Melody Matcher

Project Proposal

April 2010 Update

Jennifer Hughes

AI, Fall 2009

Franz Kurfess

Abstract (System purpose and tasks to be performed):

Unmodified:

This project aims to facilitate semi-automated composition of melody and accompanying lyrics, by way of matching the musical emphases of a piece with the textual emphases of the lyrical phrases. The program will ultimately rely upon user input for "fitness" criteria, but will initially match lyrics with melodies based upon some rules.

Usage Domain and Environment:

I've needed to change my usage domain a bit. Now, my program can only reliably be used on songs that don't change notes in the middle of phrase/clauses.

User Feedback:

I have no new user feedback. I do, however, have some new requirements specification data from running my own trials on sample data.

System Design:

Currently, my project prototype is a series of excel spreadsheets, partially generated from manual input and input from .csv files, which were partially generated by some command-line text parsing.

Prototype and Implementation

Syllable Structure

As a reminder, a syllable has a nucleus, a coda, and an onset. The nucleus consists of a vowel (or semivowel), the coda consists of zero to four consonant phones, and the onset consists of zero to three consonant phones.

Length and Weight of Individual Phones

I've developed several tables to determine the weight and length of individual phones, divided by type.

Vowels

Vowels, unlike consonants, do not have any sort of sub-variables to make extensive calculations necessary. A vowel can be long, short, a semivowel, or a diphthong. A long vowel weighs more than a short vowel, which weighs about the same as a semivowel. A diphthong's weight/length is just the sum of its parts. A diphthong is a combination of two vowels or semivowels. You just add the lengths/weights of its members to get its value.

Vowel Type	Weighted Worth	Weight
syllabic consonant semivowel	0.25	0.5
short	0.25	0.5
long	0.75	1.5
diphthong	sum of parts	

Consonants

Consonants have a few variables that affect their weight.

First, "manner of articulation": plosives, affricates, and fricatives weigh the most, with plosives being the heaviest (effectively, though, they're the same weight). Nasals, laterals, approximants, and semivowels are all effectively "weightless", in and of themselves.

Second, voicing: to "voice" a phone means to cause the vocal chords to vibrate. "Voiced" phones are singable, whereas "voiceless" phones are not. "Voiceless" phones are really nothing more than a hiss, or directing streams of escaping air.

Third, the manner of starting and ending a sound has a bearing on weight. A "block" means that you must fully stop airflow through the mouth to make the correct noise to start the sound. A block is basically a buildup-then-release of a stream of air. "Continuous frication" is simply a continuous release of a directed air stream, without the primarily buildup. An "end", which is the only type of start/end that I made up a name for, is a variation of a block without the full stop of airflow through the mouth. It's a distinctive change in the sound of a phone, but does not "build up" the air stream; it just releases it. A "continuous" start or end is fully open, and does not require any sort of change in mouth position or air direction. A "redirect" is a combination of a continuous start/end and a block: it does not allow air to escape through the mouth, but does allow a continuous sound through the nose.

CONSTANTS	Type	Un-weighted worth	Weighted worth
Manner of Articulation	plosive	7	1.000
	affricate	7	1.000
	fricative	7	1.000
	nasal	0	0.000
	lateral	0	0.000
	approximant	0	0.000
	semivowel	0	0.000
Voicing	voiceless	2	1.000
	voiced	0	0.000
Starts/ends as	block	3	3.000
	continuous frication	1	1.000
	end	2	2.000
	continuous	0	0.000
	redirect	0.5	0.500

