AN EXPERT SYSTEM FOR THE PRODUCTION OF PHONEME STRINGS FROM UNMARKED ENGLISH TEXT USING MACHINE-INDUCED RULES

Alberto Maria Segre
University of Illinois
at Urbana-Champaign
Coordinated Science
Laboratory
1101 W. Springfield
Urbana, IL 61801 U.S.A.

Bruce Arne Sherwood
University of Illinois
at Urbana-Champaign
Computer-based Education
Research Laboratory
103 S. Mathews
Urbana, IL 61801 U.S.A.

Wayne B. Dickerson
University of Illinois
at Urbana-Champaign
English as a Second Language
Foreign Language Building
707 S. Mathews
Urbana, IL 61801 U.S.A.

ABSTRACT

The speech synthesis group at the Computer-Based Education Research Laboratory (CERL) of the University of Illinois at Urbana-Champaign is developing a diphone speech synthesis system based on pitch-adaptive short-time Fourier transforms. This system accepts the phonemic specification of an utterance along with pitch, time, and amplitude warping functions in order to produce high quality speech output from stored diphone templates.

This paper describes the operation of a program which operates as a front end for the diphone speech synthesis system. The UTTER (for "Unmarked Text Transcription by Expert Rule") system maps English text onto a phoneme string, which is then used as an input to the diphone speech synthesis system. The program is a two-tiered Expert System which operates first on the word level and then on the (vowel or consonant) cluster level. The system's knowledge about pronunciation is organized in two decision trees automatically generated by an induction algorithm on a dynamically specified "training set" of examples.

I INTRODUCTION

Most speech synthesis systems in use today require that eventual utterances be specified in terms of phoneme strings. The automatic transformation of normal English texts into phoneme strings is therefore a useful front-end process for any speech synthesis unit which requires such phonemic utterance specification.

Unfortunately, this transcription process is not nearly as straightforward as one might initially imagine. It is common knowledge to nonnative speakers that English poses some particularly treacherous pronunciation problems. This is due, in part, to the mixed heritage of the language, which shares several orthographic bloodlines.

Past attempts to create orthographically-based computer algorithms have not met with great success. Algorithms such as the Naval Research Laboratory pronunciation algorithm [Elovitz76] are letter-based instead of linguistically-based. For this reason, such algorithms are excessively rigid in that they are often unable to cope with a letter pattern that maps onto more than one phoneme pattern. Extreme cases are those words which, although differing in pronunciation, share orthographic representations (an analogous problem exists in speech recognition, where words which share phonemic representations differ in orthographic representation, and therefore possibly in semantic interpretation). A notable exception is the MIT speech synthesis system [Allen81] which is linguistically-based, but not solely phoneme-based.

A desirable feature in any rule-based system is the ability to automatically acquire or modify its own rules. Previous work [Oakey81] applies this automatic inference process to the text-to-phoneme transcription problem. Unfortunately, Oakey's system is strictly letter-based and suffers from the same deficiencies as other nonlinguistically-based systems.

The UTTER system is an attempt to provide a linguistically-based transcription system which has the ability to automatically acquire its own rule base.

II METHOD

The system's basic goal is the transcription of input text into phoneme strings. The method used to accomplish this goal is based on a method taught to foreign students which enables them to properly pronounce unknown English words [DickersonF1, DickersonF2]. The method is basically a two stage process. The first stage consists in assigning major stress to one of the word's syllables. The second stage maps a vowel or consonant group with a known stress value uniquely onto its corresponding phoneme string. It is the stress-assignment process which distinguishes this pronunciation method from applying purely letter-based text-to-speech rules, as in, for example, the Naval Research Laboratory algorithm [Elovitz76].

In order to accomplish the transcription of text into phoneme strings, the system uses a set of two transcription rules which are machine generated over a set of sample transcriptions. As the system transcribes new input texts, any improper transcriptions (i.e., mispronunciations)

would be flagged by the user and added to the sample set for future generations of transcription rules.

