INDUCTIVE LEARNING OF PRONUNCIATION RULES BY HYPOTHESIS TESTING AND CORRECTION

S Oakey and R C Cawthorn*

Dept. of Computer Science, Teesside Polytechnic,
Middlesbrough, Cleveland, England

ABSTRACT

This paper describes a system that learns the rules of pronunciation inductively. It begins with a set of 26 rules for single-letter pronunciation. Individual words are presented to it, and the system uses its rule set to hypothesise a pronunciation. This is compared with a dictionary pronunciation, and if any part of the pronunciation is incorrect new rules are created to handle the word as an exception condition.

These rules are checked for similarity with others already produced, and where suitable a "general" rule is produced to deal with two or more created rules. The effect is to produce rules that are more and more general, and these approach the general pronunciation rule sets that have been produced manually by other workers.

I INTRODUCTION

Conversion of unrestricted English text into its phonetic equivalent is usually performed using context dependent rules, which dictate how a character string should be pronounced when in a particular context. The rules are arranged so that one set exists for each letter of the alphabet. The rules within these sets appear in the order in which they should be applied i.e. more specific rules appear before more general rules. Each set ends with a context independent rule (default rule) which is used if no rule matches the context.

Systems using such rules operate on one word at a time, which is scanned from left to right. The rules are scanned sequentially until one that matches is found. The character(s) handled by that rule are discarded, and the process repeated until the whole word has been processed. The result is a set of phonemes showing the system's pronunciation of that word.

The first such system of any significance was written by Ainaworth (1), who produced a speech system that used a set of 159 rules for which he claimed an error rate of about 10% A second more recent system produced by an American group (2) is rather more successful. They changed and enhanced Ainsworth's rules to produce their own set of 309 rules, and claimed a success rate of 90% of words correctly translated (and 97% of phonemes correctly translated) in an average text sample.

•Sponsored by the Science Research Council

The system described in this paper automates the production of these rules by deducing them from samples of the input and required output. It is written in LISP, and uses the following rule format:-

(<ch.string><l.context><r.context><phoneme-set>)

(Two other fields are contained at the end of each rule, but these will be described later.)
The character string being searched for is at the beginning of the rule; then come the left and right context character strings; followed by the Latin equivalent of the International Phonetic Alphabet phonemes (see Appendix 2).

As an example of rule format and the translation operation, suppose the rule set for the letter A was:-

```
(A    0     (C0NS1 E DELIM)     (EY))       1
(A    ()           ()           (AE))       2
```

(C0NS1 and DELIM are group identifiers - see Appendix 1.)

The first rule states that an A preceded by anything and followed by a single consonant, an E, and the end of the word receives the pronunciation EY. The second rule is the default, and will match any other A, giving it the pronunciation AE.

If the A of the word PALE was being processed then this would fit the first rule and receive the pronunciation EY* On the other hand, A in the word PAT does not fit this rule so the system tries the next rule, which does match, giving it the pronunciation AE.

The American group (2) produced their rule set by following an iterative procedure which involves looking at erroneously pronounced words and trying to produce rules to overcome the errors. This procedure when used to translate a large dictionary involves a great deal of work in checking the output, firstly to see if the words have indeed been pronounced correctly, then in formulating new rules to correct the errors. This process needs repeating several times to produce a good set of rules and for each iteration the individual checking of each word must take place. This process of detecting the *errors* in pronunciation could be carried out automatically if some "correct'* pronunciation (ie a standard dictionary pronunciation) was held alongside each word in the dictionary.

This led on to automatic rule generation. A system with the above information included has all the data available to enable it to develop its own rules. The main problem with such a system is deciding how to implement new rules from an examination of the pronunciation errors produced.

Basically the approach taken is to use a rule set to hypothesise a pronunciation: check this against a standard pronunciation: if the pronunciation is in error, create a special purpose rule to correct the error(s): then find similar special purpose rules and combine them to produce slightly more-general rules, which may in turn be combined together to produce more-general rules.

II ROLE PRODUCTION

This is the process of taking a word in normal Ihglish, plus its pronunciation (IP A code equivalent), and hypothesising its pronunciation by using the current rule set. In a similar fashion to rule-based systems, the rules are searched from the top until one that matches both contexts is found. This requires rules to be ordered with the more-specific preceding the more-general. (The rule set is initialised to the single-letter pronunciation rules given in Appendix 1, and this is augmented by rules produced by the system.)

