a .996 correlation. (There were many more false alarms, hence the higher correlation.) In computing the simulated processing time for the mapper in later experiments, we used figures of 0.25 seconds processing per word tested, 1.0 seconds per position of initial processing before trying any words, and 10.0 seconds per second of speech in the sentence if "island-driving" was being simulated (see discussion of Experiment 4). These figures are derived from rough measurements of the mapper running on an IBM System/370 Model 145.

EXPERIMENT 2--FANOUT

The second experiment dealt with the fanout in the language with and without acoustic constraints. "Fanout" is defined as the number of words that can be successfully appended to an initial substring of some sentence, to produce either a complete sentence or a string that can potentially be completed to form a sentence. The average fanout over a large number of initial substrings provides a measure of the uncertainty of each word, as indicated by the number of alternatives open to the system.

The fanout was measured for 11 sentences, together containing a total of 67 words. The fanout was measured only for initial substrings of the actual sentences; it was not measured along false paths. The distribution of the size of the fanout was bimodal. Using the 305-word vocabulary and ignoring acoustic constraints, 24 positions (36%) had a fanout of less than 30 words, while 33 positions (49%) had a fanout of more than 173 words. The small fanout positions were places allowing only vocabulary classes with a small number of members, classes such as preposition or verb. The large fanout positions corresponded to places where a noun could be expected. The mean fanout was 117, with a standard deviation of 90 and a maximum of 219. The fanout at the beginning of sentences was 206.

The fanout with acoustic constraints is based on the simulated mapper data. It is calculated by counting the number of words: (a) that are accepted by the simulated mapper at a position starting plus or minus 0.05 seconds from the end of an initial substring of hits, and (b) that are also in the fanout set without acoustic constraints for that substring. In addition to recording the size of the fanout, we ordered the set of words by mapper scores and computed the rank of the hit. For example, if 2 false alarms had scores higher than the hit, the rank of the hit would be 3. For the 305 word vocabulary, the mean fanout with speech was 18, and the average hit rank was 3.7. The fact that the hit rank is smaller than half of the fanout reflects the previously mentioned difference between the score distributions for hits and false alarms.

The results of this experiment help to show why the control strategy problem for speech understanding is so difficult. The results suggest that, on the average, there will be between 2 and 3 false alarms with higher scores than the actual hit to tempt the system down false paths. Luckily, there are compensating factors tending to bring the system back to the correct path. One aid is that the fanout following a false alarm is probably smaller than the fanout following a hit, and false paths thus often lead to dead ends. (This speculation needs to be tested empirically.) The decrease in fanout is most pronounced near the boundaries of an utterance, where many words are eliminated because their minimum duration is greater than the available time. Similarly, false paths may be impossible to complete because too much speech remains. For example, a path will be a dead end if it requires a one-syllable word to fill a four-syllable section of the input. Finally, even if there are complete false paths, the system may still get the sentence right if the correct path is found and is given a higher overall score than any incorrect path. The difference in hit and false alarm score distributions makes this more likely. These factors, and perhaps others not yet recognized, may offset the effect of the large number of high scoring false alarms, but speech understanding is still a difficult task, as indicated by the results of the next experiment.

EXPERIMENT 3--TWO (TOO) SIMPLE SYSTEMS

In the third experiment, two systems with extremely simple designs were tested on a set of 60 sentences. The sentences covered a wide range of vocabulary and included questions, commands, and elliptical sentence fragments. There were 10 sentences at each length of simulated speech from 0.8 to 2.3 seconds at intervals of 0.3 seconds. The sentences averaged 5.9 words in length, with a maximum of 9 words.

The first system tested used a dynamic programming method to find the sequence of words accepted by the simulated acoustic processing with the highest cumulative score. The sequence was constrained to start and end within 0.05 seconds of the utterance boundaries, and to have between-word gaps and overlaps of no more than 0.05 seconds. There were no syntactic or semantic constraints on the word sequences chosen. This system selected the complete correct sequence of words in 3 sentences (5%) and found 86 of the 323 hits (27%).

The second test system used a context-free parsing algorithm to find the sequence of words with the best cumulative score that met the same gap and overlap constraint of 0.05 seconds and also satisfied some simple phrase structure rules for a subset of English.* The additional constraint eliminated many of the false paths open to the first test system, but the algorithm still selected the correct sequence in only 6 cases (10%) and found only 100 hits (31%).

The poor performance of these simple test systems tends to confirm the need for more extensive linguistic knowledge (or large improvements in acoustic processing accuracy) and helps to justify the greater complexity of the standard system, which, as the following experiments show, is able to achieve much better results.