Calculating Consonant Phone Length

To figure out the length, based on all these variables, there's a pretty complicated formula. It didn't already exist, so I spent several days determining it via trial and error. Here's the Excel formula for determining the length of a phone in line 2 of the spreadsheet, which will be followed by psuedocode that should explain it.

```
=IF($D2=$K$2,
      $M$2,
      IF($D2=$K$3,
            $M$3,
            IF($D2=$K$4,
                  $M$4,
                  IF($D2=$K$5,
                        $M$5,
                        IF($D2=$K$6,
                              $M$6,
                              IF($D2=$K$7,
                                    $M$7,
                                    IF($D2=$K$8,
                                          $M$8,
                                          0
)     )     )     )     )     )     )
+ IF($E2=$K$9,
      $M$9,
      $M$10)
+ IF($F2=$K$12,
      $M$12,
      IF($F2=$K$13,
            $M$13,
            IF($F2=$K$14,
                  $M$14,
                  IF($F2=$K$16,
                        $M$16,
                        $M$15
)     )     )     )
+ IF($G2=$K$12,
      $M$12,
      IF($G2=$K$13,
            $M$13,
            IF($G2=$K$14,
                  $M$14,
                  IF($G2=$K$16,
                        $M$16,
                        $M$15
)     )     )     )
```

Psuedocode: (where **this** is a phone)

```
int sum = 0;
switch( this.MannerOfArticulation ){
     case plosive:
     case affricate:
     case fricative:
          sum += 1;
          break;
}
if ( this.voice == voiceless) {
     sum +=1;
}
switch( this.StartsAs ){
     case block:
          sum += 3;
          break;
     case continuousFrication:
          sum += 1;
          break;
     case end:
          sum += 2;
          break;
     case redirect:
          sum += 0.5;
          break;
}
switch( this.EndsAs ){
     case block:
          sum += 3;
          break;
     case continuousFrication:
          sum += 1;
          break;
     case end:
          sum += 2;
          break;
     case redirect:
          sum += 0.5;
          break;
}
return sum;
```

In the interest of space, I've not included conditions that would add zero to the sum.

Full Table of Phone Makeup and Length

SAMPA	Example	Type/ Manner of Articulation	Voiced/ Voiceless	Starts as	Ends as	Cat.	Weight
p	pen, spin, tip	plosive	voiceless	block	block	Con.	8
b	but, web	plosive	voiced	block	block	Con.	7
t	two, sting, bet	plosive	voiceless	block	block	Con.	8
d	do, odd	plosive	voiced	block	block	Con.	7
tS	chair, nature, teach	affricate	voiceless	block	cont fric	Con.	6
dZ	gin, joy, edge	affricate	voiced	block	cont fric	Con.	5
k	cat,kill,queen,thick	plosive	voiceless	block	block	Con.	8
g	go, get, beg	plosive	voiced	block	block	Con.	7
f	fool, enough, leaf	fricative	voiceless	cont fric	cont fric	Con.	4
v	voice, have, of	fricative	voiced	cont fric	cont fric	Con.	3
T	thing, breath	fricative	voiceless	cont fric	cont fric	Con.	4
D	this, breathe	fricative	voiced	cont fric	cont fric	Con.	3
s	see, city, pass	fricative	voiceless	cont fric	cont fric	Con.	4
z	zoo, rose	fricative	voiced	cont fric	cont fric	Con.	3
S	she,sure,emotion,leash	fricative	voiceless	cont fric	cont fric	Con.	4
Z	pleasure, beige	fricative	voiced	cont fric	cont fric	Con.	3
h	ham	fricative	voiceless	cont fric	cont fric	Con.	4
m	man, ham	nasal	voiced	redirect	redirect	Con.	1
n	no, tin	nasal	voiced	redirect	redirect	Con.	1
N	singer, ring	nasal	voiced	redirect	redirect	Con.	1
l	left, bell	lateral	voiced	continuous	continuous	Con.	0
r	run, very	approximant	voiced	continuous	continuous	Con.	0
w	we	semivowel	voiced	continuous	end	Con.	2
j	yes	semivowel	voiced	continuous	end	Con.	2
W	what (Scottish)	approximant	voiceless	continuous	end	Con.	3
x	loch (Scottish)	fricative	voiceless	cont fric	cont fric	Con.	4
A	father, not, law	short				Vow.	0.25
I	city	short				Vow.	0.25
E	bed	short				Vow.	0.25
3`/`r	bird, winner	short				Vow.	0.25
{	lad, cat, ran	short				Vow.	0.25
u	soon, through	short				Vow.	0.25
@	about	short				Vow.	0.25
jU/ju	use, pupil	diphthong		syllabic consonant semivowel	short	Dip.	0.5
i	see	long				Vow.	0.75
V	run, enough	short				Vow.	0.25
U	put	long				Vow.	0.75
e	day	long				Vow.	0.75
O	or, shore	long				Vow.	0.75
a	DNE in GenAm	long				Vow.	0.75
aI	my, height	diphthong		long	short	Dip.	1
OI	boy	diphthong		long	short	Dip.	1
oU/ou	boat	diphthong		short	long	Dip.	1
aU	now	diphthong		long	long	Dip.	1.5
=	ridden	syllabic consonant semivowel				Vow.	0.25

