Natural Language Syllable Alignment: From Conception to Implementation

Greg J. Hedlund, Keith Maddocks, Yvan Rose, and Todd Wareham

Abstract— **Several types of linguistic analyses, *e.g.*, studies of first or second language acquisition, require the alignment of the (actual) syllables produced by a speaker with those in the corresponding intended (target) utterance. Maddocks (2005) described a dynamic programming algorithm for performing such alignments. In this paper, we will describe (1) the implementation of this algorithm and its integration into Phon, a software program for analyzing language acquisition data (Rose *et al.* 2005) and (2) the development of a genetic algorithm for optimizing the performance of this algorithm relative to a user-provided corpus of actual / target utterance pairs.**

Keywords— **Software Design; User Interface Design; Dynamic Programming; Genetic Algorithm**

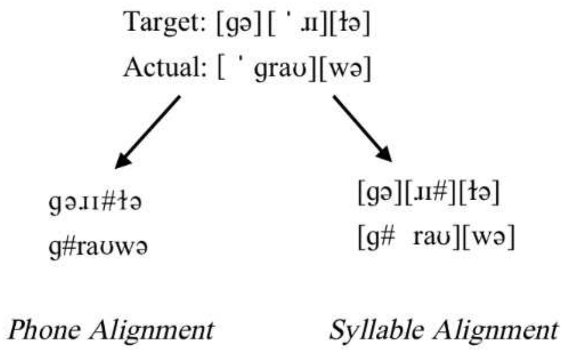

Fig. 1. Phone and Syllable Alignment

I. Introduction

Given two sequences, an **alignment** of these sequences is a display in which each symbol in each sequence is matched with either (i) a symbol in the other sequence or (ii) a special symbol called an **indel** (denoted here by '#'). An alignment graphically displays symbol correspondences and substitutions (if two symbols are matched) as well as symbol insertions and/or deletions (if a symbol is matched with an indel). An alignment is thus a succinct summary of possible processes relating a pair of sequences.

Alignment problems occur in language acquisition research, in which the sequences are syllabified learner (actual) and intended (target) utterances and both phone and syllable alignments of actual-target pairs are required to ensure valid analyses of learner-produced speech errors (see Figure 1). With the advent of large-scale analytical systems such as Child-Phon and Phon [3], [7], [8], alignment algorithms have become necessary. Given the size of the datasets involved, such algorithms need to be both efficient and accurate; moreover, as these algorithms must be embedded inside a large pre-existing software system with a non-CS user base, issues of software architecture and user interface design are also crucial.

In this paper, we will describe our experience in integrating the syllable alignment algorithm proposed by Maddocks [5] into Phon. In Section II, we give a brief overview of the syllable alignment problem and of the algorithm proposed in [5]. In Section III, we describe the software architecture and user interface underlying the implementation of this algorithm in Phon. In Section IV, we describe a genetic algorithm which optimizes parameters associated with the alignment algorithm and enables it to reach accuracies of greater than 98%. Finally, in Section V, we discuss some directions for future research.

Department of Computer Science (KM, TW) and Department of Linguistics (GJH, YR), Memorial University of Newfoundland, St. John's, NL, Canada

This work has been supported by SSHRC grant 410-2004-1836 (YR) and NSERC grant 228104 (TW).

II. Background

A. The Syllable Alignment Problem

The alignment problem in natural language operates on pairs of utterances. Utterances can in turn be subdivided into smaller units. Phon uses a five-level utterance structure (see Figure 2). Utterances are transcribed using the International Phonetic Alphabet, and individual phones are described by feature matrices based on a standard set of features which provide detailed descriptions of the phonetic makeup of these phones [3, Appendix B]. Utterances transcribed within Phon are automatically syllabified using a standard onset-rhyme syllable internal structure [3, Section 2.4.1].

The main goal of sequence alignment is to match related items in sequences while avoiding matches between unrelated items. There are many possible alignments for a given pair of sequences, so the pre-

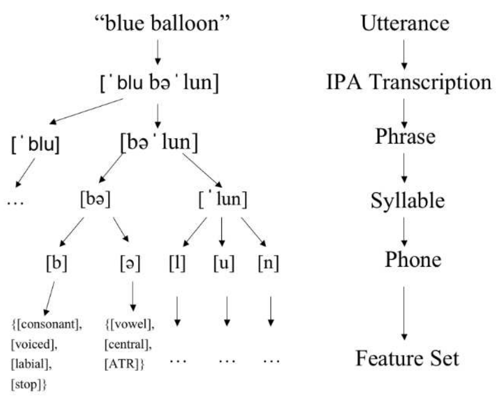

Fig. 2. Utterance Structure

ferred alignments are those that have the minimum or maximum value for an associated score-function.

