MISHEARD ME ORONYM TREE: A VISUALIZATION OF PHONETIC AMBIGUITY IN PROSE INTENDED FOR THIRD-PARTY PERFORMANCE

A Thesis

Presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Jennifer "Jenee" Gayle Hughes

COMMITTEE MEMBERSHIP

TITLE:	Misheard Me Oronym Tree: A Visualization of Phonetic Ambiguity in Prose Intended For Third-Party Performance
AUTHOR:	Jennifer "Jenee" Gayle Hughes
DATE SUBMITTED:	June 2012
COMMITTEE CHAIR:	Zoë Wood, Ph.D.
COMMITTEE MEMBER:	Franz Kurfess, Ph.D.
COMMITTEE MEMBER:	John Clements, Ph.D.

Abstract

Misheard Me Oronym Tree: A Visualization of Phonetic Ambiguity in Prose Intended For Third-Party Performance

Jennifer "Jenee" Gayle Hughes

In this project, we developed a visualization of the ambiguity of written prose. Given a textual phrase, our program determines all oronyms for that phrase, or possible ways that that phrase is likely to be misheard. It then creates an oronym parse-tree visualization, with each branch fork indicating that a phonetic sequence can be interpreted in more than one way. Each branch segment represents an orthographic word, and the branch radius is scaled to the word's frequency of use in everyday language.

Given any valid English phrase, our system will first generate all possible correct phonetic sequences for a General American accent. Then, it parses through these phonetic transcriptions depth-first, looking for valid orthographic words for each subsequent phonetic subsequence, generating full and partial phrases from these words. While it is doing so, a tree branch is generated on screen for each possible orthographic divergence. In the event that a branch's phonetic "tail is not orthographically interpretable, we visually dead-end the branch by drawing a red sphere. In the event that the entire phonetic sequence can be parsed into a valid orthographic phrase, we indicate this successfully-found oronym with a green sphere.

This visual representation allows users to see how many ways a phrase can be interpreted, and most novely, where dead-end interpretations of the phrase's phonetic sequence exist. A particularly strong orthographic partial phrase before a phonetic dead-end can mislead a listener, causing them to lose track of the

words in the rest of the phrase.

Our visual representation does not take into account n-gram word-proximity, which causes the visual representation to incorrectly weight some branch paths. However, overall, it does a fairly good job of weighting the likelihood that a listener will follow an oronym branch's particular interpretation as they listen to a phrase.

In addition to implementing the visualization, we did a multi-phase user study, incorporating over 851 data points from 208 test subjects. In it, we tested the validity of our oronym generation by having people read a oronym phrase aloud and send us the recording, and by having people transcribe pre-recorded oronyms. In the first phase, we generated oronym strings for the phrase *"a nice cold hour"*, and had over 12 people make 65 recordings of the most common oronym phrases. We then compared their pronunciations to the pronunciations we were expecting, and found that in all cases, the recorded phrase's phonemics matched our expectations. In the second phase, we selected 15 of the phase one recordings, and had 30 to 60 different people transcribe each one. In the aggregated transcriptions, the most commonly transcribed phrases roughly corresponded with our metric for the most likely oronym interpretation of the phrase in the recording.

Contents

List of Tables viii

List of Figures ix

1 Preliminary Vocabulary 1

1.1 Mondegreens . . . 1

1.2 Oronyms . . . 2

1.3 Orthography . . . 2

1.4 Phonetics and Phonology . . . 3

1.4.1 Phonetics . . . 3

1.4.2 Phonology (aka phonemics) . . . 3

1.4.3 Phonetics Vs Phonology . . . 4

1.5 Phonemic/Phonetic Alphabets . . . 5

1.5.1 SAMPA . . . 5

2 Introduction 7

2.1 You and me...and Leslie? . . . 7

2.2 Why it breaks down . . . 9

2.3 Intended audience . . . 11

3 Implementation 12

3.1 Customized Phonetic Dictionary . . . 12

3.1.1 Dictionary Options . . . 13

3.1.2 Formatting the dictionary . . . 15

3.1.3 Custom dictionary fields . . . 16

3.1.4 Transferring the dictionary to a sqlite database . . . 18

3.2 Oronym Generation . . . 18
3.2.1 Step 1: find all phonemic variations of an orthographic phrase 18
3.2.2 Step 2: Finding all Orthographic phrases for a Phonemic Sequence . . . 21
3.2.3 Word Frequency Evaluation . . . 22
3.3 Visual Representation . . . 23
4 User Study **46**
4.1 Structure . . . 46
4.2 User Sampling Population . . . 47
4.3 Methodology . . . 47
4.3.1 First Phase: Recitation . . . 47
4.3.2 Second Wave: Transcription . . . 49
4.3.3 Recording Sample Pool . . . 49
4.3.4 Transcription of recordings . . . 50
5 Results **52**
5.0.5 Transcription oronym phrase frequency vs calculated frequency . . . 52
5.0.6 Individual Recording/Transcription Breakdowns . . . 56
6 Future Work **63**
6.1 Target Audience and Goals . . . 63
6.2 Melody Matcher Alpha . . . 64
6.2.1 PRACTICAL EXAMPLE OF UNDERLYING THEORY . 67
7 Conclusion **74**
7.1 Successes . . . 74
7.2 Places for improvement . . . 74
7.2.1 Frequency Validity . . . 76
7.2.2 Higher-order frequency data . . . 77
Bibliography **82**

List of Tables

3.1 All Oronyms for 'A Nice Cold Hour' with frequency values 32

4.1 Phrases Recorded 48

4.2 Countries and responses 51

5.2 Phrase word frequency sum vs times transcribed 56

7.1 SAMPA phoneme weight breakdown 81

List of Figures

1.1 The difference between phonetics and phonology 4
1.2 Dictionary IPA screenshot 5

2.1 Annotated Oronym Parse tree generated for the phrase "fever pitch" 11

3.1 Geographic Origin of General American 13
3.2 CMU dictionary entry example 14
3.3 Custom dictionary entry example 16
3.4 queryDBwithOrthoWordForSampa example 19
3.5 19
3.6 Pseudocode for findAllPhoneSeqsForOrthoPhrase 20
3.7 22
3.8 26
3.9 Pseudocode for discoverOronymsForPhrase 27
3.10 Code for buildAndDrawFullTree 28
3.11 Code for drawBranchesAtFork 29
3.12 Oronym Parse Tree 30
3.13 Annotated Oronym Parse Tree 31

4.1 Responses Per Country 50

5.1 Most Common Transcriptions Globally 53
5.2 Most Common Transcriptions from American respondents 54
5.3 Bubble Chart of All Transcribed Phrases mapped against their predicted frequency 55

5.4 Most common transcriptions for the recorded phrase "a nice coal dower" 57
5.5 Most common transcriptions for the recorded phrase "aNiceColdOur" 57
5.6 Most common transcriptions for the recorded phrase "aNighScoldOur" 58
5.7 Most common transcriptions for the recorded phrase "aNyeScold-Hour" 58
5.8 Most common transcriptions for the recorded phrase "aNyeScoldOur" 58
5.9 Most common transcriptions for the recorded phrase "anAyeScold-Hour" 59
5.10 Most common transcriptions for the recorded phrase "anEyeScoldOur" 59
5.11 Most common transcriptions for the recorded phrase "an Ice-Cold Hour" 59
5.12 Most common transcriptions for the recorded phrase "anIceCole-Dower" 60
5.13 Most common transcriptions for the recorded phrase "anIceKohlDower" 60
5.14 Most common transcriptions for the recorded phrase "an$_{I}ceCoalDower$" 60
5.15 Most common transcriptions for the recorded phrase "a NiceColdOur" 61
5.16 Most common transcriptions for the recorded phrase "ehNiceCole-Dower" 61
5.17 Most common transcriptions for the recorded phrase "onIceCold-Hour" 61
5.18 Most common transcriptions for the recorded phrase "onIceCoal-Dower" 62

7.1 Bubble Chart comparison of Frequency for deer, does, and bucks . 77

Chapter 1

Preliminary Vocabulary

Before we start, there are a few uncommon terms we will use fairly often in this paper. We have briefly defined them here.

1.1 Mondegreens

A mondegreen is a word or phrase resulting from a misinterpretation of a word or phrase that has been heard[11]. The word was coined by American author Sylvia Wright in her article, "The Death of Lady Mondegreen", published in a 1954 issue of Harper's Bazaar. In it, she describes the origin of the word:

> When I was a child, my mother used to read aloud to me from Percy's Reliques, and one of my favorite poems began, as I remember:
>
> > Ye Highlands and ye Lowlands,
> > Oh, where hae ye been?
> > They hae slain the Earl O' Moray,
> > And Lady *Mondegreen.*

The fourth line of the quote is actually "and laid him on the green"[23].

Additional commonly-cited mondegreens include[7][19][21]:

Gladly the Cross-Eyed Bear	Gladly the Cross I'd Bear
Scuse me while I kiss this guy	Scuse me while I kiss the sky
There's a bathroom on the right	There's a bad moon on the rise

1.2 Oronyms

Oronyms are phrases that may differ in meaning or spelling, but sound near-identical when spoken. They are similar to mondegreens, and the terms are often used interchangeably. The difference, however, lies in the context. The label "mondegreen" is used more often in regards to music lyrics, where pronunciation can be affected by the addition of music and tone to the phrase. Oronyms, on the other hand, refer to spoken words, not sung lyrics.[**?**]

Common oronyms include:

i scream	ice cream
an ice cold hour	a nice cold hour
grape ants	gray pants
real eyes	realize

1.3 Orthography

The word 'orthographic' comes from the Latin *orthographia*, meaning *correct* writing. Orthography itself is the part of language study concerned with letters and spelling. More specifically, it's the standardized system of writing down words in a specific language, using a commonly-accepted set of letters according to accepted usage. [13]

The orthographic symbol set for a language is the commonly-accepted set of letters used to spell words in that language. In English, our orthographic symbol set is the Latin alphabet.

In this paper, "orthographic phrase", refers to a sequence of regularly-spelled words found in an English dictionary.

Example: "This is a orthographic phrase."

1.4 Phonetics and Phonology

To discover oronyms for a phrase, we must first to translate the root orthographic phrase to a representation that allows us to unambiguously measure pronunciation. Phonology and phonetics are branches of linguistics that deal with pronunciation.

1.4.1 Phonetics

Phonetics is a branch of *descriptive* linguistics, and refers to the study of the actual, uttered sound of human speech. It deals with describing the physical phenomena of how these sounds are produced from the vocal tract, how they are transmitted once spoken, and how they are recieved by audiences. The building blocks of phonetics are *phones*, which represent atomic sounds.

1.4.2 Phonology (aka phonemics)

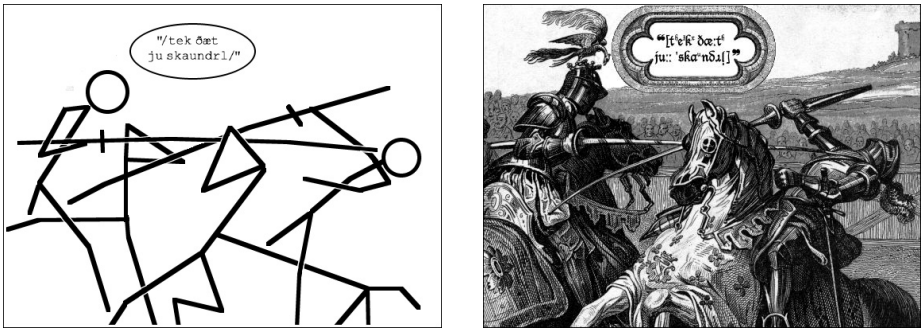
Phonology is a branch of *theoretical* linguistics, and as such, is primarily concered with the abstract grammartical characterization of sounds. It describes

the way that sounds function within a language and give meaning to words. The basis of phonological analysis is the grouping of sounds (*phones*) into distinct units within a languages. These distinct units are called *phonemes*.

These phonemes may contain different phones, depending on the accent of the speaker. For example, native speakers of General American English only generally recognize one 'L' sound phoneme. However, there are two different ways that that phoneme manifests itself: the 'l' in "male", and the 'l' in late. This difference is not noticable to a native speaker of American English, because that particular accent will parse any 'L' phone as the same 'L' phoneme.

1.4.3 Phonetics Vs Phonology

As we said previously, though the terms are sometimes used interchangably, the words 'phonemic' and 'phonetic' (and their corresponding sound building blocks, 'phone' and 'phoneme') indicate a different stages of sound parsing. *Phonemes* are idealized sounds; *phones* are the actual sounds that come out of a person's mouth. Figure 1.1 provides a final, illustrative metaphor of the difference.



(a) Phonology[8] (b) Phonetics[9]

Figure 1.1: The difference between phonetics and phonology

1.5 Phonemic/Phonetic Alphabets

As we stated in section 1.4.2, phonemes are the atomic building blocks of words. In a phonemic alphabet, every meaningful sound has its own letter. The way that we interact with phonemes in a concrete way is by using phonetic alphabets and phonetic dictionaries.

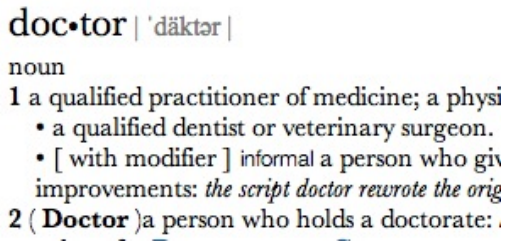

doc•tor | ˈdäktər |
noun
1 a qualified practitioner of medicine; a physi
• a qualified dentist or veterinary surgeon.
• [with modifier] informal a person who giv
improvements: *the script doctor rewrote the orig*
2 (**Doctor**)a person who holds a doctorate:

Figure 1.2: The characters to the right of the large bold word "doctor" are IPA symbols.

The most common phonetic alphabet is the IPA (Internation Phonetic Alphabet). It contains representations of every sound in every known language globally, and allows for cross-cultural pronunciation guidelines. As shown in figure 1.2, IPA representations of orthographic words are found in traditional dictionaries to aid pronunciation.

1.5.1 SAMPA

SAMPA (Speech Assessment Methods Phonetic Alphabet) is a computer-readable phonetic alphabet, based upon the symbols found in the more-standard-but-not-easily-computer-readable IPA (International Phonetic Alphabet). It uses "letters" consisting of 1-2 ASCII characters to represent each phoneme. The ASCII sequences for the SAMPA letter are designed so that any SAMPA sequence is deterministically parsible.

We chose to use SAMPA instead of IPA because its ASCII-compliance makes it easy to integrate into other systems.

See table 7.2.2 for a full table of each SAMPA phoneme, its description, and its sub-parts.

For some brief context, the SAMPA spelling of the name 'Jenee Hughes' is *dZEni hjuz*. 'Dr Zoe Wood' becomes *dAkt@`r zoui wUd*. 'Dr John Clements' becomes *dAkt@`r dZAn klEm@nts*. 'Dr Franz Kurfess' becomes *dAk@`r fr{nz k3`rfEs*.

Chapter 2

Introduction

Human brains are built to come to single conclusions about things that have more than one interpretation. The way that you come to this end conclusion is dependent upon your experiences, cutural immersion, and language familiarity [22]. When attempting to write English phrases that will be read aloud and heard by people with other linguistic biases than you, it's important to make your prose as deterministically understandable as possible. The first step towards this is understanding and identifying how many ways a particular textual phrase be misheard, and why.

2.1 You and me...and Leslie?

In the song *"Groovin' (on a Sunday Afternoon)"*, by the Young Rascals, there's a part in the bridge that many people hear as *"Life would be ecstasy, you an' me an' Leslie"*. In fact, the line is *"Life would be ecstasy, you and me endlessly"*. The confusion lies with the last three syllables of the phrase. The pronunciation of each version, if spoken normally, is as follows:

Orthographic:	and Les- lie	end- less- ly
SAMPA:	@nd “lEs li	“End l@s li

In the song, the singer is doing what many singers are taught to do, to make it easier to sustain the singing of words that end with difficult-to-sing consonants: the unsingable consonant is displaced onto the front of the next word. In this case, the consonant “d” is not singable, so he displaces it onto the next syllable, when he can: “and ME” becomes “an dME”, and “end LESS” becomes “en dLESS”.

Basically, singers are *born* to ignore syllable boundries. So, our singer can effectively think of the sung phrase as:

YOU an dME en dLESS lee

This does not cause confusion for listeners, because they are used to hearing it. This does mean, however, that lyric placement does not provide an accurate barometer to a listener of where a word actually ends.