EXPERIMENT 4--CONTROL STRATEGY DESIGN CHOICES

In the fourth experiment, the performance of the standard speech system was measured on the 60 test sentences described above, while varying four major control-strategy design choices. The choices used as experimental variables were the following:

> Island-Drive or Not -- Go in both directions from arbitrary starting points in the input versus proceed strictly left to right from the beginning: Island-driving allows the system to use words that match well anywhere in an input and to build up an interpretation around them. Left to right processing is simpler and less flexible but may still be more accurate and efficient than island-driving.

----

* The rules were taken from the language definition for the standard speech system used in the later experiments.

Map All or One -- Test all the words at once at a given location versus try them one at a time and delay further testing when a good match is found: Mapping all at once lets the system know the best candidates from the acoustics and reduces the chances of following a false path. Mapping one at a time avoids exhaustive testing and will be more efficient than mapping all at once if the system does not encounter too many false alarms.

Context Checks -- Take into account the restrictions of the possible sentential contexts as part of setting priorities versus ignore the contextual restrictions except for use in eliminating already formed structures: Context checking should give more information for setting priorities and should lead to better predictions. However, the checks can be expensive and therefore may not lead to an overall improvement in performance.

Focus by Inhibition -- Focus the system on selected alternatives by inhibiting competition versus employ an unbiased best-first strategy: Focusing allows the system to concentrate on a particular set of potential interpretations rather than thrashing among a large number of alternatives. However, if the focus of attention is too often wrong, the net effect may be harmful to system performance.

All combinations of the 4 control-strategy variables were tested on the 60 sentences. This experimental design allows us to compare the 16 combinations of control choices and to evaluate, by analysis of variance, the main effects and interactions of the control strategy variables. The main effect of a variable is the change in performance it produces, averaged over all the possibilities for the other variables. The interaction of two variables tells whether the effect of one variable is the same for all possibilities of the other. The interaction of three variables tells whether the interaction of two of them is the same for all possibilities of the third, and so on. Analysis of variance is a statistical technique for computing the probability that the observed effects or interactions are really caused by the experimental variables, rather than the result of random variation (see e.g. Winer 1971). In other words, this method aids in evaluating results of experiments influenced by substantial random factors. In our case, the random factors include the random choices of false alarms and hit scores in simulating the mapper, and the selection of a particular sample of sentences from the much larger population of possible sentences. The statistical results for a main effect or interaction are given in a form such as "F(1,5)=6.9, p < .05." This means that the F ratio (a statistic for comparing variances) for the effect or interaction has 1 and 5 degrees of freedom and has a value equal to 6.9. This in turn implies that the probability is less than .05 that the observed effect or interaction was caused by random variation alone. If the probability is given by itself in the following discussion, it is based on the these values: F(1,5)>=16.3 for p < .01, F(1,5)>=6.6 for p < .05, and F(1,5)>=4.1 for p < .10.

