An Effective Algorithm for the Alignment of Target and Actual Syllables for the Study of Language Acquisition

Keith Maddocks

Department of Computer Science
Memorial University of Newfoundland

A dissertation submitted to the Faculty of Science in partial fulfillment of the Bachelor of Science Honours in Computer Science

May 2005

Abstract

The speech of a child does not always conform to that of an adult. Children acquire language over time. This is reflected in their word pronunciations, which evolve from initial (actual) babbles to adult (target) forms. Prior to analysis, researchers must properly align the syllables pronounced by a child with their adult equivalents. For example, if the child intends to say “apricot” but actually produces “acot”, the two syllables of this actual form must be aligned to the first and last syllables of “apricot”. To ease the analysis of large child language datasets, an automatic alignment algorithm is needed. While work has been done on segmental (sound) alignment of two utterances, very little work has been done on syllabic alignment.

In this dissertation, I will discuss an algorithm designed to align both segments and syllables of target and actual utterances. I will also discuss the specific problems encountered in deriving such an algorithm, the methods employed to overcome these difficulties, and the preliminary results of tests of this algorithm against English language acquisition data.

Acknowledgments

I would like to acknowledge all the assistance I had in the development of this algorithm and in the writing of this dissertation. I would like to thank Dr. Todd Wareham for first introducing me to this project and for his continued help in the development of the algorithm. Without his unique insight and chipper demeanor, this project would not have progressed as well as it did. Also, I would like to acknowledge the great help and patience of Dr. Yvan Rose; who not only provided guidance throughout the entire process, but, also, managed to teach a computer scientist a thing or two about linguistics. And finally I would like to acknowledge the support of all my friends and family who have had amazing patience throughout all of this.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 The Problem

Children acquire language over time. From their first words until they have reached an adult level of speech, children's language skills are constantly evolving. How children acquire language is of great interest and importance to researchers. Such knowledge can be used not only to deepen the understanding of human language in general, but also to provide tangible benefits to society. Such knowledge can, for example, be used to aid speech therapists in diagnosing and treating speech problems in children, therefore; the study of language acquisition is an important pursuit.

The study of language acquisition requires a systematic comparison of what a child says with what he/she intended to say. In doing so, a linguist can determine what correlations exist between the intended (target) and actual utterances. Such alignments currently have to be done by hand relying on a linguist's knowledge, skills, and intuition. Designing an algorithm that can automate the task of alignment is the goal of this project.

1.2 The Phon System

The alignment algorithm designed in this project will be incorporated into the Phon system (Hedlund and O'Brien (2004); Rose *et al.* (2005)). Phone, which constitutes of a complete reengineering of the ChildPhon system (Rose (2003)) and is a software system for phonological data transcription and analysis.

Phon provides several tools to aid researchers in language acquisition research. Phon handles the management of research material in both video and audio form. These recordings can be segmented into smaller, more manageable

sections from within the application, and the audio portions of these segments can then be transcribed into text to allow the composition of the utterance to be analyzed. Phon also handles storage and retrieval of the transcribed data and analysis results.

The current version of Phon was designed and developed by Gregory Hedlund and Philip O'Brien under direction and supervision of Drs. Rod Byrne, Yvan Rose, and Todd Wareham. Once it is complete, Phon will be a component of the CHILDES system being developed under Dr. Brian Whinney at Carnegie Mellon University (http://childes.psy.cmu.edu).

1.3 Goals and Methodology

The goal of this project is to develop an automatic algorithm for syllable alignment that, given the syllabifications of the phonetic transcriptions of both the adult and child forms of an utterance, returns the corresponding segmental and syllabic alignments. Segmental alignment is the alignment of corresponding sounds between two related utterances while syllabic alignment is the alignment of corresponding syllables within these utterances. Because of inaccuracies produced by language learners, the algorithm must handle several alignment challenges that appear in real world situations. These include, but are not limited to, syllable deletion, syllable epenthesis, sound deletion, and sound epenthesis (see Section 2.2).

Since the aligner is part of the larger Phon project, it follows the three principles that have guided the development of Phon (Rose *et al*. (2005), p. 6).

1. The algorithm should be user-configurable via linguistically relevant parameters. This effectively makes the algorithm theory-neutral, and allows users to easily test and evaluate various proposed mechanisms for alignment.

2. The algorithm should be based on simple mechanisms and use the smallest number of user-configurable parameters possible. This not only ensures faster running time, but also restricts the number of potential unexpected and unwelcome interactions between parameters.
3. The algorithm need not always give perfect results, and there is always the option to manually override produced alignments. This prevents the accumulation of potentially interacting mechanisms to deal with infrequently occurring cases, and allows us to preserve the advantages of algorithm simplicity noted in (2) above.

In addition to these three principles, the alignment algorithm itself has two independent requirements:

1. The algorithm should produce alignments that are as accurate as possible. This means that the alignments produced, for the most part, reflect the alignments that would be produced by a linguist.
2. The algorithm should use the most efficient algorithm in order to produce results. This will allow large corpora to be aligned in the least amount of time.

These two principles are interrelated. While accuracy is very important, accuracy at the expense of time is not always beneficial; conversely, a fast algorithm that is inaccurate is not productive. The most effective algorithm is one that provides both acceptable speed and accuracy.

1.4 Organization of Thesis

Having provided a brief introduction to the problem at hand, it is necessary to review some basic linguistic concepts that are needed to understand the

functionality of the algorithm. This includes details on the composition of utterances and the structure of individual sounds (Section 2.1) as well as the role of alignment in language acquisition research (Sections 2.2 and 2.3.1).

Previous work on utterance alignment is reviewed in Section 2.3.2. While there has been some work in the area of segmental alignment of two utterances, *e.g.* Covington (1996), Somer (1998), and Kondrak (2002, 2003), there does not appear to be any published work on syllabic alignment (see, however, the preliminary work by Hedlund and O'Brien (2004) discussed in Section 4.2.2). While none of the work on segmental alignment is directly applicable to the problem at hand, it does provides valuable insights. In particular, Kondrak's work highlights the effectiveness of dynamic programming. Building on this idea, Section 2.3.3 outlines the basic principles of dynamic programming and discusses how alignment algorithms based on dynamic programming function. Knowledge of how such algorithms function, is key to understanding the alignment algorithm developed in this dissertation.

Details of our algorithm will be discussed in Section 3. Both a single-level approach (Section 3.1) and a double-level approach (Section 3.2) to syllabic alignment will be outlined. The testing process and the produced results will be discussed in Section 4.2. The single-level and the double-level algorithms are compared and contrasted, and the performance of the single-level algorithm for both segmental and syllabic alignments relative to various utterance types is examined. These results demonstrate the effectiveness of the single-level alignment algorithm.

Finally, in Section 5, an assessment is made as to how well our goals were met and how closely we complied with our guiding philosophies. Future avenues for research that have arisen from this work will also be discussed. These include several expansions to the current algorithms as well as potential functionality enhancements.

Chapter 2 Background

This chapter will provide an overview of the linguistic concepts needed to describe the utterance alignment problem. These include the concepts of utterance, phrase, syllable, phone, and feature matrix (Section 2.1). Building on these concepts, the problem of alignment will be explained (Section 2.2). Then, various alignment methods for solving this problem will be discussed (Section 2.3).

2.1 Utterance Structure: An Overview

Speech is divided into units of different size. Phon uses a five-level structure, which, from the largest unit to the smallest, is composed of Utterances, Phrases, Syllables, Phones, and Features. Figure 2.1 shows how an utterance breaks down into each of the smaller units. Each of these units is described in more detail below.

The largest unit is an ***utterance***. An utterance usually corresponds to a complete sentence, or to a collection of sentences. This level of structure represents the entire body of linguistic production that a person would utter at one time. An example of an utterance is the statement “Monkeys eat bananas”. Utterances, on their own, are generally too large for analysis. Therefore, utterances are broken down into smaller, more manageable units, each of which can be linguistically motivated from semantic, syntactic, morphological, or phonological perspectives.

Utterances break down into smaller units called ***phrases***. Phrases are groupings of words of arbitrary length that generally correspond to pre-defined parts of a sentence such as subjects, objects, and verbs. One such breakdown of our previous example gives the phrases, “monkeys”, “eat”, and “bananas”,

where “monkeys” is the subject of the sentence, “eat” is the verb, and “bananas” is the object.

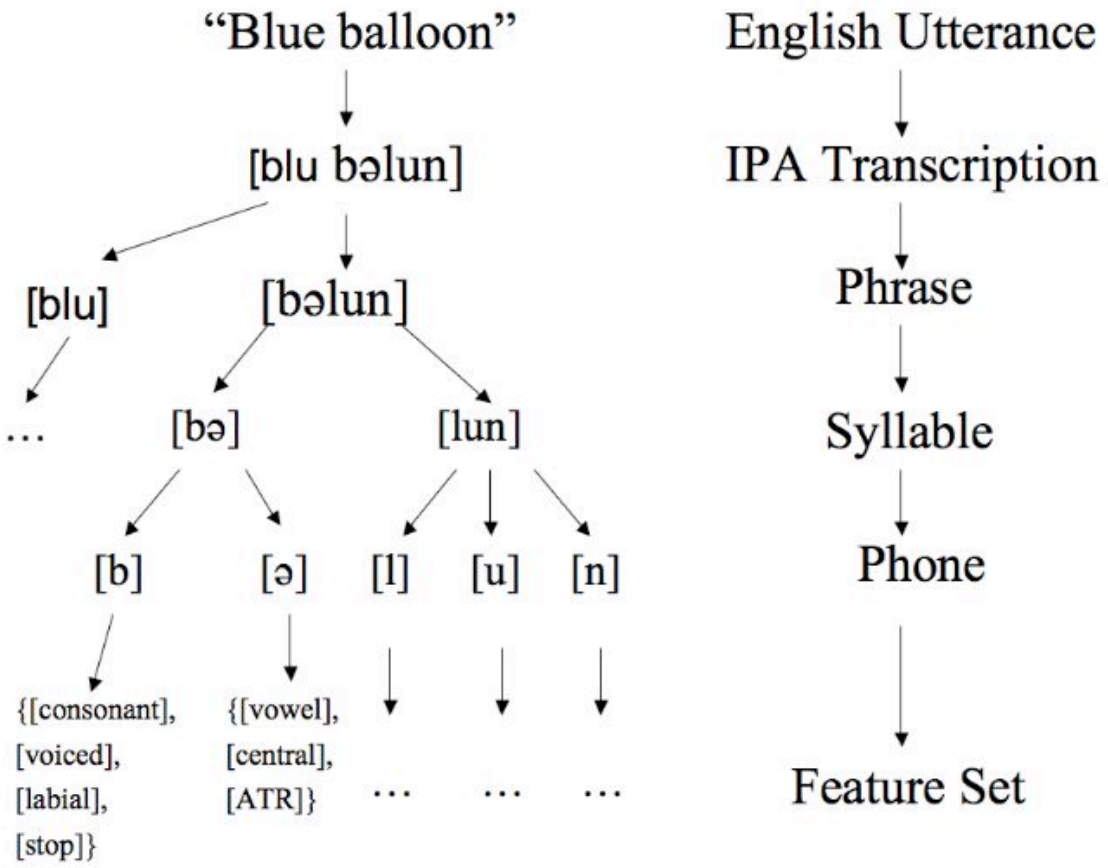

Figure 2.1 The Hierarchy of Utterance Structure

Phrases are in turn broken down into smaller units called ***syllables***. Syllables are essentially groupings of consonants around vowels. The phrase “bananas” breaks down into three syllables clustered around the three “a”s, namely “ba”, ”na”, and “nas”. Syllables can be subdivided into smaller units called constituents; however this level of structure is not necessary to understand the alignment algorithm. For further information on syllable-internal structure see Section 2.4.1 of Hedlund and O’Brien (2004).

Syllables are broken down further into ***phones***, each of which represents an individual sound. Individual phones can be decomposed into ***features***. Each

feature describes specific aspects of any given sound and combinations of features make up a feature matrix for each sound.

While there are five levels of linguistic representation used in Phon, the alignment algorithm works primarily on phones and feature sets. Therefore, it is important to provide more detail on these two levels.

In order to derive the set of sounds (and their corresponding feature matrices) contained in each linguistic production, linguists perform a phonetic transcription of these productions using a standardized set of symbols following conventions set by the International Phonetic Association (IPA). For example, the utterance "Monkeys eat bananas" broadly transcribed into IPA is [mə̃ŋkiz it bənænəz]. The resemblance between the two written forms is apparent; however, IPA transcriptions utilize symbols that are not used in written English. For example, even though the vowels in "bananas" are all orthographically written with the letter "a"; they are transcribed in IPA form using different characters, each of which corresponds to a specific vowel sound production.

Each symbol in the IPA character set represents a specific sound and each sound is composed of a collection of features grouped in a feature matrix. There is a standard set of features from which sounds are compiled and each sound has a unique set of features. The complete set of features used in this dissertation is the same as those used in Phon (see Appendix B of Hedlund and O'Brien (2004)). The individual feature matrix of each of the symbols in "bananas" ([bənænəz]) is shown in Table 2.2.