In addition, the singer is singing fudging his vowels, like singers are taught to do, so “and” and “end” sound almost indistinguishable. So, really, what listeners are hearing is this:

YOU en dME en dLESS lee

Now, the listener’s brain has to take this syllabic gobbledy-gook, and parse it into something useful. They’ve currently got this mess to deal with (represented in SAMPA syllables):

ju *En* ***dmi*** *En* ***dl@s*** *li*

They parse the first part just fine, because the emphases match:

you and **me** *En* ***dl@s*** *li*

But no one says endLESSly. People say ENDlessly. So, the listeners don't recognize it. They have to work with what they have. They already turned one "En d" into an "and", so they do it again:

you and **me** and ***l@s*** *li*

Now, they're just left with LESS lee. And that fits Leslie, a proper noun that fits in context and in emphasis placement. So, the final heard lyric is:

you and **me** and **Les-** lie

The misunderstanding can be traced back to improper emphasis placement. The songwriter probably didn't even think of that, and now he's stuck: a one-hit-wonder with a misunderstood song. We bet that in interview after interview, someone asks him who Leslie is. It's probably very frustrating — especially since he could have just moved the word an eight note later, and it would have been understood perfectly.

That's the sort of situation this program is going to help avoid.

2.2 Why it breaks down

There are two points at which the author's intendeded phrasing can be muddled : First, when the author's orthographic text becomes an orator's spoken (phonetic) interpretation, and second, when the orator's phonetic interpretation

is translated phonetically by an audience into a perceived orthographic phrase. Both of these interpretations must be made succesfully in order for the author's intended meaning to be conveyed.

The phrase "iced ink" undisputedly succeeds in the first translation, but fails on the second. Iced ink can only be pronouced one way, but it can be heard multiple ways–the most notable of which is "I stink", not "iced ink".

The phrase "a nice cold hour" can fail on both parts. First, the orator could have accidentally-capitalized the word Nice in their head, and made it sound like Nice, the city in France. An audience would likely hear this as "niece", and would be confused, at best. Even if the orator pronouces the phrase as the author intended, they audience could hear multiple orthographic phrases in the same phonetic sequence: "a nice cold hour", "an ice cold hour", or even "a nigh scold our".

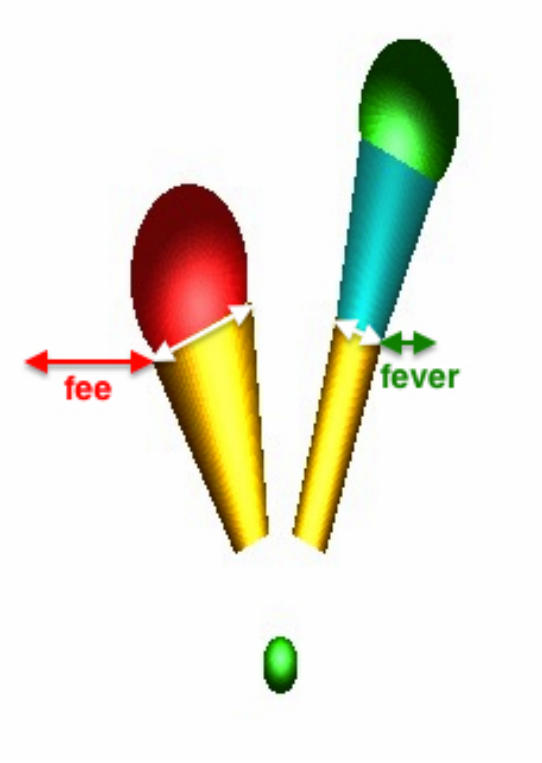
A third, more rare and nefarious type of audience misunderstanding can be caused by parse-tree misdirection, where an audience member is absolutely sure they're hearing one phrase, only to get lost halfway through the lyric because they though they were interpreting a phonetic sequence in a way that resulted in an orthographic dead end.

For example, when asked to sing along with the Adele song, Rolling in the Deep, people who were starting to sing enthusiastically dropped out around the line "reaching a fever pitch"[16]. Let us consider the phrase "fever pitch". This phrase has no exact oronyms, but it does have a potential dead end– a listener could hear the first syllable of the phrase as the word "fee", which has a frequency of 7265. That's more than double the frequency of the word "fever", which is 3095.

Looking at the oronym parse tree for the phrase "fever pitch" in figure 2.1, we can see that the branch for "fever" ends in a much smaller radius than the branch on the left for the word "fee". As you can see by the relative size of the end spheres of the branches, the word "fee" even outweighs the last word in the other branch as well (which is "pitch" with a frequency of 5104). Since the human brain is pre-disposed to parse more-familiar words, having that heavily-weighted dead-end branch is likely the cause of the casual listener not being able to memorize the lyrics.

2.3 Intended audience

- Playwrights
  - use case goes here
- Speechwriters
  - use case goes here
- Jingle writers
  - use case goes here
- Lyricists
  - use case goes here



[h]

Figure 2.1: Annotated Oronym Parse tree generated for the phrase "fever pitch"

Chapter 3

Implementation

This program has three major functional parts: a custom phonetic dictionary, a command-line oronym generator, and a OpenGL oronym-parse-tree visualization generator.

3.1 Customized Phonetic Dictionary

In order to discover oronyms for each phrase, we first needed to determine how each phrase is pronounced. Pronunciation can vary depending on the speakers accent, so it was important for us to (1) chose an accent that we could easily replicate and (2) find a dictionary that supported that accent.

We decided to utilize a General American accent, due to its ubiquity in media and news sources. The General American accent, also known as the "Standard American English" dialect, is not spoken by the majority of people in America, but is used as a sort of "average accent". It most closely resembles the Midwestern accent using in the area in Figure 3.1 and more commonly recognized as "the

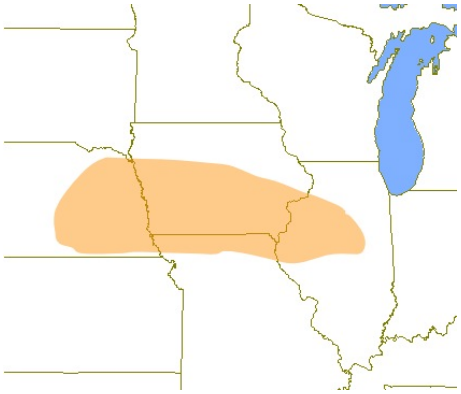


Figure 3.1: This is the geographic area whose accent most closely resembles the General American Accent [6]

newscaster accent". Newscasters learn this accent for national TV, because it is the "least-accented" of the American accents.

3.1.1 Dictionary Options

We considered using three different phonetic dictionaries: the CMU dictionary, LC-STAR dictionary and UNISYN dictionary[10] [2] [15]. We started out by looking at the LC-STAR dictionary, but quickly decided that it wasnt going to be as useful to us, because the LC-Star project is relatively focused on Speech-to-Speech or Text-to-Speech tech. Also, the website had not been maintained since 2006.

We then tried the CMU dictionary, which, for a while, seemed like it was going to work. It had a very simple way of encoding words: first the word, then the identifier number in parens (if needed), then a space, then a one-to-two char code for each sound in the word, with the numbers 0, 1, 2 appended to indicate emphasis (if needed), separated by spaces. An example of a CMU dictionary

```
ABBREVIATE  AH0 B R IY1 V IY0 EY2 T
```

Figure 3.2: Here is the CMU dictionary entry for the word "abbreviate"

entry can be seen in Figure 3.2.

The problem that arose with this format, was that there was no explicit definition of where to hyphenate the word when splitting it up. This causes problems if I want to use the word in lyrics, where each note has is own syllable underneath it, and each syllable might have many different sounds. The benefits of the CMU dictionary over some other dictionaries were that (1) it was actively maintained, (2) it included proper nouns, which are often found in lyrics, but not in dictionaries, and (3) it was ridiculously easy to read.

The downsides were that (1) it included no part of speech data or hyphenation data, and (2) it used non-standard symbols for its phonetic alphabet. With the downsides and benefits in mind, the CMU dictionary could not be used in isolation, especially if we someday want to attempt generation of original lyrics (which part of speech data would be vital for).

The UNISYN dictionary is used primarily to phonetically translate words into multiple accents. It has its own formated dictionary, with a bunch of wildcards in it. They also provide some semi-functioning perl scripts that allow you to specify a dialect youd like to use (For example, a Californian would say cooking differently than someone from the Deep South, and both would say it differently than someone from London. However, they are all speaking English. The UNISYN dictionary facilitates this translation).

It had all the information we needed, and then some. However, it was case-insensitve, meaning that it didn't make it easy to differenciate pronunciations for

some words. For example, the word "nice" is pronounced differently from the city "Nice", but they were both stored as "nice" in the orthography of UNISYN. The CMU dictionary did keep track of capitalization. The obvious conclusion, then, was to grab the capitalizations from the CMU dictionary and put them in the UNISYN dictionary, aligning them by pronunciation and part of speech.

However, we ran into a setback, mentioned in the very first article we found references to both dictionaries in: the dictionaries were inconsistent[20]. They didnt always put stresses in the same place, nor did they always have the same pronunciation. Because of this, it was difficult to match wordsespecially words that were homographic heteronyms (same writing, different sounds, like "Do you know what a buck *does* to *does*?"). Because of this, we decided to use the UNISYN dictionary exclusively.

3.1.2 Formatting the dictionary

We used the UNISYN dictionary, using General American SAMPA output, as the base for our phonetic dictionary.

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

Once we'd extracted them, we attempted to run the provided perl scripts to turn the UNISYN meta-phonetic symbols into SAMPA phonetic symbols, using the procedure specified in Section 6.1 of the UNISYN Documentation, and the towncode gam (which indicated that we want pronunciations using the *G*eneral *AM*erican accent)[18].

There were a few errors in translation we fixed manually:

Example:

```
transfer : 2 : VB/VBP : tr{ns"f3‘r : tr{nsf3‘r : {trans==fer}
: 7184
```

Figure 3.3: Here is an example an entry in our custom phonetic dictionary, using the word "transfer"

- The translation script had erroneously substituted the character *d* instead of the digit *4* , which primarily affected the unique identifiers to disambiguate words and the word-frequency counts. We did a simple search-and-replace to fix this, excluding the orthographic, extended orthographic, and SAMPA spellings.

- The translation script had erroneously substituted the digit *5* instead of the character *l* (lower case L). This interfered with the phonetic/sampa spellings of the words. Fortunately, 5 is not a valid SAMPA phoneme or orthographic letter value, so we removed all instances of it from the ortho, SAMPAspelling, and extendedOrtho columns

- The translation script had erroneously substituted the character sequence *‘r* instead of *‘r* . We were able to fix this by simply removing all \characters from our output.

3.1.3 Custom dictionary fields

Here is the format for the fields in an entry in our custom phonetic dictionary, after we were done with fixing the UNISYN output:

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

<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 speech

<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 theyre 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.

<SAMPAnoEmph> is the same as *¡SAMPASpelling¿*, but with all emphasis characters stripped out. We chose to add this field so that we could more-easily look up phonetic sequence matches.

<extendedOrtho> has no particular use to me, at this time. It indicates the morphological breakdown of the word, allowing for stemming analysis of words. Since we arent convering that, we never use this field.

<freq> is the frequency at which the word occurs in language, according to UNISYN. The frequency count is "taken from a composite of a number of on-line sources of word-frequency. It includes frequencies from the British National Corpus and Maptask, and frequencies derived from Time articles and online texts such as Gutenberg. They were weighted to give more importance to sources of spoken speech, and also to increase the numeric frequency of smaller corpuses"[18].

An example of a entry in our custom phonetic dictionary can be seen in **Figure 3.3** .

3.1.4 Transferring the dictionary to a sqlite database

Because there are several hundred thousand entries in our phonetic dictionary, it was necessary to have a database, rather than store them all in-program in a multi-dimensional array. We decided to use a SQLite database for this purpose.

To turn the colon-delimited dictionary file into a SQLite database, we decided to use a program called the SQLite Database Browser, an open source, public domain, freeware visual tool to create, design, and edit SQLite3.x database files. We specifically used version 2.0b1 of the program, which was built with version 3.6.18 of the SQLite engine[14].

From that point on, it was just a matter of changing the ':' delimiter in our phoneticDictionary.csv to a '|' delimiter, and letting the program convert the pipe-delimited csv into a sqlite database. After letting this process run for a day and a half, it finally spit out a sqlite database file.

3.2 Oronym Generation

3.2.1 Step 1: find all phonemic variations of an orthographic phrase

First, our program takes given an orthographic phrase to find oronyms for.

'a nice cold hour'

We then tokenize this phrase into its component words, using whitespaces as a delimiter.

‘a’, ‘nice’, ‘cold’, ‘hour’

For each word in the phrase, we query our phonetic dictionary for all possible SAMPA pronunciations.

‘a’ → `e`, @, `A`
‘nice’ → `naIs`, `nis`
‘cold’ → `kould`
‘hour’ → `aU`r`

Figure 3.4: In this and all subsequent diagrams, a ‘string in quotes’ indicates an orthographic word or phrase, and a `monospaced string` indicates that it is a SAMPA word or phrase.

Now that we have the pronunciation of each of the words in the form of SAMPA strings, we can list all the possible phonetic permutations of the original phrase.

e naIs kould aU`r
@ naIs kould aU`r
A naIs kould aU`r
e nis kould aU`r
@ nis kould aU`r
A nis kould aU`r

Figure 3.5:

The pseudocode for this process can be reviewed in figure 3.6.

```
vector< vector < phone > >findAllPhoneSeqsForOrthoPhrase( string orthoPhrase) {
  vector< vector < phone > > allFullPhrasePhoneSeqs
  vector<string> orthoWords = split(orthoPhrase);

  int origNumFullPhrases = 0;
  for (  int i = 0 to orthoWords.size() ) {
    string orthoWord = orthoWords[i] ;
    Vector<vector <phone > > nextWordSampaPhoneSeqs (orthoWord)

    if ( this is the first orthoWord weve looked up this loop ) {
      for( vector<phone> phoneSubSeq: nextWordSampaPhoneSeqs ) {
        allFullPhrasePhoneSeqs[i].append(phoneSubSeq);
      }
    } else {
      origNumFullPhrases = allFullPhrasePhoneSeqs.size();
      if theres more than one vector <phone> in nextWordSampaPhoneSeqs
        then we need to create duplicates of all existing allFullPhrasePhoneSeqs
    }
    for( int m = 0 to allPhrasePhoneSeqs.size() ) {
      int phraseToAppendIndex = m / origNumFullPhrases;
      vector<phone> phoneSeqToAppend = nextWordSampaPhoneSeqs[phraseToAppendIndex]
      append phoneSeqToAppend to the end of allFullPhrasePhoneSeqs[m].
    }
  }

  return allFullPhrasePhoneSeqs;
}
```

Figure 3.6: Algorithm to get all phonetic sequences for an orthographic phrase.

3.2.2 Step 2: Finding all Orthographic phrases for a Phonemic Sequence

Then, for each phonemic phrase, we want to figure out all valid orthogrpahic interpretations. For this, we have to go back to our phonetic dictionary.

The ideal way to think about searching for words in a phonetic sequence is by picturing the phoenetic sequence in a tree form. For example, if I had a phonetic tree with the entire dictionary in it, each phonetic tree node would have at least 45 child nodes: one for each phone. A node might also have "word" nodes, if the phones along the path to that node constructs a valid orthographic word:

When there are multiple orthographic interpretations at a single phonetic node, the most likely interpretation can be determined by checking the frequency of use for each word. For example, the sequence "n aI s" is much more likely to be "nice" than "gneiss". In figure 3.7, An visual representation of the process of going through an entire dictionary's phonetic tree for nodes along the paths for the SAMPA sequences 'aIs' and 'nice'.

We can use this dictionary tree method to discover valid orthographic interpretations for each phonetic sequence. Using the dictionary tree method above, we can orthographically interpret each phonetic transcription of our root orthographic phrase, as shown in figure 3.8:

Once we have grabbed all the orthographic interpretations for each phonetic sequence, we combine them all into a orthographic oronym phrase list. This process may leave us with some redundant oronyms, so we de-duplicate that list.

This process gives us a list of all unique and valid oronyms for the original root phrase.