This page was intentionally left blank because Word doesn't like me being table-happy.

Structure of a Syllable

A syllable, defined here in the traditional, colloquial sense, is represented as such:

WORD	<syllable>		
NOTE_LEN	<length of associated note>		
EMPH	<syllable emphasis value>		
SAMPSYL	<syllable in SAMPA>		
SYL_PART	**on**	**nu**	**cd**
	<-1>	<0>	<+1>
	<-2>	<1>	<+2>
	<-3>	<2>	<+3>
			<+4>

WORD *(required field)* is the orthographic (regular-spelling) syllable. If this field only has a ^ in it, then it represents a rest.

NOTE_LEN *(required field)* is the length of the note that the syllable is associated with, represented as a number from 1 to N, where N is the denominator of the shortest note length. (for example, if the shortest note is an eighth note, then N would be 8.) N represents a whole note, and 1 would represent the shortest-length note. (In our previous example, an eighth note would be 1, and a whole note would be 8).

EMPH *(conditionally-required field)* is the emphasis value of the syllable, relative to the rest of the syllables in the word. It's required only if the syllable is a part of a multi-syllable word. If the syllable is a rest or a single-syllable word, then this field should be left blank.

SAMPASYL *(conditionally-optional field)* is the SAMPA spelling of the syllable. This field can be left blank only if the syllable is a rest.

SYL_PART *(conditionally-optional multi-part field)* is a multi-part field, representing the parts of a syllable. All of the values are written in SAMPA. This field can be left completely blank only if the syllable is a rest.

on *(optional array-like field-column)* is a column of cells representing a syllable onset. It contains four cells, but only up to three may be used. Each cell has a single SAMPA phone/"letter" in it. The phone at the top of the column is closest to the syllable nucleus, and as you go further down, the closer you get to the beginning of the syllable. Not all fields must be filled out. Valid values are consonants (meaning plosives, affricatives, fricatives, nasals, laterals, approximants, and semi-vowels). Semi-vowels that are directly adjacent to the nucleus should be part of the nucleus, not the onset.

nu *(conditionally-optional array-like field-column)* is a column of cells representing the syllable nucleus. It contains four cells, but only up to three may be used. Each cell has a single SAMPA phone/"letter" in it. The phone at the top of the column is closest to the onset, and those

further down are closer to the coda. At least one field must be filled out, unless the syllable is a rest. Valid values are vowels, semivowels (j in **y**es, w in **w**e), nasals-with-semivowels (n= in "hidd**en**", .m= in "winso**me**"), and laterals-with-semivowels (l= in "waff**le**").

cd *(optional array-like field-column)* is a column of cells representing a syllable coda. It contains four cells, all of which may be used. Each cell has a single SAMPA phone/"letter" in it. The phone at the top of the column is closest to the syllable nucleus, and as you go further down, the closer you get to the end of the word. Not all fields must be filled out. Valid values are consonants (meaning plosives, affricatives, fricatives, nasals, laterals, approximants, and semi-vowels). Semi-vowels that are directly adjacent to the nucleus should be part of the nucleus, not the coda.

