Text Analysis and Word Pronunciation in Text-to-speech Synthesis

Mark Y. Liberman
Kenneth W. Church

AT&T Bell Laboratories
600 Mountain Ave.
Murray Hill, N.J., 07974

1. Introduction: the structure of a text-to-speech system

A text-to-speech (TTS) system maps a sequence of numbers representing the characters of a text into another sequence of numbers representing the samples of an acoustic waveform. It is convenient to divide this mapping into four types of processing, which may be called *text analysis*, *word pronunciation*, *phonetic interpretation*, and *signal generation*.

Text analysis includes such things as dividing the text into words and sentences, assigning syntactic categories to words, grouping the words within a sentence into phrases, identifying and expanding abbreviations, recognizing and analyzing expressions such as dates, fractions, and amounts of money, and so on. *Word pronunciation* is the problem of translating orthographic words -- words in ordinary spelling -- into phonological words -- words whose sound is expressed in a sort of rationalized spelling, using an alphabet that corresponds to the set of broad phonetic[1] segments found in the pronunciation guide of a dictionary.

The result of *text analysis* and *word pronunciation* is an explicit representation of the linguistic structure of the message encoded in the original text. The *phonetic interpretation* phase of a TTS system assigns quantitative phonetic values to the various aspects of this linguistic representation: durations of phonetic segments, F0 target values for pitch accents, and so forth. The *signal generation* phase of a TTS system then uses this detailed phonetic specification to produce time functions of the control parameters for an acoustic or articulatory speech synthesis model [examples, references], which are then used to calculate the samples of the speech waveform.

In this paper, we will discuss the *text analysis* and *word pronunciation* aspects of the text-to-speech problem. The nature and difficulty of these problems depends very much on the language and on the genre of text. For instance, in ordinary English text it is nearly trivial to divide the text into words, but in Chinese, this is a very hard problem [ref Sproat], since the average word length is about 2.3 characters, the set of words is open-ended, and boundaries between words are not indicated in the orthographic system. On the other hand, the problem of Chinese word pronunciation can be almost completely solved by table lookup on characters with a simple combination rule -- there are only a few thousand characters, each of which corresponds almost always to a fixed syllable-plus-tone, and the pronunciation of a multi-character word is almost always just the concatenation of the character pronunciations with rightward stress assignment [weasel footnote, refs]. By contrast, high-accuracy word pronunciation in English is quite hard, requiring large tables of information about the pronunciation of words and word fragments, complex rules to bring these tables to bear on particular cases, and accurate text analysis to deal with contextually variable pronunciations.

Even within a single language, different types of text can raise very different sorts of difficulties. For instance, text analysis and word pronunciation programs that do a good job in reading the newspaper may fail quite badly in dealing with some database text fields, which are all capital letters with erratic punctuation and spacing:

---

1. or surface phonemic

```
WILLIAM F BYRNE 4025 SANDY HILL RD W SPRINGFIELD MA 01075
MARTIN A PENKALA 200 THAMES PKWY APT 1C PROSPECT HTS IL 60068
VIETTA J MCSWAIN 72 1/2 KING ST CHARLESTON SC 29376

01011   GEL-KAM DTP FRUIT&BERRY 3.5OZ   0.4%
01552   NYSTATIN TOPICAL OINT    15GM100000U
01610   MULTIVITAMIN W/FL CHEW TAB031.5MG
01725   SPECTROBID SUSP         200CC125MG
01727   ANTIBIOTIC OTIC SOLN 10CC
01981   HYDROFLUMETH+RESERPINE TAB12725/.125
02052      BENZAMYCIN TOP.GEL 23.3GM
```

In our experience, applications are never "just plain text;" each application highlights a certain set of problems, and stresses (or breaks!) certain of a system's algorithms. The range of problems and solutions is very large, and we do not see a clean distinction between "standard" or "general" text on one hand, and "special" text on the other. Instead, each application has features of varying degrees of generality, from those that are shared with nearly all English text to those that are entirely unique.

This paper will feature the approaches that we know best, primarily those that have been developed in our group at AT&T Bell Laboratories and [acknowledgements, references], but [sketch of other stuff]. We will discuss a range of genres, and highlight algorithms that are adaptable or even directly trainable from text samples.

Throughout, our practice will be be to evaluate each problem in terms of its frequency in a particular type of text, and to evaluate each solution in terms of its performance and its costs. Except where explicitly noted, all examples will quoted from some source of on-line text, rather than made up.

2. Text Analysis

There are two reasons for a TTS system to do text analysis. One reason is that word pronunciation sometimes depends on usage: **I** can be a pronoun or a Roman numeral, **wind** can rhyme with "bind" or "binned," **Dr.** can be "doctor" or "drive," **2/3** can be "two thirds" or "February third" or "two slash three." A second, equally important reason for text analysis is that its results will be used to modulate the pitch, timing and amplitude of the speech so as to present the text's message clearly. In other words, we want the program to read as if it were a skilled speaker who had understood the text.

Table 1 presents some symptoms of the phonetic "shape" of a spoken phrase. It shows the effect of position in a ten-digit telephone number on the duration, second-format frequency, pitch, and amplitude of the vowel **[o]** in the Spanish digit "dos." Each number represents the mean of ten measurements, in a set of 100 10-digits numbers arranged so that each digit occurs equally often in each position, and each pair of digits occurs equally often spanning each pair of positions. Durations are given in milliseconds, and F2 and F0 values are given in Hz. The F2, F0 and RMS amplitude measures are the mean of the central 40 milliseconds of each vowel, averaged over the ten cases in each position.

Position	1	2	3	4	5	6	7	8	9	10
Duration	110	110	144	92	98	131	100	115	98	152
F2	1326	1366	1261	1342	1335	1273	1326	1282	1309	1179
F0	158	157	178	152	139	173	143	171	144	115
RMS	2711	2446	3075	1808	1985	2316	1547	2565	1697	1225

The 3+3+4 grouping of the digits is plain to see in all measures.

In general, this sort of phrasal shape expresses what we might call the "information structure" of the text [refs], and thus to shape a phrase skillfully requires understanding the text. Full machine understanding of unrestricted text remains a far-off goal, so in a text-to-speech system we do what we can, ranking the problems according to their impact on speech quality, and trying to find the approximate solutions that give the best overall results for a particular application.

2.1 Text Format and Text Structure

Obvious point. Examples: name-and-address, drug names, Challenger transcripts, email headers, dictionaries etc. In most applications, these issues now dominate performance figures. Special-purpose filters; Need for "hooks" on TTS input to support them. Is a general solution possible? or rather, how general a solution?

2.2 Sentence Division

Summary of heuristic solutions, Riley work.

2.3 Part-of-Speech Assignment

It is well-known that part of speech depends on context. The word ‘‘table,’’ for example, can be a verb in some contexts (e.g., ‘‘He will table the motion’’) and a noun in others (e.g., ‘‘The table is ready’’). A program has been written which tags each word in an input sentence with the most likely part of speech.

- He/PPS will/MD table/VB the/AT motion/NN ./.
- The/AT table/NN is/BEZ ready/JJ ./.