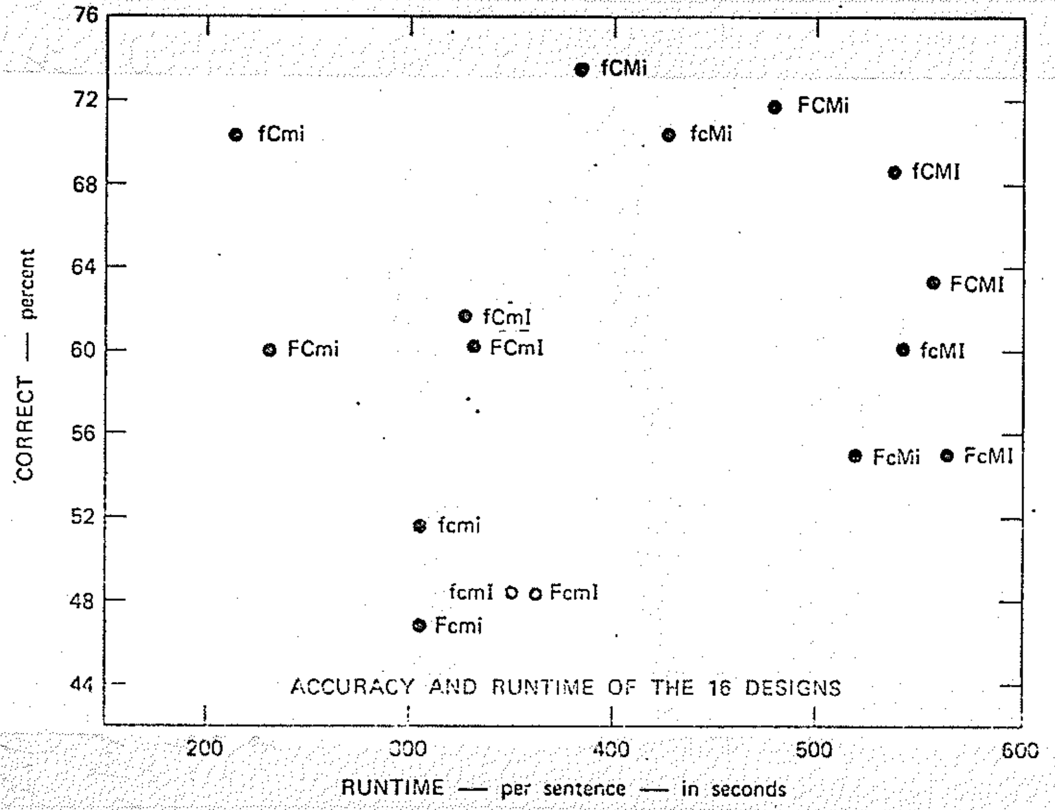

Figure 1. Accuracy and Runtime

The most important performance measures for the system are accuracy (the percentage of sentences for which the correct sequence of words is found) and runtime (the computation required by the system, including simulated acoustic processing time). For these measures, the control strategy variables had large, significant effects. Before discussing the effects, we need to introduce a notation for naming the experimental designs. The capital letter "F" will refer to focus by inhibition, lower case "f" for no focus by inhibition; "C" stands for context checks, "c" for no context checks; "M" for map all at once, "m" for not map all at once; "I" for island-driving, and "i" for no island-driving. This notation will indicate the different combinations of choices. For example, "fCMi" refers to the system that does not use focus by inhibition, does use context checks, does map all at once, and does not island-drive.

Using this notation, Figure 1 shows the accuracy and runtime of the 16 experimental systems. Notice the range of values for both measures, from 46.7% to 73.3% for accuracy, and from 221 to 559 seconds processing per sentence for runtime. These wide ranges confirm the importance of control strategy in determining system performance. With respect to the individual control variables, comparing the C-systems to the correspondng c-systems shows that context checks for priority setting result in better accuracy and faster runtimes. Similar comparisons show that mapping all at once improves accuracy but increases runtime, while focus by inhibition and island-driving both reduce the accuracy and increase the runtime. In the course of this paper, we discuss these effects and propose explanations for them.

Table 1
MAIN EFFECTS OF VARIABLES
ON PERCENT CORRECT

	WITH	WITHOUT	DIFFERENCE	
F	57.5	62.9	-5.4	*
C	66.0	54.4	11.6	*
M	64.6	55.8	8.8	*
I	58.1	62.3	-4.2	

* $p < 0.05$

Table 2
FOCUS AND ISLAND-DRIVING
INTERACTION

	I	i	I-i
F	56.7	58.3	-1.6
f	59.6	66.3	-6.7
F-f	-2.9	-8.0	5.1

Table 1 shows the effect of the control variables on accuracy. For the purposes of analysis of variance, we pooled the results on the 10 sentences of equal length to get 6 accuracy measures per system. The interaction with length was then used as the error term for calculating statistical significance. Results are reported for analysis of the raw percentages; analysis after an arcsin transformation to improve homogeneity of variance was also performed and gave the same levels of significance. As inspection of Figure 1 suggests, context checks and map all improve accuracy, while focus and island-driving make it worse. The island-driving effect was not significant statistically because of a large interaction with sentence length. For the long sentences, 1.7 to 2.3 seconds, island-driving decreased accuracy by 15.8%, but for the short ones, 0.8 to 1.4 seconds, it actually increased accuracy by 7.5%. There was a significant interaction ($p < 0.05$) between focus and island-driving. As shown in Table 2, the effect of island-driving is less with focus, and the effect of focus is less with island-driving. To explain this collection of results we must first consider how accuracy is influenced by control strategy.

The control strategy affects accuracy indirectly. All the strategies are "complete" in the sense that they only reorder, and never eliminate, alternatives. If there were no false alarms, all the systems would get 100% of the test sentences correct. Even with false alarms, the strategies would get an equal percent correct, if all the possible alternatives could be tried before the system picked an interpretation. Errors would only occur when false alarms had high enough scores to displace hits in the highest rated interpretations. However, in the actual system, the large number of alternatives makes it impossible to consider all of them in the space and time available. As a result, the order in which the alternatives are considered can affect the accuracy, and so can the demands on space and time. Control strategy thus affects accuracy indirectly by reordering alternatives and by modifying space and time requirements. To explain the accuracy effects, we must look at these other factors.

