A PARSER FOR REAL-TIME SPEECH SYNTHESIS OF CONVERSATIONAL TEXTS

Joan Bachenko
Jeffrey Daugherty[1]
Eileen Fitzpatrick

AT&T Bell Laboratories
Murray Hill, NJ 07974

ABSTRACT

In this paper, we concern ourselves with an application of text-to-speech for speech-impaired, deaf, and hard of hearing people. The application is unusual because it requires real-time synthesis of unedited, spontaneously generated conversational texts transmitted via a Telecommunications Device for the Deaf (TDD). We describe a parser that we have implemented as a front end for a version of the Bell Laboratories text-to-speech synthesizer (Olive and Liberman 1985). The parser prepares TDD texts for synthesis by (a) performing lexical regularization of abbreviations and some non-standard forms, and (b) identifying prosodic phrase boundaries. Rules for identifying phrase boundaries are derived from the prosodic phrase grammar described in Bachenko and Fitzpatrick (1990). Following the parent analysis, these rules use a mix of syntactic and phonological factors to identify phrase boundaries but, unlike the parent system, they forgo building any hierarchical structure in order to bypass the need for a stacking mechanism; this permits the system to operate in near real time. As a component of the text-to-speech system, the parser has undergone rigorous testing during a successful three-month field trial at an AT&T telecommunications center in California. In addition, laboratory evaluations indicate that the parser's performance compares favorably with human judgments about phrasing.

1. INTRODUCTION

Text-to-speech researchers and developers tend to assume that applications of their technology will focus on edited text, either "canned" material such as name and address lists, or free text like the AP newswire. There has been much effort aimed at preparing text-to-speech for applications such as caller identification and newsreading services, in which texts are generally proofed and the primary challenges come from issues of name pronunciation, intonation contouring, etc. In this paper, we concern ourselves with an application of text-to-speech for speech-impaired, deaf, and hard of hearing people. The application is unusual because it requires text-to-speech synthesis of unedited, spontaneously generated conversational text. Moreover the synthesis must occur in near real time as the user is typing.

We will describe a parser that prepares conversational texts for synthesis by first performing lexical regularization of nonstandard forms and then identifying prosodic phrase boundaries. The parser is derived from the prosodic phrase system presented in Bachenko and Fitzpatrick (1990) and has been implemented as the front end of a version of the Bell Laboratories text-to-speech synthesizer (Olive and Liberman 1985). As a component of the text-to-speech system, the parser has undergone rigorous testing during a successful three-month field trial at an AT&T telecommunications center in California. In addition, laboratory evaluations indicate that the parser's performance compares favorably with human judgments about phrasing. In Section 2 of the paper we describe the application and the texts. Section 3 provides a technical description of the parser and Section 4 discusses evaluation of the parser's performance.

2. THE APPLICATION

Users of Telecommunications Devices for the Deaf (TDD's) can communicate with voice telephone users via services such as AT&T's Telecommunications Relay Service (TRS). During a TRS call, special operators read incoming TDD text to the voice telephone user and then type that person's spoken responses back to the TDD user; this makes for a three-way interaction in which the special operator is performing both text-to-speech and speech-to-text conversion. Text-to-speech synthesis

1. AT&T Bell Laboratories, Naperville, Illinois.

	Spelling	*Punctuation*	*Case*	*Syntax*
Expected texts (e.g. AP newswire)	1% errors	standard	upper and lower case conventions	st. English dialect
TDD texts	5% errors	little or none	single case only	written language of the deaf

Figure 1: TDD vs. Expected Text Input

makes it possible to automate part of this arrangement by reading the TDD text over the telephone to the voice user. The synthesizer thus replaces an operator on the TDD half of the conversation, providing increased privacy and control to the TDD user and, presumably, cost savings to the provider of the service.