(PPS = subject pronoun; MD = modal; VB = verb (no inflection); AT = article; NN = noun; BEZ = present 3rd sg form of ‘‘to be’’; JJ = adjective; notation is borrowed from [Francis and Kucera, pp. 6-8])

Part of speech tagging is an important practical problem with potential applications in many areas including speech synthesis, where it is clear that pronunciation sometimes depends on part of speech, as demonstrated by the following three examples. First, there are a few words like ‘‘wind’’ where the noun has a different vowel than the verb. That is, the noun ‘‘wind’’ has a short vowel as in ‘‘the wind is strong,’’ whereas the verb ‘‘wind’’ has a long vowel as in ‘‘Don’t forget to wind your watch.’’ Secondly, there are a few very common function words such as ‘‘that’’ which often become cliticized (reduced), but only in certain usages. For instance, ‘‘that’’ is almost always reduced when it introduces a subordinate clause, e.g., ‘‘It is a shame that [schwa] he is leaving,’’ but only in that usage; it is never reduced when used as a demonstrative pronoun, e.g., ‘‘Did you see THAT [no schwa]?’’ Thirdly, note the difference between ‘‘oily FLUID’’ and ‘‘TRANSMISSION fluid’’; as a general rule, an adjective-noun sequence such as ‘‘oily FLUID’’ is typically stressed on the right whereas a noun-noun sequence such as ‘‘TRANSMISSION fluid’’ is typically stressed on the left. These are but three of the many constructions which would sound more natural if the synthesizer had access to accurate part of speech information.

The program uses a linear time dynamic programming algorithm to find an assignment of parts of speech to words that optimizes the product of (a) lexical probabilities (probability of observing part of speech i given word i), and (b) contextual probabilities (probability of observing part of speech i given n following parts of speech). Probability estimates were obtained by training on the Tagged Brown Corpus [Francis and Kucera], a corpus of approximately 1,000,000 words with part of speech tags assigned laboriously by hand over many years. Program performance is encouraging (95-99% ‘‘correct’’, depending on the definition of ‘‘correct’’). Performance is good enough that a growing user population has found that it meets their needs better than most alternative programs that they have access to.

It is surprising that a local ‘‘bottom-up’’ approach can perform so well. Most errors are attributable to defects in the lexicon; remarkably few errors are related to the inadequacies of the extremely over-simplified grammar (a trigram model over parts of speech). Apparently, ‘‘long distance’’ dependences are not very important, at least most of the time. For the tagging application, we believe the N-gram approximation is not as bad as some approximations that are often made in practice (e.g., ignoring probabilities, as most AI natural language parsers do).

Statistical N-gram models were quite popular in the 1950s, and have been regaining popularity over the past few years. The IBM speech group is perhaps the strongest advocate of N-gram methods, especially in other applications such as speech recognition. They have also experimented with the tagging application (Jelinek, 1985). [Leech, Garside and Atwell], also found N-gram models highly effective; they report 96.7% success in automatically tagging the Lancaster-Oslo/Bergen (LOB) Corpus, using a bigram model modified with heuristics to cope with more important trigrams. [DeRose] has observed that their implementation could have benefited from the dynamic programming optimization (and a representation of lexical probabilities that did not introduce as much quantization error).

2.4 The Proposed Method

Consider once again the sentence, ''. . I see a bird . .'' (We will not discuss the front end prepass which tokenizes the input sentence into words and pads sentences with two special tokens at each end.) The problem is to find an assignment of parts of speech to words that optimizes both lexical probabilities $Prob(p_i | w_i)$ and contextual probabilities $Prob(p_i | p_{i+1} p_{i+2})$, both of which are estimated from the Tagged Brown Corpus. Conceptually, enumerate all sequences of parts of speech $\{p_i\}$ that could correspond to the input sentence of N words $\{w_i\}$. Then score each sequence by

$$\underset{\{p_i\}}{MAX} \prod_{i=1}^{N} \frac{Prob(p_i | w_i)\, Prob(p_i | p_{i+1} p_{i+2})}{Prob(p_i)}$$

and select the highest scoring sequence. (We will not discuss the normalization factor, $Prob(p_i)$, which compensates for the fact that $Prob(p_i)$ is counted twice in the numerator, once in the lexical probabilities and once in the contextual probabilities.)

The ''I see a bird'' example is illustrated in figure 1 below. There are 8 possible part of speech sequences, labeled A1 through A8, duplicated just below for convenience.

	.	.	I	see	a	bird	.	.
A1	.	.	PPSS	VB	AT	NN	.	.
A2	.	.	PPSS	VB	IN	NN	.	.
A3	.	.	PPSS	UH	AT	NN	.	.
A4	.	.	PPSS	UH	IN	NN	.	.
A5	.	.	NP	VB	AT	NN	.	.
A6	.	.	NP	VB	IN	NN	.	.
A7	.	.	NP	UH	AT	NN	.	.
A8	.	.	NP	UH	IN	NN	.	.

That is, the pad token ''.'' has just one part of speech: ''.''; the word ''I'' can be either a pronoun PPSS or a proper noun NP (according to [Francis and Kucera]); the word ''see'' is either an uninflected verb VB or an interjection UH; ''a'' is either an article AT or a preposition IN (from French); ''bird'' is always a noun NN.

In figure 1, each part of speech sequence is followed by two two rows of numbers corresponding to the lexical and contextual probabilities. The probabilities for A1 are repeated here:

A1	.	.	PPSS	VB	AT	NN	.	.
lex	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
con	0.99	0.20	0.07	0.07	0.23	0.25	1.00	1.00

The lexical probabilities are all nearly 1. In other words, it is very likely that ''I'' is a pronoun, ''see'' is a verb, ''a'' is an article and ''bird'' is a noun. The contextual probabilities are also quite large. The third number, 0.07, for example, denotes the probability of finding a pronoun before a verb and and article. It was computed by dividing the number of PPSS, VB, AT trigrams in the training corpus by the number of VB, AT bigrams.

The bottom line score for A1 is 10^{-4}, which is computed by multiplying all of the lexical and contextual probabilities. As one can see from the right most column in table 1, A1 has a much higher score than any other part of speech sequence; A1 (pronoun, verb, article, noun) scores 10,000 times better than A2 (pronoun, verb, preposition, noun), the next best sequence.

Table 1

	.	.	I	see	a	bird	.	.	
A1	.	.	PPSS	VB	AT	NN	.	.	
lex	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	
con	0.99	0.20	0.07	0.07	0.23	0.25	1.00	1.00	10^{-4}
A2	.	.	PPSS	VB	IN	NN	.	.	
lex	1.00	1.00	1.00	1.00	10^{-4}	1.00	1.00	1.00	
con	0.99	0.20	0.08	0.03	0.13	0.25	1.00	1.00	10^{-9}
A3	.	.	PPSS	UH	AT	NN	.	.	
lex	1.00	1.00	1.00	10^{-3}	1.00	1.00	1.00	1.00	
con	0.99	1.00	0.00	0.00	0.23	0.25	1.00	1.00	0
A4	.	.	PPSS	UH	IN	NN	.	.	
lex	1.00	1.00	1.00	10^{-3}	10^{-4}	1.00	1.00	1.00	
con	0.99	1.00	0.00	0.00	0.13	0.25	1.00	1.00	0
A5	.	.	NP	VB	AT	NN	.	.	
lex	1.00	1.00	10^{-4}	1.00	1.00	1.00	1.00	1.00	
con	0.97	0.03	0.01	0.07	0.23	0.25	1.00	1.00	10^{-10}
A6	.	.	NP	VB	IN	NN	.	.	
lex	1.00	1.00	10^{-4}	1.00	10^{-4}	1.00	1.00	1.00	
con	0.97	0.03	0.01	0.03	0.13	0.25	1.00	1.00	10^{-15}
A7	.	.	NP	UH	AT	NN	.	.	
lex	1.00	1.00	10^{-4}	10^{-3}	1.00	1.00	1.00	1.00	
con	0.97	0.00	0.00	0.00	0.23	0.25	1.00	1.00	0
A8	.	.	NP	UH	IN	NN	.	.	
lex	1.00	1.00	10^{-4}	10^{-3}	10^{-4}	1.00	1.00	1.00	
con	0.97	0.00	0.00	0.00	0.13	0.25	1.00	1.00	0