Examples

For simplicity's sake, the note length will not vary on all the following examples.

Example 1: "o"

(Single Syllable, Single Phone)

.

o::IN/NNP/NN/LS: "ou :{o}

WORD	o		
NOTE_LEN	1		
EMPH			
SAMPSYL	ou		
SYL_PART	on	nu	cd
		ou	

Example 2: "no"

(Single Syllable, No Coda, Single-Phone Nucleus and Onset)

67992:no::DT/RB/UH: "nou :{no}

WORD	no		
NOTE_LEN	1		
EMPH			
SAMPSYL	nou		
SYL_PART	on	nu	cd
	n	ou	

Example 3: "on"

(Single Syllable, No Onset, Single-Phone Nucleus and Coda)

on::IN/JJ: "An :{on}

WORD	on		
NOTE_LEN	1		
EMPH			
SAMPSYL	An		
SYL_PART	on	nu	cd
		A	n

Example 4: "non"

(Single Syllable, Single-Phone Onset Nucleus and Coda)

non::FW/NN/NNP: "nAn :{non}

WORD	non		
NOTE_LEN	1		
EMPH			
SAMPSYL	nAn		
SYL_PART	on	nu	cd
	n	A	n

Example 5: "notes"

(Single Syllable, Single-Phone Onset and Nucleus, Double-Phone Coda)

notes::NNS/VBZ: "nouts :{note}>s>

WORD	notes		
NOTE_LEN	1		
EMPH			
SAMPSYL	nouts		
SYL_PART	on	nu	cd
	n	ou	t
			s

(Single Syllable, No Onset, Double-Phone Nucleus, Single-Phone Coda)

yes::UH/NN: "jEs :{yes}

WORD	yes		
NOTE_LEN	1		
EMPH			
SAMPSYL	jEs		
SYL_PART	on	nu	cd
		j	s
		E	

Example 6: "yes"

Example 7: "stringy"

(Single Syllable, Multi-Phone Onset, Single-Phone Nucleus and Coda)

stringy::JJ: "strIN$i :{string}>y>

WORD	string-			y		
NOTE_LEN	1			1		
EMPH	1			0		
SAMPSYL	strIN			i		
SYL_PART	on	nu	cd	on	nu	cd
	r	I	N		i	
	t					
	s					

Example 8: "nonmembership" (Multi-Syllable Word with Two-Tier Emphases)

nonmembership::NN: nAn"mEm$b@`r%SIp :<non<{member}>ship>

WORD	non-			mem-			ber-			ship		
NOTE_LEN	1			1			1			1		
EMPH	0			2			0			1		
SAMPSYL	nAn			mEm			b@`r			SIp		
SYL_PART	on	nu	cd	on	nu	cd	on	nu	cd	on	nu	cd
	n	A	n	m	E	m	b	@	`r	S	I	p