TDD texts present unusual challenges for text-to-speech. Except in laboratory experiments, large scale applications of text-to-speech have tended to focus on name pronunciation and "canned text" such as catalogue orders. To the best of our knowledge, the TRS text-to-speech field trial in California represents the first large scale attempt to use speech synthesis on spontaneously generated conversational texts, and also the first to use this technology on texts that are orthographically and linguistically non-standard. Unlike the written material that most text-to-speech systems are tested on, e.g. the AP newswire, TDD texts observe few of the writing conventions of English. All text is in upper case, and punctuation, even at major sentence boundaries, rarely occurs; spelling and typographical errors complicate the picture even further (Tsao 1990; Kukich, 1992). In addition, nearly all texts employ special abbreviations and lingo, e.g., CU stands for *see you*, GA is the message terminator *go ahead*. The following example illustrates a typical TDD text:

OH SURE PLS CALL ME ANYTIME AFTER SAT MORNING AND I WILL GIVE U THE NAMES AND PHONE NOS OK QGA
(Oh sure, please call me anytime after Saturday morning and I will give you the names and phone numbers. OK? Go ahead.)

Finally, many texts are written in a variety of English that departs from expected lexical and syntactic patterns of the standard dialect (Charrow 1974). For example, WHEN DO I WILL CALL BACK U Q GA is a short TDD text that we believe most native speakers of English would recognize as *When should I call you back? Go ahead.* The (attested) example below is less clear, but interpretable:

I WISH THAT DAY I COULD LIKE TO MEETING DIFFERENT PEOPLE WHO DOES THIS JOB AND THE WAY I WANT TO SEE HOW THEY DO IT LIKE THAT BUT THIS PLACES WAS FROM SAN FRANCISCO I GUESS

Syntactic variation in such texts is systematic and consistent (Bachenko 1989, Charrow 1974). Although a complete account has yet to be formulated, Suri (1991) reports that aspects of the variation may be explained by the influence of a native language--ASL--on a second language--English.

Figure 1 above summarizes the points about TDD texts. Spelling error estimates come from Kukich (1992) and Tsao (1990).

Timing creates an additional obstacle since we expect TRS text-to-speech to synthesize the text while it is being typed, much as an operator would read it at the TRS center. How to chunk the incoming text now becomes a critical question. Word by word synthesis, where the listener hears a pause after each word, is the easiest approach but one that many people find nerve-wracking. N-word synthesis, where the listener hears a pause after some arbitrary number of words, is nearly as simple but runs the risk of creating unacceptably high levels of ambiguity and, for long texts, may be as irritating as single-word synthesis. Our solution was to build a TDD parser that uses linguistic rules to break up the speech into short, natural-sounding phrases. With partial buffering of incoming text, the parser is able to work in near real time as well as to perform lexical regularization of abbreviations and a small number of non-standard forms.

3. A TEXT-TO-SPEECH PARSER

3.1. PARSER STRUCTURE AND RULES

In constructing the parser, our goal was to come up with a system that (a) substitutes non-standard and abbreviated items with standard, pronounceable words, and (b) produces the most plausible phrasing with the simplest possible mechanism. Extensive data collection has been the key to success in regularizing lexical material, e.g. the conversion of *fwy* (pronounced "fwee") to *freeway*. Phrasing is accomplished by a collection of rules derived

from the prosodic phrase grammar of Bachenko and Fitzpatrick (1990), with some important modifications. The most radical of these is that the TDD phrasing rules build no hierarchical structure. Instead they rely on string adjacency, part of speech, word subclass and length to make inferences about possible syntactic constituency and to create enough prosodic cohesion to determine the location of phrase boundaries.

The parser works deterministically (Marcus 1980, Hindle 1983). It uses a small three element buffer that can contain either words or structures; once a lexical or prosodic structure is built it cannot be undone. As TDD text is typed, incoming words are collected in the buffer where they are formed into structures by rules described below. Phrasing rules then scan buffer structures. If a phrasing rule applies, all text up to the element that triggered the rule is sent to the synthesizer while, during synthesis, the buffer is reset and the rules restart anew. Once a structure has moved out of the buffer it cannot be recovered for examination by later phrasing rules.

Our approach differs from other recent efforts to build small parsers for text-to-speech, e.g. O'Shaughnessy (1988) and Emorine and Martin (1988), where savings are sought in the lexicon rather than in processing. O'Shaughnessy (1988) (henceforth O.) describes a non-deterministic parser that builds sentence-level structure using a dictionary of 300 entries and a medium sized grammar, which we guess to be slightly under 100 rules. The lexicon is augmented by a morphological component of 60 word suffixes used principally to derive part of speech; for example, *-ship* and *-ness* are considered good indicators that a word of two or more syllables has the category 'noun'. O. gives a thorough account of his parser. Much of his exposition focusses on technical details of the syntactic analysis, and supporting linguistic data are plentiful. However, evaluation of O.'s proposals for speech synthesis is difficult since he gives us only a vague indication of how the parsed sentences would be prosodically phrased in a text-to-speech system. Without an explicit description of the syntax/prosody relation, we cannot be sure how to assess the suitability of O.'s analysis for speech applications.