2.5 The Dynamic Programming Solution

In this example, there are eight possible sequences since there were three 2-ways ambiguous tokens (''I'', ''see,'' and ''a''), and $2^3 = 8 paths$. In general, let us assume that words are no more than k ways ambiguous, where k is about 10, and that input sentences are no more than N words long, where N is about 100. Then, if the search were to literally enumerate all part of speech sequences, it might need to look at $k^N (\approx 10^{100})$ part of speech sequences. Fortunately, there is a dynamic programming solution to the search since the scoring function can see only 2 words away.

Suppose, (for convenience only), we start the search from the end of the sentence. First we consider the possible part of speech sequences for ''bird'':

0.25 bird/NN ./. ./.

The score, 0.25, was computed by multiplying the lexical probability for NN given ‘‘bird’’ (1.00) and the contextual probability of NN given the following two parts of speech (0.25). We now consider the possible part of speech sequences for ‘‘a’’:

0.06	a/AT	bird/NN	./.	./.	3×10^{-6}	a/IN	bird/NN	./.	./.

The score, 0.06, was computed by multiplying the previous score (0.25) with the lexical probabilities for AT given ‘‘a’’ (1.00) and the contextual probability for AT given the next two parts of speech (0.23). We now consider the possible part of speech sequences for ‘‘see’’: (The score for the last two paths is actually slightly more than 0, but we won’t deal with that here.)

4×10^{-3}	see/VB	a/AT	bird/NN	./.	./.
1×10^{-7}	see/VB	a/IN	bird/NN	./.	./.
0	see/UH	a/AT	bird/NN	./.	./.
0	see/UH	a/IN	bird/NN	./.	./.

Now, find assignments of ‘‘I’’ and score. Note, however, that it is no longer necessary to hypothesize that ‘‘a’’ might be a French preposition IN because all four paths, ‘‘I/PPSS see/VB a/IN bird/NN,’’ ‘‘I/NP see/VB a/IN bird/NN,’’ ‘‘I/PPSS see/UH a/IN bird/NN’’ and ‘‘I/NP see/UH a/AT bird/NN’’ score no better than some other path and there is no way that any additional input could make any difference. In particular, the path, ‘‘I/PPSS see/VB a/IN bird/NN’’ scores no better than the path ‘‘I/PPSS see/VB a/AT bird/NN,’’ and additional input will not help ‘‘I/PPSS see/VB a/IN bird/NN’’ because the contextual scoring function has a limited window of three parts of speech, which is not enough to see past the existing ‘‘I/PPSS’’ and ‘‘see/VB.’’

10^{-4}	I/PPSS	see/VB	a/AT	bird/NN	./.	./.
10^{-9}	I/NP	see/VB	a/AT	bird/NN	./.	./.
0	I/PPSS	see/UH	a/AT	bird/NN	./.	./.
0	I/NP	see/UH	a/AT	bird/NN	./.	./.

The search continues two more iterations for the two pad characters and ultimately concludes that part of speech sequence A1 is the best. Because of the dynamic programming optimization, only $k^2 (\approx 10^2)$ paths need to be kept around as each of the N input words are processed. Thus, the dynmanic programming optimization reduces the search space from k^N down to only Nk^2.

2.6 Smoothing Issues

Some of the probabilities are very hard to estimate by direct counting. Consider, for example, the lexical probabilities. We need to estimate how often each word appears with each part of speech. Unfortunately, it is not possible to estimate all of these parameters. If there were only 50,000 words and 10 parts of speech, there would be 500,000 parameters to estimate. Given that we have only 1,000,000 words of training material, it is just not possible to estimate so many parameters even if the data were nicely distributed (and they aren’t). In general, no matter how much text we look at, there will always be a large tail of words that appear only a few times. In the Brown Corpus, for example, 40,000 words appear five times or less. For instances, the word ‘‘yawn’’ appears once as a noun and once as a verb. What is the probability that it can be an adjective? It is impossible to say without more information. Fortunately, conventional dictionaries can help alleviate this problem to some extent. We add one to the frequency count of possibilities in the dictionary. For example, ‘‘yawn’’ happens to be listed in our dictionary as noun/verb ambiguous. Thus, we smooth the frequency counts obtained from the Brown Corpus by adding one to both possibilities. In this case, the probabilities remain unchanged. Both before and after smoothing, we estimate ‘‘yawn’’ to be a noun 50% of the time, and a verb the rest. There is no chance that ‘‘yawn’’ is an adjective.

In some other cases, smoothing makes a big difference. Consider the word ‘‘cans,’’ which appears 5 times as a plural noun and never as a verb in the Brown Corpus. The lexicon (and its morphological routines), fortunately, give both possibilities. Thus, the revised estimate is that ‘‘cans’’ appears 6/7 times as a plural noun and 1/7 times as a verb.

Proper nouns and capitalized words are particularly problematic; some capitalized words are proper nouns and some are not. Estimates from the Brown Corpus can be misleading. For example, the capitalized word ‘‘Acts’’ is found twice in the Brown Corpus, both times as a proper noun (in a title). It would be a mistake to infer from this evidence that the word ‘‘Acts’’ is always a proper noun. Most proper nouns are now dealt with in a prepass that labels words as proper nouns if they are ‘‘adjacent to’’ other capitalized words (e.g., ‘‘White House,’’ ‘‘State of the Union’’) or if they appear several times in a discourse and are always capitalized. This very simple prepass appears to work well enough that the main routine does not need a very good model of lexical probabilities for proper nouns.

The lexical probabilities are not the only probabilities that require smoothing. There are also problems with contextual probabilities, though they aren’t as bad because the model has many fewer parameters, and therefore, the model is somewhat better trained. Ad hoc solutions such as adding 1 to all of the frequencies seem to be acceptable. Other solutions such as [Katz] should be tried.

2.7 General Parsing

As pointed out in (Allen et al, 1987, p. 40), the text-to-speech application places somewhat unusual demands on a parser. The parser should have a broad (though possibly superficial) coverage of unrestricted text, rather than a deep analysis of a restricted domain. In addition, it is important to tune the trade off between type I and type II errors appropriately. In particular, it is probably more serious to insert a spurious boundary than to leave a real one out. Finally, there are severe constraints on time; the parser must run in real-time. Memory is also a critical resource in many practical implementations.