In this experiment, the storage limit was an important factor for accuracy. In the 960 tests (60 sentences times 16 systems), 578 (60.2%) were correct and 383 (39.8%) were wrong. Of the errors, 175 (46%) had an incorrect interpretation, while 207 (54%) had no interpretation at all. Since the systems could potentially get the correct answer, and no time limit was imposed until at least one interpretation had been found, all of the 207 sentences with no interpretation were a result of running out of storage.

The storage limit used in the tests was based on the number of phrases constructed. When the total reached 500, the system would stop trying new alternatives and, if any interpretation had been found, pick the highest rated interpretation as its answer. The average number of phrases constructed was 204 nonterminal and 63 terminal. The system with the best accuracy, fCMi, had the lowest average (113 nonterminals and 45 terminals), while the system with the worst accuracy, Fcmi, had one of the highest averages (260 nonterminals and 68 terminals). Overall, there was a strong negative correlation (-.93) between system accuracy and average number of phrases constructed.

Table 3
MAIN EFFECTS ON STORAGE
(number of phrases)

	WITH	WITHOUT	DIFFERENCE	
F	281	253	28	*
C	240	294	-54	**
M	244	290	-46	**
I	287	247	40	

** $p < .01$ * $p < .05$

Table 3 shows the effects of the control variables on the number of phrases. The pattern is the same as for accuracy; context checks and map all have good effects, while focus and island-driving have bad effects. Again, because of a large interaction with length, the island-driving effect is not significant statistically. There are significant interactions, $p < 0.05$, between focus and island-driving for storage, as seen in Table 4, and between context checking and mapping all at once, as seen in Table 5.

Table 4
FOCUS AND ISLAND-DRIVING
INTERACTION
(number of phrases)

	I	i	I-i
F	290	272	18
f	284	222	62
F-f	6	50	-44

Table 5
CONTEXT AND MAP-ALL
INTERACTION
(number of phrases)

	M	m	M-m
C	221	259	-38
c	267	322	-55
C-c	-46	-63	17

The beneficial effect of mapping all at once on the storage requirements and accuracy is caused by a reduction in the proportion of false alarm terminal phrases. Mapping all at once significantly reduces the proportion of terminal phrases that are false alarms--from 88.0% to 85.7%, $p < .01$. The false terminal proportion is in turn significantly correlated with the number of phrases (.72) and the accuracy (-.75). When the words are all mapped at once, the system is able to take advantage of the difference in false alarm and hit score distributions to reduce the likelihood of constructing false terminal phrases. Notice that a relatively small change in false terminal percentage has a large effect on system performance.

Surprisingly, context checking also results in a significant reduction in the false terminal percentage--from 87.5% to 86.2%, $p < .01$. This reduction may be evidence that context checking is giving lower priority to looking for words adjacent to false alarms than it gives to looking next to hits. This change could affect the false terminal likelihood, since there is always a hit adjacent to a hit, while false alarms often have nothing but other false alarms next to them. In addition to its effect on false terminals, context checking may also be improving the storage requirements and accuracy by generally improving the priority setting, thereby reducing the likelihood of following false paths.

Focus by inhibition slightly increases the proportion of false alarm terminal phrases (from 86.3% to 87.3%), but this increase is too small to be statistically significant. The explanation of the ill effects of focus is essentially the converse of the explanation of the effects of context checking. Context checking makes performance better by improving priorities, while focus makes it worse by distorting priorities. Focus too often changes priorities to bias the system in favor of a false alarm instead of a hit. Overall, focus changed priorities in favor of a false alarm 112 times per sentence and in favor of a hit, only 15 times per sentence. Thus the priorities, and the system performance, were better with the unbiased best-first strategy than with focus by inhibition.

Island-driving did not affect the false terminal proportion, but it did have bad effects on storage and accuracy for the longer sentences. To get a sentence correct, island-driving must start at least one island with a hit. If all the islands are false alarms, the sentence will not be interpreted correctly. The overall average was 3.7 false alarm islands per sentence and 0.9 hit islands. There were one or more hit islands in 82% of the tests using island-driving. The bad effects of island-driving on long sentences was not caused by a greater likelihood of false alarm islands. The average rank of the first hit in the sequence of words for use in forming islands was 4.8, and this did not increase with sentence length. (The correlation between rank and length was .04). For sentences 1.7 seconds or longer, instead of an increase in the number of islands necessary to get a hit, there was an increase in the amount of storage consumed per island. Perhaps the greater length allowed islands to grow in both directions, whereas in shorter sentences the sentence boundaries blocked one direction or the other.