The hypothesised phonetic spelling and the dictionary's phonetic spelling obtained from the input can then be compared. If no differences are found then the rules in the data base require no amendment. However, if differences occur, then it is necessary to produce new rules to augment the database. For example:-

Initially the word PALE gets the pronunciation
P / AE / L / EH
This is compared with the correct phonetic spelling
P / EY / L
and specific rules are produced which say that 'A' preceded by *P' and followed by 'IE' has the long vowel pronunciation, and that the final E is silent

```
 i.e.  (A     (DELIM P)     (L E DELIM)     (EY))
 and   ( E   (DELIM P A L)     (DELIM)        () )
```

(The DELIM's mark both ends of each word.) These two rules would be added to the rule sets of A and E respectively.

The comparing routines, it was found, work best by looking at the contexts for "exact matches". The system ie always able to find at least two exact matches, these being the DELIM at both ends of the word. After all of the "exact matches" within the word are found then what's left in the computer produced pronunciation must translate to what's left in the actual pronunciation.

A match is deemed to be "exact" if the dictionary phoneme matches only one machine produced phoneme in the locality of the one being considered. Checking one machine produced phoneme either side was found to be adequate. Example:-

```
          D   U   C   K                     (word)
 DELIM    D    AH   K    K   DELIM          (computer)
 DELIM    D    AH      K     DELIM          (actual)
```

The first K in the actual pronunciation is not an "exact" match with the K in the computer pronunciation as there is another K in the locality. The next exact match is found with the final DELIM's, so the rule produced is:-

```
   ( CK     (DELIM D U)     (DELIM)     (K))
```

One other factor in the process is the LOOK-AHEAD-COUNT. This global variable determines how far ahead the system should look to find an exact match. If no match is found within the limit then the system moves on to consider the next machine phoneme. The count is initially set to 1, but is dynamically increased by the system if the DELIMs at the end of each word are not encountered together. This ensures that long words will be matched correctly. Example:-

```
         S    E    A    T                   (word)
 DELIM   S     EH    AE    T   DELIM        (computer)
 DELIM   S          IY     T   DELIM        (actual)
```

Exact matches are found up to the S, but the IY and EH do not match. No IY is found on the look ahead so T and AE are compared next. These also do not match, but within the look ahead limit the exact match T T is found. This means the following rule is produced:-

```
   (  EA    (DELIM S)     (T DELIM)     (IY))
```

The above system will create rules to correctly translate all the words it has met, but as several rules are usually needed to translate each word this, as it stands, is not very useful. A method of creating rules which will translate words the system has not met is required.

III RULE INDUCTION

A. Generalisation

The purpose of this part of the system is to produce general rules from consideration of the very specific rules produced by the first part of the system. This process is performed every time a new basic rule enters the system. Rules are only considered for generalisation, if they translate the same character string to the same phoneme•

T The left and right contexts of the rules under consideration are examined and a pairing off of the context elements is undertaken.

The three different ways of combining context elements can be shown by considering the following example pairing from left to right:-

contextl	(P	CONS1	P	P	VOWEL	VOWEL)
context2	(P	CONS1	CONS1	T	CONS1)	
giving	(P	CONS1	CONS1	CONS1)		
type	A	A	B	C	D	D

Type A is an exact match between elements; type B occurs if one is a member of the other; type C if both are of the same group; and type D is a mis-match i.e. VOWEL versus CONS1 and any context elements that remain when the pairing can *go* no further.

It can be seen that the Type C combination is more of a generalisation than Types A and B. Once a group such as CONS1 has been inserted into the resultant rule then it will allow any consonant in that position, even though it was created using just two consonants.

A weighting system was used to reflect these differences in type. Types A and B are given a value of +1; type C a value of 0; and type D a value of -1.

Left and right contexts are considered separately' and the total weight is calculated for each. The number of letters in the character string of the rule (the first field) is added to both of these (these letter(s) are counted as exact matches (type A)). If the resultant values from left and right contexts are both positive, then the two rules are combined to produce a general rule.

For example, if the rules under consideration are:-

(CK (DELIM D U) (DELIM) (K))
(CK (DELIM L U) (DELIM) (K))

the left context total weight is 4, the right is 3 so combination is allowed, and the final general rule appears as:-
(CK (DELIM CONS1 U) (DELIM) (K))

This rule when added to the database will correctly translate the CK part of the words creating the rule as well as other words such as MUCK, SUCK, etc.

But the combination of the rules:-
(A (DELIM P R I V) (T E DELIM) (AX))
(A (DELIM W O M) (N DELIM) (AX))
would not be allowed. The left context weight gives -1, as does the right.