The most basic type of utterance alignment is phone alignment. If syllabifications of the utterances are available, one can also create syllable alignments, which are essentially enhanced phone alignments that include syllable boundaries placed such that syllable integrity is preserved, *i.e.*, phones that are part of one syllable in an actual or target form are not scattered across multiple syllables in the alignment. Note that syllable integrity within utterances does not forbid many-one mappings of syllables between utterances; indeed, such mappings are necessary to model syllable epenthesis, *e.g.*, "[spy]" → "[si][py]", and truncation, *e.g.*, "[ba][na][na]" → "[ba][na]".

A number of phone alignment algorithms have been proposed [1], [2], [4], [10]. However, the only previous work on syllable alignment is by Connolly [1] (in which he discusses possible algorithms and syllable-pair scores), Hedlund and O'Brien [3] (in which they give an algorithm which did not achieve more than 70% accuracy), and Maddocks [5]. The last of these, which is arguably the only successful syllable alignment proposed to date, is described in the next section.

B. A Syllable Alignment Algorithm

The algorithm proposed in Maddocks [5] is what is known as a single-level alignment algorithm, *i.e.*, utterances are viewed as sequences of phones and syllable-boundary markers and utterance alignment is treated as an alignment of these sequences that preserves syllable integrity. This algorithm is based on the standard dynamic programming algorithm for pairwise sequence alignment, which has been derived independently in several disciplines (see [9] and references therein). At the core of this algorithm is a function $sim(x, y)$ that assesses the degree of similarity of a symbol x from the first given sequence and a symbol y from the second given sequence. Given a linguistically-relevant definition of $sim()$, the classical sequence alignment algorithm can perform phone alignments. However, this algorithm can also be adapted to create syllable alignments if we (1) define a version of $sim()$ that simulates syllable integrity and matching constraints via constraints on phone matchings and (2) define a set of rules that reintroduce the actual and target form syllable boundaries back into the enhanced phone alignment. Full details of these modifications are given in [5]; however, for our purposes here, we need only describe those modifications associated with (1).

In our $sim()$ function for enhanced phone alignments, the similarity value of two phones is a function of a basic score and the associated values of a number of applicable rewards or penalties, where each reward and penalty condition encodes a linguistically-motivated constraint on the form of the alignment. Our basic score is the number of features shared by phones x and y. If any penalty condition applies, $sim()$ is the value associated with that penalty; otherwise, $sim()$ is the sum of the values associated with applicable reward conditions times the basic score. Note that we assume that no more than one penalty is applicable to any phone-pair. Our reward and penalty conditions were defined by repeatedly examining the set of incorrect alignments produced the algorithm and proposing conditions that addressed the largest possible subset of these alignments, and the associated condition-values were optimized manually by trial and error.

The scheme proposed above has two advantages: (1) it performs both phone and syllable alignments, and (2) it readily handles many-one syllable mappings. It has the disadvantage that syllable integrity is not explicitly enforced but is a product of the interactions of various mechanisms, and hence can be perturbed by the introduction of new mechanisms or the modification of old ones. This problem becomes more acute when the alignment algorithm parameters, *i.e.*, the condition-values, must be re-calibrated to handle utterances in a new natural language. However, such difficulties can be ameliorated by the addition of an automatic algorithm parameter optimization procedure such as that described in Section IV.

III. Algorithm Implementation

A. Initial Implementation and Testing

The phone and syllable alignment algorithms described in the previous section were implemented in Java 1.4.2 and a testbed was constructed using soft-

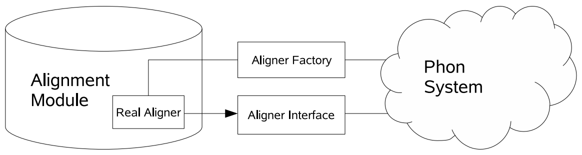

Fig. 3. Alignment Algorithm / Phon Integration Architecture.

ware components from Phon (notably the utterance-structure and syllabification mechanisms) with which these algorithms were tested against a corpus of 168 adult-child utterance pairs extracted from the literature on first and second language acquisition of English. The alignments produced for this corpus were then compared to the "correct" alignments prepared by Rose. Both phone and syllable alignments improved as conditions were added, eventually reaching accuracies of 95% (see [5] for details).

B. Integration into Phon: Software Architecture

As mentioned above, the Maddocks algorithm was developed using the Phon Java classes encoding utterance structure and the syllabification mechanism (see [3, Chapter 6] for details). Three issues arose when this algorithm was integrated into Phon:

1. As both Phon and the aligner will continue to evolve, it should be possible to change the aligner code independently of the Phon code and vice versa;
2. As Phon operates over multiple natural languages, it should be possible to provide more than one instance of the aligner at any time; and
3. As the algorithm will not always produce alignments that are linguistically accurate, it should be possible for the user to modify the alignment produced by the algorithm.