To give an example of how features relate to phones, we can look at the phone [b], which is composed of the features [consonant], [voiced], [labial], and [stop]. All sounds naturally break down into vowels and consonants. The sound [b] in this instance is a consonant. When a sound requires activation of the vocal folds, then it receives the feature [voiced]. The feature [labial] indicates that the lips are used in the pronunciation, while; finally, the blockage of the airflow

incurred by the lips being closed is indicated by the feature [stop]. The phone [b] can be contrasted with [p] in that both phones share the features [consonant], [labial], and [stop], but [b] has the feature [voiced] while [p] receives the feature [voiceless].

b	[consonant], [voiced], [labial], [stop]
ə	[vowel], [central], [ATR]
n	[consonant], [voiced], [coronal], [distributed], [anterior], [stop], [lateral], [sonorant]
æ	[vowel], [low], [front]
n	[consonant], [voiced], [coronal], [distributed], [anterior], [stop], [lateral], [sonorant]
ə	[vowel], [central], [ATR]
z	[consonant], [continuant], [voiced], [coronal], [anterior], [sibilant], [fricative]

Table 2.1 The Feature Sets for the Phones in [bənænəz]

2.2 Alignment in Language Acquisition

Syllables are amongst the basic units used in research on language acquisition. The problem in acquisition studies is that the number of syllables produced in the child (***actual***) form does not always match the number of syllables in the adult (***target***) form. This problem gives rise to two different situations.

1. **Syllable truncation (deletion):** The number of syllables produced by the child is smaller than that of the target form. Truncation is caused by the

child deleting or combining syllables. An example of a truncation is [bənænə] being pronounced as [bænə].

2. **Syllable epenthesis (addition):** This situation generally occurs when a child attempts to simplify the pronunciation of a difficult consonant cluster. To solve this problem a child will, at times, insert a vowel between two adjacent consonants. A child might, for example, break up the [ŋk] cluster in "monkeys" [mɑ̃ŋkiz]; this would transform "[mɑ̃ŋ][kiz]" into "[mɑ̃][ŋi][kiz]" and thereby create an additional syllable.

The study of language acquisition requires both phone-by-phone and syllable-by-syllable comparisons between target and actual forms. A straightforward alignment created by aligning symbols from left to right does not always produce accurate results, as truncation and epenthesis cause changes in symbol positions from target form to actual form. The mismatches generated by such an alignment can be seen in Figure 2.2.

	(a)		(b)
Target:	[bə][næ][nə]	Target:	[mɑ̃ŋ][kiz]
Actual:	[bæ][nə]	Actual:	[mɑ̃][ŋi][kiz]

Figure 2.2 Straightforward Alignment Yielding Incorrect Results

Part (a) of Figure 2.2 shows a misalignment of both syllables and phones caused by truncation. The utterance pair not only displays a deletion of a syllable in the actual form, but also the first target ([bə]) and second target ([næ]) syllables have fused. The resulting actual syllable [bæ] is composed of the consonant of the first target syllable [b] fused with the vowel [æ] of the second target syllable. In addition, in this alignment, the final syllables of both forms are identical; however, these final syllables do not align properly nor do the two [æ] phones. Part (b) of

Figure 2.2 shows an example of misalignment caused by epenthesis in which the child has inserted a vowel between the consonants [ŋ] and [k] thereby creating an extra syllable.

Thus, as syllables and phones cannot simply be lined up in a way that could warrant proper comparisons of the two forms, an alignment algorithm is necessary. Such an algorithm will need to align both phones and syllables to produce linguistically valid results such as those shown in Figure 2.3. To do so, the algorithm will need to be able to calculate the similarity between a pair of phones and the similarity between a pair of syllables. The algorithm will also need to take into account such elements as the common features of two phones, as well as other aspects such as the application of stress to a syllable.

	(a)		(b)
Target:	[bə][næ][nə]	Target:	[mɑ̃ ŋ#][kiz]
Actual:	[##][bæ][nə]	Actual:	[mɑ̃][ŋi][kiz]

Figure 2.3 Proper Syllable Alignment Examples

2.3 The Alignment of Utterances

There are many ways in which two utterances can be aligned and as many different approaches to alignment (see Setubal and Meidanis (1997), Chapter 3, for a detailed treatment of alignment algorithms). Some approaches derive a set of all possible alignments and then find the best match in that set. Others use the principle of reusing solved subproblems to speed up calculation of the optimal alignment. All of these issues will be explained and discussed in this section.

2.3.1 An Overview of Alignment

Since the problem presented in this dissertation relates to alignment, basic alignment terminology needs to be defined in advance. The main goal of alignment is to match two related items while avoiding matches between unrelated items. To do this an algorithm generates a score for each column-wise pair of items and the final score for the total alignment is the sum over all columns. Since our algorithm is aligning two utterances, the column-wise pair of items scored is a symbol in the target and a symbol in the actual as shown in Figure 2.4. The score generated depends on the type of alignment used and whether the algorithm is trying to minimize the distance between the two items or to maximize their similarity.

b	ə	n	æ	n	ə
#	#	b	æ	n	ə

Figure 2.4 A Column-wise Syllabic Alignment of Target and Actual Forms

The case of phonetic alignment involves aligning individual phonetic units in the actual form with their corresponding units in the target form. In our algorithm, the phonetic units used for alignment are phones. A phone in one form can either align with a phone in the other form or it can align with a special symbol called an ***indel*** (denoted here by the symbol "#"). An indel is used to indicate that a phone has been deleted from the target form or moved into another syllable or that a symbol has been inserted into the actual form. The final alignment, therefore, is a column-wise matching of phones with phones or phones with indels.

The numeric value of the alignment score is dependent on two things: the type of alignment used, and whether the distance or the similarity between pairs of phones has been optimized. The possible types of alignment are

demonstrated in Figure 2.5, using orthographic transcription for sake of simplicity. Each type of alignment is used in different situations to solve different problems. The first of these alignment types is ***global alignment***, which is used when the entire target form must be aligned with the entire actual form. The opposite situation is ***local alignment***, in which only the most common portion of the two utterances being aligned is considered. In this type, symbols at the beginning and the end of both forms can be deleted at no cost in order to obtain only the subset of each that is most similar to the other. The third situation is ***semiglobal alignment***, in which an arbitrary number of free insertions or deletions is allowed for any length of symbols at either the beginning or the end of either utterance for both actual and target forms. The result of these free insertions or deletions is an overlap between the two utterances.

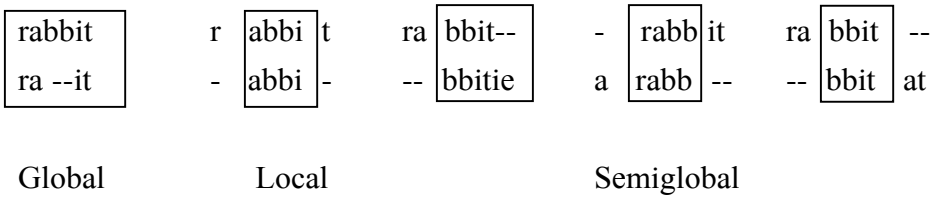

Figure 2.5 Types of Alignment

To create any of the three types of alignment, the score for two matching phones needs to be calculated. There are two ways of doing this: distance and similarity. The distance between two phones is a function of how different the two phones are, *i.e.* how far apart they are. The distance between two phones is calculated using a distance metric. A distance metric *d(x,y)* on a pair of phones *x* and *y* is a function that maintains the following properties (Setubal and Meidanis (1997), p. 91):

1. $d(x,x) = 0$ for all $x \in E$ and $d(x,y) > 0$ for $x \neq y$

   The distance between an element and it self is zero and the distance between an element and any different element is always greater than 0.

2. $d(x,y) = d(y,x)$ for all $x,y \in E$

   The distance between two elements is always symmetric.

3. $d(x,y) \leq d(x,z) + d(y,z)$ for all x,y, and $z \in E$

   The distance between two elements is always less then or equal to the distance between the first element and any combination of other distances leading to the second.

Similarity, on the other hand, is a function that indicates how closely related two symbols are. There are no restrictions on similarity functions, though they are typically symmetric.

To calculate the distance or similarity between two utterances, the values of the distance or similarity of all the symbol pairs in both utterances are summed. This produces a score for the entire alignment built from the individual symbol-to-symbol scores. Distance metrics can be used to produce only global alignments, while similarity can be used to generate any of the previously discussed types of alignment.

2.3.2 Previous Work

There are three main bodies of work whose material have influenced this dissertation: Covington (1996), Somers (1998), and Kondrak (2002). It should be noted up front that all three approaches focus on segmental alignment and not on syllabic alignment. Also, only Somers' work deals with child language acquisition while the other two algorithms were developed to deal with problems in historical language reconstruction. This, however, does not disqualify them from consideration because the algorithms they propose are adaptable to our current problem.

Covington (1996) focused on historical reconstruction of ancestral languages. The algorithm was designed to take words from different languages that share a

common ancestor and attempt to find what portion of the original word was preserved in both forms. Covington employed an exhaustive search approach to solve this problem. His algorithm generates all possible global alignments in a tree and traverses that tree in a depth-first manner. The selected alignment is the alignment that minimizes the distance between the two inputs.

Somers (1998) focused on child language acquisition. Somers utilized a greedy algorithm that relied on the observation that children tend to preserve the stressed syllable from the target form. Using this fact, Somers' algorithm relies on stressed syllables as anchor points for an alignment. The algorithm recursively aligns stressed syllables and those symbols around the anchor points, using minimized phone-to-phone distance, until a complete alignment is produced. Therefore, the resulting alignment is not the alignment that would minimize the distance between the two utterances; rather, since the anchor points are not necessarily the optimal matches, the alignments produced are not guaranteed to be the optimal alignments.

Kondrak (2002) proposed a dynamic programming (see Section 2.3.3) approach to solve the same problem as Covington, that of historical reconstruction. Kondrak's algorithm also calculates the similarity between two words by counting the number of common features, where each feature has a predefined weight. Both Covington's and Somers' algorithms calculate distance, which makes Kondrak's algorithm very different from the other two. An additional difference in Kondrak's algorithm is that it finds the best local alignment for the two words and not a global alignment.

From the descriptions above it can be seen that none of these algorithms meet our goals. However, for each of these algorithms there are three key elements that are of importance with respect to the problem addressed in this dissertation.

1. **Global Alignment** – Like Covington and Somers, our algorithm needs to align the entire target form with the entire actual form. This is because we

are not interested in the preserved core of the two utterances, like Kondrak, but want to see where additions or deletions were made.

2. **Similarity Score** – Covington's algorithm optimizes the alignment based upon minimum distance between the two words. Kondrak, on the other hand, optimizes on maximum similarity, while Somers' algorithm does not optimize the final score on either distance or similarity. Since the algorithm developed in this dissertation will endeavor to produce an optimal global alignment, either distance or similarity can be selected. Similarity provides us with more flexibility then the distance approach, as we do not have to maintain metric properties (see Section 2.3.1).
3. **Dynamic Programming** - This is very important because it directly relates to the goals set in Section 1.3. Covington's exhaustive search algorithm can be time consuming when run on long word pairs and large corpora as the number of possible alignments in such an algorithm is exponential. The dynamic programming approach proposed by Kondrak is much more efficient and runs in $O(m*n)$ time. The dynamic programming approach, therefore, is more in line with the goal of an efficient algorithm.

2.3.3 A Dynamic Programming Alignment Algorithm

Dynamic programming is a method utilized to increase the efficiency of an algorithm by reusing solved subproblems (see Cormen *et al.* (2001) Section 15, for details). This approach is advantageous in utterance alignment because it does not require the algorithm to generate all possible alignments. Instead, it calculates the best alignment of the given forms using the best alignments of smaller portions of these forms.

An example of how dynamic programming can be used to reduce the run time of an algorithm can be demonstrated using the calculation of the Fibonacci numbers. The Fibonacci numbers are a sequence of numbers where the first two

numbers in the sequence are ones and every number after is the sum of the two previous numbers. This can be summarized by the following recurrence:

$$F_i = \begin{cases} 1 & i = 1,2 \\ F_{i-1} + F_{i-2} & i > 2 \end{cases}$$

The first ten Fibonacci numbers, according to this equation, are 1, 1, 2, 3, 5, 8, 13, 21, 34, and 55. To calculate the third Fibonacci number, F3, an algorithm would need to add the two preceding numbers, F1 and F2, so F3 = F1 + F2. The value of F4, therefore, is F2 + F3. We already know, however, that F3 is equal to F1 + F2 so F4 is actually equal to F1 + F2 + F3. The key observation here is that the algorithm has already calculated F3 in the previous step and does not need to calculate it again. Hence, if F3 is stored in memory, calculating F4 is only a matter of looking up two values in a table. This way the algorithm uses the stored values of smaller solved subproblems to speed up the calculation of larger problems.

To apply this approach to pairwise global utterance alignment, the subproblems, for which the solved values are stored and reused are the prefixes of the two utterances. A ***prefix*** is any string of symbols starting at the beginning of an utterance. For a given pair of utterances, the number of possible prefix combinations is *(m+1)*(n+1)*, where *m* and *n* are the number of symbols in the target and actual forms, respectively. For each prefix combination for both the target and actual utterances the maximum similarity score for each can be calculated. This results in an *(m+1)* by *(n+1)* matrix that has to be stored in memory.