The interaction of focus and island-driving can be explained as the result of the storage limit. The limit put a ceiling on the size of the possible combined effect. Thus the combined effect was less than the sum of the individual effects. Similarly, the interaction between context checking and mapping all at once is a result of overlapping good effects, which consequently fail to add. The same pattern of context and map-all interaction appears in false terminal proportion, $p < .05$, and in accuracy, $F=4.00$ versus $F(1,5)=4.06$ for $p < .10$.

We now turn to a brief analysis of the sentences that got one or more interpretations, but were incorrect because their highest rated interpretation was wrong. As mentioned previously, this happened in 175 tests. In 109 of these (62%), the chosen interpretation was reasonable linguistically but contained incorrect words. In 10 tests (6%), the chosen interpretation could have been eliminated by a better language definition ("Was feet one builder of the Farragut?" is an example from these 10). Finally, 56 of the errors (32%) were harmless, in that the system would probably produce the same answer as if it had found the correct sequence of words (e.g., "What reactor does it have?" instead of "What reactors does it have?" was one of these harmless errors). If the harmless errors are counted as correct in calculating the accuracy, the most accurate system, fCMi, increases in percent correct from 73.3% to 80.0%, and the average accuracy for all the systems goes up 5.8%. The ten to one preponderance of linguistically reasonable errors over linguistically bad errors suggests that, although there is still room for improvement in the language definition, the major way to improve accuracy is to reduce the number of high scoring false alarms. The effects of such acoustic improvements are explored in Experiment 6.

The accuracy effects have been explained in terms of storage requirements, proportions of false terminal phrases, and priorities. The important role of storage limitations raises the question of whether the accuracy effects would have disappeared if more storage had been available. I speculate that the effects would have been smaller but still important. The effects on the proportion of false terminal phrases would remain, as would the presumed effects on priorities. Moreover, even if the storage limit were relaxed, the limit on runtime would remain to penalize inefficient systems. The effects of control strategy choices on accuracy would only vanish if space and runtime limitations were both removed, a most unlikely event in view of the current performance of speech understanding systems.

The system runtime is another important performance measure. Here, we will use the phrase "total runtime" to refer to the simulated acoustic processing, plus the actual processing time (on a DEC PDP KA-10) for the executive and the semantic components. The executive time is mainly spent setting priorities and parsing. The semantics time is used in constructing semantic translations and in dealing with anaphoric references and ellipsis. The reported total runtime does not include acoustic preprocessing or question answering, since neither are affected by the experimental variables. In analysis of variance of the runtimes, interaction with length was used as the error term, and significance levels were confirmed by reanalysis after a square root transformation to improve homogeneity of variance.

The main effects of the control variables on total runtime are given in Table 6. All except context checking increase the runtime. Partitioning the sentences into a short group (0.8 to 1.4 seconds) and a long group (1.7 to 2.3 seconds) shows that island-driving has a much worse effect on runtime for long than for short sentences. For short sentences, island-driving increased the mean runtime from 262 to 290 seconds, a difference of 28. For long sentences, the increase was from 457 to 598 seconds, a difference of 141. Recall that island-driving also had worse effects for long sentences on accuracy and storage.

Table 6
MAIN EFFECTS ON TOTAL RUNTIME
(seconds per sentence)

	WITH	WITHOUT	DIFFERENCE	
F	417	386	31	*
C	383	421	-38	**
M	498	305	193	**
I	444	359	85	#

* p < .05 ** p < .01 # p < .10

Table 7
EFFECTS ON EXECUTIVE RUNTIME
(seconds per sentence)

	WITH	WITHOUT	DIFFERENCE	
F	120	106	14	**
C	109	117	-8	#
M	90	135	-45	**
I	127	98	29	#

Table 8
EFFECTS ON ACOUSTICS RUNTIME
(seconds per sentence)

	WITH	WITHOUT	DIFFERENCE	
F	276	260	16	#
C	254	282	-28	**
M	389	147	242	**
I	295	241	54	#

** p < .01 * p < .05 # p < .10

Tables 7 and 8 show the main effects on executive runtime and simulated acoustics runtime respectively. In both cases, context checks decrease the runtime, while focus and island-driving increase it. Mapping all at once improves the executive runtime but leads to a huge increase in acoustic processing time. As usual, examination of the results according to sentence length shows that island-driving is worse for longer sentences. Since the average executive and acoustic times together account for 95% of the average total, we do not report separate effects for semantics.