The first stage operates on "words"[1] while the second stage operates on "clusters" of vowels or consonants.[2] Each word is examined individually, and "major stress"[3] is assigned to one of the "syllables".[4] Major stress is assigned on the basis of certain "features" or "attributes"[5] extracted from the word (an example of a word-level attribute is "suffix-type"). The assignment of major stress is always made uniquely for a given word. The assignment process consists of invoking and applying the "stress-rule".

The "stress-rule" is one of two machine-generated transcription rules, the other being the "cluster-rule". A transcription rule consists of a decision tree which, when invoked, is traversed on the basis of the feature values of the word or cluster under consideration. The transcription rule "test"[6] is evaluated and the proper branch is then selected on the basis of values of the word features. The process is repeated until a leaf node of the tree is reached. The leaf node contains the value returned for that invocation of this transcription rule, which uniquely determines which syllable is to receive the major stress.

---

[1] A "word" is delimited by conventional word separators such as common punctuation or blank spaces in the input stream.

[2] A "cluster" consists of contiguous vowels or contiguous consonants. The following classificatory scheme is used to determine if a letter is a vowel (-v-) or a consonant (-c-):
"a", "e", "i", and "o" are -v-,
"u" is -v- unless it follows a "g" or "q",
"l" is a special consonant represented by -l-,
"r" is a special consonant represented by -r-,
"y" is -v- if it follows -v-, -c-, -l- or -r-,
"w" is -v- if it follows -v-.

[3] "Major stress" corresponds to that syllable which receives the most emphasis in spoken English.

[4] A "syllable" will be taken to be a set of two adjacent clusters, with the first cluster of the vowel type and the second cluster of the consonant type. For syllable division purposes, if the word begins with a consonant the first syllable in that word will consist solely of a consonant cluster. Similarly, if the word ends in a vowel then the final syllable will consist of a vowel cluster alone. In all other cases, a syllable will always consist of a vowel cluster followed by a consonant cluster.

[5] The terms "feature" and "attribute" will be used interchangeably to refer to some identifiable element in a word or cluster. For more information regarding word or cluster attributes see the following section.

[6] A transcription rule "test" refers to the branching criteria at the current node.

After word stress is assigned, each cluster within the word is considered sequentially. The cluster features are extracted, and the cluster-rule is invoked and applied to obtain the phonemic transcription for that particular cluster. Note that one of the cluster features is the stress of the particular syllable to which the cluster belongs. In other words, it is necessary to determine major stress before it is possible to transcribe the individual clusters of which the word is comprised. The value returned from invoking the cluster rule is the phoneme string corresponding to the current cluster.

UTTER uses the World English Spelling [Sherwood78] phonetic alphabet to specify the forty-odd sounds in the English language. The major advantage of WES over other phonetic representations (such as the International Phonetic Alphabet, normally referred to as IPA) is that WES does not require special characters to represent phonemes. In UTTER's version of WES, WES uses no more than two Roman alphabet characters to specify a phoneme.[7]

The choice of WES over other phoneme representation systems was also motivated by the fact that Glinski's system [Glinski81], with which UTTER was designed to interface, uses WES. The choice was strictly implementational, and by no means excludes the use of a different representation system for future versions of UTTER.

III SYSTEM ORGANIZATION

The current implementation of UTTER operates in one of three modes, each of which corresponds to one of the three tasks required of the system:

(1) execution mode: the transcription of input text using existing transcription rules.

(2) training mode: flagging incorrect transcriptions for inclusion in the next generation of transcription rules.

(3) inference mode: automatic induction of a new set of transcription rules to cover the set of training examples (including any additions made in training mode).

What follows is a more detailed description of each of these three modes of operation.

A. Execution Mode

Execution mode is UTTER's normal mode of operation. While in execution mode, UTTER accepts English input one sentence at a time and produces the corresponding pronunciation as a list of phonemes.

What follows is a detailed description of each step taken by UTTER when operating in execution mode.

---

[7] For a complete listing of the World English Spelling phonetic alphabet see Appendix A.