The basic dynamic programming algorithm for utterance alignment is given in Figure 2.6. In the algorithm, *g* is the cost of inserting or deleting a character, the matrix *a[,]* is used to store the values of the solved subproblems, *target[x]* and *actual[y]* are the phones at positions *x* and *y* in the target and actual forms respectively, and *similarity(target[x],actual[y])* is the function that calculates the

similarity of two phones.

```
Algorithm Similarity
    input: sequences target and actual
    output: similarity between target and actual
    m ← |target| //length of the target utterance
    n ← |actual| //length of the actual utterance
    for i ← 0 to m do
        a[i,0] ← i * g
    for j ← 0 to n do
        a[0,j] ← j * g
    for i ← 1 to m do
        for j ← 1 to n do
            a[i,j] ← max(   a[i-1,j] + g, //an indel in the target
                            a[i-j,j-1] + similarity(target[i],actual[j]),
                            a[i,j-1] + g //an indel in the actual
                        )
```

Figure 2.6 Pseudocode for the Single-level Alignment Algorithm (Adapted from Figure 3.2 of Setubal and Meidanis (1997))

The first task the algorithm accomplishes is to initialize the top row and leftmost column with the indel cost. The indel cost for each cell is the cumulative score of all the indels before it plus *g*. Next, the algorithm iterates through the remaining cells of the matrix and calculates their values. The value for each cell is the maximum of the following:

1. The value of aligning the first *j* symbols of the actual form with the first *i-1* symbols of the target form while deleting the i^{th} symbol in the target.
2. The similarity score of the two current symbols (*target[i]* and *actual[j]*) plus the value from aligning the first *i-1* symbols in the target with the first *j-1* symbols in the actual.
3. The value of aligning the first *i* symbols of the target form with the first *j-1*

symbols in the actual from plus an insertion of a symbol at the i^{th} position.

Since the algorithm only uses values that are immediately on top of *(a[i,j-1])*, to the left of *(a[i,j-1])*, or on a diagonal to *(a[i-1,j-1])* the current cell, the entire table can be filled in starting from the top left corner.

Once the final similarity score is calculated in the bottom right hand corner, the alignment can be derived. To trace back the alignment from the given matrix, the algorithm starts from the bottom right hand corner. It then traces back the path the algorithm initially used to generate the table. The path is generated by moving backwards left, right, or diagonally to the cell that corresponds to the cell used to generate the maximum similarity score. Each horizontal or vertical move corresponds to an indel in the target or actual form, respectively. A diagonal move corresponds to an alignment of a symbol in the target and actual forms. By storing the appropriate symbol-pairs in a stack, upon reaching the top right hand corner, the alignment can be retrieved.

Chapter 3 Algorithm Development

Both methods for syllabic alignment discussed in this dissertation use a modified version of the basic dynamic programming algorithm for global alignment given in Section 2.3.3. This algorithm was chosen for its benefits in time and space complexity over more exhaustive approaches. The majority of development and computation for the algorithm is done in determining the similarity of pairs of phones, which is used in turn to determine the similarity of pairs of syllables. The simplicity of the dynamic programming recursion is intentionally retained.

While the original aligner for the Phon project was designed as a double-level dynamic programming algorithm (Hedlund and O'Brien (2003); see also Section 3.2), this dissertation focuses primarily on a single-level approach. This decision was made for two reasons. First, extending the single-level approach to a double-level approach is trivial; indeed, the double-level approach discussed in Section 3.2 uses the single-level algorithm to calculate the similarity of two syllables. Second, the single-level algorithm falls more in line with our principle of starting with the most basic structure and building upon it. While the single-level approach is not perfect, it will be shown that its advantages far outweigh its shortcomings.

3.1 Single-Level Dynamic Programming

While it might seem unusual to approach syllabic alignment by way of a modified segmental phone alignment, this approach provides three main advantages:

1. The single-level algorithm provides the simplest mechanism for

alignment. Every alignment is done at the phone level, which means that there is only one level of structure that the algorithm needs to contend with. This is in contrast to an approach that would use one algorithm to align syllables and another to align the phones within those syllables.

2. By treating utterances as a collection of phones, the algorithm is able to easily handle one-to-many syllable mappings. A one-to-many mapping is a relationship between one syllable and zero or more syllables. This represents the cases of truncation (deletion) or syllable epenthesis (insertion) discussed in Section 2.1. Since the single-level algorithm works entirely on phones, it is not concerned with the syllabic level and only matches phones based on their common features and the set of rewards. It is the appropriate rewards, then, that produce the illusion of syllabic alignment and in turn produce the proper syllable mappings.
3. This approach allows for the alignment of both syllables and segments. Since, in essence, syllabic alignment is actually a modified phone alignment, the single-level approach allows for these two different forms of alignment. This flexibility is a major advantage of the single-level algorithm and one that a double-level approach would not be able to produce.

Despite these advantages the single-level algorithm does have problems. As mentioned before, this approach ignores the overlaying syllabic structure, which has the potential to corrupt syllable integrity. This means that individual phones in a syllable may wander outside their syllable boundary, resulting in a mapping in which actual syllables align with the wrong target syllables and pairs of syllables in the actual form map to the same syllable in the target form. An example of such an incorrect alignment is given is Figure 3.1. The [t] in the second syllable of the actual form is misaligned with the [n] in the target form. The result of this is that the syllable [tɛ] is aligned with both syllables [kʌn] and

[tɛst] when is should only align with the latter. This is a serious issue and a major challenge raised by the current approach.

Incorrect Alignmnet		Correct Alignmnet	
Target:	[kʌ n][tɛ s# t#]	Target:	[kʌn][tɛ s# t#]
Actual:	[ka][t #ɛ][si][ti]	Actual:	[ka#][tɛ][si][ti]

Figure 3.1 Syllable Integrity Issues in the Alignment of "contest"

Another problem that arises with using phones to align syllables is that once the alignment has been created, syllable boundaries must be reintroduced to create a syllabic alignment. This issue is dealt with in Section 3.1.2; for the moment, it is only important to note that this can be handled by using a set of syllable boundary propagation rules.

3.1.1 Modified Segmental Alignment

To achieve alignment, rewards are given to matching phones under a variety of conditions. Different combinations of rewards are used to generate either segmental or syllabic alignments. There are also certain rewards that are applied to both cases.

It should also be noted that the values used for the rewards and penalties are not necessarily optimal. All values given in this dissertation were obtained from trial and error or from educated guesses based upon knowledge of the algorithm's performance. These values could be optimized using a variety of techniques, as will be discussed in Section 5.1. However, before any rewards are introduced it is important to explore the basic similarity score of two phones.

Basic Similarity Score

The basic similarity score of two phones is calculated using the number of features shared by these phones. The intersection of these two sets provides a count of the number of features the two phones have in common. Consider the phones [f] and [b] shown in Table 3.1. Since both [f] and [b] have the features [consonant] and [labial], we can say that the similarity score for these phones is 2. This value provides the most basic similarity score.

Phone	Feature Set
b	[consonant], [voiced], [labial], [stop]
f	[consonant], [continuant], [labial], [fricative]

Table 3.1 The Feature Sets of Phones [f] and [b]

The other value that is needed in this basic alignment approach is an indel penalty for deleting or inserting symbols. This reflects a situation where a child either deletes a sound from the target form or inserts a new sound in the actual form. A constant value of -2 is applied to all indels. This value is optimized as to discourage too many insertions or deletions while also discouraging bad matches. As the algorithm uses a global alignment scheme, initial and final indels in both target and actual forms are penalized equally.

These two values, the similarity of two phones based on the number of common features and the cost of inserting or deleting a phone, are what is required for the basic alignment algorithm. This approach will be called the basic approach for comparison with more sophisticated scoring methods discussed later.

Modified Similarity Score

To create more sophisticated similarity scores, additional values need to be added to the count of common features alone. These rewards and penalties reflect additional attributes of the phones that contribute to their similarity. These values are applied multiplicatively to the count of common features. The algorithm checks to determine if each reward is applicable to the current pair of phones. If it does, the associated reward value is added to a cumulative tally of reward values. Once all the checks are complete, the final reward tally is multiplied by the number of common features. However, if a penalty is applicable to the phone pair, then just the penalty value is returned as the similarity.

A list of these rewards and penalties along with their associated values is given in Table 3.2. The entries are listed in the order in which they were added to the algorithm.

Indel Penalty	-2
Exact Match Reward	2
Primary Syllable Stress Reward	5
Stressed Vowel Reward	16
Vowel Boost Reward	4
Stressed Vowel only Reward	20
Secondary to Primary Stress Reward	3
Secondary to Secondary Stress Reward	1
Same Syllable Constituent Reward	2
Same Place Reward	1
Diphthong Penalty	-2

Table 3.2 Alignment Rewards and Penalties with their Associated Values

Each of these mechanisms was added to resolve a situation that became evident during testing against our English language corpus (see Section 4.2 and Appendix I). Even though the entries are listed in the order of addition to the

algorithm, it is more convenient to discuss them by categories as displayed in Table 3.3.

Common Rewards	Common Penalties
• Vowel Boost	• Indel
• Syllable Constituent	• Diphthong
Segmental Rewards	**Syllabic Rewards**
• Vowel only Stress Boost	• Primary Stress
• Place	• Secondary Stress
• Exact Match	

Table 3.3 Alignment Rewards and Penalties by Category

In the following section, we look at the various mechanisms and the examples that motivated them. Since some mechanisms apply to segmental alignment and others to syllabic alignment, it is difficult to discuss them in historical order so they are discussed by category. The mechanisms that apply to both segmental and syllabic alignment will be examined first.

Common Rewards and Penalties

The mechanisms that are common to both segmental and syllabic alignment are the Vowel Boost reward, Syllable Constituent reward, and Diphthong penalty.

Vowel Boost Reward: Vowels have fewer features, in general, than consonants; thus, a common vowel boost is applied to all vowels to bring their scores more in line with those produced for consonants. This boost ensures that vowels match up and are not forced out of alignment by matching with consonants.

An example of the need for this reward is the segmental alignment of the utterance “another” given in Figure 3.2. In this example the vowel [ɚ] has aligned

with the consonant [ɹ]. The reason for this is evident when the feature sets of both phones are examined. The feature set of [ɚ] is {[rhotic], [vowel], [sonorant], [central], [mid], [voiced]}. The feature set of [ɹ], on the other hand, is {[rhotic], [sonorant], [continuant], [consonant], [approximant], [coronal], [voiced]}. By comparing these two feature sets we can determine that the common features of both phones are {[rhotic], [sonorant], [voiced]}. However, [ɚ] is a vowel and [ɹ] is not and vowels and consonants should never align. The question, then, pertains to why did [ɚ] not align with [ə] since both are vowels. The answer relates to the fact that the feature set of [ə] is {[vowel], [central]} – that is, the two vowels only have one feature in common. According to our algorithm, it makes perfect sense for the vowel [ɚ] to align with the consonant [ɹ] since they have more features in common. The Vowel Boost reward thus ensures that vowels only align with other vowels.

Incorrect Alignment		Correct Alignment	
Target:	ənʌðəɹ	Target:	ənʌðəɹ
Actual:	#nʌd#ɚ	Actual:	#nʌdɚ#

Figure 3.2 Correct and Incorrect Alignments of "another"

Syllable Constituent Reward: Syllables actually break down into smaller components called constituents. Phones that belong to the same syllable constituent within a syllable should match up with phones of the same constituent type in the matching syllable. To encourage this alignment a small reward is given to phones in matching syllable constituents.

The need for this reward is demonstrated in Figure 3.3. The phone sequence [ga] in the actual form is aligning with the phone sequence [rɑ] in the

second syllable. This causes the [ɾ] in the actual form to align with an indel. Since this is an example of truncation, the first syllable in the actual form should align with both syllables in the target form. The syllable constituent reward encourages [ga] to align with [gə] since the phones in each share the same syllable constituents.

Incorrect Alignment		Correct Alignment	
Target:	[gə][rɑ#ʒ]	Target:	[gə][rɑʒ]
Actual:	[##][gardʒ]	Actual:	[ga r#dʒ]

Figure 3.3 Correct and Incorrect Alignments of "garage"

Diphthong Penalty: A ***diphthong*** is a pair of vowels that act as one vowel with respect to phonetic alignment. The only situations that are considered in the current alignment algorithm are a "core" vowel preceded or followed by a glide (a phone possessing the feature [glide]) or a high lax vowel (a phone possessing the features [vowel] and [high] but not the feature [ATR]). These situations are broken up into two categories: a "heavy" diphthong which consists of a core vowel followed by a glide or a high lax vowel and a "light" diphthong, which is the inverse, a high lax vowel or glide followed by a core vowel. Each of these situations can be seen in Table 3.4. For both of these cases, the core vowel involved must align with another vowel.