Analysis of variance for total, executive, and acoustic runtimes reveals a significant interaction between context checking and mapping all at once ($p < .01$ for total and acoustics; $p < .05$ for executive). For total and acoustic runtime, the good effect of context checking was reduced when words were mapped all at once, and the increase in runtime caused by map-all was greater when also context checking. For executive runtime, both context and map-all had good effects, and there was actually a synergistic relation; context checking helped more when mapping all at once, and vice versa.

There was also a significant three way interaction among focus, map-all, and island-driving ($p < .01$ for total and acoustic runtimes; $p < .05$ for executive). When not mapping all at once, there was negligible interaction between focus and island-driving. However, when mapping all at once, the combined bad effect of focus and island-driving was significantly less than the sum of their individual bad effects.

The runtime results follow basically the same pattern as the accuracy and storage results. Focus and island-driving have bad effects, with worse results from island-driving for longer sentences, while context checking has consistently good effects. Map-all has a good effect on executive runtime, but, unfortunately, it causes large increases in acoustic and total runtimes. The only inconsistency with the previous pattern of effects for accuracy and storage is the bad effect of map all on the acoustic runtime. This fact is explained by pointing out that the mapper was designed for mapping words one at a time and, in the simulation, does not accumulate or share information to make subsequent tests more efficient. Finally, it is noteworthy that the extra effort for context checking resulted in a net decrease in processing time. For example, fCMi required an average of 6.3 seconds more per sentence to do context checks, but was still 41 seconds per sentence faster than fcMi.

The runtime figures above are in units of seconds used to process a sentence. A more common unit for runtime is seconds per second of speech. This is a reasonable scale if the runtime is essentially a linear function of sentence length and has a zero intercept. Both assumptions are consistent with our data. No significant nonlinearity was found by an F test of the variance of the mean for each length about the regression line, relative to the combined variance of the sentences within a given length (for instance, the data for fCMi gave F=1.37 versus F(4,54)=1.41 for p < .25). Moreover, the 95% confidence interval for the intercept of the regression line included the origin. With this justification, we used zero-intercept linear regression to calculate the processing times per second of speech and their 95% confidence intervals.

The results for the fastest system, fCmi, were 141, plus or minus 14 seconds processing per second of speech for total runtime; 66, plus or minus 9 for executive runtime; and 63, plus or minus 7 for acoustic runtime. The results for the most accurate system, fCMi, were 247, plus or minus 21 for total runtime; 34, plus or minus 6 for executive runtime; and 205, plus or minus 14 for acoustic runtime. Thus, for fCMi, 83% of the total runtime slope comes from acoustic processing, 14% from the executive, and the remaining 3% from the semantics. Clearly, the best approach to improving fCMi runtime is to redesign the mapper for mapping all at once. The potential is large for sharing work to improve efficiency in the mapper, since the data show that fCMi is mapping all the words at an average of 15 out of the 20 possible positions per second of speech.

The results of Experiment 4 have relatively clear implications for the control-strategy design choices. The effects of focus by inhibition were all bad, too often focusing the system on false alarms instead of hits. An unbiased best-first strategy is better. The effects of island-driving were also bad, and they were particularly bad for longer sentences. Island-driving was hurt by false alarm islands, especially when the sentence was long enough for the islands to grow in both directions. Perhaps island-driving can be modified to overcome this problem, but until then, simple left-to-right processing is better. Context checking had uniformly good effects. For both accuracy and runtime, it was worth the extra effort in order to get better priority setting. The only ambiguous control choice is whether or not to map all at once. Mapping all at once improves accuracy and executive runtime, but at a large cost in acoustic and total runtime. Redesign of the mapper could undoubtedly resolve this choice in favor of mapping all at once. For example, just cutting the acoustic processing in half would make the fCMi system about as fast as the fCmi system. The choice, whether to map all or not, is explored further in Experiment 6. Finally, note that changes in the mapper appear to offer the best chances for significant improvements in both accuracy and runtime.

EXPERIMENT 5--GAPS AND OVERLAPS

The data from Experiment 1 do not aid us in simulating the mapper's performance when called on to test whether two words it has accepted individually are also acceptable as a contiguous pair. Such tests are necessary whenever words and phrases are combined to form larger units. In the basic simulation of the mapper, we simply allowed gaps and overlaps between words of up to 0.05 seconds of speech. Experiment 5 tests the effect of different values of the gap-overlap parameter on the performance of the fCMi system from Experiment 4. Table 9 gives the results for a variety of measures with gap-overlap sizes of 0.00, 0.05, and 0.10 seconds.