The system described by Emorine and Martin (1988) (henceforth E&M) incorporates a 300-entry dictionary and approximately 50 rules for identifying syntactic constituents and marking prosodic phrase boundaries. The rules in this system build sentence-level structures that are syntactically simpler than those given in O. but more geared to the requirements of phrasing in that prosodic events (e.g. pause) are explicitly mentioned in the rules. Unfortunately, E&M share few technical details about their system and, like O., provide no examples of the prosodic phrasing produced by their system, making evaluation an elusive task.

Applications such as TRS, which requires near real time processing, make systems based on sentence-level analyses infeasible. In our parser, decisions about phrasing are necessarily local--they depend on lexical information and word adjacency but not upon relations among non-contiguous elements. This combined with the need for lexical regularization in TDD texts motivates a much stronger lexicon than that of O. or E&M. In addition, our parser incorporates a small number of part-of-speech disambiguation rules to make additional lexical information available to the phrasing rules. Let us briefly describe each of the three components that make up the grammar: lexicon, disambiguation rules, and phrasing rules.

3.1.1. The lexicon contains 1029 entries consisting of words, abbreviations, and two- to three-word phrases. Each entry has four fields: the input word (e.g. *u*), the output orthography (*you*), lexical category (Noun), and a list of word subclasses (destress_pronoun short_subject). Word subclasses reflect co-occurrence patterns and may or may not have any relationship to lexical categories. For example, Interjection_1 includes the phrase *byebye for now*, the adverb *however*, the noun phrase *my goodness*, and the verb *smile*, as in I APPRECIATE THE HELP SMILE THANK YOU SO MUCH. Both the lexical category and subclass fields are optional--either may be marked as NIL. Abbreviations and acronyms are usually nouns and make up 20% of the lexical entries. Nouns and verbs together make up about 50%. We expect that additions to the lexicon will consist mostly of new abbreviations and short phrases.

3.1.2. Lexical disambiguation rules identify part-of-speech and expand ambiguous abbreviations. Currently, part-of-speech disambiguation is performed by ten rules. Most apply to words lexically marked for both noun and verb, e.g. *act, call, need*, assigning a single category, either noun or verb, when a rule's contextual tests are satisfied. For example, if the third term of the buffer contains a word that is lexically marked as 'noun+verb', the word will be assigned the category 'verb' when the second buffer element is the word *to* and the first buffer element is either a verb or adverb. When applied to the word string *expect to call*, this rule correctly analyzes *call* as a verb. Other part-of-speech rules distinguish the preposition *to* from the use of *to* as an infinitive marker, and distinguish the preposition vs. verb uses of *like*.

Ambiguous abbreviations are items such as *no*, which may signify either *number* or the negative particle. Since TDD texts lack punctuation, the only clue to usage in such cases is local context, e.g. the presence of the words *the* or *phone* before *no* are used as disambiguating context to identify *no* as *number*.