(1) The input text is scanned for word and cluster boundaries, and lists of pointers to boundary locations in the string are constructed. The parser also counts the number of syllables in each word, and constructs a new representation of the original string which consists only of the letters 'v', 'c', 'l', and 'r'.

This new representation, which will be referred to as the "vowel-consonant mapping," or simply "v-c map," is the same length as the original input. Therefore, all pointers to the original string (such as those showing word and cluster boundaries) are also applicable to the v-c map. The v-c map will be used in the extraction of cluster features.

(2) Each word is now processed individually. The first step is to determine whether the next word belongs to the group of "function words".[8] If the search through the function word list is successful, it will return the cross-listed pronunciation for that word. Table look-up provides time-efficient transcription for this small class of words which have a very high frequency of occurrence in the English language, as well as highly irregular pronunciations. If the word is a function word, its pronunciation is added to the output and processing continues with the next word.

Positioning of function words provides a valuable clue to the syntax of the input. Syntactic information is essential in disambiguating certain words. Although the current version of UTTER supports part-of-speech distinctions, the current version of the parser fails to supply this information. A new version of UTTER should include a better parser which is capable of making these sorts of part-of-speech distinctions.[9] Such a parser need not be very accurate in terms of the proper assignment of words to part-of-speech classes. However, it must be capable of separating identically spelled words into different classes on the basis of function. These words often differ in pronunciation, such as "present" (N) and "present" (V) or "moderate" (N) and "moderate" (V). In other words, the parser need not classify these two words as noun and verb, as long as it makes some distinction between them.

(3) Each word is now checked against another list of words (with their associated pronunciations) called the "permanent exception list," or PEL. The PEL provides the user with the opportunity to specify common domain-specific words whose transcription would best be handled by table-look-up, without reconstructing the pronunciation of the word each time it is encountered.

The time required to search this list is relatively small (provided the size of the list itself is not too large) compared to the time necessary for UTTER to transcribe the word normally.

If the word is on the PEL, its pronunciation is returned by the search routine and added to the output. Processing continues with the next word.

(4) At this point the set of word-level features is extracted. These features are used by the stress-rule for the assignment of major stress to a particular syllable in the word. A major stress assignment is made for each word.

The set of word level attributes includes:

part-of-speech (assigned by the parser);
key-syllable (in terms of the v-c map representation);
left-syllable (in terms of the v-c map representation);
suffix type (neutral, weak or strong);
prefix/left-syllable overlap (true or false).

These features are both necessary and sufficient to assign major stress to any given word [Dickerson81].

Although a detailed account of the selection of these features is beyond the scope of this paper, an example of an input word and the appropriate attribute values should give the reader a better grasp of the word-level feature concept.

Consider the input word "preeminent".

The weak suffix "ent" is stripped.
Key-syllable (final syllable excluding suffixes) is "in".
Left-syllable (left of key-syllable) is "eem".
Prefix ("pre") overlaps left-syllable ("eem") since they share an "e".

Proper stress placement for the word "preeminent" is on the left-syllable.

(5) The word and its attributes are checked against a list of exceptions to the current stress rule (called the "stress exception list" or SEL). This list is normally empty, in which case checking does not take place. Additions to the list can only be made in training mode (see below).

If the word and its features are indexed on the SEL, the SEL search returns the proper stress in terms of the number 0 or -1. If stress is returned as 0, major stress falls on the key-syllable. If stress is returned as -1, major stress falls on the left-syllable.

---

[8] For a complete listing of function words see Appendix B.

[9] It should be possible to model a new parser on an existing parser which already makes this sort of part-of-speech distinction. For example, the STYLE program developed at Bell Laboratories provides a tool for analyzing documents [Cherry80] and yields more part-of-speech classes than would be required for UTTER's purposes.

(6) If the word does not appear on the SEL, then the current stress rule is applied. The stress rule is essentially a decision tree which is traversed on the basis of the values of the word's word level attributes. Application of the stress rule also returns either 0 or -1.

(7) Now processing continues for the current word on a cluster-by-cluster basis. The cluster-level attributes are extracted. They include:

cluster type (vowel or consonant);
cluster (orthography);
left neighbor cluster map (from v-c map);
right neighbor cluster (orthography);
right neighbor cluster map
(from v-c map);
cluster position (prefix, suffix, etc.);
stress (distance in syllables from major stress syllable).

