Chapter 33

Word Sounds

In addition to the meanings conveyed by words, words also provide distinctive sounds that can prove to be musically useful. Poets and composers often arrange or choose texts so that the sequence of sounds create alliteration, onomatopoeia, rhythm, rhyme and other sonorous effects. This chapter introduces the **IPA scheme for representing speech sounds. This representation provides a companion to the **text and **silbe representations discussed in Chapter 27. Various sonorous processes are illustrated.

The **IPA Representation

The Humdrum **IPA scheme provides a way to represent the International Phonetic Alphabet. The **IPA scheme is based on the transliteration scheme developed by linguist Evan Kirshenbaum. The scheme is suitable for representing the basic phonemes found in most of the world's languages. The table below summarizes the **IPA mappings for various phonemes.

@	**schwa**†; as in (unaccented) *banana, collide, alone or (accented) humdrum*
V	**schwa** (IPA symbol: —); as in the British pronunciation of *hut*
R	**R**‡; as in *burn, operation, dirt, urgent*
&	short **a** (IPA symbol: æ); as in *mat, map, mad, gag, snap, patch*
A	**a** (IPA symbol: *a*); as in *bother, cot*, and, with most American speakers, *father, cart*
a	**â**; *father* as pronounced by speakers who do not rhyme it with *bother*.
E	short **e** (IPA symbol: *ε* or ᴇ); as in *get, bed, peck, edge*
i	long **e** (IPA symbol: e); as in *beat, greed, evenly, easy*
I	short **i** (IPA symbol: ɪ or *ι*); as in *tip, banish, active*
o	**o** as in *oboe, trombone, banjo*
O	**ô** (IPA symbol: o or upside-down 'c'); as in *law, all, shawm*
W	**oe** digraph (IPA symbol: œ); as in the French *boeuf*, German *Holle*
u	**u**; as in *rule, youth, few, ooze*
U	**û** (IPA symbol: *υ* or ᴜ or *ω*); as in *pull, wood, book*
y	**ue**; as in the German *fullen*, *hubsch*, or French *rue*
vowel~	following a vowel* indicates a vowel or diphthong pronounced with open nasal passages; as in the French "un bon vin blanc" (`W~ bo~ va~ blA~`)

b	**b** (IPA symbol: b or c); as in *beam, cabin, rob*
d	**d**; as in *deed, dulcimer, ader*
f	**f**; as in *fugue, staff, forte*
g	**g**; as in *guitar, fagot, gig*
h	**h**; as in *hear, ahead, horn*
k	**k**; as in *cook, take, score, ache*
x	**K** (IPA symbol: k); as in the German *ich, Buch*
l	**l**; as in *libretto, Lully, pool*
m	**m**; as in *music, limb, hymn*
n	**n**; as in *no, instrument, blown*
N	**eng** (IPA symbol: 'n' with a tail); as in *sing, fingering, ink*
p	**p**; as in *piano, beeper, lip*
r	**r**; as in *reed, organ, car*
s	**s**; as in *soprano, cymbal, source, bass*
S	**sh** ["esh"] (IPA symbol: .); as in *sharp, crescendo, special, percussion*
t	**t**; as in *tempo, tie, attacca, minuet*
T	**th** ["thorn"] (IPA symbol: *θ*); as in *thin, path, ether*
D	**th** ["eth"] (IPA symbol: *đ) as in then, rhythm, smooth*
v	**v**; as in *voice, vivace, live*
w	**w**; as in *we, away*
j	**j**; as in *yes, Johann, cue, onion*
z	**z**; as in *zone, raise, xylophone, jazz*
Z	**zh** ["yogh"§]; as in *measure, vision, azure*
consonant-	following a consonant (l-, n-, m-, or N-)** indicates a consonant preceded by a *schwa* that is pronounced as an independent syllable; as in *battle, mitten, eaten*
consonant;	following a consonant,†† indicates that the front of the tongue is positioned as in the beginning of the word 'yard'
^	preceding phoneme is palatalized
´	primary stress (should precede stressed sound)
,	secondary stress (should precede stressed sound)
%	silence signifier

Summary of ****IPA** *Signifiers*

† The IPA *schwa* is notated as an upside-down 'e'.

‡ The IPA symbol consists of a *schwa* with a hook.