CLIPS definition of a phone and its subparts

```
( deftemplate mannerOfArticulation
        (slot label)
        (slot weight)
)

(deffacts moaTypes  "Types of manners of articulation"
        ( mannerOfArticulation ( label "plosive") ( weight 1 ) )
        ( mannerOfArticulation ( label "affricate") ( weight 1 ) )
        ( mannerOfArticulation ( label "fricative") ( weight 1 ) )
        ( mannerOfArticulation ( label "nasal") ( weight 0 ) )
        ( mannerOfArticulation ( label "lateral") ( weight 0 ) )
        ( mannerOfArticulation ( label "approximate") ( weight 0 ) )
        ( mannerOfArticulation ( label "semivowel") ( weight 0 ) )
        ( mannerOfArticulation ( label "syllabic consonant semivowel") ( weight 0.25 ) )
        ( mannerOfArticulation ( label "diphthong") ( weight 0 ) )
        ( mannerOfArticulation ( label "short vowel") (weight .25 ) )
        ( mannerOfArticulation ( label "long vowel") (weight .75 ) )

)

( deftemplate voicing
        (slot label)
        (slot weight)
)

(deffacts voicingTypes "Whether it's voiced or not"
        ( voicing ( label "voiceless" ) (weight 1) )
        ( voicing ( label "voiced" ) (weight 0) )
)

(deftemplate startsOrEndsAs
        (slot label)
```

```
        (slot weight)
)

(deffacts startOrEndAsTypes
        ( startsOrEndsAs ( label "block") (weight 3 ) )
        ( startsOrEndsAs ( label "cont fric") (weight 1 ) ) ;; continuous frication
        ( startsOrEndsAs ( label "end") (weight 2 ) )
        ( startsOrEndsAs ( label "continuous") (weight 0 ) )
        ( startsOrEndsAs ( label "redirect") (weight .5 ) )
        ( startsOrEndsAs ( label "syllabic consonant semivowel") ( weight 0.25 ) )
        ( startsOrEndsAs ( label "short vowel") (weight .25 ) )
        ( startsOrEndsAs ( label "long vowel") (weight .75 ) )
)

( deftemplate phoneCategory
        (slot label) ;; for vowel vs. consonant
)

(deffacts phoneCategoryTypes
        ( phoneCategory (label "consonant") )
        ( phoneCategory (label "diphthong")  )
        ( phoneCategory (label "vowel") )
)

(deftemplate sampaPhone
        (slot sampaSpelling)
        (slot phoneCategory )
        (slot mannerOfArticulation)
        (slot voicing)
        (slot startAs ) ;;(is-a startsOrEndsAs)
        (slot endAs )  ;;(is-a startsOrEndsAs)
        (slot weight ) ;;CALCULATE_ELSEWHERE
)
```

Fickle Phones in the Coda

Look, phones in the coda are like an easily-distracted dog. One moment, you think they're with you, right behind you, and then next, you turn around, and notice that they're less "behind you" than they are "in front of the person behind you". You are the syllable in this metaphor, and the person behind you is the syllable that comes after you. Your dog is the last consonant in your coda. Certain "dogs" will go join the "person" behind you, while others will stick with you! Don't judge—it's just the way they are. But you need to know if your dog's going to stick with you or not.

Phones That Stay In Coda:

"**m**" in "sa**m**-ba"

Phones That Move to Next Syllable's Onset:

"**d**" in "un-a-voi**d**-a-ble"

Calculating Weight/Length of Independent Syllables

The "Full Table of Phone Makeup and Length" table shows all the lengths and weights for all the phones, as stand-alone values. However, it does not take into account what effect that the surrounding phones might have on a phone's length. To determine that, we need to figure out the length/ weight for the whole syllable.

Psuedo Code Formula for Determining Semi-Stand-Alone Weight

```
=IF(THIS.NUCLEUSWEIGHT==0,
      0,
      (SUM OF (THIS.ONSETVALS[1-2] )/ [( COUNT OF (ONSETVALS[1-2] )
* 8)]
)
+IF(THIS.CODAVALS[0]==0,
      IF(THIS.NUCLEUSWEIGHT>1,
            1,
            THIS.NUCLEUSWEIGHT
      ),
      THIS.NUCLEUSWEIGHT
)
+ [SUM OF(THIS.CODAVALS[1-4]) ] /  [COUNT OF(THIS.CODAVALS[1-4])*
8]

- IF(NEXT.ONSETVALS[0] ==0,
      IF( THERE IS ONLY ONE CODA VAL,
            IF(THIS.NUCLEUSWEIGHT>1,
                  (THIS.NUCLEUSWEIGHT-1)+ [(THIS.CODAVALS[ last non-
empty index ]) /  8 ],
                  [ (THIS.CODAVALS[ last non-empty index ]) /  8 ]
            ),
            [ (THIS.CODAVALS[ last non-empty index ]) /  8 ]
      ),
      0
)
```