These features are necessary and sufficient to classify a cluster [Dickerson82].

As before, an example of cluster level attributes is appropriate. Consider the cluster "ee" (from our sample word "preeminent").

The cluster type is "vowel".
The cluster orthography is "ee".
The left neighbor cluster map is "cr"
(v-c map of "pr").
The right neighbor cluster is "m".
The right neighbor cluster map is "c"
(v-c map of "m").
The cluster position is
"word-prefix boundary".
The cluster is inside the syllable
with major stress (see above).

(8) The cluster and its associated attributes are checked against a list of exceptions to the cluster rule (called the "cluster exception list" or CEL). This list is normally empty, and additions can only be made in training mode (see below). If the search through the CEL is successful, it will return the proper pronunciation for the particular cluster. The pronunciation (in terms of a WES phoneme string) is added to the output, and processing continues with the next cluster in the current word, or with the next word.

(9) The cluster transcription rule is applied to the current cluster. As in the case of the stress rule, the cluster rule is a decision tree which is traversed on the basis of the values of the cluster level attributes. The cluster rule returns the proper pronunciation for this particular cluster and adds it (in terms of a WES phoneme string) to the output. Processing continues with the next cluster in the current word, or with the next word in the input.

B. Training Mode

When UTTER is operating in training mode, the system allows the user to correct errors in transcription interactively by specifying the proper pronunciation for the incorrectly transcribed word.

The training mode operates in the same manner as the execution mode with the exception that, whenever either rule is applied (see steps 6 and 9 above), the user is prompted for a judgement on the accuracy of the rule. The user functions as the "oracle" who has the final word on what is to be considered proper pronunciation.

Let us assume, for example, that the stress rule applied to a given word yields the result "stress left-syllable" (in other words, the rule application routine returns a -1) and the proper result should be "stress key-syllable" (or a result of 0). If the system were operating in execution mode, processing would continue and it is unlikely that the word would be properly transcribed. The user could switch to training mode and repeat the transcription of the problem word in the same context.

In training mode, the user has the opportunity to inspect the results from every rule application, allowing the user to flag incorrect results. When an incorrect rule result is detected, the proper result (along with the current features) will be saved on the appropriate exception list. In terms of the previous example, the current word and word-level features would be saved on the SEL.

If the given word should arise again in the same context, the SEL would contain the exception to the transcription rule, prohibiting the application of the stress rule. The information from the SEL (and from the CEL at the cluster-level) will be used to infer the next generation of transcription rules.

It is important to note that UTTER makes a given mistake only once. If the transcription error is spotted and added to the SEL (or CEL, depending on which transcription rule is at fault) it will not be repeated as long as the exception information exists. The SEL (and CEL) can only be cleared by the rule inference process (see below) which guarantees that the new generation of rules will cover any example that is to be removed from the appropriate exception list.

C. Inference Mode

Inference mode allows for the generation of new transcription rules. The inference routine is based on techniques developed in artificial intelligence for the purpose of generating decision trees based on sets of examples and their respective classifications [Quinlan79]. The basic idea behind such an inference scheme is that some set of examples (the "training set") and their proper classifications are available. In addition, a finite set of features which are sufficient to classify these examples, as well as some method for extracting these features, are also available. For example, consider the training set [dog, cat, eagle, whale, trout] where each

element is classified as one of [mammal, fish, bird]. In addition, consider the feature set [has-fur, lives-in-water, can-fly, is-warm-blooded] and assume there exists a method for extracting values for each feature of every entry in the training set (in this example, values would be "true" or "false" but this need not always be so). From this information, the inference routine would extract a decision tree whose branch nodes would be tests of the form "if has-fur is true then branch-left else branch-right" and whose terminal nodes would be of the form "the animal in question is a mammal." The premis is that such a decision tree would be capable of correctly classifying not only the examples contained in the training set but any other example whose feature values are known or extractable.[10]

What follows is a step-by-step description of the inference algorithm as applied to the generation of the stress transcription rule. Generation of the cluster transcription rule is similar, except that the cluster transcription rule returns a phoneme string rather than a number. For a more complete discussion of the inference algorithm, which would be beyond the scope of this paper, see [Quinlan79].