> ‘‘The parser for the text-to-speech system is designed to satisfy a unique set of constraints. It must be able to handle arbitrary text quickly, but does not need to derive semantic information. Many parsers attempt to build a deep structure parse from the input sentence so that semantic information may be derived for such uses as question-answering systems. The text-to-speech parser supplies a surface structure parse, providing information for algorithyms which produce prosodic effects in the output speech...
>
> It is well knwn that parsing systems which parse unrestricted text often produce numerous ambiguous or failed parses. Although it is always possible to choose arbitrarily among ambiguous parsings, a failed parse is unacceptable in the text-to-speech system. When one examines ambigous results from full sentence-level parsers, one finds... much of the structure at the phrase level has been correctly determined. The phrase-level parser takes advantage of this reliability, producing as many phrase nodes as possible for use by the MITalk prosodic component.
>
> The phrase-level parser uses comparatively few resources and runs in real-time. This is quite unusual for parsers which handle unrestricted text, but is necessary for a text-to-speech system.’’

The phrase-level parser in the MITalk system produces output such as (Allen et al, 1987, p. 51):

Noun Group	Most of the exercises
Verb Group	are
Noun Group	translations
Unclassified	.

Verb Group	There are
Noun Group	several inportant changes
Prepositional Phrase	in the way
Noun Group	the quantifier rules
Verb Group	will work
Prepositional Phrase	for the remainder of the course
Unclassified	.

We agree with Allen et al; for synthesis applications, it is currently necessary to take a very robust, conservative and unambitious approach toward parsing. In fact, we have taken an even more conservative approach than they have. It has been our experience that a very simple pattern of the form:

{function word} {context word}**

will produce results that are about as good as can be expected. For example, in this case, this pattern would produce:

Most of the exercises
are translations.
There are several important changes
in the way
the quantifier rules
will work
for the remainder
of the course.

Such a method works, of course, because English is largely a right-headed language, whose grammatical words tend to pile up at the left edge of constituents, being what syntacticians call *specifiers*. We call such a sequence of function-words followed by content words a *function-word group*, or *f-group*. Our simple f-group parser can be easily be improved, to a certain extent, at the expense of slightly greater complexity, and some blurring of the distinction between For example, a pronoun in the objective case, such as **him** or **them**, is more likely to function like a noun, ending a unit, than like a determiner, beginning one. In addition, bare tensed verb forms are more likely to function like an auxiliary, starting a unit. Thus to avoid confusion, members of the *function word* category should perhaps instead be called *chinks*, and members of the *content word* category should be called *chunks*. Then we put objective pronouns into the *chunk* category, and tensed verb forms into the *chink* category, and greedily match the pattern **{chink* chunk*}***, a process that we can now dub the *chinks ’n chunks* algorithm.[2] These modifications generally produce better behavior:

2. This nomenclature is inspired by an indexing algorithm described in (Tukey XX).

WAS:
The United States Supreme Court decision produced angry protest marches.
IS:
The United States Supreme Court decision
produced angry protest marches.

WAS:
I asked
them if they were going home
to Idaho
and they said yes
and anticipated
one more stop
before getting home
IS:
I asked them
if they were going home
to Idaho
and they said yes
and anticipated one more stop
before getting home

WAS:
and a Kansas state trooper helped
them on Interstate 70
near the Colorado border
IS:
and a Kansas state trooper
helped them
on Interstate 70
near the Colorado border

WAS:
Ellsberg testified Friday
that a protester
much like those on trial persuaded
him to leak
the Pentagon Papers.
IS:
Ellsberg
testified Friday
that a protester
much like those on trial
persuaded him
to leak
the Pentagon Papers.

Unfortunately, **her** is ambiguous between the genitive form, which usually begins a unit, and the objective pronoun, which usually ends one, so accurate tagging is a precondition for separating the cases. Likewise, tensed verb forms are not easy distinguish from past participles (in the past form) and from nouns (in the present forms). Thus to get the full benefit of the *chinks 'n chunks* algorithm, we need a robust part of speech program like that in (Church, 1988). When the parser for our text-to-speech system was first designed, its tagging algorithm was not sufficiently reliable to justify treating tensed verbs as *chinks*, since such a heuristic would produce too many inappropriate boundaries.

Comparing the extremely simple *chinks 'n chunks* approach to the outcome of more complex parsers, we have found that the division into small f-groups-sized units is nearly the same, while the complex parsers' decisions on grouping of larger units are wrong too often to be relied on.

Of course, we must make it clear that this simple algorithm is not very good. It can produce mistaken groupings, as in the example below:

RESULT:
The outspoken conspiracy theorist
will not pass
up a chance
to speak.
SHOULD BE:
The outspoken conspiracy theorist
will not pass up
a chance
to speak.

More important, the groupings that it creates are too small be very useful -- what we really need to know is how the f-groups go together to form sentence-sized units. As a result, only a rather weak amount of phonetic modulation is appropriate to mark the f-group structuring. In order to prevent long sentences from being too monotonous, we can also impose mildly alternating patterns of tonal prominence on accented words within f-groups, and f-groups within phrases. Again, we would really like to know what the text's information structure is, and in the absence of effective methods to find it, we must follow the Hippocratic principle of "first, do no harm."

We believe that current research in text analysis, combining sensible models of linguistic structure with techniques for learning from very large text corpora [refs], is likely to produce techniques that work well enough to be used safely in TTS applications.

2.8 A Focused Approach: Attributive Tags

Summary of Kathy Baker paper.

2.9 Another Idea

Summary of Eva paper.

2.10 How to Speak Trees

Summary and evaluation of Bachenko paper.

3. Text Normalization

When we try to calculate the pronunciation of words, we find that a certain fraction of them are not "ordinary" words, but are special in some way. Some obvious examples are words made up of letters not in the 26 from A to Z -- digits, dollar signs, and other such things. Another set of "special" words, whose pronunciation has to be treated somewhat differently, consists of acronyms and abbreviations. In newswire text, 3-4% of all words are "special" in one of these ways. Of course, other genres of text may have much larger or much smaller proportions.

In addition to having special properties from the point of view of word pronunciation, these cases also often require some special interaction with the text analysis component. For example, we can't expect every date and dollar amount to be found in a dictionary of lexical probabilities for our part-of-speech analyzer -- we need a pattern matcher that can recognize such things when they occur, and give them an appropriate value in the input to the sequence optimizer.

Treatment of such cases becomes crucial to statistics on word pronunciation performance once we are getting high accuracy on fully-spelled alphabetic words.

3.1 Acronyms

There are two ways to write acronyms in English -- with periods, as in **I.R.S.**, and without periods, as in **IRS**. Both patterns can occur, even for the same acronym in the same style of text. In a corpus of about 12 million words of Associated Press newswire text sampled during 1987, we find:

I.R.S/IRS	S.E.C/SEC	L.A./LA	N.Y./NY	U.S./US	U.S.A./USA	U.S.S.R./USSR	
With periods	22	0	107	209	19873	112	65
No periods	428	413	8	14	95	294	19

Overall, there were about 70 thousand acronyms without periods, and about 31 thousand acronyms with periods. Obviously the proportions (whether overall or for a particular acronym) might be quite different in a different sort of text -- the point is just that both patterns can occur.

