Using Search to Identify Phone Sequences in Lyrics with Multiple Interpretations

Jenee Hughes, 480 Makeup Work

Main use cases are from the site in the footnote.[1]

Oronyms: Sentences that can be interpreted by the ear as two different sequences of words, without any of the sounds or emphases changing.

Example:

I'm taking a nice cold shower.

I'm taking an ice cold shower.

WORD	a			nice			cold			show er		
SAMP SYL	V			naIs			koul d			SaU` r		
SYL_ PART	n	u	d	n	u	d	n	u	d	n	u	d
					I			u			U	r

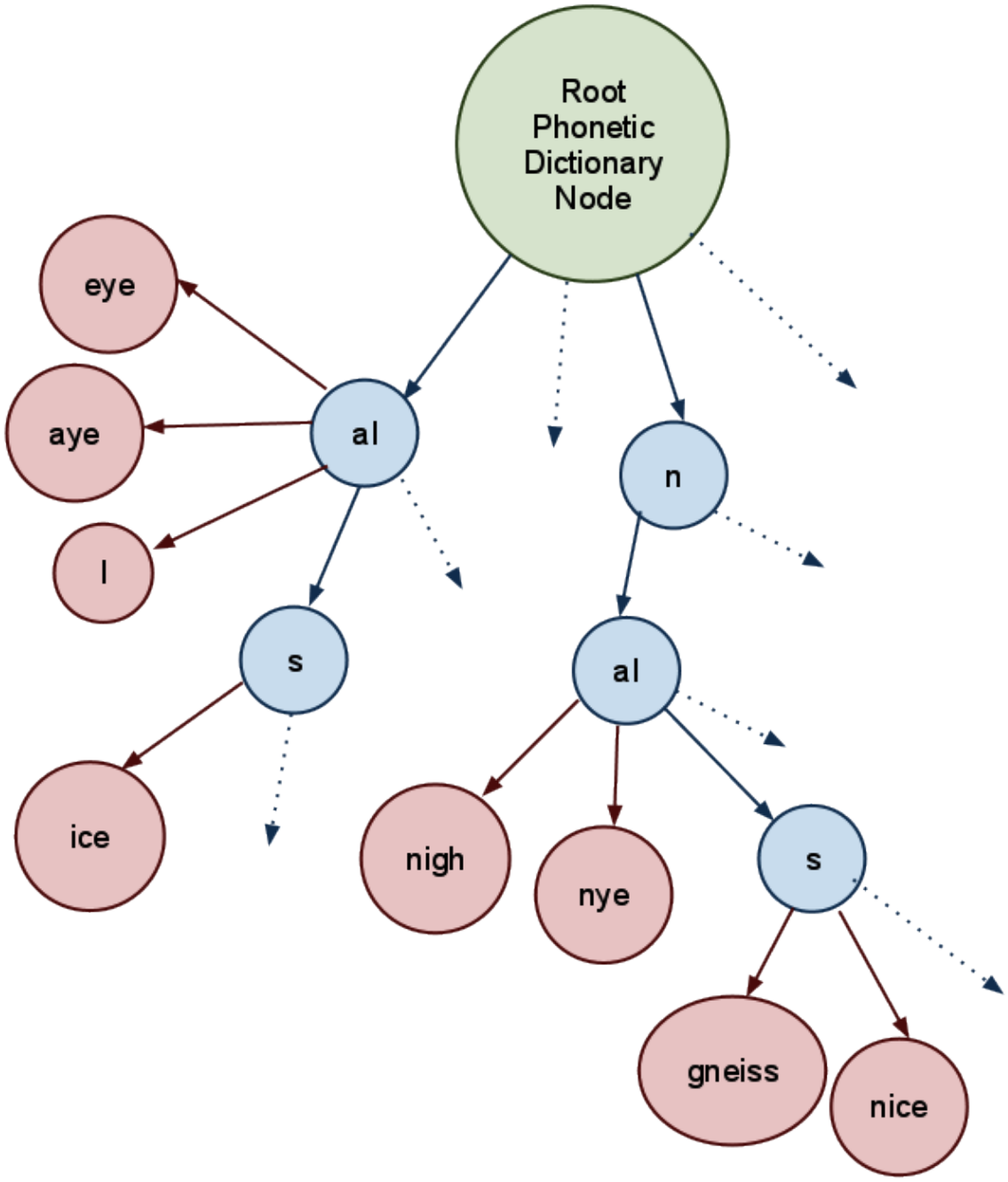
WORD	an			ice			cold			show er		
SAMP SYL	Vn			aIs			koul d			SaU` r		
SYL_ PART	n	u	d	n	u	d	n	u	d	n	u	d
					I			u			U	r

The phone sequence for both is:

V n aI s k ou l d S aU 'r

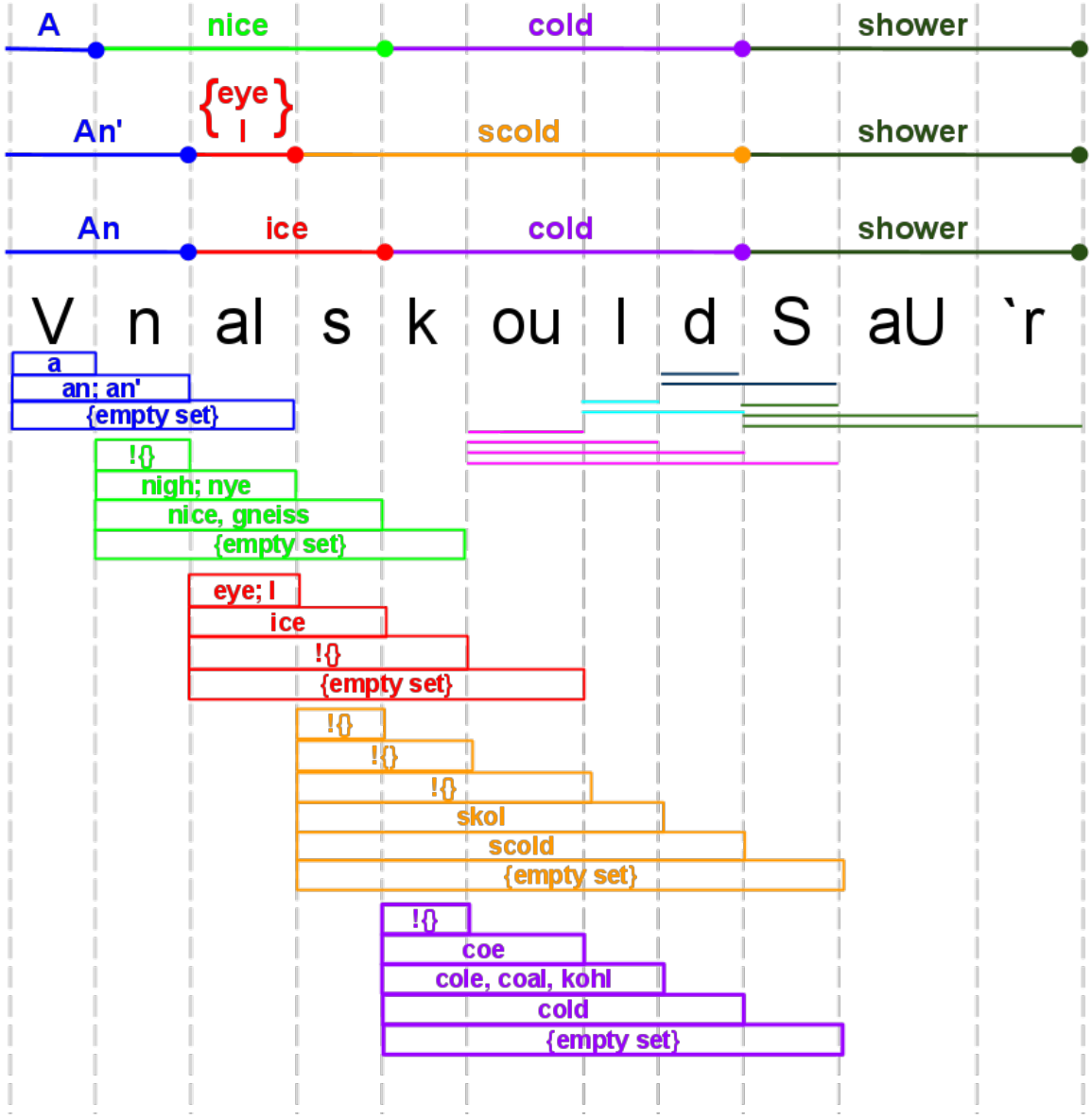
[1] Most of my use cases came from here:
http://www.wordinfo.info/words/index/info/view_unit/3347/

The ideal way to think about searching for patterns is by picturing the phoenetic dictionary in a tree form. Each node has at least 45 child nodes: one for each phone. A node might also have "word" nodes, if the node's path constructs a valid word:



When there are ties for word nodes, you can determine the most likely interpretation by checking the frequency of each word. For example, the sequence "n aI s" is much more likely to be "nice" than "gneiss".