There are two other fields in the rules. The first of these is the list of rules used to create a general rule. In the case of specific rules (basic rules) this field is null, but in all others it holds the rules which were picked out and combined to form the general rule. These sub-rules may themselves contain the rules they were created from. Thus a general rule contains a complete history of how it was formed.

The second field is a rule usage count. This number indicates how many times a rule has been successfully used. On creation of a new rule it takes the value 1 and on creation of a general rule it takes on the sum of the counts of its composing rules. Each time a rule successfully translates a letter sequence, its count is incremented by 1. Effectively, then, the count shows how many words the rule has translated correctly. For example:-

1. (CK (DELIM CONS1 VOWEL) (DELIM) (K)) 3
2. (CK (DELIM B A) (DELIM) (K)) 1
3. (CK (DELIM CONS1 U) (DELIM) (K)) 2
k. (CK (DELIM D U) (DELIM) (K)) 1
5. (CK (DELIM L U) (DELIM) (K)) 1

Rules 4 and 5 were combined to give rule 3, which was combined with (basic) rule 2 to give rule 1. The usage count appears following the rule. This format shows pictorially the way a rule has been derived, which is very useful when debugging and investigating other areas of the system.

B. Rule Conflict

Rule conflict occurs when the left and right context of two rules for the same letter are the same with the rules producing DIFFERENT phonemes. When this condition is detected the system evaluates the importance of the offending general rules by seeing which satisfies the most words (using the rule usage count). This one is kept, and the other discarded and replaced by its composing rules one level down. Rules may be made and broken several times, depending on the order of the words accepted into the system, but usually a clear "winner" emerges.

Unfortunately, there are other, more-complex, types of conflict. Given two rules

A. (stringl (A-left) (A-right) (phonemel))
B. (string2 (B-left) (B-right) (phoneme2))

where phonemel does not equal phoneme2, then the different combinations of left and right contexts may be represented using the following table:-

Table 1: types of Conflict

		LEFT				
		A<B	A=B	A>B	A≠B	A*B
R	A<B	1	1	3	1	4
I	A=B	1	0	2	3	4
G	A>B	3	2	2	2	4
H	A≠B	1	3	2	3	4
T	A*B	4	4	4	4	4

```
where  0 - conflict
       1 - rule A must precede rule B
       2 - rule B must precede rule A
       3 - ambiguity
       4 - no conflict (order unimportant)
     A<B - A is a special case (subset) of B
     A/B - priority ambiguity (see below)
     A*B - A is distinct from B
```

Priority ambiguity can be illustrated as follows:-

```
          (     A     )    ----
        (DELIM VOWEL)      ----  2
```

Which is the more general? (1) is not restricted to having a delimiter before it and (2) is not restricted to just the vowel "A". Action taken on finding conflict is outlined below.

1. No conflict (type 4)

When a new rule enters the system it is compared with each rule in its letter list. This comparison starts at the end of the list (with the default rule) and finishes at the top. If no conflict condition occurs then the new rule is inserted at the top of the list.

2. Proper conflict (type 0)

If during the above process conflict is discovered then the count of the new rule is compared with that of the conflicting rule. The rule with the highest count is retained, and the other rule is broken into its constituents and input to the system without trying to combine them with other rules in the list.

3. Partial conflict (type 1 or 2)

If a type 1 conflict occurs as the comparison operation is taking place, then the incoming rule is inserted into the rule list immediately following the rule it conflicts with (but see Part IV). If however it is a type 2 conflict the comparison passes over (and therefore above) the conflicting rule and continues on until conflict occurs again, or the top of the list is found and the rule inserted there. This partial conflict handling therefore causes the rules to be ordered with the more general following the less general.

4. Ambiguity (type 3)

While this type is reported by the system, no special action is taken. No adverse effects seem to occur by ignoring it in this way, as stabilisation (see Part IV) corrects any resultant rule misplacement.

IV RULE STABILISATION

The system so far described reads in a set of words, creates basic rules to translate these words and attempts to form general rules from examination of these basic rules.

Occasionally, rules produced towards the end of reading a batch of words can negate the effect of rules produced earlier. To overcome such errors and check the rules more generally, the batch of words are again input to the system. All the words Bhould be pronounced correctly but if this is not the case then the rule set must be incorrect. The following example shows this.