In English, an acronym can be pronounced in two ways: it can be spelled out, as when **HUD**[3] is pronounced "aitch you dee," or it can be pronounced as if it were a more ordinary word, as when **HUD** is pronounced as a monosyllable rhyming with "dud." Although some acronyms, like **HUD**, are variable, most acronyms strongly prefer one pronunciation or the other. Thus we have never heard **AIDS** spelled out, but we have never heard **GOP** pronounced to rhyme with "pop." It seems rather hard to predict how any particular case will go -- thus **DEC** (Digital Equipment Company) is "wordified" and pronounced like "deck," and **OPEC** (Organization of Petroleum Exporting Countries) is pronounced like "O-peck," while **NEC** (Nippon Electric Company), **SEC** (the Securities and Exchange commission), and **EEC** (European Economic Community) are all normally spelled out.

It is also sometimes hard to predict how a "wordified" acronym will actually be pronounced -- **SCSI** (Small Computer Standard Interface [check...]) might have been "sexy" instead of "scuzzy," if the culture of the computer industry had been a bit different. Thus the pronunciation of acronyms requires a lexicon, both for chosing acronyms to "wordify," and for deciding how to pronounce them, if we are to handle it with high accuracy. A reasonable approach is to treat the the spelled-out pronunciation as the default, and to use a pronouncing dictionary for those cases that are normally "wordified," a small sample of which is given below:

```
AIDS    ASCAP   CBEMA   MIDI    NATO    OPEC    SAC     UNESCO
ANSI    AWACS   CSLI    NASA    NORAD   PAC     SALT    UNICEF
ARCO    CAD/CAM ESOP    NASDAQ  NOW     RICO    SAM
```

In AP newswire text, about 10% of acronym tokens are "wordified," and a dictionary with a few hundred entries will capture the bulk of these. In some genres, such as scientific or bureaucratic prose, acronyms are coined and used more freely, and mingle with coined proper names formed by other methods. Such cases form about 4% of the words in a sample of scientific abstracts, for instance, and many of the examples seem appropriate to pronounce rather than spell out, even when they are unknown to the reader:

---

3. the acronym for "Housing and Urban Development."

```
lar to that embodied in the laser fusion code   MEDUSA. The theory presented here can be a
ata. The 70 group constants were generated by   MINX code with the self-shielding factor t
e isotope uranium-235) for a research reactor   (TRIGA Mark III) in Thailand. 1 table.
ons has been studied in the experiment on the   SKAT bubble chamber with freon filling exp
of analyses made using this system are given,   SINTER has been designed and produced with
broadening of low-energy electron diffraction   (LEED) spot profiles. This spacing is then
er in which these problems were solved in the   MANTECH facility is illustrated. Finally a
```

Before leaving the topic of acronyms we should note that they can freely occur in the plural and the genitive case. These are easy to treat by rule: the final **s** or **'s** should not be spelled out: thus **CEOs** is [*ipa*(*sE.E.Oz*)] *not*[ipa(sE.E.O.es)]. Agentive and participial forms are rarer, but do occur:

```
        They were working with police on  IDing.
 1944, four years later became the first  DFLer elected to the U.S. Senate.
             CBS recently signed another  ABCer, Kathleen Sullivan, for its new m
rmer U.S. Attorney in Manhattan, in 1984  RICOed oil trader Marc Rich essentially
s at Justice were wary of Mr. Giuliani's  RICOing of Princeton/Newport, but acqui
                          Aside from the  RICOing of Drexel, prosecutorial abuses
OK ministers had received hefty sums for  OKing the purchase of F-16 Fighting Fal
```

In the above discussion, we have acted as if acronyms can be unambiguously detected. In fact, there are considerable possibilities of confusion, with Roman numerals, with abbreviations, and especially in text that is capitalized for emphasis or as part of a title, or in text that is mono-case (whether upper or lower).

As examples of the Roman numeral case, compare **IV drug user** with **Henry Hunter Hudnut IV**. Repeated references to "World War Aye Aye" are especially annoying. Some abbreviations are often given in all caps, especially American two-letter state postal codes such as CT, MA, AL. Such codes are typically not followed by a period -- and of course a period, if present, may also be considered to be a mark of sentence punctuation rather than a signal of an abbreviation. [example].

Some examples of capitalization in titles, reported signs, brand names and so on are shown below:

```
 circuit is the   AP WEEKEND ENTERTAINMENT AND ARTS package for weekend editions,
                  WINDOW TINTING in autos is reviewed by the U.S.; states peer in
                  DOWNEY SAVINGS & LOAN ASSOCIATION, Newport Beach, Calif., declar
                  CORPORATE DOWNSIZING digs deeper.
the other day:    "COW MILK NOT DOW MILK" and "AGENT ORANGE WORKS -- ASK MERRELL D
                  DOWN PAYMENTS have become the biggest barrier to homeownership,
```

Note that there can be actual acronyms interspersed among ordinary words in these capitalized sections. In addition, text fields of everyday business databases are often all caps -- and often lack punctuation as well:

```
01011   GEL-KAM DTP FRUIT&BERRY 3.5OZ   0.4%
01205   YOHIMEX TABS (YOHIMBINE HCL) 5.4MG
01406   SOMOPHYLLIN ORAL LIQ DYE-FREE
01539   VITAMIN B CMPLX RX(M-PLEX)076
01552   NYSTATIN TOPICAL OINT    15GM100000U
01610   MULTIVITAMIN W/FL CHEW TAB031.5MG
01725   SPECTROBID SUSP         200CC125MG
01727   ANTIBIOTIC OTIC SOLN 10CC
01766   ERYTHRITYL TETRANITRATE-DISC10MG
01973   TRICHLORMETHIAZIDE W/RES. TAB4MG
01981   HYDROFLUMETH+RESERPINE TAB12725/.125
02052      BENZAMYCIN TOP.GEL 23.3GM
```

```
WILLIAM F BYRNE 4025 SANDY HILL RD W SPRINGFIELD MA 01075
MARTIN A PENKALA 200 THAMES PKWY APT 1C PROSPECT HTS IL 60068
VIETTA J MCSWAIN 72 1/2 KING ST CHARLESTON SC 29376
```

Result: we need (nearly) complete lexical coverage to achieve optimal performance.

3.1.1 Cost-benefit Analysis for Acronym Pronunciation The simplest thing to do is to spell out all words that are all capital letters, or a sequence of capital letters separated by periods. Two simple modifications will improve performance: dealing with plural and possessive forms of acronyms, and letting Roman numerals up to (say) XVIII pre-empt the spell-out convention. This approach is simple and compact. Like any other algorithm, it will make two sorts of errors: some things will not be spelled out that should be spelled out, and some things will be spelled out that should not be spelled out. The first type of errors will include ordinary words that are capitalized for other reasons -- some emphasized words, words in titles, certain brand names, words in telegrams and old-fashioned computerese, and so on -- as well as those Roman numerals that are falsely spelled out. The second type of errors will include cases like **AIDS** and **OPEC** -- about 10% of actual acronyms in AP newswire text. Another source of type II errors will be ordinary words that are capitalized for other reasons, such as being in titles -- [estimate of size of this source of errors].

step 1 -- 500-word dictionary for pronouncing acronyms -- fixes most type I errors, fairly robust. step 2A -- introduce heuristics for etc. -- reduces both types of errors -- brittle. step 2B -- big dictionary of all known acronyms, all known words of all types, adaptive statistical methods for differentiating cases. Likely to involve large tables [estimate], moderate training effort -- available improvement: about 1 in 1000.