§ The IPA *yogh* is written like a flat-topped numeral '3' that has been lowered in height.

* In IPA such vowels are marked by the presence of a tilde above the vowel.

** In IPA such consonants are marked by the presence of a vertical bar below the consonant.

†† The IPA symbol consists of a superscript letter 'j' either following or hooked beneath the consonant.

Humdrum does not provide a tool for translating from **text or **silbe representations to **IPA. However, there are a number of commercial text-to-phoneme translators available for most common languages.

Alliteration

A common sonorous use of words is found in alliteration where several successive words commence with the same sound. A famous example of alliteration is found at the beginning of William Shakespeare's *Tempest*:

```
**text   **IPA
Full     ful
fathom   f&D@m
five     fAiv
thy      DAi
father   fADR
lies     lAiz
*-       *-
```

Given an **IPA input, occurrences of alliteration can be found by first isolating the initial phoneme for each word using **humsed**. This task requires some additional knowledge about using **humsed**. Both **sed** and **humsed** provide a "back reference" construction that allows users to manipulate a matched expression without knowing the precise matched sequence of characters. The expression to be matched is indicated via parentheses preceded by back-slash characters, i.e., `\(` and `\)`. Several such expressions can be defined and each successive expression is internally labelled with an integer beginning with 1. The marked expression can then be "back-referenced" by using the integer label preceded by a back-slash. Hence, `\1` refers to the first referenced expression. Consider, by way of illustration, the following command:

```
sed 's/^\(.\).*/\1/'
```

In this command, the referenced expression consists of the period (match any single character). Notice that this expression is back-referenced in the replacement string — `\1`. In other words, this **sed** command carries out the following operation: find a single character at the beginning of a line followed by zero or more characters. Replace this entire string by just the first character in the line.

Let's now use this back-reference technique in our alliteration search. First we extract the **IPA spine and use **humsed** to eliminate all but the first character in each data record:

```
extract -i '**IPA' Tempest | humsed 's/^\(.\).*/\1/'
```

The result is:

```
**IPA
f
f
f
D
f
l
*-
```

We can now amalgamate successive initial phonemes by using the **context** command. Suppose we are interested in identifying alliterations where three or more words begin with the same initial phoneme. For this, we would use the **-n 3** option for **context**. Having amalgamated three phonemes on each data record we can use **humsed** to eliminate the spaces between the multiple stops:

```
extract -i '**IPA' Tempest | humsed 's/\(.\).*/\1/' \
    | context -n 3 | humsed 's/ //g'
```

The revised output is:

```
**IPA
fff
ffD
fDf
Dfl
.
.
*-
```

Now we need to identify any data records that contain three identical sigifiers. Once again, we can use the back-reference feature for **humsed**.

```
extract -i '**IPA' Tempest | humsed 's/\(.\).*/\1/' \
    | context -n 3 | humsed 's/ //g'; s/\(.\)\1\1/allit: \1/'
```

The resulting output is:

```
**IPA
allit: f
ffD
fDf
Dfl
.
.
*-
```

Let's add one further refinement which illustrates yet another feature provided by **sed** and **humsed**. Recall that operations such as substitions (`s`) and deletions (`d`) can be preceded by a regular expression that limits the operation only to those lines that match the expression. For example, the command `sed '/=/s/[0-9]//g'` will eliminate all numbers found on lines containing an equals sign. The leading regular expression can be followed by an exclamation mark which reverses the sense of the action. For example, the command `sed '/=/!s/[0-9]//g'` will eliminate all numbers *except* those found on lines containin an equals sign.

This feature can be usefully applied in our alliteration task to eliminate all other data in our spine except alliteration markers. Our final revised pipeline transforms non-alliteration data to null tokens:

```
extract -i '**IPA' Tempest | humsed 's/\(.\).*/\1/' \
   | context -n 3 | humsed 's/ //g'; s/\(.\)\1\1/allit: \1/; \
   /allit/!s/.*/./'
```

The final output is:

```
**IPA
allit: f
.
.
.
.
.
*-
```

Classifying Phonemes

Linguists have devised innumerable ways for classifing phonemes. For example, phenomes such as *f*, *s*, *sh*, *th*, *v*, etc. are classified as fricatives. Bi-labial plosives include the *p* and *b* sounds. The *m* and *n* sounds are classified as nasals. And so on.