Heavy Diphthongs		Light Diphthongs	
paɪz	pajz	pɪaz	pjaz
pe#z	pe#z	p#ez	p#ez

Table 3.4 Diphthong Correct Alignment Examples

Several approaches were considered for handling diphthongs. These were preprocessing, postprocessing, modifying the recurrence, or altering the cost function. Preprocessing was ruled out because of the large changes needed for the algorithm to handle a change in input. Postprocessing was viable; however, this solution would move the responsibility outside the actual alignment algorithm. Finally, changes to the recurrence would violate our goal of using the simplest mechanism possible.

Having eliminated the first three possibilities the only option remaining was to attempt to handle the situation in the cost function. The key to this approach is recognizing that it is always the core vowels of diphthongs that align. If a reward were given to vowels that were part of a diphthong, then the similarity boost would be enough to encourage alignment, however, this approach could also force core vowels to lock on to an incorrect vowel if the reward were too great.

Another option is to penalize incorrect matches within a diphthong. When a vowel aligns with the glide or high lax vowel portion of the diphthong, a penalty is incurred which forces the algorithm to align the core vowels. This approach has the benefits not only of aligning the correct portions of the diphthong combinations but also of not producing dubious matches.

The Vowel Boost reward, Syllable Constituent reward, and Diphthong penalty are applied across the board. The effectiveness of each can be seen in Section 4.2; for the moment, it is sufficient to say that they provide significant increases in accuracy.

Segmental Rewards

Since the alignment algorithm is intended to produce two different alignments, segmental and syllabic, there are collections of mechanisms that apply exclusively to each situation. Segmental alignment focuses on aligning a phone with the phone that is most similar to it. For example, a [b] in the target form might be pronounced as an [m] in the actual form; in this case the [m]

effectively corresponds to the [b], such that the alignment algorithm must align these two phones.

Exact Match Reward: Phones that do not change between the target and actual form should align. To encourage this idea, phones that exactly match in both target and actual utterances are encouraged to align. Only a small reward is given here as the number of features shared by two matching phones will generally be high and the resulting score will also be high, regardless of this reward.

This is evident in the [bənænə] / [bænə] alignment discussed earlier (see Section 2.2) and shown in Figure 3.4. In this example each phone in the actual form matches an identical phone in the target. The exact match reward encourages this correct alignment.

Incorrect Alignment		Correct Alignment	
Target:	bənænə	Target:	bənænə
Actual:	bænə##	Actual:	b##ænə

Figure 3.4 Correct and Incorrect Alignments of "Banana"

Stressed Vowel Reward: Vowels that occur in stressed syllable must lock on to other vowels in stressed syllables. A significantly large boost is applied to two vowels that both occur in primary stressed syllables. This reward forces stressed vowels to align regardless of how dissimilar they are. A large reward is needed as vowels, in general, have very few features and different vowels will have even fewer in common. This reward is similar to the approach of Somers (1998) when he used stressed vowels as anchor points. Our algorithm does not go as far as that; however, it does recognize that the preservation of stressed vowels is key to creating correct alignments.

The effect of this reward is demonstrated in Figure 3.5. In the correct alignment the first [a] in the actual form aligns with [e] in the target form. This is because both phones are stressed. However, in the incorrect alignment, [a] aligns with [ə]. This is caused by the similarity of the two [t] phones. The vowel only stress boost forces the phones [a] and [e] to align thereby causing the [t] to correctly align with [ɾ] despite their featural differences.

Incorrect Alignmnet		Correct Alignmnet	
Target:	pəteɪɾə	Target:	pəteɪɾə
Actual:	pat###ə	Actual:	p##a#tə

Figure 3.5 Correct and Incorrect Alignments of "Potato"

Place Reward: Our algorithm, for the most part, does not that take into account the notion of feature salience, where salience is essentially a weighting of features. Kondrak's algorithm (Kondrak 2002), however, relies heavily on these weights. In our algorithm, all features are equally weighted with one exception. Features that refer to place of articulation must be taken into account when aligning phones. The features [labial], [velar], and [coronal] are all considered place features. Children easily swap phones that share the same place so the algorithm must take this into account. The effect of the place reward can be seen in Figure 3.6.

Incorrect Alignment		Correct Alignment	
Target:	bənænə	Target:	bənænə
Actual:	##mænə	Actual:	m##ænə

Figure 3.6 Correct and Incorrect Alignments of "Banana"

Syllabic Rewards

In contrast to segmental alignment, syllabic alignment is concerned with aligning whole syllables, not just individual phones. As such, the matching of individual phones it not as important as the matching of entire syllables.

Primary Stress Reward: Syllabic alignment takes into account the stress of an entire syllable, whereas segmental alignment rewards only the vowel portion. Syllables can be transcribed as having primary stress, secondary stress, or no stress at all. Stress represents an emphasis that is placed on a syllable when pronounced by the speaker. Primary stressed syllables should align with other primary stressed syllables. More generally, as noted above, stressed vowels should align with other stressed vowels. The primary stress reward encourages this alignment by rewarding the column-wise matching of all phones inside primary stressed syllables, thereby encouraging the entire stressed syllables to align.

Figure 3.7 demonstrates the correct segmental and syllabic alignment of the target utterance [bəˈluːn] and the actual utterance [ˈbun]. In the segmental alignment the [b] of [bun] aligns with the [b] of [bəlun]. This is to be expected, as they are both identical. However, in the syllabic alignment the [b] of [bun] aligns with the [l] of [bəlun]. The reason for this is that [bun] has only one syllable and that single syllable is more similar to the second syllable in [bəlun] than to the first as both [bun] and [lun] have two phones in common. In contrast, [bun] and [bə] only have one phone in common and [bə] has one fewer symbol than [bun]. In addition to these similarities both [bun] and [lun] are primary stressed syllables.

Segmental Alignment	Syllabic Alignment
bəlun	[bə][lun]
b##un	[##][bun]

Figure 3.7 Segmental and Syllabic Alignments of [bəˈluːn] and [ˈbun]

Secondary Stress Reward: Under certain circumstances, primary stressed syllables will align with secondary stressed syllables and secondary stressed syllables with other secondary stressed syllables. This reward reflects the tendency of a child to only pick up on the emphasis and not differentiate between primary and secondary stress, *i.e.* to treat all stress as the same.

A bonus is applied when a phone from a primary stressed syllable aligns with a phone from a secondary stressed syllable. In cases where the primary stressed syllable of the actual form is very similar to the secondary stressed syllable of the target form, this boost encourages these syllables to lock on to each other. In addition to a primary / secondary stress match, a small boost is provided to two phones which both occur in a secondary stressed syllable. This encourages both primary and secondary stressed syllables to align. By doing so, this reward aids in maintaining syllabic integrity because it encourages entire syllables to align. Once the stressed syllables align, unstressed syllables will be properly distributed.

In Figure 3.8 the first [a] in the actual form is misaligned with the phone [ɛ] in the target form. Although on the basis of common features the alignment is correct, it does not take into account that both the vowels [a] and [ɑ] are in stressed syllables. Once the Secondary Stress reward is put in place, the vowels correctly align and in turn encourage [k] in the target and [t] in the actual form to align correctly despite their feature differences.

Incorrect Alignment		Correct Alignment	
Target:	[hɛ][lɪ][kɑp][tər]	Target:	[hɛ][lɪ][kɑp][tər]
Actual:	[ta][##][###][ta#]	Actual:	[##][##][ta#][ta#]

Figure 3.8 An Incorrect Alignment of "helicopter" Caused by Alignment of Primary Stressed Syllables

3.1.2 Syllable Reconstruction and Boundary Propagation

The stress rewards described in the last section, in addition to the common rewards, are sufficient to properly align syllables. However, since we have chosen a single-level approach, the algorithm really only aligns phones. Once the algorithm has aligned the phones of the syllables, it then needs to reconstruct the original syllable boundaries.

This is achieved in two ways. First, a mapping is created between all syllables in both target and actual forms where a mapping is simply a listing of all pairs of syllables that align with each other. This mapping can then be used to retrieve all syllables that align with a given syllable. Syllables that align with indel syllables are excluded from the mapping.

Second, in addition to a mapping of syllable pairs, the algorithm also generates a pair of strings that can be displayed to the user. The main difficulty in doing so is figuring out how to properly reconstruct the syllables from the aligned phones and how to correctly propagate the boundary markers.

There are four situations for boundary propagation as shown in Figure 3.9. In the figure, X and X' are adjacent phones in the target form while Y and Y' are adjacent phones in the actual form. The first situation of boundary propagation arises when there exists a boundary in both target and actual forms. In this situation the boundary is maintained. The second situation is when there is neither a boundary in the first or in the second form. In this situation no new boundary is introduced. The third and fourth situations occur whenever there is a

boundary in the target or in the actual form but a matching boundary does not exist in the other.

Type 1	Type 2	Type 3	Type 4
X|X'	XX'	X|X'	XX'
Y|Y'	YY'	YY'	Y|Y'

Figure 3.9 The Four Types of Syllable Boundary Propagation Situations

If a boundary exists in either target or actual form, the algorithm must determine if that boundary is to be propagated. An example of an alignment where the boundary exists in one form and not in the other is given in Part (a) of Figure 3.10. In this alignment, there exists a boundary between the second (A2) and third (A3) syllables of the actual form that does not align with a boundary in the target form. To determine if this boundary should propagate up into the target form, the algorithm checks for a mapping on either side of the boundary. In this case, there exists a mapping from the second and third syllable of the actual form to the first (T1) syllable of the target form. Since A2 and A3 map to T1, we cannot propagate the boundary, as it would increase the number of syllables in the target form. If a mapping did not exist for either A2 or A3, then it would indicate that one syllable maps to an indel and the boundary could propagate. This situation is shown in Part (b) of Figure 3.10. The boundary between the first (A1) and second (A2) syllables of the actual form do not map to any syllable in the target form. Given the rule stated above, it is safe to propagate this boundary.

(a)

Alignment

Target: [##][t# rit]

Actual: [sə][tə][rit]

Mapping

T1 → A2

T1 → A3

(b)

Alignment

Target: [##][##][stræp]

Actual: [sa][sa][strap]

Mapping

T1 →A3

Figure 3.10 An Example of Boundary Propagation

3.2 Double-Level Dynamic Programming

The double-level dynamic programming algorithm was developed both for comparison with the single-level algorithm and as an alternative for overcoming unforeseen difficulties in reconstructing syllables after alignment in the single-level algorithm. The double-level approach is a simple extension of the single-level algorithm in that the double-level algorithm deals only with syllables and delegates the alignment of the phones within those syllables to the single-level algorithm.

The Pseudocode for the double-level algorithm in Figure 3.11 is very similar to the algorithm for the single-level approach found in Figure 2.3. However, some of the variables and parameters have changed. In the algorithm, *g* is the cost of inserting or deleting a symbol, the matrix *a[,]* is used to store the values of the solved subproblems, *target[x]* and *actual[y]* are the syllables (not the phones as in the single-level algorithm) at positions *x* and *y* in the target and actual forms respectively, and *similarity(target[x],actual[y])* is the function that calculates the similarity of two syllables, which is actually the single-level algorithm derived in Section 3.1.

```
Algorithm Similarity
    input: sequences target and actual
    output: similarity between target and actual
    m ← |target| //length of the target utterance
    n ← |actual| //length of the actual utterance
    for i ← 0 to m do
        a[i,0] ← i * g
    for j ← 0 to n do
        a[0,j] ← j * g
    for i ← 1 to m do
        for j ← 1 to n do
            a[i,j] ← max(   a[i-1,j] + g * |target[i]|, //an indel in the target
                            a[i-j,j-1] + similarity(target[i],actual[j]),
                            a[i,j-1] + g * |actual[j]|  //an indel in the actual
                        )
```

Figure 3.11 Pseudocode for the Double-level Alignment Algorithm

The double-level algorithm functions in the same manner as the single-level algorithm with two exceptions. First, the algorithm delegates the calculation of the substitution cost of two syllables to the single-level algorithm. The single-level algorithm simply treats the syllables as two occurrences of one syllable and returns the best similarity score for them. Second, the double-level algorithm does not have a simple indel value but calculates one based on the length of the syllable being skipped.

This approach has the advantage of always maintaining syllable integrity, which has to be reconstructed in the single-level algorithm. The double-level algorithm provides this advantage with no additional cost in time or space complexity. However this approach has two main shortcomings.

First, the double-level algorithm is more complex than the single-level alternative. The cost of an indel is no longer a static value but a function of the length of the syllable being inserted or deleted. Also, the similarity function is a comparison of entire syllables, which requires an additional level of structure to

calculate. This is far more complex than the approach used by the single-level algorithm of phone-to-phone comparisons and involves many more parameters.

Second, because the algorithm aligns syllables with syllables, it is not able to align more than one syllable with another. This means that the double-level algorithm is not capable of handling a one-to-many mapping and, as such, cannot handle cases of syllable epenthesis or truncation (see Section 2.2).

The alignment shown in Figure 3.12 displays the issue that arises with the double-level algorithm encounters a one-to-many syllable mapping. In this example, the first syllable of the actual form aligns with the entire target form. However, in the correct alignment, all three syllables in the actual form align with the first syllable of the target form.