3.2 Abbreviations

3.3 Numbers

Cardinals Decimals Fractions Dates

Examples of ranges:

```
               deviations (up to -10%) from the NBS curve in the    6-12 MeV range.
                                                    At    14 -- 15 MeV of neutrons,
        He played with the New Orleans Saints from    1969-71 and ended his career
              serious forms and is more prevalent nowadays than    10-15 years ago.
me minister Margaret Thatcher asserts Moscow has a    9-1 advantage over the North
```

```
                                 c sampling of 1,402 adults across the country Jan.    16-24.
                                                  The wedding amendment was defeated    5-4.
                                             Increases for meat ranged from    16-22 percent,
                                  are from the Maya’s Classic era, dating from A.D.    300-900.
                     ggle for an independent homeland for the estimated    15-20 million Kurds
```

The last, for instance, should read "...for the estimated fifteen to twenty million Kurds;" without the "to," the phrase would be distinctly odd.

AP use for fractions:

```
ry bond, down more than a point late Tuesday, rose    13-32 point, or just under $5
oint, intermediate maturities fell in the range of    7-16 point to 9-16 point and
terest on overnight loans between banks, traded at    6 5-16 percent.
```

Examples of hyphens in number strings for grouping, or at least in uses where "to" is inappropriate:

```
                                       ‘‘ With    20-20 hindsight it is now pretty clear
d by Levin into $13 million, to be split    50-50.
 Big Beaver, P.O. Box 3951, Troy, Mich.,    48007-3951.
         Or they can call, toll-free, at    800-822-8987.
mputers, but Lotus is best known for its    1-2-3 electronic spreadsheet, while Mic
```

Money amounts When we write **$5.27** we might read **five dollars and twenty seven cents**, or perhaps, in a context where the monetary units are redundant, **five twenty seven**. A phrase written **$5.27 billion** should normally be read as if it were written **five point two seven billion dollars**. To read **$5.27 billion** as **five dollars and twenty seven cents billion** is likely to cause significant confusion. In general, the interaction of the dollar sign and a string of digits with the words around is somewhat intricate:

```
 gave Sterling an indicated value in the   $80- to $90-a-share range.
es needed to fianance this nation’s huge   $200-billion-a-year budget deficit.
opposition argued that nationalizing the   $750-million operation would kill the g
165 an ounce, the actress announced at a   $100-per-person cocktail party to raise
 a replacement, although funding for the   $18-20 million bridge has not been work
k and Decker saber saws, ’’ which go for   $30-$40, said assistant manager Margare
 Before the show, the duchess attended a   $500-dollar-a-ticket cocktail party at
     U.S. autoworkers earn an average of   $13.50 an hour in straight wages and $8
p nationwide effort to crack down on the   $200-$500 million child pornography ind
aka can’t wait, because they believe the   $7-billion plan, one of the biggest con
Egypt’s 8,800-square-mile delta, where a   $20-million control program is just beg
er makes about $25 an hour, with another   $12-$13 worth of benefits.
        ‘‘ It would be nice to be making   $30-$35,000 a year, plus a company car
 there has always been a need ’’ for the   $25-million-a-year worker notification
              Fuchs estimates there is an   $80-100 million market for special educ
       The Senate would require a minimum   $10-per-acre bid; the House drafters se
                                   Once a   $2-an-hour sandwich maker in Connecticu
              Exports in October stood at   $5.29 billion, a mere 0.7% increase fro
when it sold at auction in Stockholm for   $2.44 million.
```

4. Word Pronunciation

High-accuracy word pronunciation is a challenging problem, especially in languages like English and Japanese, where the writing system is not phonetically transparent. We will describe a set of algorithms for pronunciation of English words, as they occur in unrestricted text, that are able to achieve error rates of a few tenths of a percent for ordinary text, about two orders of magnitude better than the word error rates of fifteen percent or so that were common a decade ago. Since many "words" (as much as 5% of newswire text, for instance) may be pronounced in a way that depends on their context of use, some form of text analysis is crucial to achieving these low error rates. Even in languages such as Spanish, where standard word prounciation can be determined fairly accurately by simple rules, the pronunciation of some "words" (such as numbers in dates, fractions, amounts of money, addresses, etc.) may depend on their use.

Typically, the conversion of spelling (orthography) to an IPA-like phonologyical representation is accomplished in one of two ways: either (1) by looking the words up in a dictionary (with possibly some limited morphological analysis), or (2) by sounding the words out from their spelling using basic principles.

Both approaches have their advantages and disadvantages. Most speech synthesizers adopt a hybrid strategy: employing letter-to-sound rules for most words, and catching the most common irregular words with a small ‘‘exceptions dictionary’’ of 5,000 words or less. MITalk took a radical dictionary-based approach for its day. A dictionary of 10,000 morphemes (Allen, Hunnicutt, Klatt, 1987, p. 25) covered the vast majority of the input words. Only 5% of the input words could not be handled by the *decomp* module and had to be passed to Hunnicutt’s letter-to-sound rules (Allen, personal communication). The Bell Laboratories Text-to-Speech system, *TTS*, takes an even more radical dictionary-based approach; dictionary methods are used for 99.9% of the input words, and only the remaining 0.1% will be passed to *namsa*, a letter-to-sound rule system designed for surnames (Church, 1986). Now that the dictionary is the rule and not the exception, the term ‘‘exceptions dictionary’’ seems somewhat dated.

The main motivation for moving to a dictionary-based approach is accuracy. In general, table lookup is *much* more accurate than starting from basic principles. Dictionary-based systems make a few errors per 10,000 words. In contrast, a good letter-to-sound system such as Hunnicutt’s (Allen, Hunnicutt, Klatt, 1987, chapter 6) will make about 100 times as many errors. Self-organizing/connectionist systems such as (Sejnowski and Rosenberg, 1987) are so much worse that they report error rates by letter, not by word. For instance, Rosenberg [ref] reports that his best system achieved 92% by letter (including the spaces between words) on words outside its training set, which corresponds to a word error rate of approximately 50%.

In the early days of speech synthesis, the dictionary-based approach faced two problems: memory and coverage. The memory problem has been much alleviated with declining memory prices, though for some applications, the memory requirements (approximately 0.25 megabytes) are still a concern. Coverage is a more serious issue, especially for surnames, which are generally thought to be more difficult than ordinary words that one might find in a collegiate dictionary.

Names are particularly hard because there are so many of them. The Donnelly Marketing list contains 1.5 million names (covering 72 million households in the United States). In order to appreciate just how large 1.5 million is, note that it is three times larger than the number of entries in an unabbridged dictionary (e.g, *Merriam Webster’s Third New International* (1961)), which is considerably larger than a collegiate dictionary. In addition, it takes a much larger list of names to achieve a certain amount of coverage. For example, to cover half of the surnames of in the United States requires more than 2300 names. In contrast, 50% of non-names are drawn from a list of only 141 words. And, names are thought to be less amenable to derivation techniques. Names come from many different languages; the methods of derivation are more diverse and language-specific.