Consider the two following ordered "S" rules (where the indented rules listed below them are the rules that were combined to form the general rule):-

```
  1    (S   (CONS1 VOWEL)    (DELIM)     (S))
    1a    (S    (DELIM B U)     (DELIM)      (S))
    1b    (S    (DELIM T H I)   (DELIM)      (S))
and
  2    (S       (VOWEL)       (DELIM)     (Z))
    2a    (S    (DELIM W A)      (DELIM)      (Z))
    2b    (S    (DELIM H A)      (DELIM)      (Z))
    2c    (S    (DELIM I)        (DELIM)      (Z))
    2d    (S    (DELIM A)        (DELIM)      (Z))
```

When "WAS" is reinput, it is translated by rule 1 rather than rule 2 (since 1 precedes 2 in the rule set), giving the wrong phoneme for the "S". This requires the following (specific) rule to be created:-

```
(S     (DELIM W A)     (DELIM)      <Z)>
```

and this is checked against the current rule set in the usual way to look for possible generalisations. It of course finds rule 2, and attempts to combine the two.

At this point, the system checks to see if the rule being combined with rule 2 has already been combined with other rules to produce rule 2. Finding that it has (as 2a), rule 2 is broken into its (immediate) lower-level components, and these are re-entered individually into the system (as if they had just been created).

This in itself is insufficient, in that if no other action was taken these components would merely be recombined to give the original (over-general) rule. To stop this, during recombination the system will only combine those rules which have exactly the same pattern.

Using the above example, then, rule 2 is broken to give the four separate rules 2a to 2d. With exact matches now needed for generalisation, the "WAS" and "HAS" rules can be combined to give:-

```
(S    (DELIM CONS1  A)     (DELIM)    (Z))
   (S    (DELIM  W  A)       (DELIM)    (Z))
   (S    (DELIM  H  A)       (DELIM)    (Z))
```

and then the "IS" and "AS" rules give:-

```
(S    (DELIM VOWEL)          (DELIM)    (Z))
   (S    (DELIM I)             (DELIM)    (Z))
   (S    (DELIM A)             (DELIM)    (Z))
```

There are occasions when this is insufficient and the same rule is still formed. This situation is handled by breaking the other rule (ie the one that interfered with the rule that should have been used - rule 1 in the example).

If this also recombined to form the same rule (even under this stricter control) then the least used rule is broken and its components re-entered into the rule set without allowing any combination.

This process of re-inputting the words is repeated until all the words are translated correctly and the system is stable. This usually takes 2 or 3 iterations. The system is then complete in that it will correctly translate all the words given to it.

V RESULTS

The system has three modes of operation. The first is the normal reading of words and creation of rules; the second is the stabilisation procedure; and the third is a rule evaluation mode. This third mode enables the system to read words, show their pronunciation using its rule set and compare this with the actual pronunciation to enable statistics to be produced. These show the number of words the system has pronounced correctly and the number of phonemes correctly produced. During this process the rule set is not changed in any way.

If all the words that created a set of rules were run in this way, then the statistics would always show 100* correctness of pronunciation. To obtain more meaningful results all the basic rules are deleted before evaluation. The resultant statistics give an indication of how good the general rules produced are. It may also be the case that some of the basic rules may not be needed anyway as the general rules may have developed far enough to make them redundant.

The usage counts of all the rules are zeroised before an evaluation run and incremented during it, so the figures for usage are correct at the end of a run. Any unused rules (usage count of zero) can be deleted.

The result is a set of figures showing the proportion of phonemes and words correctly translated by the rule set. In addition, the list of rules used is given together with an indication (via the usage count) of the usefulness of individual rules.

Evaluation, runs have been carried out on the rules produced from reading the frequency based words from the Ladybird key word books (5), and the dictionary entries (4) for the first two letters of the alphabet (run separately). To act as a comparison, the same data have been run through a manually produced set of rules. This was obtained by Anglicising the rules produced by the American group (2). The results are shown in the following table:-

Table 2: Results

Input	Total	Manual correct	Automatic correct
"Ladybird"			
Phonemes	1745	91%	84%
Words	477	84%	59%
(Rules)		(171)	(65)
"A"			
Phonemes	6378	64%	74%
Words	1015	16%	21%
(Rules)		(128)	(64)
"B"			
Phonemes	4630	79%	85%
Words	866	47%	43%
(Rules)		(121)	(108)

The original American rule set was claimed to have a success rate of 90* words correct and 97* phonemes correct, but it should be stressed that these figures were derived from listening tests, not comparison with dictionary definitions. Their figures are also frequency weighted i.e. correct translation of a frequent word was more highly rated than an infrequent word.