Incorrect Alignment		Correct Alignment	
Target:	[bruwz][##][###]	Target:	[b# r# #uwz]
Actual:	[bə][rə][wuz]	Actual:	[bə][rə][wu#z]

Figure 3.12 Examples of how Epenthesis is Handled by the Double-level Algorithm

While the algorithm given in Figure 3.11 is not able to handle the misalignment caused by epenthesis and truncation, it can be modified to handle these situations. By modifying the recurrence to not only check the similarity of the current pair of syllables, but to also check the similarity of all possible combinations of previous syllables in both forms, the algorithm can determine if a combination of syllables provides a better similarity score. In Figure 3.12, the combination of all three syllables in the actual form would provide a higher similarity score, when compared to the single syllable in the target form, than any of these syllables could alone. However, if this condition were introduced into the algorithm, the result, in essence, would be an exhaustive search. Not only does this violate our principle of using the simplest method, but it also affects the speed and efficiency of the algorithm.

Chapter 4 Implementation and Testing

4.1 Implementation

The alignment algorithm is implemented in Java 1.4.2, the same language used in the development of Phon. The entire structure of the implemented algorithm can be seen in the class diagram in Figure 4.1. The diagram is not explained and is only provided to give a general idea of how the algorithm was implemented. It is important to note in the diagram that the alignment algorithm shares the same data structure as Phon and relies on the Phon Syllabifier to provide the syllable boundaries used in alignment.

Up to this point we have simply talked about creating syllables from utterances and phones from syllables. However, the algorithms used to perform these task are nontrivial, especially the syllabification algorithm which is very detailed and robust. Syllabification is described in detail in Hedlund and O'Brien (2004). The Syllabifier functions by taking, as input, a string of symbols in IPA format and returns an array of syllable Objects. For the purpose of explaining alignment, the inner workings of the Syllabifier need not be given in detail. The syllable Object not only provides access to the individual syllable constituents, but also indicates whether or not the current syllable has a primary stress or a secondary stress. Syllable constituents, meanwhile, contain the collection of feature sets for the individual symbols.

Since the single-level alignment algorithm works on the phone level, a new phone Object was created (see Figure 4.1). This object simply keeps a reference to the syllable and the syllable constituent to which the phone belongs as well as the position of that phone in the constituent. By doing so, the phone Object is able to access all the information for both the syllable and the syllable constituent without having to duplicate this information.

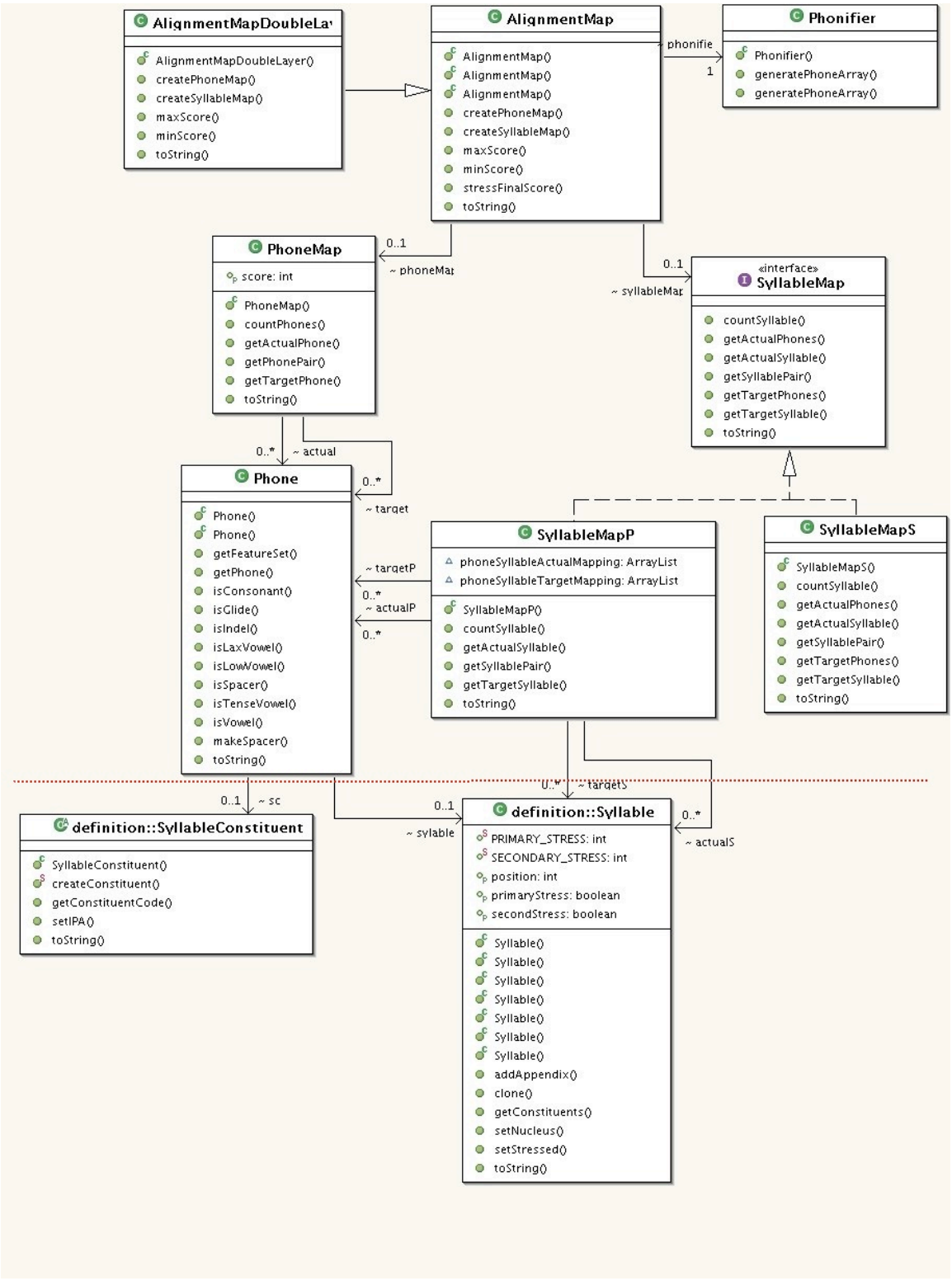

Figure 4.1 Class Diagram for the Alignment Algorithm Implementation

Classes below the dotted line are those implemented in Phon (see Hedlund and O'Brien (2004)).

4.2 Testing

The alignment algorithm was tested against a corpus of 168 utterance pairs drawn from existing corpora of English linguistic data (see Appendix I). The resulting alignments were then automatically checked against a corpus of correct alignments created by a trained linguist (Dr. Yvan Rose). Incorrect alignments were written into a separate file containing both the generated result and the expected result. The incorrect alignments were also broken down by correctness of segmental and/or syllabic alignment so the algorithms performance on both could be assessed independently.

4.2.1 Single-Level vs. Double-Level Results

In Figure 4.2, the percentage of syllabic alignments that match the predefined correct results are graphed. The *y*-axis shows the percentage correct and the *x*-axis shows the various mechanisms as they were added to the algorithm. Both algorithms start with a result over 70%. It is interesting to note that the double-level algorithm initially performs better than the basic single-level algorithm. However, once the primary stress reward is applied to the basic algorithm, the accuracy of the single-level approach surpasses that of the double-level and as more and more mechanisms are added to the algorithm, the gap continues to widen. With all the proposed mechanism in place the single-level algorithm is over 95% accurate while the double-level approach remains constant at 75%.

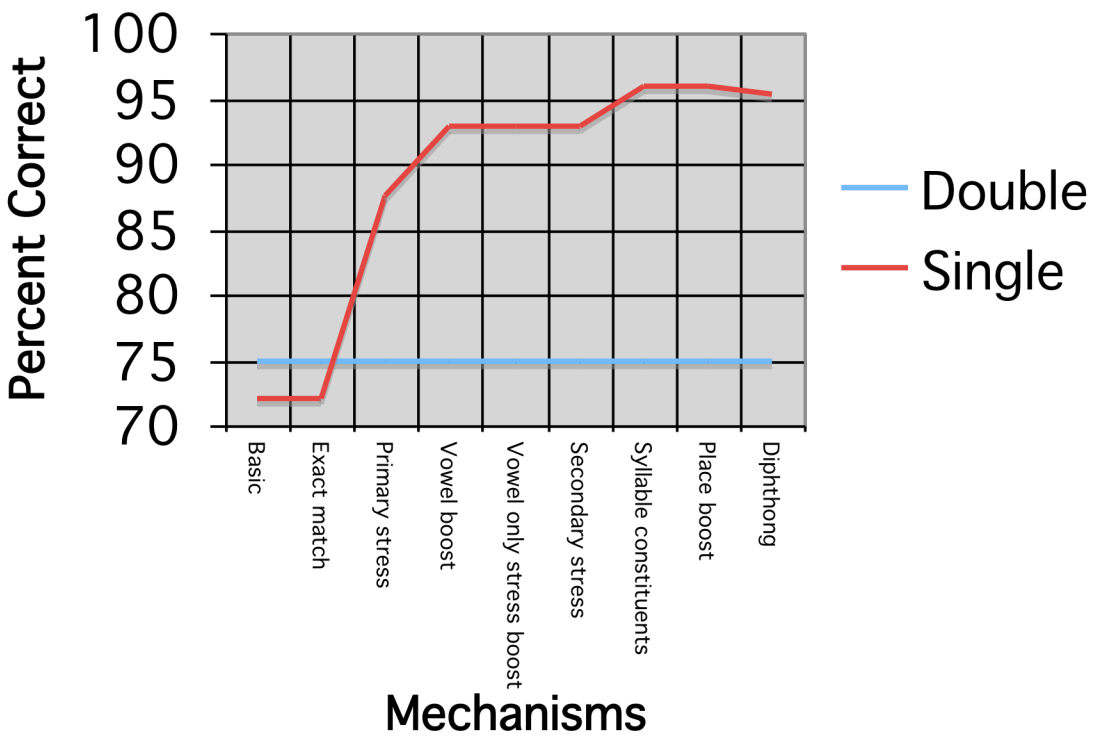

Figure 4.2 A Comparison of the Correct Alignments Produced by the Single-Level and Double-Level Algorithms

From this graph two conclusions can be made:

1. In the absence of additional mechanisms, the double-level algorithm performs better than the single-level algorithm. However, with all mechanisms in place, the single-level algorithm performs over twenty percent better than the double-level algorithm.
2. Second, the double-level algorithm does not improve in performance. The double-level algorithm is very accurate when only a simple feature comparison is used to determine similarity; however, when a one-to-many mapping is needed, it falls short. Because of this, the double-level alignment does not improve beyond its initial success.

4.2.2 Single-Level Results

In the previous section, we showed how the single-level algorithm produces more accurate results than the double-level algorithm. Because the single-level algorithm can be applied to both segmental and syllabic alignment, it is necessary to analyze the results of the algorithm for both types of alignment. It is also necessary to analyze not only the overall performance of the algorithm, but also the performance improvements produced by each individual reward and penalty mechanism.

The results for segmental and syllabic alignment from the single-level algorithm can be seen in Figure 4.3. Again, the *y*-axis shows the percentage of correct alignments and the *x*-axis shows the mechanisms in the order in which they were added to the algorithm. As in Figure 4.2, both segmental and syllabic alignments for the single-level algorithm are over 70% correct for the most basic form of alignment, with the segmental alignment algorithm actually producing results of over 80% for the basic approach. Each alignment improves with the addition of mechanisms. There are, however, exceptions to this statement that need to be discussed.

As more mechanisms are added, the percent of alignments that are correct also increases. The notable exception is the Diphthong penalty. This reward is very complicated and is still a work in progress. With additional development, it is hoped that this reward will produce a positive increase in matches. It will be more evident later how the Diphthong penalty actually affects results when we do a breakdown of alignments by example type (see Table 4.2); for the moment, it is sufficient to note that there is a slight drop in performance when this reward is added.

Two further observations can be drawn from this graph. The first is that the mechanisms specific to either segmental or syllabic alignment affect only those specific types of alignment (see Section 3.1.1). For example, the Vowel Only Stress Boost affects only segmental alignment, as can be verified by inspecting

the graph.