3.1.3. Phrasing rules consider part-of-speech, word subclass and length (as measured by word count) to

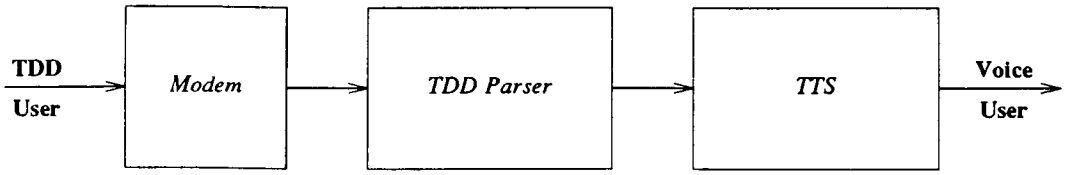

Figure 2: Block Diagram of TDD/TTS System

identify phrase boundary locations. These rules are strictly ordered. In general, they instruct the synthesizer to set off interjections (e.g. *wow, oh ok,* etc.), and to insert a phrase boundary before non-lexical coordinate conjunctions (e.g. *and* in *I don't recall that and am not sure,* see Bachenko and Fitzpatrick (1990:163)), before sentences, and before subordinate conjunctions (*after, during,* etc.). Boundaries are also inserted at noun-verb junctures unless the noun is short, and at prepositional phrase boundaries unless the prepositional phrase is short. A short noun is a single word noun phrase such as a pronoun or demonstrative (*this, that*); a short prepositional phrase is one with a pronominal object (*with me, about it,* etc.). Hence the noun-verb rule will produce the phrasings below, where double bars mark phrase boundaries (this and the prepositional phrase rule are adaptations of the verb and length rules, respectively, given in Bachenko and Fitzpatrick (1990)).

MY CAR || IS HAVING A TRANSMISSION PROBLEM

IT IS HAVING || A TRANSMISSION PROBLEM

Our formulation of the phrasing rules assumes that, in the absence of syntactic structure, the subclass membership, part-of-speech and string position can provide sufficient information to infer structure in many cases. For example, we are assuming that the subclass 'nominative_pronoun', which includes *he, she, we,* etc., acts consistently as the leading edge of a sentence, so that the parser can compensate somewhat for the lack of punctuation by identifying and setting off some top-level sentential constituents. Similarly, prepositions are assumed to act consistently as the leading edge of a prepositional phrase; the parser guesses about prepositional phrase length by checking the word class of the element following the preposition to see if the object is pronominal.

The phrase rules thus attempt to seek out major syntactic constituents. If there is evidence of constituency, the parser may look for a short constituent or it will simply insert a prosodic boundary at a presumed syntactic boundary (e.g. a verb phrase, sentence or subordinate conjunction).

3.2. PARSER IMPLEMENTATION

3.2.1. SYSTEM ARCHITECTURE

The quickest way to incorporate a TDD parser into a ser vice using text-to-speech (TTS) synthesis is to implemen the parser in a separate front-end module to the text-to speech system. The parser filters the input stream from : TDD modem and sends the processed text to the text-to speech system where it is synthesized for the voice tele phone user, as shown in the block diagram in figure 2 This architecture minimizes the need to modify an existing equipment or system. Also, it allows us t maintain and change the parser module without introduc ing substantial, or unpredictable, changes elsewhere i the system.

3.2.2. IMPLEMENTATION

Integrating the TDD parser into a near real time systen architecture is a difficult task. To achieve it, the parse must (a) filter the TDD input stream in real-time in orde to identify tokens, i.e. words, abbreviations, and expres sions, that are suitable for processing by parser rules, an (b) group these tokens into natural sounding phrases tha can be sent to the text-to-speech system as soon as the are formed.

In an ideal situation, it is desirable to parse the entir TDD input before sending the processed text to the text to-speech synthesizer. But the practical situatio demands that the voice user hear TDD text synthesize as soon as it is reasonably possible so that long period of silence can be prevented. Figure 3 below shows th basic architecture chosen to implement the parse described in this paper.

3.2.2.1. The canonical input filter process has to dea with the TDD input characters as they are being typed The output of the canonical filters consists of TDD wor tokens i.e. groups of characters separated by whit spaces. Input characters arrive at irregular speeds wit nondeterministic periods of pauses due to uneven typin by the TDD user. Also incidences of spelling error typographical mistakes, and attempts to amend previousl typed text occur at very irregular rates. Even the TDI modem can contribute text to the input stream that i

seen by the canonical input filter. For instance, the TDD modem might periodically insert a carriage-return character to prevent text wraparounds on the special operator's terminal. Unfortunately, these carriage-return characters could split words typed by the TDD user into incoherent parts, e.g., *advantage* might become *adva*<CR>*ntage*.

Since the voice telephone user needs to hear TDD text synthesized after some, hopefully short, interval of time, the input filter cannot wait indefinitely for TDD characters that are delayed in arriving, as might occur when the TDD user pauses to consider what to type next. Hence, the filter includes an input character timeout mechanism. The timeout interval is set to an appropriately short duration to ensure the timely synthesis of available TDD text, but still long enough to prevent the exclusion of forthcoming input characters.