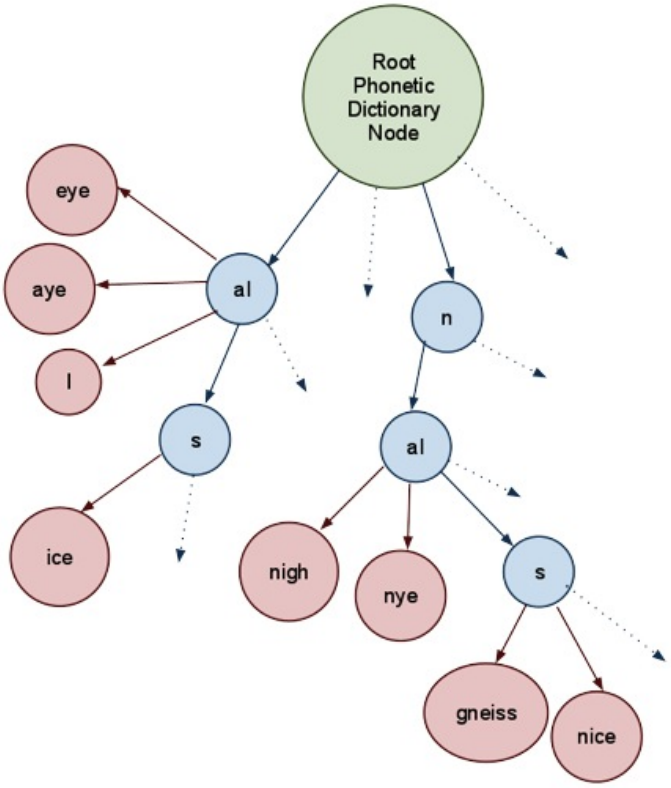


Figure 3.7:

In the case of "a nice cold hour", this returns 290 oronyms, as seen in the first column of figure 3.1.

The pseudocode for this process can be reviewed in figure 3.9

3.2.3 Word Frequency Evaluation

Next, we want to evaluate all our oronyms based on how common each oronym's component words are. For example, "a nice cold hour" is much more likely to be heard "a gneiss cold hour", even though both are phonetically identical.

To do this, we tokenize each oronym phrase into its component words, using once again using non-newline whitespaces as a delimiter.

Then, we query our phonetic dictionary with each word for the word's frequency value. We store each word's value separately. When we have retrieved the frequencies for all the words in a phrase, we then add all the frequencies up to give a combined-frequency of the entire phrase.

You can see these frequency counts for the phrase "a nice cold hour" in figure 3.1.

3.3 Visual Representation

We go about building the visual representation of the oronym parse tree in much the same way that we build the textual list of oronyms, with one important difference: our oronym parse trees may contain oronym fragments. To deal with these we've got to keep track of all our abandoned sub-phrases.

Our algorithm for doing this recursive, called from a parent function that draws the tree's 'seed' sphere. This parent function is documented in figure 3.10

We start in the parent function by getting all the oronyms of our orthographic phrase, using the process in sections 3.2.1 3.2.2. However, instead of ignoring any incomplete orthographic interpretation of a phonetic sequence, as we do in section 3.2.2, we add them to the list of oronyms, keeping track of them by appending 'xxx' or 'fff' to the end of the incomplete oronym string. Then, we tokenize our phrases by whitespace, and look up the frequency of each word, keeping track of only the maximum and minumum values. We will later scale our branches' radiuses using these values.

Once we have all the partial and complete oronyms and the max and min word frequency values for them, we pass them into our recursive function, along

with the radius of the seed sphere. That radius will be the beginning radius of each first-level branch.

Inside our recursive function, we pull the first word out of every orthographic phrases we were passed, and create a set of unique first words.

We then go through this set of unique first words iteratively.

For each word, we look up frequency in the phonetic dictionary. Then, we use the max and min frequencies that we found in our parent function, plus constants for max and min radius size, to scale that frequency into a usable radius size.

Then, we check the contents of the word.

If the word is "xxx" or "fff", then it's not a word at all–just an indication of the dead end of a partial oronym. In this case, we draw a red sphere with the radius of the branch's ancestor, using the parameter past into our recursive function for 'lastRadius'.

If the word is "___SUCCESS!___", that is also not a real word. It indicates that a full oronym has been successfully found, and is terminating at that point. This time, we draw a green sphere using the 'lastRadius' parameter for size.

If the word is neither of these, then it must be a real word. We then draw a cylinder "branch" representing that word. The cylinder's bottom radius is equal to *lastRadius*, and the top radius is equal to the scaled radius that we got from the word's frequency.

After we draw the cylinder, we then go through the full list of phrases, and compile a list of all phrases that start with the word we just drew the cylinder for. Then, we remove the first word from each of those phrases, deduplicating the resulting list of "tail" phrases.

Then, we change our material color (so that different levels of branches will be different colors), and make a recursive call to our current function, passing as parameters the scaled radius and the list of tail phrase.

After this recursive call, we change our color material back to whatever it was before the call, and then continue on to the next unique first word in our set.

Once we have looped through all our unique first words, we know we're done drawing that set of branches, and we return.

This gives us the oronym parse tree seen in figure 3.12. As shown in figure 3.13 (the annotated version of figure 3.12) each branch on the tree represents a single orthographic word.

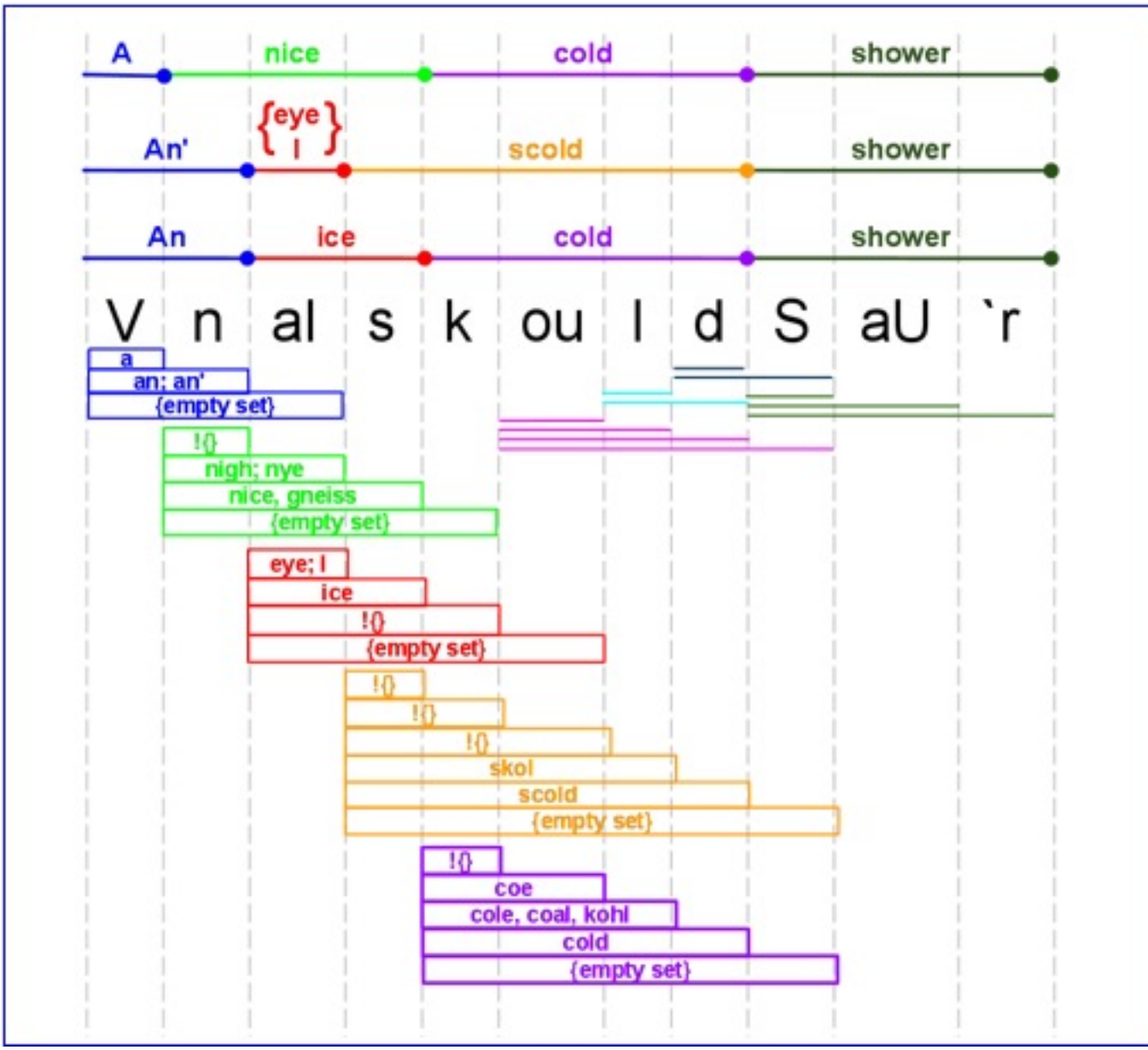

A
nice
cold
shower
{eye
I}
An'
scold
shower
An
ice
cold
shower
V n al s k ou l d S aU `r
a
an; an'
{empty set}
!{}
nigh; nye
nice, gneiss
{empty set}
eye; I
ice
!{}
{empty set}
!{}
!{}
!{}
skol
scold
{empty set}
!{}
coe
cole, coal, kohl
cold
{empty set}

Figure 3.8:

```
vector<string> discoverOronymsForPhrase( string origOrthoPhrase, bool includeDeade
   vector<string> orthoMisheardAsPhrases;
   vector<vector<phone> > allPhoneSeqsOfOrigPhrase = findAllPhoneSeqsForOrthoPhras

   int numUniquePhoneticInterpretations = allPhoneSeqsOfOrigPhrase.size();
   for(int i = 0; i < numUniquePhoneticInterpretations; i++) {
      vector<phone> curPhoneSeqWithEmph( allPhoneSeqsOfOrigPhrase.at(i) );
      string strOfCurPhoneSeq = phoneVectToString( curPhoneSeqWithEmph );

      //remove emphasis marking for easier lookups
      vector<phone> curPhoneSeq = getNoEmphsPhoneVect( curPhoneSeqWithEmph );

      vector<string> altOrthoPhrases = findOrthoStrsForPhoneSeq( curPhoneSeq );

      for( int j = 0; j < altOrthoPhrases.size(); j++) {
         string altOrthoPhrase = altOrthoPhrases.at(j);

         //ensure it contains valid ortho text in all cases, and if includeDeadend
         if ( ( includeDeadends == true && altOrthoPhrase != deadEndDelim1
                  && altOrthoPhrase != deadEndDelim2 )
            || ( altOrthoPhrase.find( deadEndDelim1 ) == string::npos
                  && altOrthoPhrase.find( deadEndDelim2 ) == string::npos ) ) {

            orthoMisheardAsPhrases.push_back( altOrthoPhrase );
         }
      }
   }
   //deduplicate orthoMisheardAsPhrases by putting in a set and back again
   set<string> tempSetForDeduplication( orthoMisheardAsPhrases.begin(), orthoMishe
   orthoMisheardAsPhrases.assign( tempSetForDeduplication.begin(), tempSetForDedup
   return orthoMisheardAsPhrases;
}
```

Figure 3.9: Algorithm to get all oronyms for an orthographic phrase.

```
void buildAndDrawFullTree( string orthoPhrase ) {
   vector< string > fullPhrases = discoverOronymsForPhrase( orthoPhrase , true );
   getMaxAndMinFreqForAllOrthoPhrases( fullPhrases, &maxWordFreq, &minWordFreq);


   //draw the tree's seed

   glPushMatrix();
   {
      glTranslated(0.0, -1.0* DEFAULT_BRANCH_LEN , 0.0);
      materials(GreenShiny);
      drawSphere(DEFAULT_RADIUS);
      materials(allMaterials.at( mat % allMaterials.size () ) );

      drawBranchesAtFork ( fullPhrases, DEFAULT_RADIUS );

   }
   glPopMatrix();
}
```

Figure 3.10: Given an orthographic phrase, this function prepares to draw the tree

```
void drawBranchesAtFork( vector< string > fullPhrases, double lastRadius) {

   if( fullPhrases.size() == 0 ) {
      return;
   }

   //use a set to ensure no duplicates
   set< string > firstWords;

   //put the first word of each phrase into the set
   for(int i = 0; i < fullPhrases.size(); i++){
      if( fullPhrases.at(i).size() > 0 ) {
         string firstWord = FirstWord( fullPhrases.at(i) );
         firstWords.insert( firstWord );
      }
   }

   //calculate positioning variables for the spread of branches for firstWord

   set<string>::iterator curFirstWordIter;
   int i = 0;

   //for each firstWord in the set
   for ( curFirstWordIter = firstWords.begin(); curFirstWordIter != firstWords.end
      string curFirstWord = *curFirstWordIter;

      //calculate the branch radius using the scaled frequency of curFirstWord
      int firstWordFreq = queryDBwithOrthoForFreq ( curFirstWord );
      double firstWordRadius = scaleFreqToRadius( firstWordFreq );

      double newAdditiveRadius = firstWordRadius;

      glPushMatrix();
      {

         //translate and rotate into place


         //if firstWord indicates a dead end ( xxx or fff, defined in wordBreakdow
         if(curFirstWord == deadEndDelim1  || curFirstWord == deadEndDelim2 ) {
            //draw a red sphere at the end of the last branch
         } else if (curFirstWord == successDelim ) {
            //draw a green sphere at the end of the last branch
         } else {

            //draw a branch
            drawBranch( radiansToDegrees( tiltAngle ), curXOffset, curYOffset, new
```

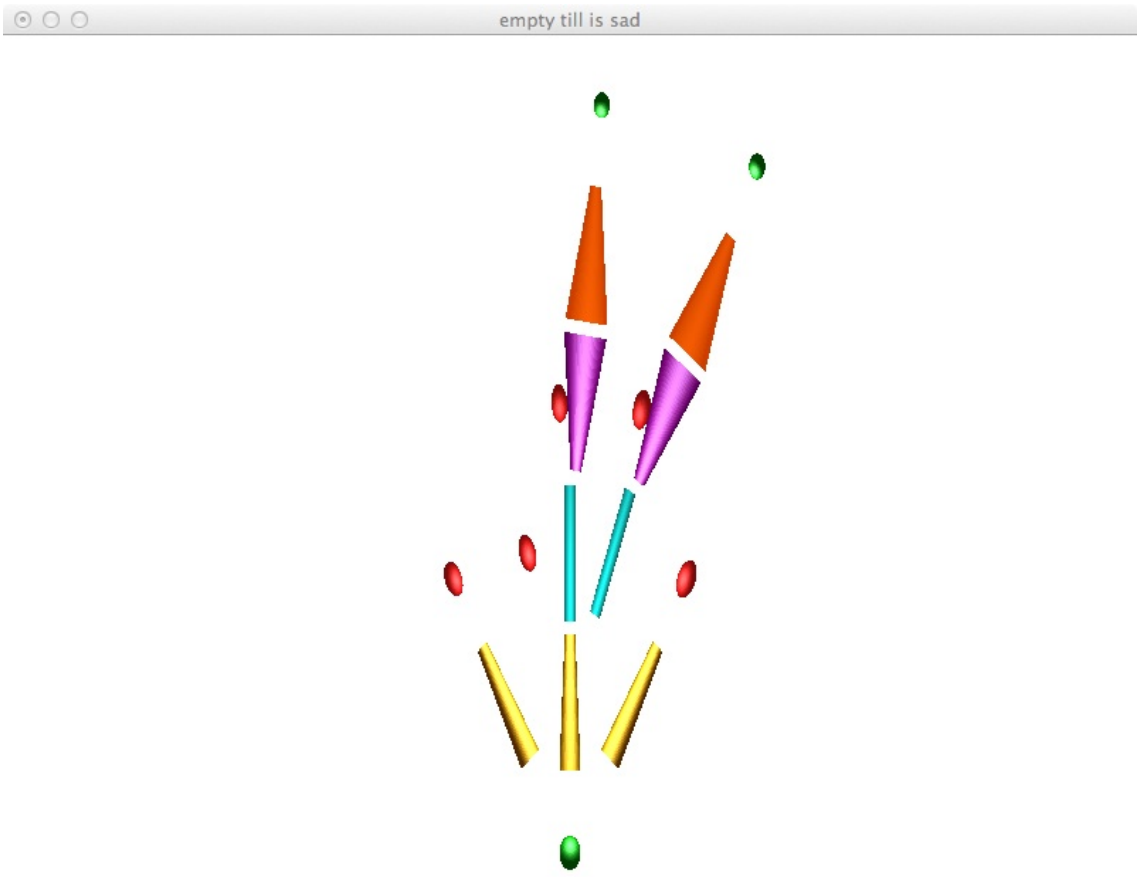


Figure 3.12: This is the parse tree for the phrase "empty till is sad"