David Schulz and Beth Schulz (AT&T Bell Laboratories, Indian Hill Park) have recently constructed a dictionary of the 50,000 most frequent surnames in the United States so that it is no longer necessary to use the letter-to-sound system *namsa* for these names. This greatly improves performance on a corpus of names such as the Kansas City Telephone Book. It is believed that *namsa* by itself produces good results 50% of the time and acceptable results about 85% of the time (Schulz, personal communications). The dictionary itself covers 87% of Kansas City. Thus, if *namsa* is somewhere between 50% and 85% correct by frequency, then the combination of the dictionary plus *namsa* should yield between 87% + (50%)(13%) = 93.5% and 87% + (85%)(13%) = 98% performance, a significant improvement over *namsa* alone.

The argument becomes considerably stronger when we consider morphology and analogical extensions to the dictionary. Names such as *Walters* and *Lucasville* can be derived from other names by very simple morphological processes. These stress-neutral processes increase the coverage of the names dictionary by 25%, as indicated in the table below. More complicated stress shifting processes such as *Jordan* → *Jordanian* and *Washington* → *Washingtonian* have also been implemented. One might note, with some disappointment, that they do not produce great benefits in coverage. The table below shows that stress shifting morphology (primary-stress endings, suffix-exchange,[4] *ity*-class endings, *al*-class endings) contributes considerably less than stress neutral morphology.

Sometimes surprisingly simple processes provide the greatest benefits. The rhyme analogy method is one such case. The pronunciation of an unknown name such as *Plotsky* is determined by analogy with *Trotsky*, which happens to be in the names dictionary. The pronunciation of *Plotsky* is computed from the pronunciation of *Trotsky* by removing the initial /tr/ of *Trotsky* and replacing it with /pl/. It is remarkable just how many names can be pronounced in this way. As the table below shows, the rhyme method covers more names than many of the more complicated morphological processes.

There is, of course, some chance of error. For example, we wouldn't want to derive *Jose* from *hose*. It is not possible to know for sure if two words rhyme by looking at their spelling alone. The heuristic employed by the rhyme analogy method is correct about 90% of the time. Although far from perfect, this heuristic is more reliable than letter-to-sound rules. Given a choice between the rhyming heuristic and letter-to-sound rules, it is much safer to choose the rhyming heuristic.

The table below gives the coverage for the 1/4 million most frequent names in the Donnelly Marketing List. The table also shows the result of an informal evaluation by a single human judge, Jill Burstein. The judge listened to almost 1000 names and graded them on a 3-way scale: (1), good (‘‘like I would have said it’’), (2), OK or don't know, and (3), poor (‘‘yuck’’). This evaluation shows that compounding is considerably more risky than the other processes. In general, surnames are very hard; the error rate for ordinary words are much smaller.

4. We introduce the term *suffix-exchange* to refer to a process (like Aronoff's truncation operation) of substituting one affix for another (in the same class) such as *nominate* → *nominee*.

Methods used for Frequent Names in Donnelly Marketing List

Method	Raw Counts Type	Raw Counts Token	Percentage Type	Percentage Token	Evaluation Good	Evaluation OK/?	Evaluation Poor
Direct Hit	40,208	40,354,563	16.08%	59.34%	95	3	2
Name + Stress-Neutral Ending	42,454	16,996,558	16.98%	24.99%	98	6	4
Name + Primary-Stress Ending	627	69,016	.25%	.10%	94	6	6
Name + *ity*-class Ending	6,760	984,662	2.70%	1.45%	127	4	8
Name + *al*-class Ending	2,393	341,301	.96%	.50%	90	8	6
Name + Name (Compound)	16,619	1,663,444	6.65%	2.45%	76	3	13
Name + Suffix-Exhange	14,523	1,010,813	5.81%	1.49%	101	5	3
Rhyme with Name	29,924	1,579,499	11.97%	2.32%	84	8	4
Partial Letter-to-Sound	13,796	500,923	5.52%	.74%	71	8	5
Prefix	1,230	117,857	.49%	.17%			
Combinations of Above	43,874	3,023,149	17.55%	4.44%			
All Dictionary-based Methods	212,408	66,641,785	84.96%	97.99%			
Remainder (to be handled by *namsa*)	37,592	1,364,118	15.04%	2.01%			
Totals	250,000	68,005,903	100.00%	100.00%			

The following list gives a number of examples of the more common decomposition methods:

- stress-neutral ending: *abandons = abandon + s; abandoning = abandon + ing; abandonment = abandon + ment* (names) *Abbotts = Abbott + s; Abelson = Abel + son*
- primary-stress ending: *addressee = address + ee; abductee = abduct + ee; accountability = account + ability; activization = active + ization; adaptation = adapt + ation*
- *ity*-class ending: *abortion = abort + ion; abnormality = abnormal + ity; academician = academic + ian* (names) *Adamovich; Ambrosian; Anagnostakis*
- *al*-class ending: *accidental = accident + al; adjectival = adjective + al; combative = combat + ive*
- suffix-exchange: *auditoria = auditorium - um + a; collusive = collude - ude + usive; eldress = elder - er + ress* (names) *Agnano = Agnelli - elli + ano; Bierstade = Bierbaum - baum + stadt*
- prefix: *adjoin; cardiovascular; chlorofluorocarbon* (names) *O'brien; Macdonald; Distephano*
- compound: *airfield; anchorwoman; armrest* (names) *Abdulhussein; Baumgaertner*
- Rhyming: (names) *Alifano* (from *Califano*); *Anuszewski* (from *Januszewski*)

The following table shows the coverage of the various methods for words distributed over the Associated Press Newswire during 1988. This corpus is very different than the Donnelly list of surnames. There are a large number of uppercase words in the AP corpus, only some of which are names. For the purposes of this paper, a word is considered to be a name if it appears in uppercase at least one hundred times more often than it appears in lowercase.

Methods used in 1988 Associated Press Newswire

Method	Ordinary Words (non-names)				Capitalized Words (names)			
	Raw Counts		Percentage		Raw Counts		Percentage	
	Type	Token	Type	Token	Type	Token	Type	Token
Direct Hit	13,356	20,646,656	18.27%	75.13%	26,965	4,147,321	22.87%	70.24%
Stress-Neutral	24,050	4556078	32.90%	16.58%	25,638	811,751	21.75%	13.75%
Primary-Stress	679	76,222	.93%	.28%	433	17,128	.37%	.29%
ity-Class	1,943	332,587	2.66%	1.21%	2,209	96,469	1.87%	1.63%
al-Class	1,174	237,979	1.61%	.87%	1,119	67,817	.95%	1.15%
Suffix-Exchange	497	36,763	.68%	.13%	2,409	23,356	2.04%	.40%
Rhyme					6,888	137,054	5.84%	2.32%
Prefix	2,907	436,839	3.98%	1.59%	780	12,945	.66%	.22%
Partial L-to-S					5,133	23,833	4.35%	.40%
Compound	3,718	170,877	5.09%	.62%	5,591	79,753	4.74%	1.35%
Combinations	15,151	966,033	23.46%	3.51%	32,705	496,026	27.76%	8.40%
All Methods	63,475	27,460,034	89.58%	99.92%	97,849	5,752,566	83.01%	97.43%
Remainder	9,618	21,076	10.42%	.06%	20,032	151,951	16.99%	2.57%
Totals	73,093	27,481,110	100.00%	100.00%	117,881	5,904,517	100.00%	100.00%

5. Partial Letter-to-Sound

The difficulty with letter-to-sound rules is that the same sequence of letters can show up in so many different circumstances: in stressed syllables or unstressed ones; with different following consonants, in names from different languages, etc. With suffix-exchange, we have most of these variables pinned down. We know for certain that the ending is fairly common in the proposed language group. We also know that the part of the word excluding the ending has shown up with another ending from the same language group. And so, without a formal attempt to identify the language, we have done so implicitly, and found a dictionary counterpart.