Actual Excel Formula:

```
=IF(O2=0,
      0,
      (SUM(L2:M2)/(COUNT(L2:M2)*$AE$2))
)
+ IF(P2=0,
      IF(O2>1,
```

```
            1,
            O2
      ),
      O2
)
+(SUM(Q2:T2)/(COUNT(Q2:T2)*$AE$2))

-IF(N3=0,
      IF(OFFSET(T2,0,-COUNTIF(P2:T2,0))=P2,
            IF(O2>1,
                  (O2-1)+((OFFSET(T2,0,-
COUNTIF(P2:T2,0)))/((COUNT(Q2:T2)-COUNTBLANK(Q2:T2))*$AE$2)),
                  OFFSET(T2,0,-COUNTIF(P2:T2,0))/ ((COUNT(Q2:T2)-
COUNTBLANK(Q2:T2))*$AE$2)
            ),
            OFFSET(T2,0,-COUNTIF(P2:T2,0))/((COUNT(Q2:T2)-
COUNTBLANK(Q2:T2))*$AE$2)
      ),
      0
)
```

Calculating Length of Syllables in Relation to Each Other

Just like the weight of phones cannot be fully quantified in isolation, syllable length/weight is affected by surrounding syllables.

If a syllable's last phone is the same as the next syllable's first phone, then it you can disregard the weight of one of them. Singers often only sing one of them, and use it for both words. For example: Bake cookies becomes : bay-cookies Stars shining becomes : star-shining. Yelp peppers becomes: yell-peppers.

There are several other rules, which I've put in the algorithm/formula, but haven't written out yet. That's on the menu for a later update.

Psuedo Code Formula for Determining Syllable “fit” with the music.

```
=IF(U18 > A18,
      IF(D19=F18,
          IF( (U18 - 1 <= A18 + A19 ) ,
                1,
               0
          ),
          IF(D19 = 0,
               IF(U18 <= A18 + A19,
                     1,
                    0
               ),
               0
          )
     ),
     1
)
=IF(U18 >= A18,  IF(D19=F18, IF( (U18 - 1 <= A18 + A19 ) , 1, 0),
IF(D19 = 0, IF(U18 <= A18 + A19, 1, 0),   0) ),   1)

=IF(U18 > A18,  IF(D19=F18,    IF( (U18 - 1 < A18 + A19 ),  1, 0),
IF(D19 = 0,  IF(U18 < A18 + A19, 1, 0),   0),   1)

=IF(U18 > A18,
     IF(N19 = 0,
          IF(U18 < A18 + A19,
               1,
               0
          ),
          0
     ),
     1
)
```

Patterns for Musical Emphases

RULES:

Single syllable words are useful as “filler”, in that they can be used to fill emphasized or un-emphasized “slots” in the music.

Multi-syllable words must have their emphasized and unemphasized syllables match up with the emphasized or unemphasized “slots” in the music.

Effect of Note Intervals on Musical Emphases

In my continued research for the rules of emphases calculation, I realized that there was another factor that influences emphasis patterns: Note intervals. "Intervals" are changes in note values: that is, not changes in length, but in the actual sound's frequency. I will not be applying this in my algorithm for this project, which is okay, because I will attempt to only apply my algorithm to songs that only have a single note.

I realized my omission when attempting to apply my algorithm to the chorus of The Bloodhound Gang's, "The Bad Touch". The first threes clauses of the chorus are as follows:

"You and me" "baby ain't" "nothing but mammals"

This breaks down like this:

WORD	You			and			me			ba-			by			ain't		
NOTE_LEN	1			1			2			1			1			2		
EMPH										1			0					
SAMPSYL	ju			@nd			mi			be			bi			ent		
SYL_PART	**on**	**nu**	**cd**	**on**	**nu**	**cd**	**on**	**nu**	**cd**	**on**	**nu**	**cd**	**on**	**nu**	**cd**	**on**	**nu**	**cd**
		ju			@	n	m	i		b	e		b	i			e	n
						d												t