(1) The current stress exception list is combined with the training set used to generate the previous stress transcription rule. The old training set is referred to as the "stress classified list," or SCL, and is stored following rule generation.[11] Since the SCL is not used again until a new rule is generated, it can be stored on an inexpensive remote device, such as magnetic tape. The SCL (as well as the CCL) tends to become quite large.[12]

(2) Features are extracted for each of the entries in the training set. Features which cannot be extracted in isolation, such as the part-of-speech of a given word, are stored along with the entry and its result in the SEL. These unextractable attributes rely on the context the entry appeared in rather than on the entry itself and, therefore, cannot be reconstructed "a posteriori."

The training set now consists of all of the entries from the SCL and the SEL, as well as all of the features for each entry. At this point an initial "window" on the training set is chosen. Since the inference algorithm's execution time increases combinatorially with the size of the training set, it is wise to begin the inference procedure with a subset of the training set. This is acceptable since there is often a relatively high rate of redundancy in the training set. The selection of the window may be done arbitrarily (as in the current version of UTTER), or one might try to select an initial window with the widest possible set of feature values.[13]

(3) For each "attribute-value"[14] in the current window a "desirability index" is computed. This index directly reflects the ability of a test on the attribute-value to split the window into two relatively even subwindows.

The current version of UTTER uses a desirability index which is defined as:

$$\frac{\#\text{ samples with this attribute-value}}{\#\text{ distinct final values in this subset.}}$$

Different desirability indices might be substituted to reflect the information content of attribute-values.

When generating rules using UTTER the user has the option of using either only a test for equality in the decision tree, or a larger set of tests containing "equals," "not-equals," "less-than," and "greater-than". If the larger set of possible tests is used, then the inference routine takes

---

[10] The inference algorithm need not be time- or space-efficient. In fact, in the current implementation of UTTER, it is neither. This observation is not particularly alarming, since inference mode is not used very often, in comparison to execution or training modes (where space- and time-efficiency are particularly vital to fast text transcription). There are some inference systems [Oakey81] in which the inference routine is somewhat streamlined and not nearly as inefficient as in the case of the current implementation. Future versions of UTTER might consider using a more streamlined inference routine. However, since the inference routine need not be invoked very often, its inefficiency does not have any effect on what the user percieves as transcription time.

[11] The equivalent list in the cluster transcription rule case is called the "cluster classified list," or CCL.

[12] It should be possible to use an existing computer encoded pronunciation dictionary (or a subset thereof) to provide the initial SCL and CCL. The current version of UTTER uses null lists as the initial SCL and CCL, and therefore forces the user to build these lists via the SEL and CEL. This implies a rather time consuming process of running text through UTTER in training mode. An existing pronunciation dictionary would allow training mode to be used rather infrequently, and then only to make more subtle corrections to the transcription rules.

[13] The selection of all those examples which have unique combinations of feature values should reduce the number of iterations required in the inference routine by eliminating redundant entries in the training set. This type of training set pruning should be done at the same time the training set is scanned for clashes (discussed below).

[14] An "attribute-value" refers to the value of a feature or attribute for the given example. For instance, let the attribute in question be the word-level attribute "part-of-speech" and assume it may take one of five possible values (noun, verb, adjective, adverb, or function word). If this attribute appears with only three values (such as noun, verb, adjective) in the current window, then only those three attribute-values need be considered.

much longer to execute. However, the decision trees generated using the larger set are often smaller and therefore usually faster to traverse.

(4) The attribute-value with the greatest desirability index is chosen as the next test in the decision tree. This test is added to the decision tree. In this manner, examples occurring most frequently will take the least amount of time to classify and, thus, to transcribe.[15]

(5) The current window is split into two subwindows. The split is based on which examples in the window contain the attribute-value selected as the new test, and which examples do not.