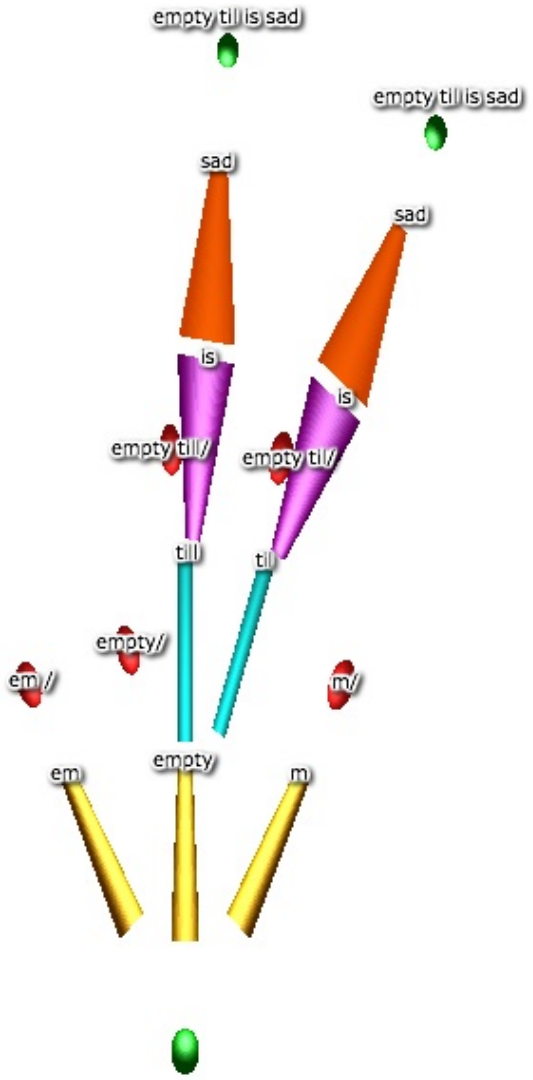


Figure 3.13: This is the annotated parse tree for the phrase "empty till is sad"

Table 3.1: All Oronyms for 'A Nice Cold Hour' with frequency values

phrase	total freq	word1	freq1	word2	freq2	word3	freq3	word4	freq4
on i scold our	13185760	on	2774243	i	9937877	scold	217	our	473423
on i scold hour	12784150	on	2774243	i	9937877	scold	217	hour	71813
on i skol dour	12712244	on	2774243	i	9937877	skol	5	dour	119
on i skol dower	12712217	on	2774243	i	9937877	skol	5	dower	92
an i scold our	11205686	an	794169	i	9937877	scold	217	our	473423
an i scold hour	10804076	an	794169	i	9937877	scold	217	hour	71813
an i skol dour	10732170	an	794169	i	9937877	skol	5	dour	119
an i skol dower	10732143	an	794169	i	9937877	skol	5	dower	92
'n' i scold our	10411517	'n'	0	i	9937877	scold	217	our	473423
'n' i scold hour	10009907	'n'	0	i	9937877	scold	217	hour	71813
'n' i skol dour	9938001	'n'	0	i	9937877	skol	5	dour	119
'n' i skol dower	9937974	'n'	0	i	9937877	skol	5	dower	92
a nice cold our	8253272	a	7536297	nice	190708	cold	52844	our	473423
a niece cold our	8064257	a	7536297	niece	1693	cold	52844	our	473423
a gneiss cold our	8062585	a	7536297	gneiss	21	cold	52844	our	473423
a ne scold our	8017040	a	7536297	ne	7103	scold	217	our	473423
a knee scold our	8016076	a	7536297	knee	6139	scold	217	our	473423
a nigh scold our	8011331	a	7536297	nigh	1394	scold	217	our	473423
a nye scold our	8009974	a	7536297	nye	37	scold	217	our	473423
a nice cold hour	7851662	a	7536297	nice	190708	cold	52844	hour	71813
a nice coal dour	7747572	a	7536297	nice	190708	coal	20448	dour	119
a nice coal dower	7747545	a	7536297	nice	190708	coal	20448	dower	92

Continued on next page

Table 3.1 – continued from previous page

phrase	total freq	word1	freq1	word2	freq2	word3	freq3	word4	freq4
a nice cole dour	7729197	a	7536297	nice	190708	cole	2073	dour	119
a nice cole dower	7729170	a	7536297	nice	190708	cole	2073	dower	92
a nice kohl dour	7728036	a	7536297	nice	190708	kohl	912	dour	119
a nice kohl dower	7728009	a	7536297	nice	190708	kohl	912	dower	92
a niece cold hour	7662647	a	7536297	niece	1693	cold	52844	hour	71813
a gneiss cold hour	7660975	a	7536297	gneiss	21	cold	52844	hour	71813
a ne scold hour	7615430	a	7536297	ne	7103	scold	217	hour	71813
a knee scold hour	7614466	a	7536297	knee	6139	scold	217	hour	71813
a nigh scold hour	7609721	a	7536297	nigh	1394	scold	217	hour	71813
a nye scold hour	7608364	a	7536297	nye	37	scold	217	hour	71813
a niece coal dour	7558557	a	7536297	niece	1693	coal	20448	dour	119
a niece coal dower	7558530	a	7536297	niece	1693	coal	20448	dower	92
a gneiss coal dour	7556885	a	7536297	gneiss	21	coal	20448	dour	119
a gneiss coal dower	7556858	a	7536297	gneiss	21	coal	20448	dower	92
a ne skol dour	7543524	a	7536297	ne	7103	skol	5	dour	119
a ne skol dower	7543497	a	7536297	ne	7103	skol	5	dower	92
a knee skol dour	7542560	a	7536297	knee	6139	skol	5	dour	119
a knee skol dower	7542533	a	7536297	knee	6139	skol	5	dower	92
a niece cole dour	7540182	a	7536297	niece	1693	cole	2073	dour	119
a niece cole dower	7540155	a	7536297	niece	1693	cole	2073	dower	92
a niece kohl dour	7539021	a	7536297	niece	1693	kohl	912	dour	119
a niece kohl dower	7538994	a	7536297	niece	1693	kohl	912	dower	92

Continued on next page

Table 3.1 – continued from previous page

phrase	total freq	word1	freq1	word2	freq2	word3	freq3	word4	freq4
a gneiss cole dour	7538510	a	7536297	gneiss	21	cole	2073	dour	119
a gneiss cole dower	7538483	a	7536297	gneiss	21	cole	2073	dower	92
a nigh skol dour	7537815	a	7536297	nigh	1394	skol	5	dour	119
a nigh skol dower	7537788	a	7536297	nigh	1394	skol	5	dower	92
a gneiss kohl dour	7537349	a	7536297	gneiss	21	kohl	912	dour	119
a gneiss kohl dower	7537322	a	7536297	gneiss	21	kohl	912	dower	92
a nye skol dour	7536458	a	7536297	nye	37	skol	5	dour	119
a nye skol dower	7536431	a	7536297	nye	37	skol	5	dower	92
on aye scold our	3378386	on	2774243	aye	130503	scold	217	our	473423
on e scold our	3356846	on	2774243	e	108963	scold	217	our	473423
on ice cold our	3312712	on	2774243	ice	12202	cold	52844	our	473423
on eye scold our	3274633	on	2774243	eye	26750	scold	217	our	473423
on ay scold our	3254516	on	2774243	ay	6633	scold	217	our	473423
on ice-cold our	3247715	on	2774243	ice-cold	49	our	473423		
on aye scold hour	2976776	on	2774243	aye	130503	scold	217	hour	71813
on e scold hour	2955236	on	2774243	e	108963	scold	217	hour	71813
on ice cold hour	2911102	on	2774243	ice	12202	cold	52844	hour	71813
on aye skol dour	2904870	on	2774243	aye	130503	skol	5	dour	119
on aye skol dower	2904843	on	2774243	aye	130503	skol	5	dower	92
on e skol dour	2883330	on	2774243	e	108963	skol	5	dour	119
on e skol dower	2883303	on	2774243	e	108963	skol	5	dower	92
on eye scold hour	2873023	on	2774243	eye	26750	scold	217	hour	71813

Continued on next page

Table 3.1 – continued from previous page

phrase	total freq	word1	freq1	word2	freq2	word3	freq3	word4	freq4
on ay scold hour	2852906	on	2774243	ay	6633	scold	217	hour	71813
on ice-cold hour	2846105	on	2774243	ice-cold	49	hour	71813		
on ice coal dour	2807012	on	2774243	ice	12202	coal	20448	dour	119
on ice coal dower	2806985	on	2774243	ice	12202	coal	20448	dower	92
on eye skol dour	2801117	on	2774243	eye	26750	skol	5	dour	119
on eye skol dower	2801090	on	2774243	eye	26750	skol	5	dower	92
on ice cole dour	2788637	on	2774243	ice	12202	cole	2073	dour	119
on ice cole dower	2788610	on	2774243	ice	12202	cole	2073	dower	92
on ice kohl dour	2787476	on	2774243	ice	12202	kohl	912	dour	119
on ice kohl dower	2787449	on	2774243	ice	12202	kohl	912	dower	92
on ay skol dour	2781000	on	2774243	ay	6633	skol	5	dour	119
on ay skol dower	2780973	on	2774243	ay	6633	skol	5	dower	92
an aye scold our	1398312	an	794169	aye	130503	scold	217	our	473423
an e scold our	1376772	an	794169	e	108963	scold	217	our	473423
an ice cold our	1332638	an	794169	ice	12202	cold	52844	our	473423
an eye scold our	1294559	an	794169	eye	26750	scold	217	our	473423
an ay scold our	1274442	an	794169	ay	6633	scold	217	our	473423
an ice-cold our	1267641	an	794169	ice-cold	49	our	473423		
an aye scold hour	996702	an	794169	aye	130503	scold	217	hour	71813
an e scold hour	975162	an	794169	e	108963	scold	217	hour	71813
ah nice cold our	946271	ah	229296	nice	190708	cold	52844	our	473423
an ice cold hour	931028	an	794169	ice	12202	cold	52844	hour	71813

Continued on next page

Table 3.1 – continued from previous page

phrase	total freq	word1	freq1	word2	freq2	word3	freq3	word4	freq4
an aye skol dour	924796	an	794169	aye	130503	skol	5	dour	119
an aye skol dower	924769	an	794169	aye	130503	skol	5	dower	92
an e skol dour	903256	an	794169	e	108963	skol	5	dour	119
an e skol dower	903229	an	794169	e	108963	skol	5	dower	92
an eye scold hour	892949	an	794169	eye	26750	scold	217	hour	71813
an ay scold hour	872832	an	794169	ay	6633	scold	217	hour	71813
an ice-cold hour	866031	an	794169	ice-cold	49	hour	71813		
an ice coal dour	826938	an	794169	ice	12202	coal	20448	dour	119
an ice coal dower	826911	an	794169	ice	12202	coal	20448	dower	92
an eye skol dour	821043	an	794169	eye	26750	skol	5	dour	119
an eye skol dower	821016	an	794169	eye	26750	skol	5	dower	92
an ice cole dour	808563	an	794169	ice	12202	cole	2073	dour	119
an ice cole dower	808536	an	794169	ice	12202	cole	2073	dower	92
an ice kohl dour	807402	an	794169	ice	12202	kohl	912	dour	119
an ice kohl dower	807375	an	794169	ice	12202	kohl	912	dower	92
an ay skol dour	800926	an	794169	ay	6633	skol	5	dour	119
an ay skol dower	800899	an	794169	ay	6633	skol	5	dower	92
eh nice cold our	783938	eh	66963	nice	190708	cold	52844	our	473423
ah niece cold our	757256	ah	229296	niece	1693	cold	52844	our	473423
ah gneiss cold our	755584	ah	229296	gneiss	21	cold	52844	our	473423
et nice cold our	723706	et	6731	nice	190708	cold	52844	our	473423
o’ nice cold our	717438	o’	463	nice	190708	cold	52844	our	473423

Continued on next page

Table 3.1 – continued from previous page

phrase	total freq	word1	freq1	word2	freq2	word3	freq3	word4	freq4
ah ne scold our	710039	ah	229296	ne	7103	scold	217	our	473423
ah knee scold our	709075	ah	229296	knee	6139	scold	217	our	473423
ah nigh scold our	704330	ah	229296	nigh	1394	scold	217	our	473423
ah nye scold our	702973	ah	229296	nye	37	scold	217	our	473423
'n' aye scold our	604143	'n'	0	aye	130503	scold	217	our	473423
eh niece cold our	594923	eh	66963	niece	1693	cold	52844	our	473423
eh gneiss cold our	593251	eh	66963	gneiss	21	cold	52844	our	473423
'n' e scold our	582603	'n'	0	e	108963	scold	217	our	473423
eh ne scold our	547706	eh	66963	ne	7103	scold	217	our	473423
eh knee scold our	546742	eh	66963	knee	6139	scold	217	our	473423
ah nice cold hour	544661	ah	229296	nice	190708	cold	52844	hour	71813
eh nigh scold our	541997	eh	66963	nigh	1394	scold	217	our	473423
eh nye scold our	540640	eh	66963	nye	37	scold	217	our	473423
'n' ice cold our	538469	'n'	0	ice	12202	cold	52844	our	473423
et niece cold our	534691	et	6731	niece	1693	cold	52844	our	473423
et gneiss cold our	533019	et	6731	gneiss	21	cold	52844	our	473423
o' niece cold our	528423	o'	463	niece	1693	cold	52844	our	473423
o' gneiss cold our	526751	o'	463	gneiss	21	cold	52844	our	473423
'n' eye scold our	500390	'n'	0	eye	26750	scold	217	our	473423
et ne scold our	487474	et	6731	ne	7103	scold	217	our	473423
et knee scold our	486510	et	6731	knee	6139	scold	217	our	473423
et nigh scold our	481765	et	6731	nigh	1394	scold	217	our	473423

Continued on next page

Table 3.1 – continued from previous page

phrase	total freq	word1	freq1	word2	freq2	word3	freq3	word4	freq4
o' ne scold our	481206	o'	463	ne	7103	scold	217	our	473423
et nye scold our	480408	et	6731	nye	37	scold	217	our	473423
'n' ay scold our	480273	'n'	0	ay	6633	scold	217	our	473423
o' knee scold our	480242	o'	463	knee	6139	scold	217	our	473423
o' nigh scold our	475497	o'	463	nigh	1394	scold	217	our	473423
o' nye scold our	474140	o'	463	nye	37	scold	217	our	473423
'n' ice-cold our	473472	'n'	0	ice-cold	49	our	473423		
ah nice coal dour	440571	ah	229296	nice	190708	coal	20448	dour	119
ah nice coal dower	440544	ah	229296	nice	190708	coal	20448	dower	92
ah nice cole dour	422196	ah	229296	nice	190708	cole	2073	dour	119
ah nice cole dower	422169	ah	229296	nice	190708	cole	2073	dower	92
ah nice kohl dour	421035	ah	229296	nice	190708	kohl	912	dour	119
ah nice kohl dower	421008	ah	229296	nice	190708	kohl	912	dower	92
eh nice cold hour	382328	eh	66963	nice	190708	cold	52844	hour	71813
ah niece cold hour	355646	ah	229296	niece	1693	cold	52844	hour	71813
ah gneiss cold hour	353974	ah	229296	gneiss	21	cold	52844	hour	71813
et nice cold hour	322096	et	6731	nice	190708	cold	52844	hour	71813
o' nice cold hour	315828	o'	463	nice	190708	cold	52844	hour	71813
ah ne scold hour	308429	ah	229296	ne	7103	scold	217	hour	71813
ah knee scold hour	307465	ah	229296	knee	6139	scold	217	hour	71813
ah nigh scold hour	302720	ah	229296	nigh	1394	scold	217	hour	71813
ah nye scold hour	301363	ah	229296	nye	37	scold	217	hour	71813

Continued on next page

Table 3.1 – continued from previous page

phrase	total freq	word1	freq1	word2	freq2	word3	freq3	word4	freq4
eh nice coal dour	278238	eh	66963	nice	190708	coal	20448	dour	119
eh nice coal dower	278211	eh	66963	nice	190708	coal	20448	dower	92
eh nice cole dour	259863	eh	66963	nice	190708	cole	2073	dour	119
eh nice cole dower	259836	eh	66963	nice	190708	cole	2073	dower	92
eh nice kohl dour	258702	eh	66963	nice	190708	kohl	912	dour	119
eh nice kohl dower	258675	eh	66963	nice	190708	kohl	912	dower	92
ah niece coal dour	251556	ah	229296	niece	1693	coal	20448	dour	119
ah niece coal dower	251529	ah	229296	niece	1693	coal	20448	dower	92
ah gneiss coal dour	249884	ah	229296	gneiss	21	coal	20448	dour	119
ah gneiss coal dower	249857	ah	229296	gneiss	21	coal	20448	dower	92
ah ne skol dour	236523	ah	229296	ne	7103	skol	5	dour	119
ah ne skol dower	236496	ah	229296	ne	7103	skol	5	dower	92
ah knee skol dour	235559	ah	229296	knee	6139	skol	5	dour	119
ah knee skol dower	235532	ah	229296	knee	6139	skol	5	dower	92
ah niece cole dour	233181	ah	229296	niece	1693	cole	2073	dour	119
ah niece cole dower	233154	ah	229296	niece	1693	cole	2073	dower	92
ah niece kohl dour	232020	ah	229296	niece	1693	kohl	912	dour	119
ah niece kohl dower	231993	ah	229296	niece	1693	kohl	912	dower	92
ah gneiss cole dour	231509	ah	229296	gneiss	21	cole	2073	dour	119
ah gneiss cole dower	231482	ah	229296	gneiss	21	cole	2073	dower	92
ah nigh skol dour	230814	ah	229296	nigh	1394	skol	5	dour	119
ah nigh skol dower	230787	ah	229296	nigh	1394	skol	5	dower	92