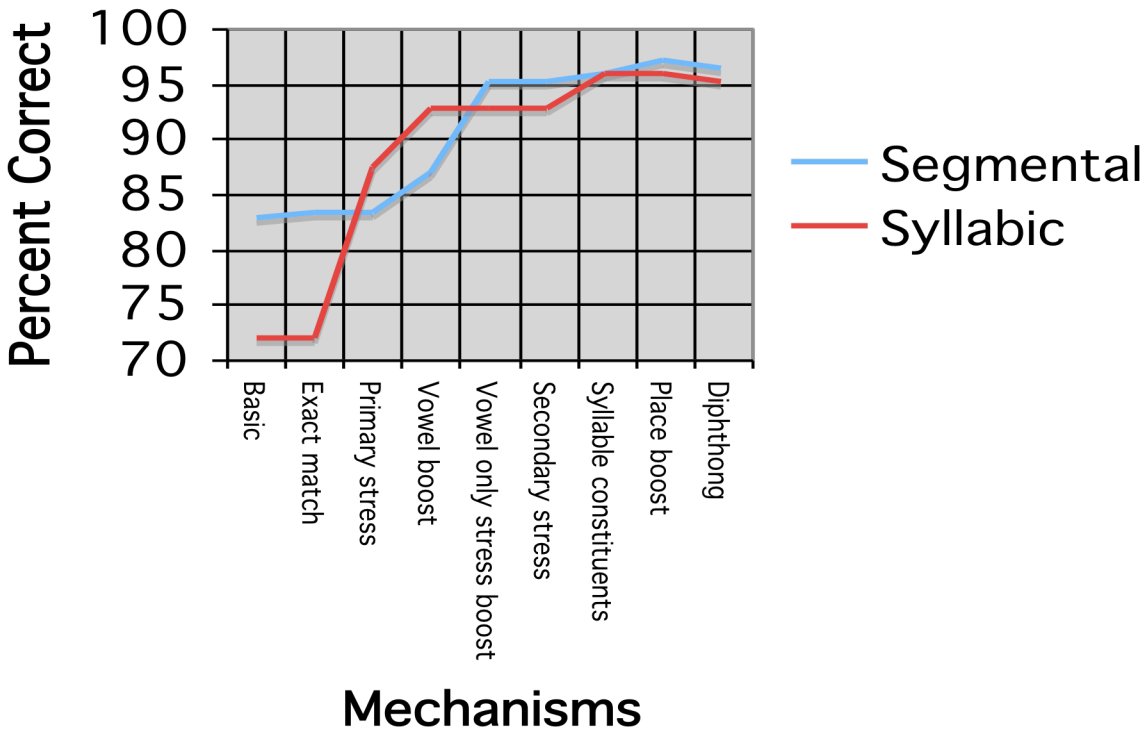

Figure 4.3 A Comparison of the Correct Segmental and Syllabic Alignment for the Single-level Algorithm

The second observation is that the Secondary Stress reward does not produce any change in the number of correct results. This reward was introduced to deal with situations where the actual form had translated a secondary stress into a primary stress; however, it does not appear to produce an increase in the number of correct alignments. Two possible reasons for this are that either the reward value may not be optimized to encourage such an alignment result or there may be a subset of different mechanisms that are producing the same effect as the Secondary Stress reward. Future investigation in this area is outlined in Section 5.2.

While Figure 4.2 provides a good comparison between segmental and syllabic alignment results, it is helpful to take a closer look at the actual values

used in graphing the figures. Tables 4.1 and 4.2 show the number of incorrect alignments for a given utterance pair type at each stage of mechanism addition broken down by example type. All of the examples in the test corpus were categorized into types based upon the hypothesized process linking the target and actual forms. The abbreviations used in the table are defined below:

- **1SF** (First Syllable Fusion): The first syllable of an utterance has been fused with an adjacent syllable such that sounds from both syllables are found in the initial syllable of the actual form, *e.g.,*

  Target Syllables: [pə][teɪ][ɾə]
  Actual Syllables: [pa][tə]

- **2SF** (Second Syllable Fusion): The second syllable of an utterance has been fused with an adjacent syllable such that sounds from both syllables are found in the initial syllable of the actual form, *e.g.,*

  Target Syllables: [ɪ][tə][reɪ][ʃən]
  Actual Syllables: [tə][ʃə]

- **IT** (Initial Truncation): The initial syllable of an utterance is deleted, *e.g.,*

  Target Syllables: [pə][teɪ][ɾə]
  Actual Syllables: [ta][to]

- **FT** (Final Truncation): The final syllable of an utterance is deleted, *e.g.,*

  Target Syllables: [ə][nʌ][ðəɹ]
  Actual Syllables: [ə][nə]

- **Str Fin** (Stressed and Final retained): The stressed syllable and the final syllable are all that is retained in the actual form, *e.g.,*

  Target Syllables: [hɛ][lɪ][kɑp][tər]
  Actual Syllables: [ha][pa]

- **VE** (Vowel Epenthesis): A vowel is inserted into an utterance thereby creating an extra syllable, *e.g.,*

  Target Syllables: [bruwz]
  Actual Syllables: [bə][rə][wuz]

- **CH** (Consonant Harmony): Two consonants in the actual form share a place (e.g. [labial]) or a manner (*e.g.* [voiced]) feature, *e.g.,*

  Target Syllables: [gɑr][bədʒ][bin]
  Actual Syllables: [ba][bi]

- **Other**: Rarely occurring processes or the combination of several processes.

There are three interesting points in Table 4.1 that are worth discussing. First, as with Figure 4.2 mechanisms that are added to deal with syllabic alignment, such as the Primary Stress reward, do not alter the segmental alignment results. Second, mechanisms such as the Vowel Only Stress Boost result in dramatic decreases in incorrect results overall and even eliminate all the incorrect results for a given type. In this case, incorrect examples of types *IT & FT* and *Str Fin* are reduced to zero when this reward is added. Third, the Diphthong penalty, while correcting one error in the *Other* category, breaks two more alignments causing a slight increase in incorrect results.

	Basic	Exact match	Primary stress	Vowel boost	Vowel only stress boost	Secondary stress	Syllable constituents	Place boost	Diphthong
All	29	28	28	22	8	8	7	5	6
1SF & 2SF	8	7	7	7	3	3	3	1	1
IT & FT	10	13	13	8	0	0	0	0	2
Str Fin	3	3	3	3	0	0	0	0	0
VE	1	0	0	1	2	2	1	1	1
CH	1	1	1	1	1	1	1	1	1
Other	6	4	4	2	2	2	2	2	1

Table 4.1 Incorrect Alignment Results by Category Type (Single-level / Segmental)

The same three points also arise in relation to a similar breakdown for syllabic alignment (see Table 4.2). In addition, the observation made earlier about the Secondary Stress reward is reinforced by the data in the table; when this reward is added, the total number of incorrect results remains constant at 12. The most significant reward for syllabic alignment is the Primary Stress reward that reduces the number of incorrect results by over half. This is consistent with the assumption made earlier that stress was an important indicator for alignment.

	Basic	Exact match	Primary stress	Vowel boost	Vowel only stress boost	Secondary stress	Syllable constituents	Place boost	Diphthong
All	47	47	21	12	12	12	7	7	8
1SF & 2SF	25	25	3	3	3	3	1	1	1
IT & FT	10	10	5	0	0	0	0	0	2
Str Fin	3	3	1	1	1	1	0	0	0
VE	2	2	2	1	1	1	1	1	1
CH	2	2	1	1	1	1	0	0	0
Other	5	5	9	6	6	6	5	5	4

Table 4.2 Incorrect Alignment Results by Category Type (Single-level / Syllabic)

In total, out of 168 segmental alignments the final number of incorrect alignments was 6, for a final accuracy of 96%. The total number of incorrect alignments out of 168 syllabic alignments was 8, which gave an accuracy of 95%. It is important to note that, with all mechanisms in place, no category of examples remains unaffected or retains a high percentage of incorrect alignments.

Chapter 5 Conclusions and Future Work

5.1 Conclusions

In the Introduction, five goals were set for the development of the alignment algorithm:

- The algorithm should be user-configurable;
- The algorithm should be based on the simplest mechanisms possible;
- The algorithm is allowed to generate incorrect results;
- The algorithm should be designed to produce the highest percentage of accurate alignments; and
- The algorithm should be programmed with the most efficient algorithm possible.

In this section, we will evaluate how well these goals were met.

The current design of the alignment algorithm allows the user to alter all reward and penalty values. In addition, the user is also able to activate or deactivate any mechanism they choose. This flexibility allows researchers to produce custom alignments.

The current algorithm uses only the mechanisms that were introduced to deal with situations that occurred during development. Mechanisms designed to handle specific situations or rarely occurring abnormalities were not introduced. This produced an algorithm that is simpler and more general-purpose than it would have been if additional mechanisms had been introduced. Also, the algorithm uses only one level of alignment and deals only with phones, which are simpler units then syllables. In these ways the algorithm held firm to the goal of maintaining simplicity.

While the final rate of correct alignments was high, *i.e.* over 95% of actual-target pairs are correct for both segmental and syllabic alignment, the algorithm

was still allowed a five percent miss rate. Since our initial goal stated that manual correction was possible, the algorithm did not need to obtain 100% correctness. However, the high match rate achieved met out goal of a high rate of accuracy, which minimizes the number of alignments that have to be corrected manually.

Finally, the aligner uses a dynamic programming algorithm, which results in a fast and efficient algorithm. The current implementation will align the given test corpus of 168 examples in 2.7 seconds for segmental, syllabic, and double-level alignment combined. This time was obtained by running the algorithm on a 1.5GHz PowerPC processor with 768MB RAM. Note that approximately one second of that time is the result of file access and is constant for any size corpus. With these figures, on a similar machine, an entire 20,000 utterance-pair corpus (the size of the largest publicly available corpus) would take approximately two and a half minutes. This is a reasonable time for such a large dataset.

The algorithm developed herein has met the five goals initially set out. While there are still many areas of research and improvement to be explored (see Section 5.2), the alignment algorithm produces acceptable results and can be used to complete the tasks for which it was designed.

5.2 Future Work

Potential additional research can be summed up in five areas:

- Testing the alignment algorithm on non-English corpora;
- Optimizing reward and penalty values;
- Finding the minimum subset of required mechanisms;
- Determining the minimum test corpus size; and
- Generating an alignment correctness ranking.

Each area will provide useful benefits in flexibility, accuracy, and simplicity, and

will contribute to an increase in the overall effectiveness and usefulness of the aligner.

Non-English Corpora: Currently, the alignment algorithm has only been tested with English utterance pairs. It is not known how well the aligner will function on examples outside of the English language. More research needs to be done in this area.

This first part of this research will consist of obtaining utterances from different languages and attempting to align the target and actual forms. This could initially be done on a language closely related to English, such as Dutch. Depending on the results of this alignment, the algorithm would be tested on data from languages that differ significantly from English. One such language is French, which differs from English both in terms of its phone inventory and with regards to its stress system.

Optimized reward values: If the aligner works on multiple language sets, then we can conclude that the principles governing syllable truncation and syllable epenthesis in language acquisition apply similarly across languages. However, if the aligner does not work on different language sets, then we will need to develop a method to quickly and easily generate values that result in alignments for any given language.

Provided with a large enough corpus of sample alignments, an automatic reward value calculation algorithm could generate optimal values for rewards and penalties. Automatic reward value calculation would provide several advantages to the system. First, it would allow the aligner to optimize the set of values for rewards and penalties. The current reward values are not optimized; they merely reflect values that appear to work in practice based on our test corpus. Second, it would allow the aligner to adapt the set of values for a wide variety of languages.

The reward values currently work for English utterances, but as mentioned above, it is not yet known how well these will hold for different language sets.

With an algorithm that can automatically calculate optimal reward values, adding a new language to the system would only require a corpus of sample alignments to be provided by a researcher. Once this corpus is in place, the system would be able to calculate the optimal reward values.

An algorithm designed for this task could draw on a variety of optimization techniques. For example, such an algorithm could take the form of a genetic algorithm, in which each member of the population would be a collection of reward values with different integer assignments and the fitness of each individual would be based upon the number of correct segmental and syllabic alignments it produced. Such an approach should produce the most fit individual over time, hence the best set of reward values for the alignment algorithm.

Minimum subset of required mechanisms: In addition to finding an optimized set of reward values, it is important to find the optimal subset of alignment mechanisms. This would involve investigating whether or not all the current mechanisms are needed to produce an acceptable rate of correct alignments. The advantage of this research is that, by reducing the number of mechanisms, we reduce the complexity of the algorithms. This is in line with our philosophy of developing the simplest algorithm. Also, if we discover that the aligner does not work for all languages, we can use the same algorithm to find what mechanisms are needed for different languages.

As mentioned in Section 4.2.2, the Secondary Stress reward does not contribute to the overall accuracy of the alignment and may not be necessary. This research would try to find the most complete subset of the current mechanisms that provide the optimal, or an acceptable, alignment score. This task can be completed via an exhaustive search. While not the optimal approach, for a small set of possible combinations, the running time may be acceptable. For example, with the current set of mechanisms, the total possible number of subsets of these mechanisms is 2^8 or 256 different mechanism combinations. Such a search would run in just over ten minutes given the current configuration.

Minimum test corpus size: The other important area of research with respect to the functionality of the aligner is the minimum size of the required test corpus. If there is a need to train the aligner for any new language, a researcher would need to know what the minimum test corpus size should be. This would require testing the aligner on smaller and smaller subsets of the test corpus and discovering how many examples are needed in total as well as how many examples of each type of utterance mutation are needed. This is probably the most difficult of the three optimization areas discussed here. However, it is important because correct alignments in many languages are difficult to obtain due to a lack of available phonological data.

Alignment Rankings: While the goal of the project is to provide the highest possible alignment rate, it is naive to think that we will ever achieve one hundred percent accuracy. We also stated in our initial philosophies that our algorithm did not need to be perfect and that there was always the option of manual correction. This presents us with a problem, if the aligner is generating results that are not correct, then we need a method for determining which results are correct and which are not.