Table 9
EFFECTS OF GAP-OVERLAP PARAMETER

	GAP-OVERLAP SIZE		
	0.00	0.05	0.10
Fanout with acoustics (words)	6.6	18.0	29.4
Rank of hit in fanout	1.9	3.7	5.6
Raw accuracy %	96.7	73.3	48.3
Forgiving accuracy %	98.3	80.0	58.3
False terminal %	58.2	83.2	89.1
Number of nonterminals	31	113	217
Total runtime (sec/sec-speech)	140	247	333
Executive runtime "	10	34	69
Acoustics runtime "	128	205	243

The performance is much better for 0.00 and much worse for 0.10 seconds of gap-overlap. This is strong evidence for the importance of special acoustic tests to verify word-pair junctions. Such tests can lead to a large reduction in the average hit rank and, consequently, to significant improvements in both accuracy and runtime.

EXPERIMENT 6--INCREASED VOCABULARY AND IMPROVED ACOUSTICS

Experiment 6 studies the effect on system performance of increased vocabulary size and improved acoustic-processing accuracy. As test systems, we use fCMi and fCmi from Experiment 4. These were the best systems for accuracy and speed, respectively, and would also give us more information about the map-all control strategy choice. Thus there were three experimental variables: vocabulary size, acoustics, and map-all. Data for two of the eight combinations, map-all or not for smaller vocabulary and regular acoustics, came from Experiment 4. For Experiment 6, the other six combinations were tested to provide a complete set of data for analysis of the effects of the variables.

The large vocabulary is a 451-word superset of the 305-word vocabulary used in the other experiments. The data gathered in Experiment 1 showed that with the 451-word vocabulary the mapper made 2026 false alarms and had a false alarm rate of 142 false alarms per second of speech (compared to 114 for the 305-word vocabulary). Using this information, the mapper performance was simulated for the large vocabulary on the same set of 60 test sentences.

Improved acoustic-processing accuracy was simulated by a 7% downward stretch of the false alarm score distribution, while leaving the hit scores unchanged. In other words, a false alarm score X, in the range 45 to 100, was replaced by 1.07X-7. If the result was below the threshold of 45, the false alarm was eliminated. This process reduced the number of false alarms for the 305-word vocabulary from 1564 to 1204, and for the 451-word vocabulary, from 2026 to 1541. Because the subthreshold scores were eliminated, the simulated improvement left the average false alarm score almost unchanged: for the 305-word vocabulary, it went from 59.4 to 60.2, and for the 451-word vocabulary, it went from 58.2 to 58.8. We feel that an improvement in acoustic accuracy of the magnitude simulated here could have been achieved by careful tuning of the mapper.

Table 10 records the accuracy results using the notation "M" for tests with mapping all at once, "m" for those without, "A" for systems with improved acoustics, "a" for those without, "V" for systems with increased vocabulary, and "v" for those without. Improved acoustics raises fCMi accuracy from 73.3% to 85.0%, or from 80.0% to 95.0% if harmless errors are forgiven. However, if vocabulary size is also increased, accuracy drops slightly from 73.3% to 71.6%. Thus, in this experiment, a 7% improvement in acoustics almost compensates for a 48% increase in vocabulary. Comparison of the M-results to the m-results shows that map-all consistently helps accuracy.

Table 10
ACCURACY RESULTS
(percent)

	AMv	Amv	aMv	AMV	amv	AmV	aMV	amV
Raw	85.0	78.3	73.3	71.6	70.0	68.3	68.3	53.3
Forgiving	95.0	85.0	80.0	78.3	76.7	76.7	75.0	58.3

The main effects on accuracy and several other measures are given in Table 11. Improved acoustics leads to big gains in accuracy, storage, and runtime. Increased vocabulary makes performance worse, but at least the system does not collapse. As in Experiment 4, mapping all at once improves everything except acoustic and total runtimes.