Continued on next page

Table 3.1 – continued from previous page

phrase	total freq	word1	freq1	word2	freq2	word3	freq3	word4	freq4
ah gneiss kohl dour	230348	ah	229296	gneiss	21	kohl	912	dour	119
ah gneiss kohl dower	230321	ah	229296	gneiss	21	kohl	912	dower	92
ah nye skol dour	229457	ah	229296	nye	37	skol	5	dour	119
ah nye skol dower	229430	ah	229296	nye	37	skol	5	dower	92
et nice coal dour	218006	et	6731	nice	190708	coal	20448	dour	119
et nice coal dower	217979	et	6731	nice	190708	coal	20448	dower	92
o’ nice coal dour	211738	o’	463	nice	190708	coal	20448	dour	119
o’ nice coal dower	211711	o’	463	nice	190708	coal	20448	dower	92
’n’ aye scold hour	202533	’n’	0	aye	130503	scold	217	hour	71813
et nice cole dour	199631	et	6731	nice	190708	cole	2073	dour	119
et nice cole dower	199604	et	6731	nice	190708	cole	2073	dower	92
et nice kohl dour	198470	et	6731	nice	190708	kohl	912	dour	119
et nice kohl dower	198443	et	6731	nice	190708	kohl	912	dower	92
o’ nice cole dour	193363	o’	463	nice	190708	cole	2073	dour	119
o’ nice cole dower	193336	o’	463	nice	190708	cole	2073	dower	92
eh niece cold hour	193313	eh	66963	niece	1693	cold	52844	hour	71813
o’ nice kohl dour	192202	o’	463	nice	190708	kohl	912	dour	119
o’ nice kohl dower	192175	o’	463	nice	190708	kohl	912	dower	92
eh gneiss cold hour	191641	eh	66963	gneiss	21	cold	52844	hour	71813
’n’ e scold hour	180993	’n’	0	e	108963	scold	217	hour	71813
eh ne scold hour	146096	eh	66963	ne	7103	scold	217	hour	71813
eh knee scold hour	145132	eh	66963	knee	6139	scold	217	hour	71813

Continued on next page

Table 3.1 – continued from previous page

phrase	total freq	word1	freq1	word2	freq2	word3	freq3	word4	freq4
eh nigh scold hour	140387	eh	66963	nigh	1394	scold	217	hour	71813
eh nye scold hour	139030	eh	66963	nye	37	scold	217	hour	71813
'n' ice cold hour	136859	'n'	0	ice	12202	cold	52844	hour	71813
et niece cold hour	133081	et	6731	niece	1693	cold	52844	hour	71813
et gneiss cold hour	131409	et	6731	gneiss	21	cold	52844	hour	71813
'n' aye skol dour	130627	'n'	0	aye	130503	skol	5	dour	119
'n' aye skol dower	130600	'n'	0	aye	130503	skol	5	dower	92
o' niece cold hour	126813	o'	463	niece	1693	cold	52844	hour	71813
o' gneiss cold hour	125141	o'	463	gneiss	21	cold	52844	hour	71813
'n' e skol dour	109087	'n'	0	e	108963	skol	5	dour	119
'n' e skol dower	109060	'n'	0	e	108963	skol	5	dower	92
'n' eye scold hour	98780	'n'	0	eye	26750	scold	217	hour	71813
eh niece coal dour	89223	eh	66963	niece	1693	coal	20448	dour	119
eh niece coal dower	89196	eh	66963	niece	1693	coal	20448	dower	92
eh gneiss coal dour	87551	eh	66963	gneiss	21	coal	20448	dour	119
eh gneiss coal dower	87524	eh	66963	gneiss	21	coal	20448	dower	92
et ne scold hour	85864	et	6731	ne	7103	scold	217	hour	71813
et knee scold hour	84900	et	6731	knee	6139	scold	217	hour	71813
et nigh scold hour	80155	et	6731	nigh	1394	scold	217	hour	71813
o' ne scold hour	79596	o'	463	ne	7103	scold	217	hour	71813
et nye scold hour	78798	et	6731	nye	37	scold	217	hour	71813
'n' ay scold hour	78663	'n'	0	ay	6633	scold	217	hour	71813

Continued on next page

Table 3.1 – continued from previous page

phrase	total freq	word1	freq1	word2	freq2	word3	freq3	word4	freq4
o' knee scold hour	78632	o'	463	knee	6139	scold	217	hour	71813
eh ne skol dour	74190	eh	66963	ne	7103	skol	5	dour	119
eh ne skol dower	74163	eh	66963	ne	7103	skol	5	dower	92
o' nigh scold hour	73887	o'	463	nigh	1394	scold	217	hour	71813
eh knee skol dour	73226	eh	66963	knee	6139	skol	5	dour	119
eh knee skol dower	73199	eh	66963	knee	6139	skol	5	dower	92
o' nye scold hour	72530	o'	463	nye	37	scold	217	hour	71813
'n' ice-cold hour	71862	'n'	0	ice-cold	49	hour	71813		
eh niece cole dour	70848	eh	66963	niece	1693	cole	2073	dour	119
eh niece cole dower	70821	eh	66963	niece	1693	cole	2073	dower	92
eh niece kohl dour	69687	eh	66963	niece	1693	kohl	912	dour	119
eh niece kohl dower	69660	eh	66963	niece	1693	kohl	912	dower	92
eh gneiss cole dour	69176	eh	66963	gneiss	21	cole	2073	dour	119
eh gneiss cole dower	69149	eh	66963	gneiss	21	cole	2073	dower	92
eh nigh skol dour	68481	eh	66963	nigh	1394	skol	5	dour	119
eh nigh skol dower	68454	eh	66963	nigh	1394	skol	5	dower	92
eh gneiss kohl dour	68015	eh	66963	gneiss	21	kohl	912	dour	119
eh gneiss kohl dower	67988	eh	66963	gneiss	21	kohl	912	dower	92
eh nye skol dour	67124	eh	66963	nye	37	skol	5	dour	119
eh nye skol dower	67097	eh	66963	nye	37	skol	5	dower	92
'n' ice coal dour	32769	'n'	0	ice	12202	coal	20448	dour	119
'n' ice coal dower	32742	'n'	0	ice	12202	coal	20448	dower	92

Continued on next page

Table 3.1 – continued from previous page

phrase	total freq	word1	freq1	word2	freq2	word3	freq3	word4	freq4
et niece coal dour	28991	et	6731	niece	1693	coal	20448	dour	119
et niece coal dower	28964	et	6731	niece	1693	coal	20448	dower	92
et gneiss coal dour	27319	et	6731	gneiss	21	coal	20448	dour	119
et gneiss coal dower	27292	et	6731	gneiss	21	coal	20448	dower	92
'n' eye skol dour	26874	'n'	0	eye	26750	skol	5	dour	119
'n' eye skol dower	26847	'n'	0	eye	26750	skol	5	dower	92
o' niece coal dour	22723	o'	463	niece	1693	coal	20448	dour	119
o' niece coal dower	22696	o'	463	niece	1693	coal	20448	dower	92
o' gneiss coal dour	21051	o'	463	gneiss	21	coal	20448	dour	119
o' gneiss coal dower	21024	o'	463	gneiss	21	coal	20448	dower	92
'n' ice cole dour	14394	'n'	0	ice	12202	cole	2073	dour	119
'n' ice cole dower	14367	'n'	0	ice	12202	cole	2073	dower	92
et ne skol dour	13958	et	6731	ne	7103	skol	5	dour	119
et ne skol dower	13931	et	6731	ne	7103	skol	5	dower	92
'n' ice kohl dour	13233	'n'	0	ice	12202	kohl	912	dour	119
'n' ice kohl dower	13206	'n'	0	ice	12202	kohl	912	dower	92
et knee skol dour	12994	et	6731	knee	6139	skol	5	dour	119
et knee skol dower	12967	et	6731	knee	6139	skol	5	dower	92
et niece cole dour	10616	et	6731	niece	1693	cole	2073	dour	119
et niece cole dower	10589	et	6731	niece	1693	cole	2073	dower	92
et niece kohl dour	9455	et	6731	niece	1693	kohl	912	dour	119
et niece kohl dower	9428	et	6731	niece	1693	kohl	912	dower	92

Continued on next page

Table 3.1 – continued from previous page

phrase	total freq	word1	freq1	word2	freq2	word3	freq3	word4	freq4
et gneiss cole dour	8944	et	6731	gneiss	21	cole	2073	dour	119
et gneiss cole dower	8917	et	6731	gneiss	21	cole	2073	dower	92
et nigh skol dour	8249	et	6731	nigh	1394	skol	5	dour	119
et nigh skol dower	8222	et	6731	nigh	1394	skol	5	dower	92
et gneiss kohl dour	7783	et	6731	gneiss	21	kohl	912	dour	119
et gneiss kohl dower	7756	et	6731	gneiss	21	kohl	912	dower	92
o’ ne skol dour	7690	o’	463	ne	7103	skol	5	dour	119
o’ ne skol dower	7663	o’	463	ne	7103	skol	5	dower	92
et nye skol dour	6892	et	6731	nye	37	skol	5	dour	119
et nye skol dower	6865	et	6731	nye	37	skol	5	dower	92
’n’ ay skol dour	6757	’n’	0	ay	6633	skol	5	dour	119
’n’ ay skol dower	6730	’n’	0	ay	6633	skol	5	dower	92
o’ knee skol dour	6726	o’	463	knee	6139	skol	5	dour	119
o’ knee skol dower	6699	o’	463	knee	6139	skol	5	dower	92
o’ niece cole dour	4348	o’	463	niece	1693	cole	2073	dour	119
o’ niece cole dower	4321	o’	463	niece	1693	cole	2073	dower	92
o’ niece kohl dour	3187	o’	463	niece	1693	kohl	912	dour	119
o’ niece kohl dower	3160	o’	463	niece	1693	kohl	912	dower	92
o’ gneiss cole dour	2676	o’	463	gneiss	21	cole	2073	dour	119
o’ gneiss cole dower	2649	o’	463	gneiss	21	cole	2073	dower	92
o’ nigh skol dour	1981	o’	463	nigh	1394	skol	5	dour	119
o’ nigh skol dower	1954	o’	463	nigh	1394	skol	5	dower	92

Continued on next page

Table 3.1 – continued from previous page

phrase	total freq	word1	freq1	word2	freq2	word3	freq3	word4	freq4
o' gneiss kohl dour	1515	o'	463	gneiss	21	kohl	912	dour	119
o' gneiss kohl dower	1488	o'	463	gneiss	21	kohl	912	dower	92
o' nye skol dour	624	o'	463	nye	37	skol	5	dour	119
o' nye skol dower	597	o'	463	nye	37	skol	5	dower	92

Chapter 4

User Study

4.1 Structure

We created a multi-wave user study to examine the effectiveness of different parts of our program.

In the first phase, we had 12 people record over 65 different phrases, to see how they pronouced them. This phase served two purposes: one, to gather recordings for the second phase, and two, to see if our phonemic transcriptions were valid.

In the second phase, we took 15 recordings of oronyms from phase one, and gathered 30 to 60 transcriptions for each recording, resulting in a total of 851 transcriptions. These transcriptions were provided by 208 unique users (127 from the United States). We then compared the transcriptions of the recorded oronym phrases to the calculated oronyms for the original root phrase.

In the third and last phase, we compared the frequency of the

4.2 User Sampling Population

We drew our test subjects from a pool of Amazon Mechanical Turk workers (hired for $ 0.02 to $ 0.10 per task) and, for part of phase 1, volunteers from Reddit.com [4] [5].

Amazon Mechanical Turk is an online crowdsourcing service where requesters can hire workers to complete Human Intelligence Tasks, or HITs. The efficacy of using mechanical turk for user studies has been widely studied in academia, and specifically proven in the linguistic community[**?**].

4.3 Methodology

Get ready for figures-without-explanation bombardment. I'm getting there, I promise.

4.3.1 First Phase: Recitation

In this wave of the user study, we used a combination of 12 Mechanical Turk workers (hired for $ 0.10 per task) to record 65 different phrases. These phrases were oronyms of one of two phrases: phrase A, "a nice cold hour" or phrase B, " fourth wry to". To keep track of the phrases, we assigned each phrase an phraseID, built off of the phrase letter, phrase length, and phrase text. We gave Mechanical Turk workers three minutes to record each phrase and email it to us with the phrase identifier in the subject of the email. The number of recordings, along with their identifiers, can be seen in table 4.1.

Table 4.1: Here are the phrases we recorded, how many times they were recorded, and the identifiers we used for each phrase

orthoPhrase	numRecordings	phraseID
a nice cold our	3	A.17.51 a nice cold our
an ice cold our	2	A.17.135 an ice cold our
a nye scold our	2	A.17.69 a nye scold our
ah nye scold our	2	A.18.109 ah nye scold our
an eye scold our	2	A.18.125 an eye scold our
on aye scold our	2	A.18.267 on aye scold our
a nigh scold our	2	A.18.65 a nigh scold our
a nye skol dower	2	A.18.71 a nye skol dower
an aye skol dower	2	A.19.119 an aye skol dower
an eye skol dower	2	A.19.127 an eye skol dower
an ice coal dower	2	A.19.133 an ice coal dower
eh nice coal dower	2	A.20.159 eh nice coal dower
ah nice coal dower	2	A.20.89 ah nice coal dower
fourth wry to	2	B.15.19 fourth wry to
fourth wry too	2	B.16.20 fourth wry too
forth right ooh	2	B.17.1 forth right ooh
fourth rite ooh	2	B.17.13 fourth rite ooh
forth wright ooh	2	B.18.6 forth wright ooh
on i scold our	1	A.16.279 on i scold our
an i scold hour	1	A.17.128 an i scold hour
an i skol dower	1	A.17.131 an i skol dower
an ice-cold our	1	A.17.141 an ice-cold our
on i scold hour	1	A.17.278 on i scold hour
on i skol dower	1	A.17.281 on i skol dower
an aye scold our	1	A.18.117 an aye scold our

We then transcribed the phonetics of each of the recording in SAMPA by ear. In a stunning example of a use case for our project, we discovered that we had unintentionally included some phrases for recordings were not deterministically phonetically parsible, meaning that our oronyms had multiple pronunciations, not all of which mapped back to the original phrase. For example, the orthographic word “a” can be interpreted as the phoneme ‘A’, and that ‘A’ phoneme can be combined with the subsequent ‘n’ phoneme from the word “nice” to create the SAMPA sequence ‘An’. That being said, this fit with our model, and we found no unexpected anomalies when comparing our transcriptions to the expected SAMPA spellings of each phrase.

4.3.2 Second Wave: Transcription

4.3.3 Recording Sample Pool

We had originally intended to use all the phase one recordings in phase two, but eventually had to discard all but 15 of the recordings for various reasons, the most common being that the recording was too loud and we wanted to spare our user’s ears, or the person recording left excessive amounts of space between words that overly-segmented the phrase. The recordings for the ”fourth rye to” oronyms were all unusable for phase two, because our users tended to insert exclamation points any time they said ”ooh” or ”too”, overloading their microphones or over-segmenting the phrase.

All 15 recordings we used were oronyms for the phrase ”a nice cold hour”, and were recorded by one man with remarkably smooth diction from the midwest,

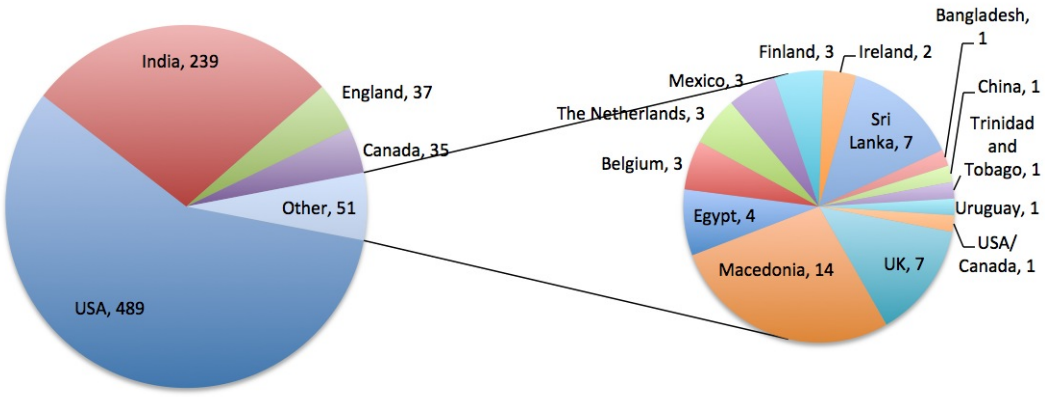


Figure 4.1: Our user study primarily polled people from the United States and India, as can be seen by the number of responses originating from each country.

which made him the best approximation we could get for a General American accent.