For some tasks, it is often useful to reduce the phonemes to phonetic classes. For example, in our example from Shakespeare's *Tempest*, the 'th' in 'thy' is part of the fricative alliteration.

In Chapter 22 we saw how **humsed** can be used to classify things. A simple reassignment script can be defined which collapses the various phonemes into a smaller set of phonetic classes. For example, a suitable script might contain the following assignments:

```
s/[bdtk]/<PLOS>/g
s/[mn]/<NASL>/g
s/[fsSTDv]/<FRIC>/g
etc.
```

Classifying phonemes in this way will allow us to broaden our searches for alliterative passages.

Properties of Vowels

Vowels are particular important in music since notes can be sustained only by increasing the duration of the vowels. In speech, more time is occupied by vowel sounds than by consonants. In vocal music, an even greater proportion of the time is taken up by vowels. This means that the quality of the vowels can have a marked impact on the overall timbre or tone color of a work. For example, a long note sustained with an 'ee' can sound very different from the same note sung with a sustained 'ah'.

Like consonants, vowels can also be classified in many ways. One common classifying dimension is the front/back distinction. The vowel 'oo' is a front position vowel whereas the vowel 'ah' is a back position vowel. Similarly, vowels can be classified according to the high/low distinction.

The vowel 'ee' is a high vowel whereas the vowel 'oh' is a low vowel.

A preponderance of high vowels is often associated with sarcasm, irony or humor. Taunting sounds made by children ("nya, nya ...") commonly use high vowels mixed with nasals. Similarly, high/nasal vocal sounds are often used by comedians and actors to produce a 'funny' voice.

Suppose we want to test the idea that a certain piece in a Gilbert and Sullivan operetta exhibits a preponderance of high vowels. We might begin by creating a re-assignment file where the estimated height of each vowel is given a value between 1 (low) and 10 (high). We can estimate the overall average vowel height for a piece by averaging these values together. The basic pipeline will extract the pertinent **IPA spine, eliminate all non-vowel phonemes, add spaces between each vowel, and then assign estimated heights to each vowel. Finally, non-data records are eliminated using **rid** and the data values averaged using the **stats** command:

```
extract -i '**IPA' Penzance | humsed 's/[^@VR&AaEiIoOWuUy]//'\
   | humsed 's/./& /g; s/  / /; s/ $//' | humsed -f vowel.map\
   | rid -GLId | stats
```

This procedure can be repeated for several movements or pieces to provide a contrast for the piece of interest.

Vowel Coloration

When translating vocal texts from one language to another, it is often difficult for translators to preserve the vowel coloration. A vocal work can be considerably maligned if a prominent (high/long) note is changed from an 'ah' sound to an 'ee' sound.

Suppose we want to determine which of several English translations of a song by Schubert best preserves the vowel coloration. As above, let's limit our notion of coloration to vowel height. (Of course any other similarity mapping or dimension can be used.) As in our Gilbert and Sullivan example, we could simply compare the overall vowel height for the original Schubert song with each of the translations. However, not all notes are equally important. In the first instance, the vowels on longer sustained notes will be more noticeable than the vowels attending shorter notes. A simple remedy is to use the **timebase** command to expand the input so that longer notes are proportionally more influential in our measure of overall vowel height. We can use **ditto** to repeat sustained vowels:

```
timebase -t 16 Schubert | extract -i '**IPA' \
   | humsed 's/[^@VR&AaEiIoOWuUy]//' | humsed 's/./& /g; \
   s/  / /; s/ $//' | humsed -f vowel.map | rid -GLId | stats
```

Since translators have plenty of other issues to consider when translating a vocal text, we might focus our comparisons solely on a small collection of especially important notes. We might for example use a longer value for **timebase**. Alternatively, we might use the Humdrum **accent** command (described in Chapter 25) to identify notes have a particularly high noticeability.

Rhymes and Rhyme Schemes

Rhymes are common poetic devices throughout the world's cultures. Rhymes involve the use of similar or identical word-final phonemes. Typically, rhymes are based on the final phonemes of phrase-terminating words, but rhymes commonly occur in mid-phrase and other positions in poetry from various cultures. Consider the rhymes in the following traditional nonsense verse:

We're all in the dumps,
For diamonds are trumps,
The kittens are gone to St. Paul's
The babies are bit,
The moon's in a fit
And the houses are built without walls.
-Anon.