So, back to our original oronym: "A nice cold shower/An ice cold shower". Using the dictionary tree method above, we can break down the phones according to the following diagram:



First Option for Algorithm/Approach

```
MatchLists is an array contains n lists of Word Matches, where n is the number of
phones in the origPhoneSequence.

foreach(Phone p in origPhoneSequence)
{
	set SubSequence to be empty;
	set the list of Matches to be empty;
	set the list of Candidates to be the whole dictionary
	set Phone* curLast to &p
	do
	{
		append curLast to SubSequence
		remove words in Candidates that don't begin with SubSequence
		if any word in Candidates is an exact match of SubSequence
		{
			add word to Matches
		}
		increment curLast to point to the next phone in origPhoneSequence
	}
	while ( curLast != NULL) (there are phones left in origPhoneSequence)
	copy Matches list to a list of MatchLists
}
Word* findValidOronyms( SequenceToInterpret)
foreach (Phone p in SequenceToInterpret)
{
	clear LyricOption (a list of possible wordLyrics);
	get p's associated Matches from MatchLists
	foreach ( Word w in p's Matches)
	{
		extract phones from w into wordPhones
		call findValidOronyms on (sequenceToInterpret - wordPhones at front)
		if findValidOronyms returns NULL, move on to next word W
		else ////SOMEHOW GET THE WORD SEQUENCE….
	}
}
```

Second Approrach option

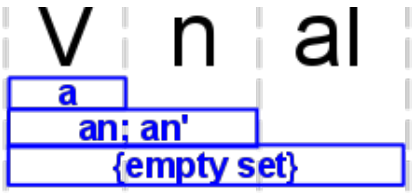
Variables and Types of Data Structures:

origPhonePhrase: The lyrics of the song, broken down into phonemes.

(Could be encompass all the lyrics, or just a lyric segment)

allPhoneSubPhrases: A list of phoneSubPhrase structures. There's one structure per phoneme in the origPhonePhrase.

phoneSubPhrases: Per phoneme in the origPhonePhrase, create a



phoneSubPhrase structure. In this structure, for each phone following the phone that is the basis for that phoneSubPhrase, concatenate them to create a phoneSubSubPhrase. Stop adding phones when the interpretation of the phoneSubSubPhrase is the empty set.

(We'll eventually compute the likelihood of aural interpretation of each subsubphrase and store it in this data structure / use this data structure to see which aural interpretation is most likely).

PhoneSubPhrase for "V" (basically a list of phones and their associated likelihoods)			
Phoneme	V	n	aI
Complete matching words	a (freq = 7536297)	an (freq = 794169) an' (freq = 794169)	{empty set}
# of words that are prefix-ed with this	########	########	########
Likelihood of hearing this as a prefix or as complete	weighted ratio of (sum/avg of frequency of all matches) to (sum/avg of frequency of all prefix-ed words)	weighted ratio of (sum/avg of frequency of all matches) to (sum/avg of frequency of all prefix-ed words)	weighted ratio of (sum/avg of frequency of all matches) to (sum/avg of frequency of all prefix-ed words)

```
For each phone P in origPhonePhrase:
        Construct a phoneSubPhrase structure.
        set phone S = P
        add phone S to the phoneSubPhrase structure
        set i = 0
        clear list-of-partial-matches
        while (there is a partial or full dictionary match for phoneSubPhrase )
                Add exact word matches and their frequencies to the phoneSubSubPhrase
                        structure for  S
                Add partial/prefix word matches' dictionary entries to list-of-partial-
                        matches
                compute heuristic to determine continuation or not
                i++
                continue
        compare heuristics of each phoneSubSubPhrase  in this phoneSubPhrase,
                weighting them accordingly.  Determine which is most likely.
        continue;
For each word in the original lyrics:
        find the phoneSubPhrase  that corresponds to the first phone in the word
        Determine how likely the phone breakdown is to be heard correctly by comparing
                the heuristics.
        Report findings to user.
```

Heuristics for Aural Interpretation

Note on measuring aural interpretation:

Keep track of the likelihood that the listener will interpret a phone phrase:

(1) as a complete word

(a.k.a. stop, completion, or **complete**)

(2) as the beginning of a longer word.

(a.k.a. continue, continuation or **prefix**)

Do this by keeping track of a few factors:

- the frequency of occurrence in the English Language[2] of words that are **full** matches.
- the frequency of occurrence in the English Language[3] of words that are **partial/prefix** matches.
- Possible addition later: account for the (hypothesized) human tendency to expect certain-length words. (For my project, I was thinking about assuming that, all other factors equal, we're more biased towards hearing two-syllable words, as opposed to longer or shorter words. It's likely more complicated than this, and most probably not universal, which is why I decided not to implement it this way.)

[2] according to UNISYN

[3] according to UNISYN

Related Test Cases (Unfinished Section)

There are cases in which it would be possible for a sequence of phones to have a higher prefix likelihood than a complete likelihood. Example:

"All I wanna do, is have some fun. I got a feeling, I'm not the only one.
All I wanna do, is have some fun, until the sun comes up **over Santa Monica Boulevard"**
until the sun comes up or i'll settle for a couple of hours
All I want to do is have some fun until the sun comes up on the sentimental look at the farm
"sun comes up on a sana on a couple of arrs"
Actually says "San**na Mon**aca Boulevard"

Though I haven't computed heuristic for this case yet, the phrase. The problem here is accidental symmetry. The Sanna Monaca thing make the listener think that the second word ends sooner than it actually does.

Dude looks like a lady
Do it like a lady

Give me the Beach Boys and free my soul.
Give me the beat, boys, and free my soul.