(6) For each subwindow, it is determined whether there is only one result value in a given subwindow (i.e., is the result uniform on the window?) or whether there is more than one result.

(7) If there is more than one result in a subwindow, this procedure is applied recursively with the subwindow as the new window.

If there is only one result across a given subwindow, then generate a "terminal" or "leaf" node for the decision tree which returns this singular result as the value of the tree at that terminal. Terminal nodes are thus easily recognized since they have only one distinct result.

(8) When the original window is completely classified the resulting decision tree is the new rule which is guaranteed to cover the original window.

The newly generated rule is applied to the remaining examples in the training set. From the examples it fails to correctly classify, a subset of the failures is chosen for addition to the previous iteration's starting window. The inference algorithm is reapplied using this new starting window.

(9) When no failures exist, the most recently generated decision tree completely covers the training set. In this case, the training set then becomes the SCL, and is stored in remote storage until the next rule generating session. The most recently generated decision tree becomes the new rule and the SEL is zeroed.

It is, of course, possible to terminate the inference algorithm before it completely classifies the training set. In this case, UTTER simply places all of the "failures" on the SEL and all of the properly classified examples from the training set on the SCL. In this fashion it is possible to reduce the size of the SEL without exhaustively classifying the entire training set. The procedure for creating a cluster rule is identical.

In the course of rule generation, an inconsistency called a "clash" may arise when the attributes are insufficient to classify two or more examples. A clash manifests itself as a window with uniform values for all of the attributes, but with more than one result present in the window. The current version of UTTER aborts the rule generation process when a clash occurs. Future versions of UTTER should screen the entire training set for clashes before starting the rule generation process, as well as allow the user to remove or correct the entries responsible for the clash.

Clashes are usually the result of an error made by the user in training mode. If a clash should arise which is not the result of a user error, it would indicate that the attribute set is insufficient to characterize the set of transcriptions. Additional attributes would have to be added to UTTER in order to handle this event.

For example, the word "read" is pronounced differently in present tense than it is in past tense. Since UTTER cannot extract contextual or semantic information, the distinction cannot be made. Therefore, two entries in the training set might be present with the same attributes, but different transcriptions. This situation results in a clash which cannot be resolved without the addition of another attribute, such as "tense." Fortunately, such cases account for a very small portion of the English language.

IV CONCLUSION

This paper has described a newly developed system for the transcription of unmarked English text into strings of phonemes for eventual computer speech output. The current implementation of the system has shown this technique to be feasible in terms of speed of execution and storage requirements, and desirable in terms of transcription accuracy.

One of the unique features of UTTER is the possibility of creating "mini-implementations" of UTTER for use on evermore popular micro computers. These reduced versions of UTTER would only need to provide execution mode. The two transcription rules could be developed on a full-scale system, and provided to the user on floppy diskettes for use on a micro computer. The micro systems need not provide a training mode, so no SEL or CEL need be retained (or checked during the transcription process). The PEL should still be provided so the user could tailor the operation of the system to the particular application by adding domain-specific words to this list. The micro systems need not supply an inference mode which requires the most processor time and memory space of all the modes of operation. Updated rules (on floppy diskettes) could be provided periodically from the

[15] In certain pathological cases, the tree generated is not optimal in terms of traversal time. This problem has not yet occurred with real transcription data, and, in any case, would still yield an acceptable, though less than optimal, decision tree.

main system -- thus keeping memory and storage requirements well within the capabilities of today's micro computers.

Accurate phoneme string transcription from unmarked text will become increasingly vital as speech synthesis technology continues to improve. Better speech synthesis tools will encourage the trend from digitally-encoded recorded messages (as well as other phrase- or word-based computer speech methods) towards sub-word synthetic speech methods (such as diphone or phoneme based synthesis). The UTTER system is an example of a new approach to this old problem, embodying features from both the linguistic and artificial intelligence communities.

REFERENCES

[Allen81]
Allen, Jonathen, "Linguistic Based Algorithms Offer Practical Text-to-Speech Systems," Speech Technology, pp12-16: Fall 1981.