WORD	no-			thing			but			mam-			mals		
NOTE_LEN	1			1			1			1			1		
EMPH	1			0						1			0		
SAMPSYL	nV			TIN			bVt			m{			ml=z		
SYL_PART	**on**	**nu**	**cd**	**on**	**nu**	**cd**	**on**	**nu**	**cd**	**on**	**nu**	**cd**	**on**	**nu**	**cd**
	n	V		T	I	N	b	V	t	m	{		m	l=	z

The main thing to notice here is the emphasis pattern:

you – and – me – ba – by – aint – no – thing – but – mam - mals

x – x – xx – 1 – 0 – xx – 1 – 0 – x – 1 – 0

(**x** is a wild-card syllable from a single-syllable word. **1** is an emphasized syllable in a multi-syllable word. **0** is an un-emphasized syllable in a multi-syllable word.)

Emphasis patterns should always be 1 – 0 – 1 – 0 , at least for the purposes of this project. With this pattern, though, there's no way to make it fit that structure, because the two-syllable words "nothing" and "mammals" are only separated by a single syllable. This means that our emphasis pattern according to our current algorithm where we only look at syllable length and emphasis patterns, must be one of the following:

you – and – me – ba – by – aint – no – thing – but – mam - mals

1 – 0 – 10 – 1 – 0 – 10 – 1 – 0 – 1 – 1 – 0

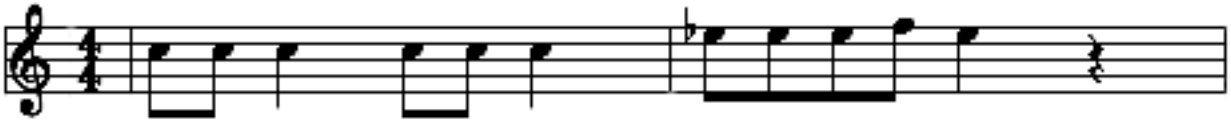
OR

you – and – me – ba – by – aint – no – thing – but – mam - mals

1 – 0 – 10 – 1 – 0 – 10 – 1 – 0 – 0 – 1 – 0

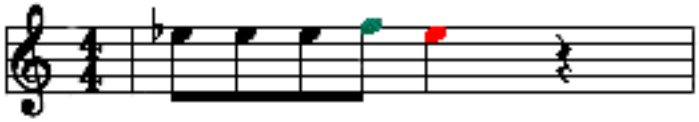
Either way, in the third clause, "nothing but mammals", we have two adjacent syllables of the same emphasis value. Under our current system of rules, this isn't allowed. However, the song's emphases still *sound* right, so there's another factor affecting emphasis.

As it turns out, the music for this part sheds light on the situation:



You and me_____ba- by, ain't_____ noth-ing but mam-mals

The first two clauses, "You and me", and "baby, ain't", have no note intervals; that is, the note doesn't change throughout the clause. In the third clause, there is a positive note interval between "but mam-", and a negative note interval between "mam-mals":



noth-ing but mam-mals

1 – 0 – 1 – 1 – 0

It appears, based upon this, that if two words clash such that the emphases are not alternating, it can be overcome by recreating the emphasis with note values: an emphasized syllable will be indicated by a positive note interval.

If time permits, I'll work this into my algorithm, but as it stands now, I'd need to do a lot more research and try a lot more use cases to determine if this is a fully-generalizable rule. For now, I'll only apply my algorithm to songs that do not have any interval changes in their lyric clauses.

Parsible Phonetic Dictionary

I used the UNISYN dictionary, using General American SAMPA output, as the base for my phonetic dictionary.

First, I downloaded the dictionary and related files from the UNISYN website.

Once I'd extracted them, I attempted to run the provided perl scripts to turn the UNISYN phonetic symbols into SAMPA phonetic symbols, using the procedure specified in Section 6.1 of their Documentation1_3.pdf, and towncode "gam".