3.2.2.2. Lexigraphical analysis examines the TDD word tokens to identify contiguous words that should be grouped together as individual units. The multi-word expressions include contractions (e.g. *"it" "s"* which becomes *"it's"*) and commonly used short phrases that can be viewed as single lexical units (e.g.*"my goodness"*, *"as long as"*, and *"mother in law"*). A simple stacking mechanism is used to save tokens that are identified as potential elements of multi-word expressions. The tokens are stacked until the longest potential multi-word expression has been identified, with three words being the maximum. After which the stack is popped and the corresponding structures (described below) are constructed.

3.2.2.3. The lexical lookup process builds a **tdd-term** structure (record) from these tokenized words and multi-word expressions in preparation for invoking the phrasal segmentation rules. Fields in the structure include the tokenized input text (the original orthographic representation), the output orthography, lexical category (Noun, Verb, Adverb, NIL, etc.), word subclass, and other fields used internally by the phrasal segmentation process. At this point in the processing only the input text field has any non-null information. The output orthography, lexical category, and word subclass fields are filled via lexical lookup.

The lexicon is organized into the four fields mentioned above. The tdd-term input text field is compared with the corresponding field in the lexicon until a match is found and the three remaining fields in the matched entry are then copied into the tdd-term structure. If no match is found, then the input text field is copied into the output text field and the other two lexicon fields are set to NIL.

As an illustration, if the single letter *u* is identified as our TDD token, the lexical lookup process might return with a tdd-term structure that looks like:

input text:	"u"
output text:	"you"
lexical category:	NOUN
subclasses:	(DESTRESS_PRONOUN SHORT_SUBJECT)
other fields:	NIL.

For the input text *oic*, the structure might look like:

input text:	"oic"
output text:	"oh, I see"
lexical category:	INTJ
subclasses:	INTERJECTION_1
other fields:	NIL.

3.2.2.4. The phrasal segmentation process applies a modest set of disambiguation and phrasing rules to a sliding window containing three contiguous tdd-term structures. In the start condition the sliding window will not have any tdd-term structures within it. Each new tdd-term structure generated by lexical lookup enters the first term position in the window, bumping existing terms forward one position with the last (third term) discarded after its output orthography is copied into a text buffer awaiting transmission to the text-to-speech synthesizer. The various rules described in Section 3.1 above are then applied to the available tdd-term structures. After a pronounceable phrase is identified, the output orthography of all active tdd-terms is then copied to the TTS text buffer which is subsequently sent to the synthesizer for playback to the voice telephone user. Also, the invocation of a timeout alarm due to tardy TDD input text causes flushing of the sliding window and text buffer into the synthesizer. The sliding window and TTS text buffer are cleared and the rules restarted anew.

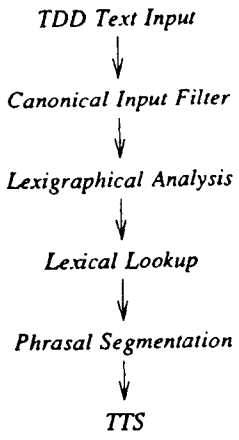

Figure 3: TDD Parser Architecture

Listed below are a few examples of TDD text processed by the parser.

TDD: I DONT THINK SO I WILL THINK ABOUT IT GA
TTS:
I don't think so
I will think about it.
Go ahead.

TDD: HELLO HOW ARE U Q GA
TTS:
hello
how are you?
Go ahead.

TDD: OK YES I AM WILLING TO GIVE INFO GA
TTS:
okay
yes
I am willing
to give information.
Go ahead.

TDD: MY GOODNESS UR MOTHER IN LAW IS HERE GA
TTS:
my goodness
your mother in law
is here.
Go ahead.

4. EVALUATION OF PERFORMANCE

Evaluation of the parser has involved two quite different forms of testing: a field trial and laboratory evaluation. First, the parser was implemented as a component in a version of the Bell Labs text-to-speech synthesizer (Olive and Liberman 1985). The synthesizer forms the core of a telecommunications system that ran for three months as a feature of TRS in California. Several thousand TDD texts were processed by the system. Although restrictions on confidentiality prevented us from collecting actual TDD text data, results of the field trial far surpassed expectations: disconnect rates for text-to-speech calls averaged less than 20% and follow-up surveys indicated a high degree of interest in and acceptance of the technology.