4.3.4 Transcription of recordings

We hired 208 unique Mechanical Turk workers to transcribe our oronym recordings for $ 0.02 to $ 0.03 per transcription. Each of the 15 recordings was transcribed 30 to 60 times, resulting in a total of 851 transcriptions. These transcriptions were provided by 208 unique users (127 from the United States). In addition to transcribing the recording, in each task, the worker was asked what country they were from. We did this to help differentiate native American English speakers from non-native speakers.

Response By Country	Num Responses
USA	489
India	239
England	37
Canada	35
UK	7
Macedonia	14
Egypt	4
Belgium	3
The Netherlands	3
Mexico	3
Finland	3
Ireland	2
Sri Lanka	7
Bangladesh	1
China	1
Trinidad and Tobago	1
Uruguay	1
USA/Canada	1

Table 4.2: Here's a table with the number of responses per country

Chapter 5

Results

5.0.5 Transcription oronym phrase frequency vs calculated frequency

Though the most commonly transcribed phrases were found by our oronym generation, figure 5.3 shows an unexpected distribution of the number of times each phrase was recorded versus the frequency metric that we calculated. We hypothesized that a simple summation of the UNISYN-provided word frequency for each word in a phrase would give a semi-meaningful indicator of whether a phrase's likelyhood to be heard.

PUT NGRAMS DIAGRAM HERE: http://books.google.com/ngrams/graph?content=nice+colo

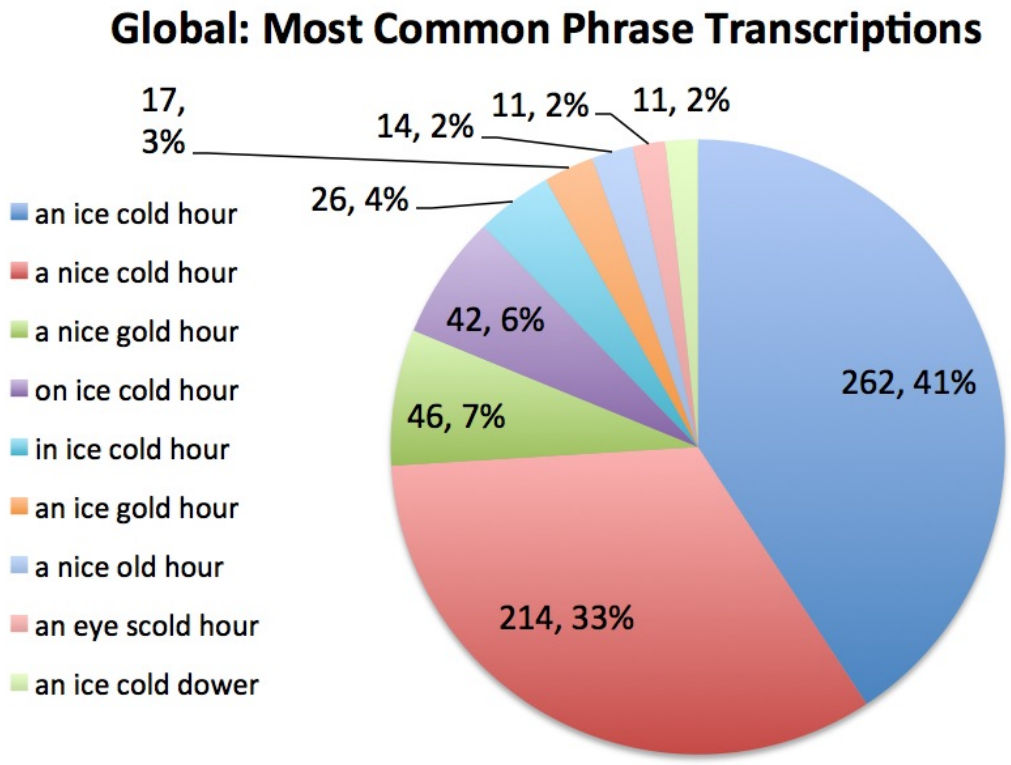


Figure 5.1: Our top two transcriptions were "a nice cold hour" and "an ice cold hour"

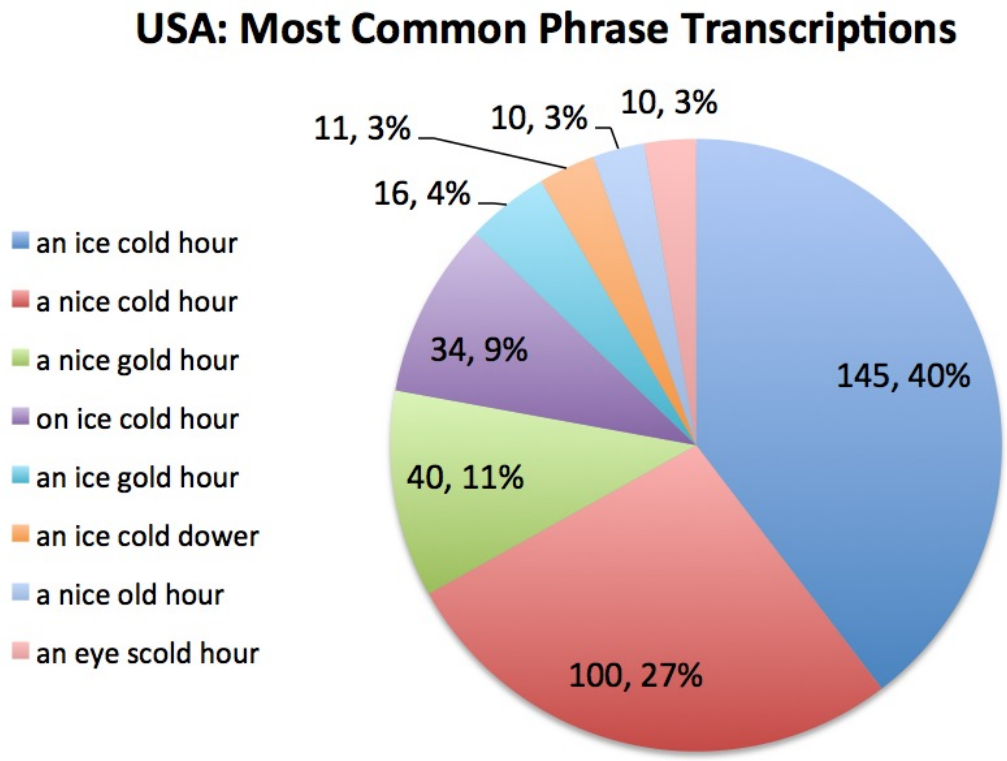
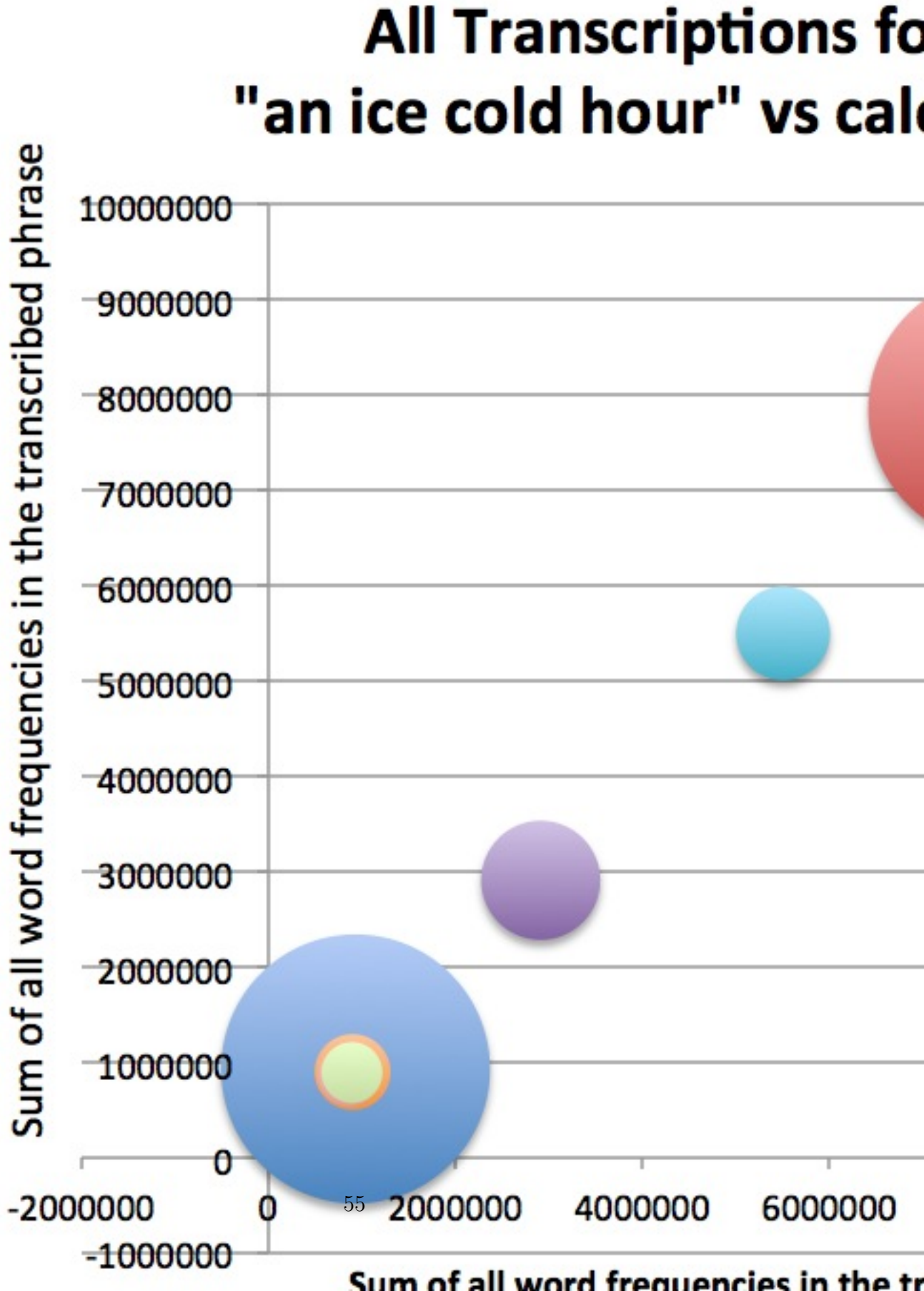


Figure 5.2: Though the breakdown is a bit different than the global transcription breakdown, you can still see the clear trend of "a nice cold hour" and "an ice cold hour" being the most common. There is a slightly larger gap between these two phrase, we hypothesize, because the American transcribers are familiar with what words normally are in proximity to others.


All Transcriptions fo
"an ice cold hour" vs cal
Sum of all word frequencies in the transcribed phrase
10000000
9000000
8000000
7000000
6000000
5000000
4000000
3000000
2000000
1000000
0
-1000000
-2000000
0
2000000
4000000
6000000
Sum of all word frequencies in the tr

freq	phrase transcribed	total answers
931028	an ice cold hour	262
7851662	a nice cold hour	214
7820004	a nice gold hour	46
2911102	on ice cold hour	42
5503158	in ice cold hour	26
899370	an ice gold hour	17
8013781	a nice old hour	14
892949	an ice cold dower	11
859307	an eye scold hour	11

Table 5.2: You can see, our data isn't too horrible

5.0.6 Individual Recording/Transcription Breakdowns

A recording-by-recording transcription breakdown reveals more interesting data. Each bubble charts represent the most common transcriptions for the recorded phrase listed at the top. The size of the bubble and the position along the x axis is indicative of the number of times that phrase was transcribed for this particular recording. The position along the y axis shows how often this phrase was transcribed over all the recordings.

Note that, for the purposes of clarifying these graphs, we did not chart any anomalous transcriptions–that is, transcriptions with only one occurance in this recording, or transcriptions with less than 5 percent of the recording's transcriptions with no other occurances over the entire set of recordings. Doing so did not give us useful visual data, because the bubbles stacked and obscured eachother. A

full table of the transcriptions per recorded phrase can be found in the appendix

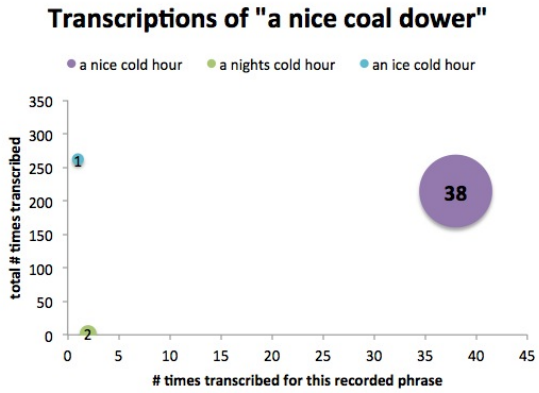


Figure 5.4:

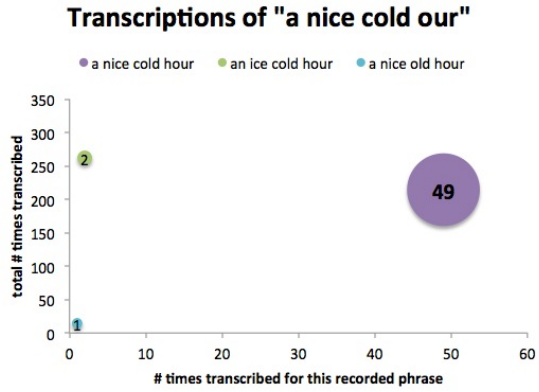


Figure 5.5:

This phonetic sequence deterministically parses to the word “on”. Unsurprisingly, in all recording with the word “on”, it was nearly always heard as “on”

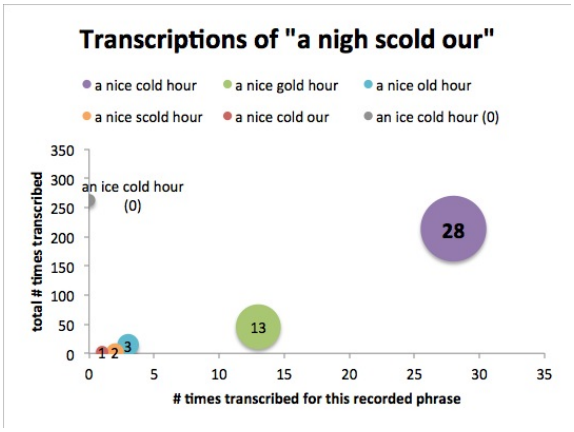


Figure 5.6:

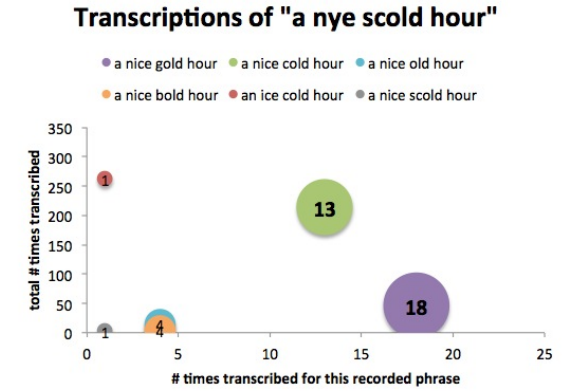


Figure 5.7:

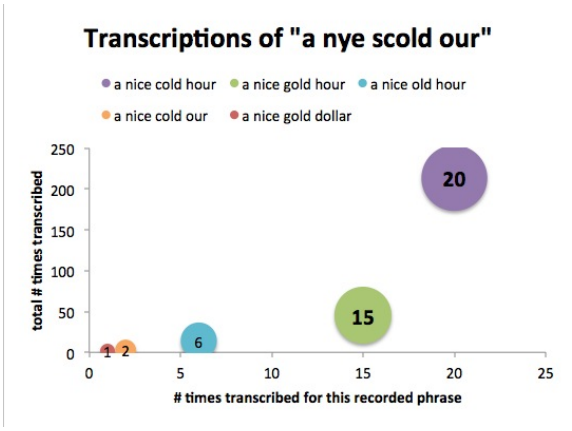


Figure 5.8:

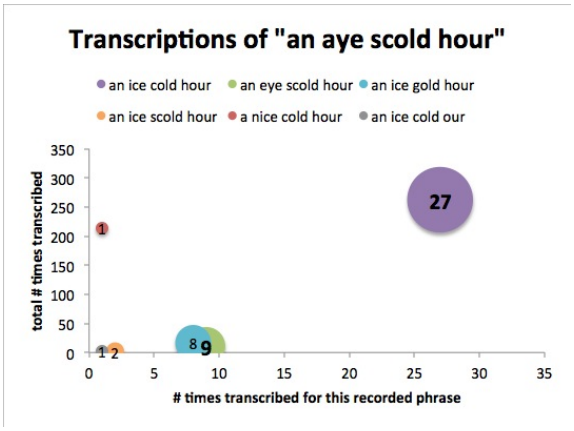


Figure 5.9:

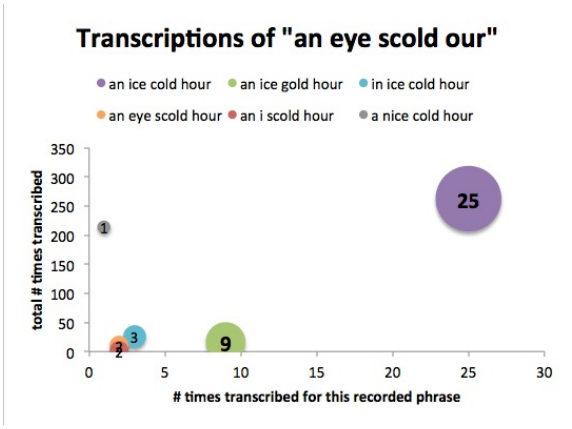


Figure 5.10:

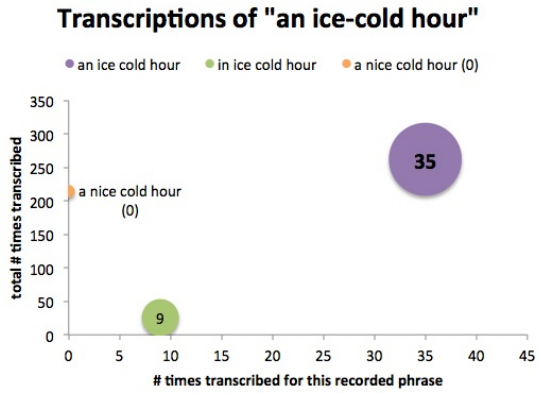


Figure 5.11:

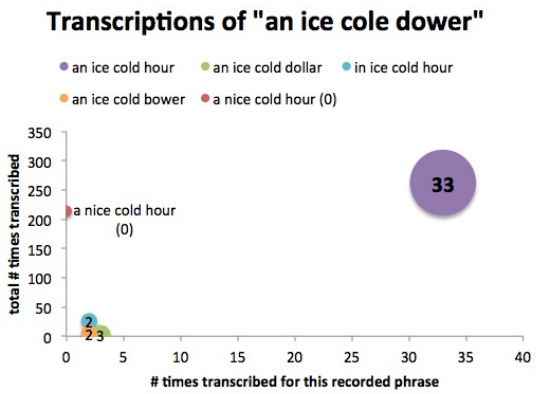


Figure 5.12:

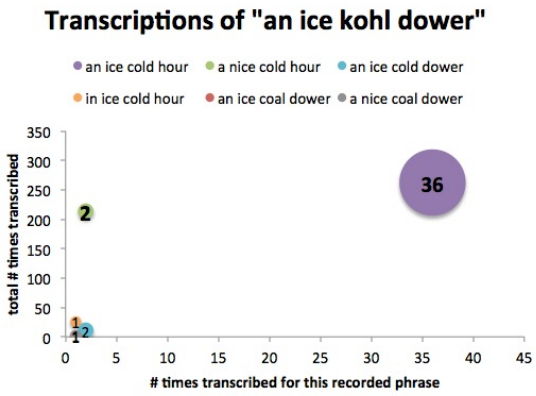


Figure 5.13:

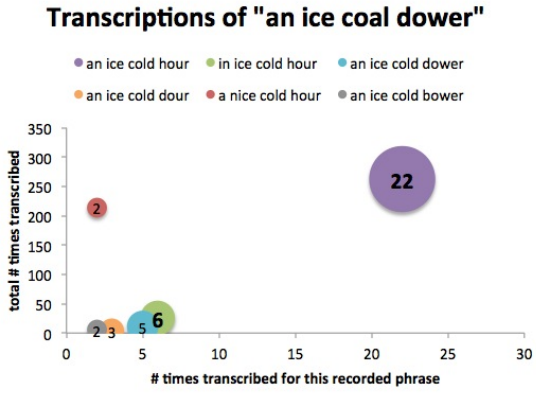


Figure 5.14:

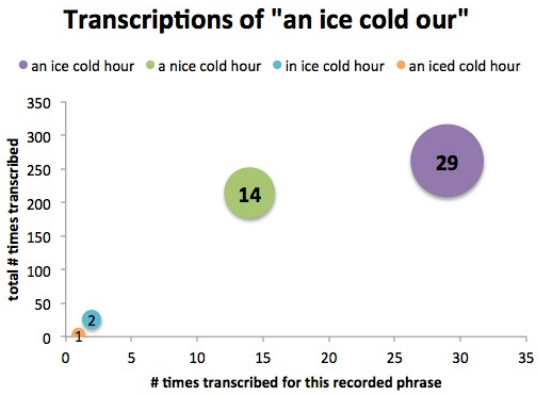


Figure 5.15:

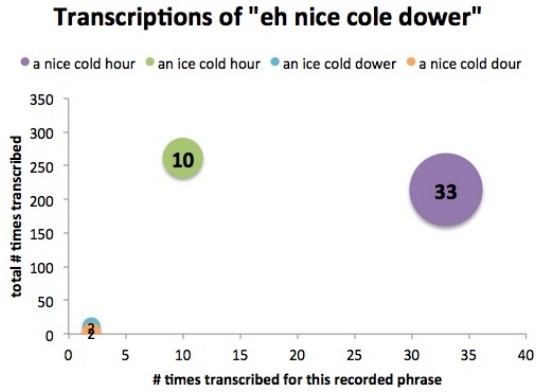


Figure 5.16:

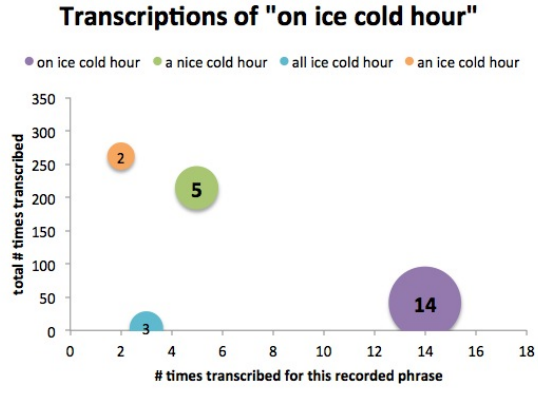


Figure 5.17:

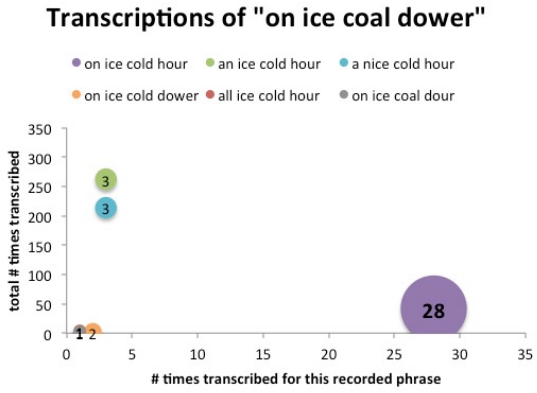

Transcriptions of "on ice coal dower"
on ice cold hour
an ice cold hour
a nice cold hour
on ice cold dower
all ice cold hour
on ice coal dour
total # times transcribed
times transcribed for this recorded phrase
0
50
100
150
200
250
300
350
0
5
10
15
20
25
30
35
3
3
28
1
2

Figure 5.18:

Chapter 6

Future Work

Though we are happy with our findings, we believe that we could create even better likelihood metrics with the integration of n-grams, and would suggest this for future work.

this section not yet edited or read through for consistency. It is likely to change and grow quite drastically and without notice

6.1 Target Audience and Goals

This program is to be used as a compositional aid by anyone who wants to write songs and make them sound good, technically. It should allow the song writer to focus on more subjective criteria of what makes a song "good", because it will make the structural rules of lyric composition immediately apparent.

Our hope for this project is that it will be useful to burgeoning songwriters, who have the creative spark to make wonderfully poetic lyrics, but lack the "ear" to match their lyrics successfully to music. It should be particularly helpful to

songwriters who place a high emphasis on understandability of lyrics (such as parody song writers, or lyricists for musical theater).

Additionally, Melody Matcher will be useful for songwriters for whom English is a second language. While they may be a master lyricist in their native language, writing lyrics in English can be a particular challenge, since so much of lyric-writing is dependent upon knowing the cadence of the language you're writing lyrics in, and since English has no easily-discernible rules for emphasis placement in words.

6.2 Melody Matcher Alpha

This thesis is a part of the Melody Matcher suite. Melody Matcher is a semi-automated music composition support program. It analyzes English lyrics along with a melody, and alerts the composer of the locations in the song where the lyrics are not deterministically understandable. Basically, it's grammar- and spell-check for songs. This is significant, because very little research has been done specifically on the quantifiable measurement of English-language lyric intelligibility, other than our project.

Melody Matcher aims to replicate the human ability to identify lyrics in a song that are easily misheard. We started on this project, thinking that there would be carefully-specified research on how lyrics match melodies, mathematically. As it turned out, there was very little objective literature on the subject. Because of the lack of objective information of the subject, we had to develop our method from scratch. As we progressed through our work, we went from thinking that understandability depended only on emphasis-matching, to realizing that syllable

length played a huge part as well, to realizing that there are many other musical, harmonic, and linguistic factors.

Melody Matcher analyzes the intelligibility of song lyrics by investigating several root causes:

- Lyric/Music emphasis mismatch, due to:
  - Note intervals
  - Phrase emphases
  - Word emphases
- Word "cramming", due to:
  - Syllable lengths that exceed that of note length
  - Mouth movement delta time intervals
- Word misidentification, due to:
  - Altered pronunciation of words
  - Phone similarity
    - Voicing (voiced vs. voiceless)
    - Beginning/end mouth positions
    - Type (Plosive, Fricative, affricate, nasal, lateral, approximant, semivowel)
  - Phone sequences with multiple syntactically-correct interpretations

The fully-implemented Melody Matcher program will eventually take into account all of these causes of unintelligibility. In this abstract, we will focus on lyric/emphasis mismatch, which has already been implemented and is fully functional in primary testing. The other sections have been implemented, but are not fully tested and/or integrated into the main program.

This is where the abstract goes. Hopefully this document will serve as an example for preparing a Cal Poly Master's thesis. It was thrown together pretty quickly. A lot more neat LaTeX features, help, and examples can be found on the web. Here is one: http://en.wikibooks.org/wiki/LaTeX

For developing LaTeX documents in the Windows environment, I use TeXnic-Center (http://www.toolscenter.org/). A simple WYSIWYG LaTeX editor (though I had problems getting it to work with this thesis format) is **LyX** (http://www.lyx.org/).

I use **InkScape** (http://www.inkscape.org/) to create any drawings/figures needed. It is a free vector graphics editor that is very powerful and popular. There is an example figure produced with InkScape in Figure **??**. InkScape can export images in many different formats. Export your images as PDF or EPS and put into your LaTeX document. If you're creating a PDF document with **pdflatex**, then export as a PDF image. If you're creating PostScript then export as EPS. Rasterized images such as JPEG can also be easily included in LaTeX.

LaTeX can also produce nice equations. Did you know that $\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4}$? A non-inline equation can be found in Figure **??**. I treated my equations as figures but they can be treated specially as Equations.

An example of a table can be found in Table **??**.

The bibliography section is very easy to create. When gathering references, I used the ACM digital library (http://portal.acm.org/portal.cfm) to grab the

Bibtex entries. Papers in the digital library have Bibtex entries ready to be copied and pasted into your bibliography. Create a separate file called something like "bibliography.bib" and paste in your Bibtex entries. LaTeX (and Bibtex) generate your bibliography section for you – very easy! I can cite references very easily. Here is a paper called *Dual contouring of hermite data* [**?**]. Here is a paper called *Surface simplification using quadric error metrics* [**?**]. I've also cited software located at some websites [**?**] [**?**].

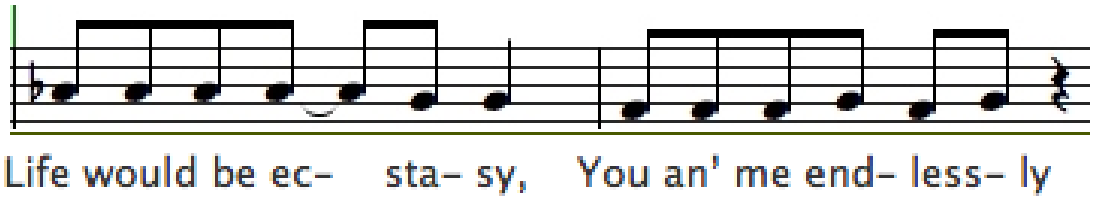
6.2.1 PRACTICAL EXAMPLE OF UNDERLYING THEORY

The structural rules of lyric placement are important, because without them, lyrics can become muddled and/or unin- telligible. For example, in the song *"Groovin' (on a Sunday Afternoon)"*, by the Young Rascals, there's a part in the bridge that many people hear as *"Life would be ecstasy, you an' me an' Leslie"*. In fact, the line is *"Life would be ecstasy, you and me endlessly"*. The confusion lies with the last three syllables of the phrase. The pronunciation of each version, if spoken normally, is as follows:

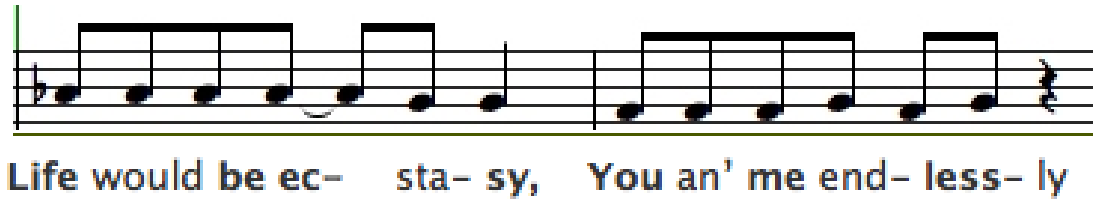
Alphabetic:	and Les- lie	end- less- ly
SAMPA:	@nd "lEs li	"End l@s li

So, in the first phrase, we see that the emphasis pattern can be simplified to "dum DUM-dum", where the first syllable of "Leslie" is emphasized. The second phrase's emphasis pattern is "DUM-dum-dum", so the first syllable of "endlessly" is emphasized. When words are put to music, however, the musical emphasis overrides the textual emphasis. Sometimes, the meaning of the phrase can change, if a previously un-emphasized syllable becomes emphasized, or a

previously emphasized syllable loses its emphasis. For "Groovin"', the lyrics match up to the music in the song as follows:



In this musical phrase, the emphasis always goes on the first part of a beat (for the purposes of this example, a "beat" is defined as a quarter note). In this case, the first measure is emphasized for the notes that correspond to the lyrics, "Life", "be", "ec-"(as in ec-sta- sy) and "sy"(again, as in ec-sta-sy) (This is a vast oversimplification, but it works for now). So, the lyrics would be emphasized as such:



Or, more simply:

1. Life would be ec-sta-sy
2. Life would be ec-sta-sy
3. LIFE would BE EC-sta-SY
4. *Life* would *be* *ec*-sta-*sy*
5. LIFE WOULD BE EC-STA-SY

6. LIFE would BE EC-sta-SY

7. LIFE would BE EC-sta-SY

8. LIFE would BE EC-sta-SY

9. LIFE would BE EC-sta-SY

10. *Life* WOULD *be ec*-STA-*sy*

11. LIFE *would* BE EC-*sta*-SY

12. **Life** *would* **be ec**-*sta*-**sy**

13. **Life** WOULD **be ec**-STA-**sy**

14. **Life would be ec-sta-sy**

15. **Life** would **be ec**-sta-**sy**

16. *Life* would *be ec*-sta-*sy*

17. Life would be ec-sta-sy

18. *Life would be ec-sta-sy*

19. Life *would* be ec-*sta*-sy

20. *Life* would *be ec*-sta-*sy*

21. LIFE would BE EC-sta-SY

22. *Life* would *be ec*-sta-*sy*

23. LIFE would BE EC-sta-SY

24. LIFE would BE EC-sta-SY

This musical emphasis matches the spoken emphasis of the phrase, so it is intelligible as a lyric. (Though ecstasy's first syllable doesn't start on the first part of beat three, it is still on the first part of beat three, and therefore still emphasized. Alternatively, since the first part of beat two didn't have a hard stop to it, the emphasis could have rolled over to the second part, "ec", which does have a hard stop.)