Table 11
MAIN EFFECTS OF ACOUSTICS, VOCABULARY, AND MAP-ALL

		WITH	WITHOUT	DIFFERENCE	
Raw Accuracy (percent)					
	A	75.8	66.3	9.5	**
	V	65.4	76.7	-11.3	#
	M	74.6	67.5	7.1	*
Phrases (total number terminal and nonterminal)					
	A	155	208	-53	**
	V	204	159	45	**
	M	156	206	-51	**
False Terminals (percent)					
	A	80.6	85.9	-5.3	**
	V	84.3	82.1	2.2	
	M	81.7	84.8	-3.1	**
Total Runtime (seconds/sentence)					
	A	266	320	-54	**
	V	312	275	37	*
	M	383	204	179	**
Acoustic Runtime (seconds/sentence)					
	A	187	213	-26	**
	V	205	195	10	
	M	315	84	231	**
Executive Runtime (seconds/sentence)					
	A	66	89	-23	**
	V	88	67	21	**
	M	55	101	-46	**

** p < .01   * p < .05   # p < .10

There were few significant interactions. Vocabulary size and mapping all at once interacted significantly for acoustic runtime (p < .05) and for total runtime (p < .10). Table 12 shows that the increase caused by map-all is greater for the bigger vocabulary, and, surprisingly, that the increase in vocabulary size leads to a reduction in processing, if the system is not mapping all at once.

Table 12
VOCABULARY AND MAP-ALL INTERACTION
for ACOUSTICS RUNTIME
(seconds/sentence)

	M	m	M-m
V	335	75	260
v	296	94	202
V-v	39	-19	58

Table 13
ACOUSTICS AND MAP-ALL INTERACTION
for FALSE TERMINALS
(percent)

	M	m	M-m
A	78.6	82.6	-4.0
a	84.8	87.0	-2.2
A-a	-6.2	-4.4	-1.8

Mapping all at once also interacted significantly with acoustics for acoustic runtime ($p < .01$), total runtime ($p < .01$), and false terminal percentage ($p < .05$). All cases were similar to the one shown in Table 13. There was a synergistic interaction causing mapping all at once to be more effective with better acoustics, and vice versa. This result is readily explained since map all is designed to take advantage of the difference between false alarm and hit score distributions, and improving the acoustics enhances that difference by reducing the number of high scoring false alarms.

In addition to the main tests for Experiment 6, we also ran two other tests to study the effect of improved acoustics on systems using island-driving and focus by inhibition. The best island-driving system from Experiment 4 was fCMI. When tested on the 305-word vocabulary with 7% simulated improvement in acoustics, fCMI gained in accuracy from 68.3% to 78.3%. It was still below the non-island-driving fCMi, and the gap between them remained large. (Recall that fCMi went from 73.3% to 85.0%.) The best focus by inhibition system was FCMi. Improved acoustics raised its accuracy and reduced its runtime, but it was still less accurate than fCMi (80.0% versus 85.0%) and also slower (235 seconds per sentence, versus 200). The accuracy difference vanished with a 24% simulated improvement in acoustics, but, even with a 51% simulated improvement, a slight runtime advantage for fCMi remained.

In summary, this experiment has given us information about how badly the system is hurt by increased vocabulary, and how much it is helped by improved acoustics. With respect to the control-strategy design choices, further evidence appeared in favor of mapping all at once, and against island-driving and focus by inhibition.

CONCLUSION

Reviewing the series of experiments, the first experiment showed that the acoustic processing component called the mapper" had a high false alarm rate, but tended to give better scores to hits than to false alarms. In the second experiment, we measured the number of alternatives open to the system for extending segments of sentences. The size of the fanout helps to explain the difficulty of speech understanding. The third experiment found that two simple system designs were too simple, a result that helps to justify the complexity of the standard system. The fourth experiment studied the effects on system performance of four control-strategy design choices. Focus by inhibition and island-driving had bad effects, while context checks for priority setting had good effects. Mapping all at once had good effects on everything except acoustic and total runtime, and these bad effects could probably be eliminated by redesign of the mapper. In fact, mapper changes appear to offer the best hope for large gains in both accuracy and runtime. The fifth experiment varied the size of allowed gaps and overlaps between words and showed the potential value of special acoustics tests to verify word-pair junctions. Finally, the sixth experiment gave quantitative measures of how badly the system is hurt by increased vocabulary, and how much it is helped by improved acoustic accuracy. The experiment also provided more information about the control choices. Overall, the series of experiments gives insights into system performance and control strategy that should be useful in designing future speech understanding systems.

REFERENCES

Paxton, William H. A Framework for Language Understanding. In COLING 76, Preprints of the 6th International Conference on Computational Linguistics, Ottawa, Canada, 28 June - 2 July 1976. [Technical Note 131, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California, June 1976.]

Paxton, William H. A Framework for Speech Understanding. Stanford University Ph.D. dissertation, Stanford, California. forthcoming.

Ritea, H. Barry. Automatic Speech Understanding Systems. Proceedings, Eleventh Annual IEEE Computer Society Conference, Washington, D.C., 9-11 September 1975.

Winer, B. J. Statistical Principles in Experimental Design, second ed. McGraw-Hill Book Company. New York, New York. 1971.