A second type of testing that has enabled us to focus on the parser involves the collection of data from a questionnaire given to TDD users. Phrasing for these data was assigned manually by a linguist unfamiliar with the rules of the parser to allow for comparison with the parser's output.

Several issues arise in the comparison of human judgements of phrasing with those of a phrase parser's output. One of the more ubiquitous is that of phrasal balancing. Apparently acting under rhythmic constraints speakers tend to aim for equivalent numbers of stressed syllables on either side of a break. However, the incorporation of rhythm into phrasing varies from speaker to speaker, as well as being partially dependent on semantic intent. For example, the sentence *so I feel there should be a better system to say bye*, taken from our data, could be phrased either as (a), (b), or (c):

(a) so I feel there should be || a better system to say bye

(b) so I feel || there should be || a better system to say bye

(c) so I feel || there should be a better system || to say bye

If the parser assigns, for example, the phrasing in (a) while the human judge assigns (b) it must be counted as qualitatively different from the parser's assignment of a misleading boundary, where the hearer's understanding of the import of the utterance is altered because of the erroneous boundary placement. An example of misleading boundary placement as assigned by the parser is given below, where the hearer is incorrectly led to interpret *well* as a modification of *see*, rather than as discourse comment.

oh i see well || so i call my boss

In a similar vein, giving equal weight in an evaluation to the locations where pauses do and do not occur is misleading. The absence of a phrasal boundary between two words is much more common than the presence of a boundary, so that predicting the absence of a boundary is always safer and leads to inflated evaluation scores that make comparison of systems difficult. For example, in the (a) sentence above there are 12 potential prosodic events, one after each word. If a given system assigns no breaks in this sentence, and if non-events are given equal weight with events, then the system will get a score for this sentence of 91.6 percent since it gets 11 of the 12 judgments right. Also, if a system assigns one break in this utterance, but puts it in a clearly inappropriate place, say before the word *bye*, it will get a score of 83 percent since it gets 10 of the 12 judgements right. While 83 percent sounds like a decent score for a system that must capture some subjective performance, the method of evaluation has completely failed to capture the fact that assigning an inappropriate prosodic break in this instance has completely misled the listener. Therefore we need to evaluate a phrasing system on the basis of positive occurrences of phrase boundaries only.

Assigning phrases to TDD output is not a clear-cut task. The output is not intended to be spoken and because of the device, it has telegraphic characteristic In addition, many TDD users do not have standard spoken English at their command. Nevertheless, an effort

CATEGORY	ERROR	EXAMPLE
Adverbial modification	75	*why not * surely || i think need interview*
Ambiguous pronoun	59	*who i long || to talk to * it will be great*
Ambiguous Interjection (*sorry, no*)	53	*no * other than that ||*
Verbal Complement	44	*let me *|| hear ||*
Relative Clause	43	*give your calling number * and number *|| you want*
Non-Standard Syntax	39	*there a pause*
Conjunction	36	*that's all * and just once || did we get*
Copular verb	31	*i'm *|| a nice person*
Subordinate clause	20	*as i said before * I feel that way || because*
Idioms	19	*i think the survey interview || is all *|| right with me*
Nominal modification	18	*if i use || pay *|| phone*
Appositive NP	13	*i think *|| they *|| the crs ||*

Figure 4: Distribution of TDD Production Errors

was made to approximate the performance of TRS operators who speak the TDD output to voice users. Care was also taken to mark as parser errors those prosodic events that would mislead the listener. This is a task that, in itself, is problematic because of the human error involved in the judgments. We regard the field trial as the appropriate method of evaluation here, and we use the results of the laboratory test to help us characterize the parser's failures rather than to evaluate the parser.

After the phrasing was assigned manually, the TDD data were run through the parser, and the parser's phrasing was compared with the human judgments. Approximately 20% of the corpus had been used to extrapolate rules for the parser, while the remainder of the corpus was saved for testing only; there was no appreciable performance difference between the two data subsets. The corpus contained 8788 words and, according to the human judgement, 2922 phrases. Since punctuation in these data is sparse, very few of the actual phrase boundaries come "for free." Even so, the parser performed well: in the 2922 phrases it produced 573 misleading errors, rendering 80.4% of the phrases acceptably. (There were 896 sites where the parser produced a phrasing different from that of the human judge, but which we judged to be not misleading.)