[Cherry80]
Cherry, L. L. .and Vesterman, W., "Writing Tools - The STYLE and DICTION Programs," UNIX Programmer's Manual, Seventh Ed., Vol. 2C, Computer Science Division, Department of Electrical Engineering and Computer Science, University of California at Berkeley: 1980.

[Dickerson81]
Dickerson, Wayne B., "A Pedagogical Interpretation of Generative Phonology, II. The Main Word Stress Rules of English," TESL Studies, Vol 4, pp27-93: 1981.

[Dickerson82]
Dickerson, Wayne B., "A Pedagogical Interpretation of Generative Phonology, III. Vowels in the Key and Left Syllables," TESL Studies, Vol. 5: 1982.

[DickersonF1]
Dickerson, Wayne B., Learning English Pronunciation, Volume III, "Word Stress and Vowel Quality," Part I, forthcoming.

[DickersonF2]
Dickerson, Wayne B., Learning English Pronunciation, Volume IV, "Word Stress and Vowel Quality," Part II, forthcoming.

[Elovitz76]
Elovitz, H. S., Johnson, R., McHugh, A. and Shore, J. E., "Letter-to-Sound Rules for Automatic Translation of English Text to Phonetics," IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol 24, p446-459: 1976.

[Glinski81]
Glinski, Stephen C., Diphone Speech Synthesis Based on a Pitch Adaptive Short Time Fourier Transform, Ph.D. thesis, University of Illinois at Urbana-Champaign: 1981.

[Kenyon53]
Kenyon, John S. and Knott, Thomas A., A Pronouncing Dictionary of American English, G. C. Miriam Company: 1953.

[Oakey81]
Oakey, S. and Cawthorn, R. C., "Inductive Learning of Pronunciation Rules by Hypothesis Testing and Correction," Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) 1981, pp109-114: 1981.

[Quinlan79]
Quinlan, J. R., "Discovering Rules by Induction from Large Collections of Examples," Expert Systems in the Micro Electronic Age, (Ed. D. Michie), Edinburgh University Press, pp168-201: 1979.

[Segre83]
Segre, Alberto Maria, A System for the Production of Phoneme Strings from Unmarked English Texts, M.S. thesis, University of Illinois at Urbana-Champaign: 1983.

[Sherwood78]
Sherwood, Bruce Arne, "Fast Text-to-Speech Algorithms for Esperanto, Spanish, Italian, Russian, and English," International Journal of Man-Machine Studies 10, pp669-692: 1978.

APPENDIX A - World English Spelling

a	fat	ie	tie	s	set
aa	far	j	jam	sh	shed
ae	Mae	k	kit	t	tin
au	taut	l	let	th	this
b	but	m	met	tx	thin
ch	chum	n	net	u	up
d	dig	ng	sing	ur	fur
e	set	nk	sink	uu	book
ee	see	oe	toe	ux	above
er	adder	oi	oil	v	van
f	fat	oo	too	w	win
g	gum	or	for	wh	when
h	hat	ou	out	y	yes
i	in	p	pet	z	zoo
ix	engage	r	run	zh	vision

APPENDIX B - Function Words

a	can	I	ought	under
about	could	if	our	unless
across	did	in	ours	until
against	do	into	ourselves	up
although	does	is	over	us
am	down	it	shall	was
among	during	its	she	we
an	each	itself	should	were
and	either	like	since	what
any	ever	may	so	whatever
anybody	every	me	some	when
anyone	everybody	might	somebody	whenever
anything	everyone	mine	someone	where
are	everything	must	something	wherever
around	for	my	than	whether
as	from	myself	that	which
at	going	neither	the	while
be	had	never	their	who
because	has	no	them	whom
been	have	nobody	themselves	whose
before	he	noone	then	why
behind	her	nor	therfore	will
below	hers	not	these	with
beneath	herself	nothing	they	without
beside	him	off	this	would
between	himself	on	those	you
beyond	his	one	though	your
but	how	onto	through	yours
by	however	or	to	yourself