Of particular interest here is the case where the main stress is on the first syllable of the ending. We know that the vowel preceding a main stress will be schwa-like. From this we know that the intervening consonants, if they form a recognizable syllable onset cluster, will attach to the stressed vowel following them. There is no situation more favorable for letter-to-sound rules to work than this. The proposal, then, is to allow an unstressed syllable between the stem and the suffix. Letter-to-sound rules are used to infer the pronunciation of the unstressed syllable; dictionary-based methods are used for the remainder. We refer to this hybrid strategy as the ‘‘partial letter-to-sound’’ method.

6. Conclusion

The pronunciation problem has traditionally been divided into two very separate modules: letter-to-sound rules and the exceptions dictionary. The focus has been on letter-to-sound rules which work from first principles. In contrast, the present work resorts to letter-to-sound rules only when all alternatives have been exhausted. The most reliable inference is table lookup. Failing that, the system tries to make as safe an inference as possible from two words in the dictionary. Stress neutral morphology is considered fairly safe; rhyming is more dangerous, but far more reliable than letter-to-sound rules. Our approach breaks down the traditional barriers between letter-to-sound rules and dictionary-based methods. The rhyme method, for example, uses letter-to-sound rules to pronounce the initial consonant onset and dictionary methods to pronounce the remainder. The partial letter-to-sound method is a more ambitious hybrid approach.

Appendix: Evaluation

Evaluation of Capitalized Words in AP (by Type)

Method	(Good) 1	2	3	4	5 (Bad)	Typo/?
Direct Hit	89	3	2	4	2	1
Stress Neutral Ending	96	0	1	1	1	1
Primary-Stress Ending	99	6	2	8	0	3
ity-Class Ending	91	6	0	4	1	1
al-Class Ending	89	8	2	4	0	1
Compound	63	9	7	16	0	4
Suffix-Exchange	85	10	1	3	0	1
Rhyme	90	5	5	3	1	7
Partial Letter-to-Sound	76	9	3	10	0	4
Prefix	90	6	4	11	6	3

Evaluation of Ordinary Words in AP (by Type)

Method	(Good) 1	2	3	4	5 (Bad)	Typo/?
Direct Hit	102	2	0	2	0	1
Stress Neutral Ending	97	0	2	1	1	3
Primary-Stress Ending	92	2	0	4	2	13
ity-Class Ending	92	2	2	2	1	5
al-Class Ending	91	1	0	5	0	8
Compound	89	2	2	1	4	9
Suffix-Exchange	95	4	2	8	1	10

Evaluation of Capitalized Words in AP (by Token)

Method	(Good) 1	2	3	4	5 (Bad)	Typo/?
Direct Hit	11729	11	10	88	47	4
Stress Neutral Ending	2503	0	1	1	1	1
Primary-Stress Ending	3666	84	82	45	0	6
ity-Class Ending	3409	120	0	15	14	1
al-Class Ending	25114	5819	10	10	0	6
Compound	377	56	26	160	0	8
Suffix-Exchange	462	90	2	10	0	1
Rhyme	5610	47	35	46	2	19
Partial Letter-to-Sound	258	15	9	47	0	96
Prefix	1410	47	9	37	12	8

Evaluation of Ordinary Words in AP (by Token)

Method	(Good) 1	2	3	4	5 (Bad)	Typo/?
Direct Hit	38270	4	0	2	0	1
Stress Neutral Ending	22947	0	2	8	12	12
Primary Stress Ending	4837	5	0	8	28	48
ity-Class Ending	18780	257	5	56	20	10
al-Class Ending	23418	5	0	39	0	29
Compound	7680	35	2	1	192	243
Suffix-Exchange	9252	7	3	3928	11	96

7. Program organization and data structures

In the English-language TTS system developed in our laboratory [refs], we begin by turning the input word sequence into a complex tree-like structure whose nodes are marked with sets of features. For instance, words have a part-of-speech feature, which takes on values such as *singular proper noun* or *subordinating* *conjuction*; expressions involving numbers are given a functional-category feature that can take on values such as *date* or *telephone number* or *dollar amount*. As processing proceeds, this structure is gradually "decorated" with more and more information -- pitch accents and boundary tones are assigned to words and phrases, a pronunciation is computed for each word, durations are computed for the phonetic segments in the pronunciation, and so forth. Gradually, the properties that we calculate become less like symbol sequences and more like sampled time-functions, such as time-functions of fundamental frequency or of spectral parameters. The final outcome is a speech waveform.

8. Future Directions

References

Allen, J., Hunnicutt, M., and Klatt, D., ‘‘From text to speech: The MITalk system,’’ Cambridge University Press, 1987.

Aronoff, M., ‘‘Word Formation in Generative Grammar,’’ MIT Press, 1976.

Church, K., ‘‘Stress Assignment in Letter to Sound Rules for Speech Synthesis,’’ ICASSP 1986.

Chomsky, N., ‘‘Three Models for the Description of Language,’’ IRE Transactions on Information Theory, vol. IT-2, Proceedings of the Symposium on Information Theory, 1956.

‘‘Collins Cobuild English Language Dictionary,’’ William Collins Sons & Co Ltd, 1987.

DeRose, S., ‘‘Grammatical Gategory Disambiguation by Statistical Optimization,’’ Computational Linguistics, Vol. 14, No. 1, 1988.

Ejerhed, E., ‘‘Finding Clauses in Unrestricted Text by Stochastic and Finitary Methods,’’ Second Conference on Applied Natural Language Processing, available from ACL, Don Walker, Bellcore, 445 South St., Morristown, NJ., 1988.

Francis, W., and Kucera, H., ‘‘Frequency Analysis of English Usage,’’ Houghton Mifflin Company, Boston, 1982.

Jelinek, F. (1985) ‘‘Self-organized Language Modeling for Speech Recognition,’’ IBM Report.

Kahan, S., Pavlidis, T., and Baird, H., ‘‘On the Recognition of Printed Characters of any Font or Size,’’ IEEE Transactions PAMI, pp. 274-287, March, 1987.

Katz, S., ‘‘Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer,’’ IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-35, No. 3, March, 1987.

Leech, G., Garside, R., Atwell, E., ‘‘The Automatic Grammatical Tagging of the LOB Corpus,’’ ICAME News 7, 13-33, 1983.

Marcus, M., ‘‘A Theory of Syntactic Recognition for Natural Language,’’ MIT Press, Cambridge, Massachusetts, 1980.

Sejnowski, T., and Rosenberg, C., ‘‘Parallel networks that learn to pronounce English text,’’ *Complex Systems*, vol. 1, pp. 144-168, 1987.

''Webster's Seventh New Collegiate Dictionary,'' Merriam Company, Springfield, Massachusetts, 1972.