The parser's error rate reflects the constraints of its construction as a real-time system, in particular, its three term buffer size, its lack of hierarchical structure building rules and its pared down lexicon. Figure 4 gives a characterization of the most frequently encountered parsing errors, along with their frequency of occurrence, and an example for each characterization. In the examples, '||' represents a prosodic pause and '*' indicates that the parser performed incorrectly at this site.

Most of the parsing errors given in Figure 4 would be resolvable if the parser were to incorporate non-local structural information. For example, the pronouns *it* and *you* function as both subject and object. In a three element buffer, then, the status of *it* in *to it will* is undecidable, since it can be the object of *to* or the subject of *will*. In the context of the sentence *i have friends who i long to talk to it will be great*, where an element corresponding to *who* functions as the object of *to*, the function of *it* as subject of *will be great*, and the concomitant prosodic break before *it*, are apparent, but only when the structure of the *who* relative clause is available.

The errors involving non-standard syntax would require sublanguage rules that indicate the possibility of non-overt subjects (*oh i see understand*) and copulas (*there a pause*) among other things, but again, given the limitation to local information and the lack of punctuation, this is not straightforward. For example, *oh i see understand* could continue as *that i don't speak well.*

A smaller set of errors is undecidable even given non-local structural information, and require further pragmatic knowledge of the discourse. For example, the decision as to which clause the adverb *occasionally* modifies in *other than that the services is great occasionally some operators are pretty slow* depends on knowing that one does not give expansive praise to something that happens only occasionally.

In general, it appears that the parser's accuracy in phrasing the incoming messages cannot be improved without a severe loss in real time efficiency that the storage of hierarchical structure would involve. Given this situation, it is worthwhile to consider that, despite what is probably about a 20% error rate in the system,

consumers used it successfully and willingly. It may be that the system did no worse than the trs operators who, unlike our laboratory linguist, do not have the luxury of stopping to consider the felicity of a particular phrasing. This may be compounded with the possibility that users may be able to compensate more easily for machine degradation of an utterance than for an operator's error, since their expectations of the latter's performance are greater.

5. CONCLUSION

We have described a text-to-speech parser for conversational texts generated by users of TDD's. The parser's main tasks are to provide some regularization of non-standard items and to determine prosodic phrasing of the text. Phrasing allows processing to take place in near real time because the synthesizer can generate speech while the TDD message is being typed instead of waiting for the finished text. Finally, although incorporating a relatively small vocabulary and rule set, the parser has proven unexpectedly successful in both laboratory and field tests.

REFERENCES

Bachenko, J. A Taxonomy of Syntactic Variation in Written Language of the Deaf. Unpublished data, 1990.

Bachenko, J. and E. Fitzpatrick. A Computational Grammar of Discourse-Neutral Prosodic Phrasing in English. *Computational Linguistics*, 16:155-17, 1990.

Charrow, V. *Deaf English*. Technical Report 236, Institute for Mathematical Studies in the Social Sciences, Stanford University, 1974.

Emorine, O. M. and P. M. Martin. The Multivoc Text-to-Speech System. *Proceedings of the Second Conference on Applied Natural Language Processing (ACL)*: 115-120, 1988.

Hindle, D. User Manual for Fidditch, a Deterministic Parser. *NRL Technical Memorandum*#7590-142. 1983.

Kukich, K. Spelling Correction for the Telecommunications Network for the Deaf. *Communications of the ACM*, 1992.

Marcus, M. *A Theory of Syntactic Recognition for Natural Language*. Cambridge, MA: MIT Press, 1980.

Olive, J. P. and Liberman, M. Y. Text-to-Speech--An Overview. *Journal of the Acoustic Society of America*, Supplement 1:78, S6, 1985.

O'Shaughnessy, D. D. Parsing with a Small Dictionary for Applications such as Text-to-Speech. *Computational Linguistics*, 15:97-108, 1988.

Suri, L. *Language Transfer: A Foundation for Correcting the Written English of ASL Signers*. University of Delaware Technical Report #91-19, 1991.

Tsao, Y.-C. A Lexical Study of Sentences Typed by Hearing-Impaired TDD Users. *Proceedings of 13th International Symposium, Human Factors in Telecommunications*, Torino, Italy, 197-201, 1990.