There were a few errors in translation I fixed manually:

' d ' instead of ' 4 '

' 5 ' instead of ' l '

' `r\ ' instead of ' r '

' 3 ' instead of ' 3` '

In the case of the last one, I actually didn't fix it, and just decided to use a modified sampa. Similarly, with the second to last, I left the ` in, and ignored them as they came along during parsing.

Here is the format for the raw fields, after I was done with fixing the UNISYN output:

<ortho>:<uniqueID>:<partOfSpeech>:<SAMPAspelling>:<extendedOrtho>:<freq>

Example:

transfer:2:VB/VBP: tr{ns"f3`r :{trans==fer}:7184

<ortho> is the regular spelling of the word

<uniqueID> is a number (and optional string) used to differentiate homographs.

<partOfSpeech> is used to identify the specific part of speechi

<SAMPAspelling> is the breakdown of the word, phonetically. It uses the SAMPA alphabet, and separators to show where breaks in the word are, and how they're emphasized. If a separator is ' $ ', the following phones (until the next separator) are not emphasized. If it's ' % ', then they are the secondary emphasis. If it's ' " ', then they are the primary emphasis.

<extendedOrtho> and <freq> have no particular use to me, at this time. They are an alternate breakdown of the word's roots, and the frequency at which it occurs in language, according to UNISYN.

Original Dictionary-Parsing Design Decision

My next step, to get these into the format that I need for my program , was to split up the fields into sub-fields to make them into a .csv (but using tabs as the field delimeter). I decided to do that like this (I'm using | as the field delimeter here, because it looks better, even though I'm using tabs for parsing)

```
          <ortho> | : | <uniqID#> | <uniqIDString> | : |
      <partOfSpeech> | : | <extendedOrtho> | : | <freq> |
<FullSAMPAstring> | : | syllableBreakdown(see below)...[1] | '\n' |

syllableBreakdown:
          <charSeparator> | <sampaSyllableSubstring>
```

The end of each word entry is indicated by a newline.

After I got the dictionary in this format, I opened it as a CSV in excel, inserted a column of commas between each field, a column of '));' after the last field, and a column with with string "dict.add(Word(" before the first field.

This was my hack-y way of making constructors for all the words in my dictionary. All I had to do was highlight the table, then copy and paste it as plain text into my code's dictionary constructor.

Adjustment for the CLIPS format

For the second iteration of Melody Matcher, we used a rule-based system based on CLIPS and C++, so some adjustments had to be made. We decided that, instead of a Dictionary class which populated itself upon construction, we would store the dictionary words as CLIPS facts.

I created a template for a "word" fact as the primary housing for my dictionary entries.

```
( deftemplate word
        ( slot orthoSpelling )
        ( slot uniqueID )
        ( slot partOfSpeech )
        ( slot FullSAMPAstring )
        ( multislot syllables )
        ( slot etymology )
        ( slot freq )
)
```

---

[1] The number of "SyllableBreakdown"s corresponds to the number of syllables in the word.

I did some text manipulation to get the dictionary file into this form, and then loaded it into CLIPS.

Incorporating CLIPS data into the Melody Match Program

*Importing Environment

Parsing a Song into Melody Matcher from MusicXML

My colleague, Ilona Sparks, joined me for a quarter to work on Melody Matcher. Her main contribution was the addition of automated song parsing, using input adhering to the MusicXML [2]format.

[2] I should have some sort of link here to MusicXML stuff

Evaluation Plan:

At this time, evaluators should read through this document, and check out the Excel Spreadsheet with the examples in it.

Timeline:

(Modified)

Again, the research and development of the algorithm has taken much more time than I thought it would. At this point, my goal is to get some sort of demonstration of the algorithm together by Week 10, along with full documentation of how it works.

---

[i] INCLUDE THE PARTS OF SPEECH TABLE!