This is shown in the table, where the best results were obtained from the Ladybird words (the most frequent 500 words of juvenile reading). Here the percentage correct produced by the manual system approaches the figures quoted. The deficiency can be accounted for by the different method of defining correctness and also because the Anglicising may have led to deterioration of the rules.

It can be seen that there is a large discrepancy between the results for the A's and B's for both the automatic and the manual systems. This is caused by the difficulty of finding a rule to define the pronunciation of initial A's. The automatic system spent nearly 40 minutes trying to come up with a suitable rule to define which word should start with AE and which with AX. The problem is basically one of stress and ,it is interesting that no set of rules appears to exist to handle this situation.

The "B" run is a better test as it is not so affected by the stress problem. The automatic system took only 10 minutes to come up with a set of rules that perform much better than the "A" set.

In both the A's and the B's, the automatic and manual system are close in their results and only in the Ladybird set do they differ significantly. Frequent words often have unusual pronunciation (i.e. they do not follow set rules) so in most systems they are usually handled by a small exceptions dictionary. This is similar to the system's basic rules, so by deleting them the system's performance has been seriously affected for frequency based words input to it. This is also reflected in the rules used, where the manual system has used three times as many as the automatic system.

However, the system does successfully learn rules governing the pronunciation of words from examples. A number of rules produced are exactly the same as some in the manually produced set, while others show considerable similarity. And the rule set produced from a section of dictionary performs approximately as well as the manually produced set.

APPENDIX I: Constants used in the system

DELIM : word delimiter (space, full stop, etc.)

VOWEL : A, X, I, 0, U

C0NS1 : B,C,D,F,Q,H,J,K,L,M,N,P,Q,R,S,T,V,V,X,Y,Z

INITIAL RULE SET:

```
(A  NIL  NIL   (AE))
(B  NIL  NIL   (B) )
(C  NIL  NIL   (K) )
(D  NIL  NIL   (D) )
(E  NIL  NIL   (EH))
(F  NIL  NIL   (F) )
(G  NIL  NIL   (G) )
(H  NIL  NIL   (HH))
(I  NIL  NIL   (IH))
(J  NIL  NIL   (JH))
(K  NIL  NIL   (K) )
(L  NIL  NIL   (L) )
(M  NIL  NIL   (M) )
(N  NIL  NIL   (N) )
(O  NIL  NIL   (OX))
(P  NIL  NIL   (P) )
(Q  NIL  NIL   (K) )
(R  NIL  NIL   (R) )
(S  NIL  NIL   (S) )
(T  NIL  NIL   (T) )
(U  NIL  NIL   (AH))
(V  NIL  NIL   (V) )
(W  NIL  NIL   (W) )
(X  NIL  NIL  (K S))
(Y  NIL  NIL   (IY))
(Z  NIL  NIL   (Z) )
```

APPENDIX II: Phonetics used in the system

Vowels		**Consonants**	
IY	l EA n	P	a PP le
IH	k I tten	B	a B le
EH	b E tter	T	T eeth
AE	l A dder	D	hi D ing
AA	l AR der	K	C ar
OX	c O lumn	G	ar G ue
AO	b OU ght	CH	CH ased
UH	g OO d	JH	J udge
UW	st OO p	M	M ime
AH	m O ther	N	o N ion
ER	b I rd	NX	ri NG
AX	p O tato	L	L ittle
EY	ch AI n	F	F amily
OW	gh O st	V	o V en
AY	m I ser	TH	brea TH
AW	n OW	DH	brea TH e
OY	b OI l	SH	SH ould
		ZH	lei S ure
		R	R oaring
		HH	a H ead
		W	W eed
		Y	Y acht
		S	S ix
		Z	Z inc
		WH	WH ere

REFERENCES

(1) Ainsworth, V. A. "A System for Converting English Text into Speech." IEEE Transactions on Audio and Electroacoustics, Vol AU-21 pp. 228-290,June 1974.

(2) Elovitz, 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 ASSP-24, No. 6 pp. 446-459, Dec.1976.

(3) Wijk, A. "Rules of Pronunciation for the English Language." Oxford University Press, 1966.

(4) Winsor Lewis, J. "A Concise Pronouncing Dictionary of British and American English." Oxford University Press.

(5) Murray, V. "The Ladybird Key Words Reading Scheme." Ladybird Books Ltd, Loughborough.

(6) Cawthorn, R. C. "Automatic Rule Production." Internal Report, 1980.

(7) Oakey, 5. and Cawthorn, R. C. "A Rule Learning System and its Application to Machine Speech." In Proceedings of Acoustics '81, Spring 1981, pp. 295-298.