Suppose we want to automatically identify the rhyme scheme for this (or some other) text. Our first order of business is to identify phrase-terminating points. Let's assume we already have some phrase indicators (via curly braces { }). Our input might begin as follows:

```
**text   **IPA
We're    {wRr
all      Al
in       In
the      D@
dumps,   d@mps}
etc.
```

Using **extract, context** and **rid** we can isolate each poetic phrase:

```
extract -i '**IPA poem | context -b { -e } | rid -GLId
```

The result is as follows:

```
{wRr Al In D@ d@mps}
{foR dAim@nds Ar tr@mps}
{D@ kIt@ns Ar gAn tu seint pAUls}
{D@ beibiz Ar bIt}
{D@ munz In @ fIt}
{&nd D@ h&uz@z Ar bIlt wITAut wAUls}
```

The rhyming portion of words typically consist of a final vowel plus any subsequent consonants. We can isolate these phonemes using **sed**. Notice our use of back reference to preserve the final phonemes:

```
... | sed 's/.*\([@VR&AaEiIoOWuUy][^@VR&AaEiIoOWuUy]*}$\)/\1/'
```

The resulting output is:

```
@mps}
@mps}
Uls}
It}
It}
Uls}
```

A little further processing can remove the closing braces using **sed**, and eliminate the duplicate lines using **sort** and **uniq**.

```
... | humsed 's/}//' | sort | uniq
```

The output can then be changed into a set of substitutions for a **humsed** script. A suitable file would contain the following substitutions:

```
s/.*@mps$/A/
s/.*It$/B/
s/.*Uls$/C/
s/.*/./
```

This script will label all words ending with "umps" to 'A'. Word ending with "its" will be labelled 'B', and so on. All other words will be output as null tokens. Using this script, a suitable pipeline for processing our original file would be as follows:

```
extract -i '**IPA poem | context -b { -e } | humsed 's/}//' \
    | humsed -f rhyme | rid -GLId
```

The corresponding output would indicate the rhyme scheme for this poem:

```
A
A
B
C
C
B
```

Note that the entire analytic procedure can be placed in a shell script and applied to any input containing **IPA text. The following script adds a number of refinements.

```
# RHYME
#
# This script determines the rhyme scheme for an input file containing
# an **IPA spine.  This script assumes that the input contains curly
# braces indicating phrase endings.
#
# USAGE:  rhyme <filename>

extract -i '**IPA' $1 | extract -f 1 | context -b { -e } | rid -GLId \
    | sed 's/.*\([@VR&AaEiIoOWuUy][^@VR&AaEiIoOWuUy]*}$\)/\1/' \
    | sort | uniq | sed 's/^/s\/.*/; s/$/\/XXX\//' \
    | awk 'BEGIN {alphabet[1]="A"; alphabet[2]="B"; alphabet[3]="C";
       alphabet[4]="D"; alphabet[5]="E"; alphabet[6]="F";
       alphabet[7]="G"; alphabet[8]="H"; alphabet[9]="I";
       alphabet[10]="J"; alphabet[11]="K"; alphabet[12]="L";
       alphabet[13]="M"; alphabet[14]="N"; alphabet[15]="O";
       alphabet[16]="P"; alphabet[17]="Q"; alphabet[18]="R";
       alphabet[19]="S"; alphabet[20]="T"; alphabet[21]="U";
       alphabet[22]="V"; alphabet[23]="W"; alphabet[24]="X";
       alphabet[25]="Y"; alphabet[26]="Z"; alphabet[27]="ERROR"}
       {temp=$0
       gsub("XXX",alphabet[NR],temp)
       print temp
       }' > rhyme.sed.$$
extract -i '**IPA' $1 | extract -f 1 | context -b { -e } | rid -GLId \
    | sed -f rhyme.sed.$$
rm rhyme.sed.$$
```

Reprise

By focusing on phonetic signifiers, the **IPA representation provides opportunities for analyzing many sonorous aspects of vocal sounds — including alliteration, vowel coloration, rhyme, and other effects. Although we did not illustrate it in this chapter, the **IPA representation can be used in conjunction witht the **silbe representation to characterize complex aspects of rhythm and rhyme in vocal texts.