This can be achieved in one of two ways. The first is to flag any results that the aligner does not believe to be correct. The second is to give each alignment a ranking that represents how accurate the alignment is, with lower rankings indicating less likely alignments and higher rankings indicating more likely alignments. Both approaches can provide users with a mechanism for determining which alignments are possibly incorrect. A ranking is more versatile than a simple flag, so it would most likely be the approach investigated.

To generate a ranking, the aligner would need certain criteria by which to determine the correctness of the alignment. Since the alignment algorithm always produces the best alignment for any pair of utterances provided, relative to the similarity scoring function, it is difficult to have to have the algorithm comment on

the correctness of that alignment. For example, if the aligner were given two unrelated utterances such as [kaʊ] and [mu], it would find their optimal alignment. However, since the two utterances being aligned are different words, the resulting alignment cannot be correct. The optimal alignment, therefore, is not always the correct alignment.

A ranking could be as simple as comparing the resulting similarity score with the maximum possible similarity score produced by aligning the target form with itself. However, the ranking could be more complex, involving the programmer hard coding characteristics that the algorithm would check for. These characteristics would be areas in which the aligner repeatedly has difficulties. Such characteristics would be generated from repeated execution of the aligner on different corpora and post hoc analysis of the results.

References

Cormen, T., Leiserson, C., and Rivest, R. 1997. *Introduction to Algorithms*. MIT Press: Cambridge, MA.

Covington, M. A. 1996. "An Algorithm to Align Words for Historic Comparison". *Computational Linguistics,* 22(4), 481-496

Hedlund G. and O'Brien, P. 2004. *A Software System for Linguistic Data Capture and Analysis*. B.Sc.h. dissertation, Department of Computer Science, Memorial University of Newfoundland.

Kondrak, G. 2002. *Algorithms for Language Reconstruction*. Ph.D. thesis, Department of Computer Science, University of Toronto.

Kondrak, G. 2003. "Phonetic alignment and similarity". *Computers and the Humanities*, 37(3), 273-291.

Rose, Y. 2003. "*ChildPhon*: A Database Solution for the Study of Child Phonology". In B. Beachely, A. Brown, and F. Conlin (eds) *Proceedings of the 27th Annual Boston University Conference on Language Development*. Cascadilla Press: Somerville, MA. 674 - 685

Rose, Y., Byrne, R., Hedlund, G., O'Brien, P., and Wareham, T. 2005. "Phon: A Tool for Transcribing and Analyzing Phonological Corpora". Manuscript

Setubal, J. and Meidanis, J. 1997. *Introduction to Computation Molecular Biology*. Brooks/Cole Publishing Company: Pacific Grove, CA.

Somers, H.L.1998. "Similarity metrics for aligning children's articulation data". In: *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*. 1227-1231.

Appendix I: English Language Dataset

This Appendix consists of 168 adult / child utterance pairs. The table contains the English utterance, the target (adult) and actual (child) IPA transcriptions and syllabifications, the correct alignment, and the category of the alignment situation (see Section 4.2.2). The type abbreviations are defined below:

Types Used in Tables 4.1 and 4.2

1SF – First Syllable Fusion

2SF – Second Syllable Fusion

IT – Initial Truncations

FT – Final Truncation

SF – Stressed and Final Retained

CH – Consonant Harmony

Categorized as *Other* in Tables 4.1 and 4.2

GV – Glide Vocalization

SS – Stress Shift

OF – Onset Fusion

2VE – Second Vowel Epenthesis

Number	*Utterance*	*Target Syllables*	*Actual Syllables*	*Alignment*	*Type*
1	Hippopotamus	[hɪ][pə][pʌ][ɾə][məs]	[po][mus]	[hɪ][pə][pʌ][ɾə][məs] [##][##][po][##][mus]	SF
2	potato	[pə][teɪ][ɾə]	[pa][tə]	[pə][teɪ][ɾə] [##][pa#][tə]	1SF
3	potato	[pə][teɪ][ɾə]	[ta][to]	[pə][teɪ][ɾə] [##][ta#][to]	IT
4	blue	[blu]	[bə][lu]	[b# lu] [bə][lu]	VE
5	smoke	[smok]	[fok]	[smok] [f#ok]	OF
6	strap	[stɹæp]	[sa][sa][strap]	[##][##][stɹæp] [sa][sa][strap]	VE
7	street	[strit]	[sə][tə][rit]	[s# t# rit] [sə][tə][rit]	VE
8	strength	[strɛŋθ]	[si][ta][rent]	[s# t# rɛŋθ] [si][ta][rent]	VE
9	elephant	[ɛ][lə][fənt]	[i][fant]	[ɛ][lə][fənt] [i][##][fant]	SF

Number	*Utterance*	*Target Syllables*	*Actual Syllables*	*Alignment*	*Type*
10	helicopter	[hɛ][lɪ][kɑp][tər]	[a][ka]	[hɛ][lɪ][kɑp][tər] [#a][##][ka#][###]	SF
11	helicopter	[hɛ][lɪ][kɑp][tər]	[ta][ta]	[hɛ][lɪ][kɑp][tər] [##][##][ta#][ta#]	Other
12	helicopter	[hɛ][lɪ][kɑp][tər]	[ha][ta]	[hɛ][lɪ][kɑp][tər] [ha][##][###][ta#]	SF
13	helicopter	[hɛ][lɪ][kɑp][tər]	[ha][pa]	[hɛ][lɪ][kɑp][tər] [ha][##][pa#][###]	SF
14	bruise	[bruwz]	[bə][rə][wuz]	[b# r# #uwz] [bə][rə][wu#z]	VE
15	iteration	[ɪ][tə][reɪ][ʃən]	[tə][ʃə]	[ɪ][tə][reɪ][ʃən] [#][##][tə#][ʃə#]	2SF
16	garbage bin	[gɑr][bəʤ][bin]	[ba][bi]	[gɑr][bəʤ][bin] [ba#][###][bi#]	CH
17	baloney	[bə][lo][nɪ]	[bo][nɪ]	[bə][lo][nɪ] [##][bo][nɪ]	1SF
18	intrigued	[ɪn][trigd]	[tig]	[ɪn][trigd] [##][t#ig#]	IT
19	intrigued	[ɪn][trigd]	[i][ti][gid]	[ɪn][tri][g#d] [i][t#i][gid]	VE
20	balloon	[bə][lun]	[ba][bun]	[bə][lun] [ba][bun]	CH + SS
21	giraffe	[ʤə][ræf]	[ʤə][ræ]	[ʤə][ræf] [ʤə][ræ#]	SS
22	Arizona	[æ][rə][zoʊ][nə]	[æ][rə][zoʊ][nə]	[æ][rə][zoʊ][nə] [æ][rə][zoʊ][nə]	SS
23	cow	[kaʊ]	[mu]	[kaʊ] [mu#]	Other
24	pee	[pi]	[lu]	[pi] [lu]	Other
25	sheep	[ʃip]	[bɛ]	[ʃip] [bɛ#]	Other
26	another	[ə][nʌ][ðəɹ]	[nʌ][dɚ]	[ə][nʌ][ðəɹ] [#][nʌ][dɚ#]	IT
27	another	[ə][nʌ][ðəɹ]	[jə][wo]	[ə][nʌ][ðəɹ] [#][jə][wo#]	IT
28	another	[ə][nʌ][ðəɹ]	[jə]	[ə][nʌ][ðəɹ] [#][jə][###]	IT + FT
29	another	[ə][nʌ][ðəɹ]	[ə][nə]	[ə][nʌ][ðəɹ] [ə][nə][###]	FT
30	another	[ə][nʌ][ðəɹ]	[əʊ]	[ə][nʌ][ðəɹ] [ə #ʊ][###]	Other
31	another	[ə][nʌ][ðəɹ]	[nʌ][dɚ]	[ə][nʌ][ðəɹ] [#][nʌ][dɚ#]	Other

Number	*Utterance*	*Target Syllables*	*Actual Syllables*	*Alignment*	*Type*
32	another	[ə][nʌ][ðəɹ]	[ʌ][ə]	[ə][nʌ][ðəɹ] [#][#ʌ][#ə#]	Other
33	another	[ə][nʌ][ðəɹ]	[a][nʌ]	[ə][nʌ][ðəɹ] [a][nʌ][###]	FT
34	another	[ə][nʌ][ðəɹ]	[nʌ][θɜ˞]	[ə][nʌ][ðəɹ] [#][nʌ][θɜ#]	IT
35	another	[ə][nʌ][ðəɹ]	[nʌ][ɚ˞]	[ə][nʌ][ðəɹ] [#][nʌ][#ɚ#]	IT
36	another	[ə][nʌ][ðəɹ]	[nʌ][dɚ˞]	[ə][nʌ][ðəɹ] [#][nʌ][dɚ#]	IT
37	another	[ə][nʌ][ðəɹ]	[ə][nɚ˞]	[ə][nʌ][ðəɹ] [ə][nɚ˞][###]	FT
38	banana	[bə][næ][nə]	[næ][nə]	[bə][næ][nə] [##][næ][nə]	IT
39	banana	[bə][næ][nə]	[mæ][nə]	[bə][næ][nə] [##][mæ][nə]	1SF + CH
40	banana	[bə][næ][nə]	[bæ][nə]	[bə][næ][nə] [##][bæ][nə]	1SF
41	banana	[bə][næ][nə]	[blæ][na]	[bə][næ][nə] [b# læ][na]	Other
42	banana	[bə][næ][nə]	[næ][næ]	[bə][næ][nə] [##][næ][næ]	IT
43	gorilla	[gə][ɹɪ][ɬə]	[graʊ][wə]	[gə][ɹɪ#][ɬə] [g# raʊ][wə]	1SF
44	gorilla	[gə][ɹɪ][ɬə]	[go][wæ]	[gə][ɹɪ][ɬə] [##][go][wæ]	1SF
45	gorilla	[gə][ɹɪ][ɬə]	[wʌ][ga]	[gə][ɹɪ][ɬə] [##][wʌ][ga]	IT
46	gorilla	[gə][ɹɪ][ɬə]	[gʌ][wa]	[gə][ɹɪ][ɬə] [##][gʌ][wa]	1SF
47	gorilla	[gə][ɹɪ][ɬə]	[wi][lə]	[gə][ɹɪ][ɬə] [##][wi][lə]	IT
48	modesto	[mɑ][dɛs][tow]	[dɛs][to]	[mɑ][dɛs][tow] [##][dɛs][to#]	IT
49	Nathaniel	[nə][θæ][njəɬ]	[fæ][fu][e]	[nə][θæ][nj əɬ] [##][fæ][fu][e#]	IT + CH
50	Nathaniel	[nə][θæ][njəɬ]	[fæ][ŋo]	[nə][θæ][njəɬ] [##][fæ][ŋ#o#]	IT
51	Nathaniel	[nə][θæ][njəɬ]	[fæ][ŋos]	[nə][θæ][njəɬ] [##][fæ][ŋ#os]	IT
52	piano	[pi][jæ][nʊ]	[pæ][no]	[pi][jæ][nʊ] [p# #æ][no]	1SF
53	piano	[pi][jæ][now]	[pæ][no]	[pi][jæ][now] [p# #æ][no#]	SF

Number	*Utterance*	*Target Syllables*	*Actual Syllables*	*Alignment*	*Type*
54	remember	[rə][mɛm][bəɹ]	[mɛ][mə]	[rə][mɛ m][bəɹ] [##][mɛ][m #ə#]	IT + CH
55	remember	[rə][mɛm][bəɹ]	[mɛm][bə]	[rə][mɛm][bəɹ] [##][mɛm][bə#]	IT
56	spaghetti	[pə][gɛ][ɾij]	[dɪ][bi]	[pə][gɛ][ɾij] [##][dɪ][bi#]	IT
57	spaghetti	[pə][gɛ][ɾij]	[gɛ][bi]	[pə][gɛ][ɾij] [##][gɛ][bi#]	IT
58	spaghetti	[pə][gɛ][ɾij]	[kɛ][bi]	[pə][gɛ][ɾij] [##][kɛ][bi#]	IT
59	spaghetti	[pə][gɛ][ɾij]	[ke][ti]	[pə][gɛ][ɾij] [##][ke][ti#]	IT
60	spaghetti	[pə][gɛ][ɾij]	[gɛ][di]	[pə][gɛ][ɾij] [##][gɛ][di#]	IT
61	apartment	[ə][paɹt][mənt]	[part][mɛnt]	[ə][paɹt][mənt] [#][part][mɛnt]	IT
62	baloney	[bə][lo][nɪ]	[bwo][ni]	[bə][lo][nɪ] [b# wo][ni]	1SF
63	delicious	[də][lɪ][ʃəs]	[dɪ][ʃəs]	[də][lɪ][ʃəs] [##][dɪ][ʃəs]	1SF
64	eleven	[ə][lɛ][vən]	[dɛ][bən]	[ə][lɛ][vən] [#][dɛ][bən]	IT + CH
65	eleven	[ə][lɛ][vən]	[jɛ][bən]	[ə][lɛ][vən] [#][jɛ][bən]	IT
66	eleven	[ə][lɛ][vən]	[jɛ][mɪn]	[ə][lɛ][vən] [#][jɛ][mɪn]	IT + CH
67	eleven	[ə][lɛ][vən]	[jɛ][vən]	[ə][lɛ][vən] [#][jɛ][vən]	IT
68	maracas	[mə][rɑ][kəs]	[ma][kas]	[mə][rɑ][kəs] [##][ma][kas]	1SF
69	pajamas	[pə][ʤæ][məs]	[da][məs]	[pə][ʤæ][məs] [##][da][məs]	IT
70	pajamas	[pə][ʤæ][məs]	[ʤæ][məʃ]	[pə][ʤæ][məs] [##][ʤæ][məʃ]	IT
71	pajamas	[pə][ʤæ][məs]	[da][məs]	[pə][ʤæ][məs] [##][da][məs]	IT
72	pajamas	[pə][ʤæ][məs]	[ʤa][mas]	[pə][ʤæ][məs] [##][ʤa][mas]	IT
73	potato	[pə][teɪ][ɾə]	[pe][do]	[pə][teɪ][ɾə] [##][pe#][do]	1SF
74	potato	[pə][teɪ][ɾə]	[te][to]	[pə][teɪ][ɾə] [##][te#][to]	IT
75	potato	[pə][teɪ][ɾə]	[te][to]	[pə][teɪ][ɾə] [##][te#][to]	IT