In contrast, take the second measure: the syllables "You", "me", and "less" are emphasized in the music. This leads to conflicting musical and spoken phrasing:

Musical Phrasing: **You** and **me** end**less**ly

Spoken Phrasing: **You** and **me end**lessly

The singer is now singing the phrase, syllable by syllable, which they think of as syllable-note combinations:

YOU and ME end LESS lee

The singer, for his part, is doing what many singers are taught to do, to make it easier to sustain the singing of words that end with unsingable consonants: the unsingable consonant is displaced onto the front of the next word. In this case, the consonant "d" is not singable, so he displaces it onto the next syllable, when he can: "and ME" becomes "an dME", and "end LESS" becomes "en dLESS". So, the singer can effectively think of the sung phrase as:

YOU an dME en dLESS lee

This doesn't cause confusion for listeners, because they're used to hearing it. This does mean, however, that lyric placement does not provide an accurate barometer to a listener of where a word actually ends.

In addition, the singer is singing fudging his vowels, like singers are taught to do, so "and" and "end" sound almost indistinguishable. So, really, what listeners are hearing is this:

YOU en dME en dLESS lee

Now, the listener's brain has to take this syllabic gobbledy-gook, and parse it into something useful. They've currently got this mess to deal with (represented in SAMPA syllables):

ju *En* ***dmi*** *En* ***dl@s*** *li*

They parse the first part just fine, because the emphases match:

you and **me** *En* ***dl@s*** *li*

But no one says endLESSly. People say ENDlessly. So, the listeners don't recognize it. They have to work with what they have. They already turned one "En d" into an "and", so they do it again:

you and **me** and ***l@s*** *li*

Now, they're just left with LESS lee. And that fits Leslie, a proper noun that fits in context and in emphasis placement. So, the final heard lyric is:

you and **me** and **Les-** lie

The misunderstanding can be traced back to improper emphasis placement. The songwriter probably didn't even think of that, and now he's stuck: a one-hit-wonder with a misunderstood song. We bet that in interview after interview, someone asks him who Leslie is. It's probably very frustrating — especially since he could have just moved the word an eight note later, and it would have been understood perfectly.

That's the sort of situation this program is going to help avoid.

LaTeX is a document markup language and document preparation system for the TeX typesetting program.

It is widely used by mathematicians, scientists, philosophers, engineers, and scholars in academia and the commercial world, and by others as a primary or intermediate format (e.g. translating DocBook and other XML-based formats to PDF) because of the quality of typesetting achievable by TeX. It offers programmable desktop publishing features and extensive facilities for automating most aspects of typesetting and desktop publishing, including numbering and cross-referencing, tables and figures, page layout and bibliographies.

LaTeX is intended to provide a high-level language to access the power of TeX. LaTeX essentially comprises a collection of TeX macros, and a program to process LaTeX documents. Since TeX's formatting commands are very low-level, it is usually much simpler for end-users to use LaTeX.

LaTeX was originally written in 1984 by Leslie Lamport at SRI International and has become the dominant method for using TeXfew people write in plain TeX anymore.

LaTeX is based on the idea that authors should be able to focus on the meaning of what they are writing, without being distracted by the visual presentation

of the information. In preparing a LaTeX document, the author specifies the logical structure using familiar concepts such as chapter, section, table, figure, etc., and lets the LaTeX system worry about the presentation of these structures. It therefore encourages the separation of layout from content, while still allowing manual typesetting adjustments where needed. This is similar to the mechanism by which many word processors allow styles to be defined globally for an entire document, or the CSS mechanism used by HTML.

Chapter 7

Conclusion

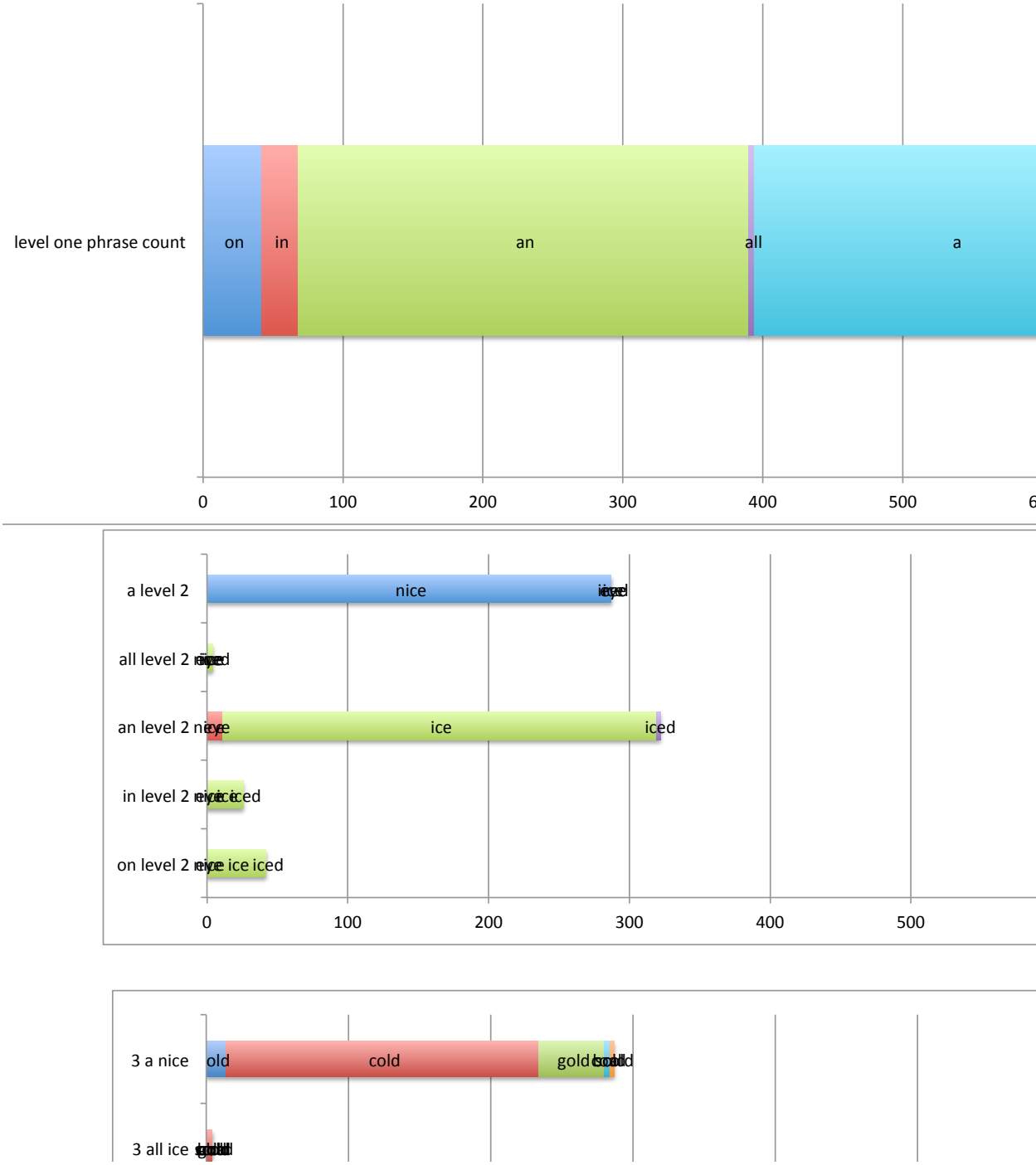
Our oronyms trees effectly display all the phonetically-matched oronyms that our users came up with. Unfortunately it also came up with a few that no human in their right mind would think of, and incorrectly weighted some others.

7.1 Successes

As seen in figure ??, all our transcribed oronyms follow our oronym parse tree, other than those that deal with phonemic substitution (such as the responses that include the word "gold" in lieu of "scold").

7.2 Places for improvement

In some cases, our phrase-frequency metric did not accurately line up with the actual transcription frequencies from our user studies. We believe that there are two possible reasons for this.


level one phrase count
on
in
an
all
a
0
100
200
300
400
500
a level 2
nice
all level 2
an level 2
ice
iced
in level 2
on level 2
ice
iced
0
100
200
300
400
500
3 a nice
old
cold
gold
3 all ice

7.2.1 Frequency Validity

Our frequency source data ended up being less than statisfactory. The lack of phonemic frequency data is a known defeciency in our source dictionary, UNISYN. According to the authors of the UNISYN lexicon documentation:

> Unfortunately there is currently no method for distinguishing between homographs by frequency. Furthermore, it should be noted that the frequency field, as it was obtained from simple word lists, is not particularly reliable.

[18] The UNISYN frequency count is based upon a large but not exhausting corpus of text. It has some particularly glaring deficiencies in the medical arena. We find this frustrating, because knowledge about common medical mondegreens could be used to prevent mistakes in patient's treatment plans[17]. Also, it meant that the word "colitis" wasn't in our dictionary, and we therefore couldn't use the example "the girl with colitis goes by/the girl with kaleidescope eyes".

Also, the fact that our program cannot distinguish between words that may be homographs (that is, words that sound different but are spelled the same) makes it improperly weight some phrases over others. For example, take the words for the animals "bucks" and "does". "Bucks" has a frequency of 1133, and "does" has a frequency of 508386. For reference, "deer" has a frequency of 1896. You can see the relative scale of these in figure 7.1. It seems highly unlikely that the male and female labels for a species would be more common than the actual name of the species, given that we don't see this for sheep (sheep , 13572 , ewe , 186 , ram , 681) or horses (horse , 27559 , mare , 1055 , stallion , 644). What is much more likely is that "bucks" is getting extra hits through its meaning as a slang synonym for dollars (dollars, 8927), and "does" is getting most of its frequency count for the 3rd person present tense of the verb "to do". That seems

Figure 7.1: Bubble Chart comparison of Frequency for deer, does, and bucks

very likely, given that the frequency for the singular "doe" is only 1077.

In the future, we'd like to find a dictionary with some way of distinguishing homographs when counting frequency, and that takes a larger, more-diverse dataset into its frequency count, such as the frequency lists from the Corpus of Contemporary American English[3].

7.2.2 Higher-order frequency data

Right now, our program only takes into account the frequency of standalone words, without taking their context into consideration. In the future, we'd like to integrate n-grams into our program. N-grams are a probabilistic model of predicting the next item that will follow in a sequence, based upon frequencies of how often those N items occur in sequence in a corpus of text[12]. A word-level 4-gram, for example, would be a series of four words. Here are some 4-gram phrases, along with counts of how often they occur, from the Google Ngram corpus:

serve as the informational 41 serve as the infrastructure 500 serve as the initial 5331 serve as the initiating 125 serve as the initiation 63 serve as the initiator 81 serve as the injector 56 serve as the inlet 41 serve as the inner 87 serve as the input 1323

[1]

and several phoneme-swapping strategies mentioned in future work.

SAMPA	Example	Manner of Articulation/ Type	Voiced/ Voiceless	Starts as	Ends as	General type	Weight
p	pen, spin, tip	plosive	voiceless	block	block	Consonant	8
b	but, web	plosive	voiced	block	block	Consonant	7
t	two, sting, bet	plosive	voiceless	block	block	Consonant	8
d	do, odd	plosive	voiced	block	block	Consonant	7
tS	chair, nature, teach	affricate	voiceless	block	continuous frication	Consonant	6
dZ	gin, joy, edge	affricate	voiced	block	continuous frication	Consonant	5
k	cat, kill, skin, queen, thick	plosive	voiceless	block	block	Consonant	8
g	go, get, beg	plosive	voiced	block	block	Consonant	7
f	fool, enough, leaf	fricative	voiceless	continuous frication	continuous frication	Consonant	4
v	voice, have, of	fricative	voiced	continuous frication	continuous frication	Consonant	3
T	thing, breath	fricative	voiceless	continuous frication	continuous frication	Consonant	4
D	this, breathe	fricative	voiced	continuous frication	continuous frication	Consonant	3
s	see, city, pass	fricative	voiceless	continuous frication	continuous frication	Consonant	4
z	zoo, rose	fricative	voiced	continuous frication	continuous frication	Consonant	3
S	she, sure, emotion, leash	fricative	voiceless	continuous frication	continuous frication	Consonant	4
Z	pleasure, beige	fricative	voiced	continuous frication	continuous frication	Consonant	3

SAMPA	Example	Manner of Articulation/ Type	Voiced/ Voiceless	Starts as	Ends as	General type	Weight
h	ham	fricative	voiceless	continuous frication	continuous frication	Consonant	4
m	man, ham	nasal	voiced	redirect	redirect	Consonant	1
n	no, tin	nasal	voiced	redirect	redirect	Consonant	1
N	singer, ring	nasal	voiced	redirect	redirect	Consonant	1
l	left, bell	lateral	voiced	continuous	continuous	Consonant	0
r	run, very	approximant	voiced	continuous	continuous	Consonant	0
w	we	semivowel	voiced	continuous	end	Consonant	2
j	yes	semivowel	voiced	continuous	end	Consonant	2
W	what (Scottish)	approximant	voiceless	continuous	end	Consonant	3
x	loch (Scottish)	fricative	voiceless	continuous frication	continuous frication	Consonant	4
A	father, not, law	short				Vowel	0.25
I	city	short				Vowel	0.25
E	bed	short				Vowel	0.25
3‘	bird, winner	short				Vowel	0.25
‘r	bird, winner	short				Vowel	0.25
{	lad, cat, ran	short				Vowel	0.25
u	soon, through	short				Vowel	0.25
@	about	short				Vowel	0.25
jU	use, pupil	diphthong		syllabic consonant semivowel	short	Diphthong	0.5
ju	use, pupil	diphthong		syllabic consonant semivowel	short	Diphthong	0.5
i	see	long				Vowel	0.75

SAMPA	Example	Manner of Articulation/ Type	Voiced/ Voiceless	Starts as	Ends as	General type	Weight
V	run, enough	short				Vowel	0.25
U	put	long				Vowel	0.75
e	day	long				Vowel	0.75
O	or, shore	long				Vowel	0.75
a	DNE in GenAm	long				Vowel	0.75
aI	my, height	diphthong		long	short	Diphthong	1
OI	boy	diphthong		long	short	Diphthong	1
oU	boat	diphthong		short	long	Diphthong	1
ou	boat	diphthong		short	long	Diphthong	1
aU	now	diphthong		long	long	Diphthong	1.5
=	ridden	syllabic consonant semivowel				Vowel	0.25

Table 7.1: Here are the metrics that we use to determine the weights of the phonemes

Bibliography

[1] All our n-gram are belong to you | research blog. http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html.

[2] The CMU pronouncing dictionary. http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

[3] Corpus-based word frequency lists, collocates, and n-grams. http://www.wordfrequency.info/comparison.asp.

[4] Dear R/Assistance, i'm about to finish my master's thesis, but i need your help! (tasks are online; i'm in san luis obispo, CA). : Assistance. http://www.reddit.com/r/Assistance/comments/ubty1/dear_rassistance_im_about_to_finish_m

[5] Dear RecordThis: i'm finishing up my masters thesis, and i need your help! : recordthis. http://www.reddit.com/r/recordthis/comments/ubt9f/dear_recordthis_im_finishing_up_my_ma

[6] File:General american.png - wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/File:General_American.png.

[7] Keep thou my way, hymnlyrics.org. http://www.hymnlyrics.org/newlyrics_k/keep_thou_my_way

[8] knights_emic.gif (GIF image, 552 407 pixels) - scaled (0%).

[9] knights_phonetic.jpg (JPEG image, 552 407 pixels) - scaled (0%).

[10] LCStar project web – schedule. http://www.lc-star.com/schedule.htm.

[11] Mondegreen | define mondegreen at dictionary.com. http://dictionary.reference.com/browse/mondegreen?s=t.

[12] N-grams: corpus based (COCA, COHA, spanish, portuguese). http://www.ngrams.info/.

[13] Orthography | define orthography at dictionary.com. http://dictionary.reference.com/browse/orthography.

[14] SQLite database browser. http://sqlitebrowser.sourceforge.net/.

[15] Unisyn lexicon. http://www.cstr.ed.ac.uk/projects/unisyn/.

[16] (1) "Rolling in the deep" cover, front porch band, Jan. 2012.

[17] J. Aronson. When i use a word words misheard: Medical mondegreens. *QJM*, 102(4):301–302, Apr. 2009.

[18] S. Fitt. Documentation and user guide to UNISYN lexicon and post-lexical rules. *Center for Speech Technology Research, University of Edinburgh, Tech. Rep*, 2000.

[19] J. Hendrix. Purple haze, June 1967.

[20] T. Polyakova and A. Bonafonte. Fusion of dictionaries in voice creation and speech synthesis task. In *Proc. of SPECOM*, 2007.

[21] C. C. Revival. Bad moon rising, Apr. 1969.

[22] G. P. Smith. Music and mondegreens: extracting meaning from noise. *ELT Journal*, 57(2):113121, 2003.

[23] S. Wright. The death of lady mondegreen. *Harpers Magazine*, 209(1254):4851, 1954.