Number	*Utterance*	*Target Syllables*	*Actual Syllables*	*Alignment*	*Type*
76	salami	[sə][lɑ][mi]	[ma][mi]	[sə][lɑ][mi] [##][ma][mi]	IT + CH
77	Theresa	[tə][ɹij][sə]	[ri][sə]	[tə][ɹij][sə] [##][ri#][sə]	IT
78	together	[tə][gɛ][ðəɹ]	[gɛ][dəˠ]	[tə][gɛ][ðəɹ] [##][gɛ][dəˠ#]	IT
79	tomorrow	[tə][mɔ][ɹow]	[mo][wo]	[tə][mɔ][ɹow] [##][mo][wo#]	IT
80	tomorrow	[tə][mɔ][ɹow]	[mo][ro]	[tə][mɔ][ɹow] [##][mo][ro#]	IT
81	vagina	[və][ʤaj][nə]	[ʤa][i][nə]	[və][ʤa j][nə] [##][ʤa][i][nə]	IT
82	tomato	[tə][mej][ɾow]	[me][no]	[tə][mej][ɾow] [##][me#][no#]	IT
83	tomato	[tə][mej][tow]	[me][to]	[tə][mej][tow] [##][me#][to#]	IT
84	tomato	[tə][mej][tow]	[me][do]	[tə][mej][tow] [##][me#][do#]	IT
85	umbrella	[ʌm][bɹɛ][lə]	[bwa]	[ʌm][bɹɛ][lə] [##][bwa][##]	IT + FT
86	umbrella	[ʌm][bɹɛ][lə]	[bɛ][la]	[ʌm][bɹɛ][lə] [##][b#ɛ][la]	IT
87	umbrella	[ʌm][bɹɛ][lə]	[brʌ][gæ]	[ʌm][bɹɛ][lə] [##][brʌ][gæ]	IT
88	umbrella	[ʌm][bɹɛ][lə]	[bre][wa]	[ʌm][bɹɛ][lə] [##][bre][wa]	IT
89	umbrella	[ʌm][bɹɛ][lə]	[bʌ][wə]	[ʌm][bɹɛ][lə] [##][b#ʌ][wə]	IT
90	umbrella	[ʌm][bɹɛ][lə]	[bwɛ][wəz]	[ʌm][bɹɛ][lə#] [##][bwɛ][wəz]	IT
91	again	[ə][gɛn]	[gɛn]	[ə][gɛn] [#][gɛn]	IT
92	again	[ə][gɛn]	[gɛ]	[ə][gɛn] [#][gɛ#]	IT
93	apart	[ə][paɹt]	[part]	[ə][paɹt] [#][part]	IT
94	away	[ə][wej]	[we]	[ə][wej] [#][we#]	IT
95	away	[ə][wej]	[waɪ]	[ə][wej] [#][waɪ]	IT
96	behind	[bə][hajnd]	[ha][ind]	[bə][ha jnd] [##][ha][ind]	IT + GV
97	behind	[bə][hajnd]	[ha][in]	bə][ha jnd] [##][ha][in#]	IT + GV

Number	*Utterance*	*Target Syllables*	*Actual Syllables*	*Alignment*	*Type*
98	caboose	[kə][buws]	[gus]	[kə][buws] [##][gu#s]	1SF
99	caboose	[kə][bus]	[kuz]	[kə][bus] [##][kuz]	SF
100	denise	[də][nijs]	[dis]	[də][nijs] [##][di#s]	1SF
101	enough	[ə][nʌf]	[nʌf]	[ə][nʌf] [#][nʌf]	IT
102	alone	[ə][lown]	[won]	[ə][lown] [#][wo#n]	IT
103	alone	[ə][lown]	[i][on]	[ə][lown] [i][#o#n]	IT
104	around	[ə][rawnd]	[o][und]	[ə][ra wnd] [#][#o][und]	IT + GV
105	around	[ə][rawnd]	[wa][un]	[ə][ra wnd] [#][wa][un#]	IT + GV
106	balloon	[bə][lun]	[bu]	[bə][lun] [##][bu#]	SF
107	balloon	[bə][lun]	[bun]	[bə][lun] [##][bun]	SF
108	Balloon	[bə][lun]	[lun]	[bə][lun] [##][lun]	IT
109	balloon	[bə][lun]	[bʊn]	[bə][lun] [##][bʊn]	SF
110	balloon	[bə][lun]	[bʌ]	[bə][lun] [##][bʌ#]	SF
111	balloon	[bə][lun]	[bum]	[bə][lun] [##][bum]	1SF + CH
112	balloon	[bə][lun]	[bum]	[bə][lun] [##][bum]	1SF + CH
113	balloon	[bə][lun]	[bun]	[bə][lun] [##][bun]	SF
114	belong	[bə][lɑŋ]	[bɔŋ]	[bə][lɑŋ] [##][bɔŋ]	1SF
115	belong	[bə][lɑŋ]	[ɔŋ]	[bə][lɑŋ] [##][#ɔŋ]	IT
116	belong	[bə][lʌŋ]	[bʌŋ]	[bə][lʌŋ] [##][bʌŋ]	1SF
117	cement	[sə][mɛnt]	[mɛnt]	[sə][mɛnt] [##][mɛnt]	1SF
118	dessert	[də][zəɹt]	[zɚt]	[də][zəɹt] [##][zɚ#t]	Other
119	excuse me	[ɛk][skjuz][mi]	[ku][zə][mi]	[ɛk][skju z#][mi] [##][#k#u][zə][mi]	IT + VE

Number	*Utterance*	*Target Syllables*	*Actual Syllables*	*Alignment*	*Type*
120	garage	[gə][rɑʒ]	[gwaʤ]	[gə][rɑʒ] [g# waʤ]	1SF
121	garage	[gə][rɑʒ]	[gaʤ]	[gə][rɑʒ] [##][gaʤ]	1SF
122	garage	[gə][rɑʒ]	[garʤ]	[gə][rɑʒ] [ga r#ʤ]	Other
123	garage	[gə][rɑʒ]	[graʤ]	[gə][rɑʒ] [g# raʤ]	1SF + CL
124	garage	[gə][rɑʒ]	[graʤ]	[gə][rɑʒ] [g# raʤ]	1SF + CL
125	guitar	[gə][taɹ]	[tar]	[gə][taɹ] [##][tar]	IT
126	guitar	[gə][taɹ]	[gi]	[gə][taɹ] [##][gi#]	1SF
127	guitar	[gə][taɹ]	[ga]	[gə][taɹ] [##][ga#]	1SF
128	guitar	[gə][taɹ]	[gar]	[gə][taɹ] [##][gar]	1SF
129	Marie	[mə][ɹij]	[mi]	[mə][ɹij] [##][mi#]	1SF
130	Michele	[mə][ʃɛɬ]	[ʃɛ][u]	[mə][ʃɛ ɬ] [##][ʃɛ][u]	IT
131	pretend	[pɹə][tɛnd]	[tɛnd]	[pɹə][tɛnd] [###][tɛnd]	IT
132	today	[tə][dej]	[de]	[tə][dej] [##][de#]	IT
133	giraffe	[ʤə][ɹæf]	[ʤwæf]	[ʤə][ɹæf] [ʤ# wæf]	1SF
134	giraffe	[ʤə][ɹæf]	[dræf]	[ʤə][ɹæf] [d# ræf]	1SF
135	giraffe	[ʤə][ɹæf]	[dwæf]	[ʤə][ɹæf] [d# wæf]	1SF
136	giraffe	[ʤə][ɹæf]	[wæf]	[ʤə][ɹæf] [##][wæf]	IT
137	machine	[mə][ʃijn]	[ʃɪ][ʃɪm]	[mə][ʃijn] [ʃɪ][ʃɪ#m]	CH
138	machine	[mə][ʃijn]	[o][ʃin]	[mə][ʃijn] [#o][ʃi#n]	Other
139	machine	[mə][ʃijn]	[so][ə][ʃim]	m# ə][ʃijn] [so][ə][ʃi#m]	Other
140	Merced	[məɹ][sɛd]	[sɛd]	[məɹ][sɛd] [###][sɛd]	IT
141	police man	[pə][lijs][mæn]	[pis][mæn]	[pə#][lijs][mæn] [p## #i#s][mæn]	Other

Number	*Utterance*	*Target Syllables*	*Actual Syllables*	*Alignment*	*Type*
142	police	[pə][lijs]	[plis]	[pə][lijs] [p# li#s]	SF
143	surprise	[sə][pɹajz]	[pwaɪz]	[sə][pɹajz] [##][pwaɪz]	IT
144	gas	[gæs]	[gæ][si]	[gæ s#] [gæ][si]	VE
145	tent	[tɛnt]	[tɛn][tɛ]	[tɛn t#] [tɛn][tɛ]	VE
146	test	[tɛst]	[tɛ][si][ti]	[tɛ s# t#] [tɛ][si][ti]	2VE
147	contest	[kʌn][tɛst]	[ka][tɛ][si][ti]	[kʌn][tɛ s# t#] [ka#][tɛ][si][ti]	2VE
148	contest	[kʌn][tɛst]	[ka][tɛ][si][ti]	kʌn][tɛ s# t#] [ka#][tɛ][si][ti]	Other
149	stop	[tɑp]	[si][tap]	[##][tɑp] [si][tap]	VE
150	stop	[tɑp]	[si][tap]	[##][tɑp] [si][tap]	VE
151	straps	[tɹaps]	[si][ta][raps]	[##][t# ɹaps] [si][ta][raps]	VE
152	straps	[tɹaps]	[si][ta][ra][pis]	[##][t# ɹa p#s] [si][ta][ra][pis]	VE
153	straps	[tɹaps]	[si][ta][ra][pis]	[##][t# ɹa p#s] [si][ta][ra][pis]	VE
154	print	[pɹɪnt]	[pi][rɪn][ti]	[p# ɹɪn t#] [pi][rɪn][ti]	VE
155	print	[pɹɪnt]	[pi][rɪ][na][ti]	[p# ɹɪ n# t#] [pi][rɪ][na][ti]	Other
156	administer	[æd][mɪs][təɹ]	[a][du][mi][ni][si][ta]	[æ d#][mɪ ## s#][təɹ] [a][du][mi][ni][si][ta#]	Other
157	strike	[stɹaɪk]	[su][to][raɪ][ku]	[s# t# ɹaɪ k#] [su][to][raɪ][ku]	Other
158	gambler	[gæm][bləɹ]	[ga][ba][la]	gæ m][b ləɹ] [ga][b a][la#]	Other
159	fit	[fɪt]	[tɪf]	[fɪt] [tɪf]	Other
160	sit	[sɪt]	[tɪs]	[sɪt] [tɪs]	Other
161	couple	[kɑ][pəɬ]	[pɑ][kə]	[kɑ][pəɬ] [pɑ][kə#]	Other
162	doggie	[dɑ][gi]	[ga][gi]	[dɑ][gi] [ga][gi]	CH
163	doggie	[dɑ][gi]	[ga][gi]	[dɑ][gi] [ga][gi]	CH

Number	*Utterance*	*Target Syllables*	*Actual Syllables*	*Alignment*	*Type*
164	doggie	[dɑ][gi]	[ga][ga]	[dɑ][gi] [ga][ga]	CH
165	doggie	[dɑ][gi]	[ga][ga]	[dɑ][gi] [ga][ga]	CH
166	doggie	[dɑ][gi]	[ga][ga]	[##][dɑ][gi] [ga][ga][##]	CH + VH
167	tammy	[tæ][mi]	[mi][mi]	[tæ][mi] [mi][mi]	CH
168	bringing	[bɹɪ][ŋɪŋ]	[mi][ni]	[bɹɪ][ŋɪŋ] [m#i][ni#]	CH