These issues were resolved using the software architecture sketched in Figure 3. Issue (1) was dealt with by making the aligner a separate module that implements and communicates with Phon via the Aligner interface. Issue (2) is dealt with by implementing an Alignment Factory class, which is responsible for creating concrete instances of aligners and returning them to Phon. Issue (3) was partially dealt with by ensuring that the Aligner interface is rich enough to allow access to all internal structures of a produced alignment as well as the ability to construct a new alignment in a piecewise fashion without having to invoke the Maddocks algorithm. However, as will be described in the next section, resolution of this issue also required the development and implementation of a straightforward user interface for modifying algorithm-produced alignments.

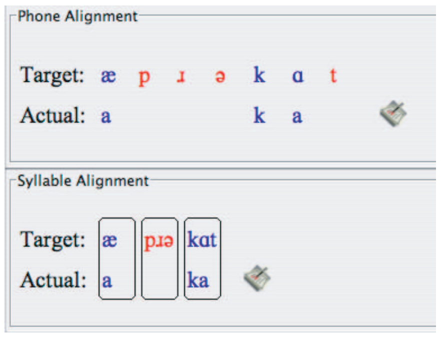

Fig. 4. Phone and Syllable Alignment Display

C. Integration into Phon: User Interface Design

In Phon, phone and syllable alignments are displayed such that aligned target and actual phones are placed on the same vertical axis and aligned target and actual syllables are similarly grouped within boxes (see Figure 4). As algorithm-produced alignments may be linguistically inaccurate, users need to be able to perform the following two activities:

1. Modify existing alignments by modifying individual linkages between phones and/or syllables; and
2. Validate user-modified alignments against standard constraints to prevent nonsensical alignments, *e.g.*, two phone- or syllable-pair linkages cannot cross.

The first activity has been implemented by allowing users to void the alignment (by mouse-clicking an icon displayed at the right end of the alignment) and then reconstruct the alignment from left to right by using mouse-clicks to successive select and link entities from both sequences and add these linkages to the modified alignment (see Figures 5 and 6). In the case of phone alignments, a linkage will always be between one phone in the actual sequence and one phone in the target sequence; however, in the case of syllable alignments, many-one linkages can be made between one syllable in either the actual or target sequence and one or more (adjacent) syllables in the other sequence. Note that phone and syllable alignments can be modified independently; however, as syllable alignments depend on phone alignments, the syllable alignment algorithm is automatically invoked after modification of a phone alignment to revise the syllable alignment if necessary. The second activity will be implemented by allowing users to validate an alignment by mouse-clicking a second icon displayed at the right end of the alignment, which will then graphically highlight portions of the alignment (if any) that violate constraints and need to be further modified.

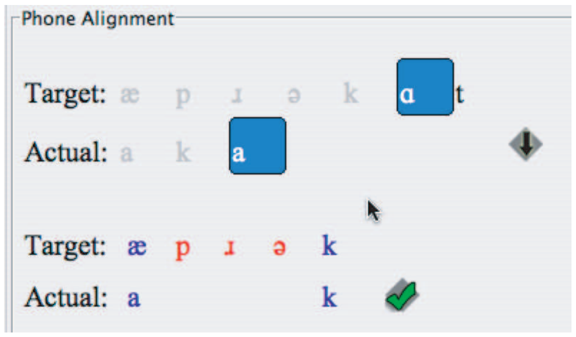

Fig. 5. Phone Alignment Modification

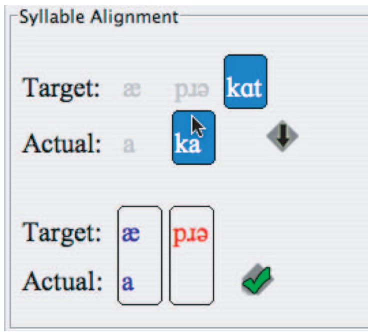

Fig. 6. Syllable Alignment Modification

IV. Algorithm Parameter Optimization

Since the accuracy of the phone and syllable alignment algorithms depend on the values assigned to the reward and penalty condition parameters, finding the optimal values for these parameters relative to a given corpus is of the utmost importance. Given a set of linguistically correct alignments, it is possible to assess the relative worth of any particular parameter-setting by seeing how many alignments in that set can be done correctly using that parameter-setting. Unfortunately, given the large number of parameters and the very large number of possible values for each parameter (namely, the set of all integers), a brute force approach to optimization is not viable.

We choose to overcome this parameter-optimization problem using a genetic algorithm (GA) (see [6] and references therein). Essentially, a GA tries to find optimal solutions to a problem of interest by mimicking the evolutionary process by which populations of organisms adapt to their environments. In a biological population, the traits inherited by each organism from its parents determine that organism's fitness and hence the likelihood that this organism will survive long enough to pass those traits on to the next generation. In a GA, the organisms in a population correspond to possible solutions to the

```
Generate initial population of size X
WHILE population fitness increases DO
    Generate new population of size X' > X
    Evaluate fitness of new population
    Trim new population to include only
     the X fittest members
Output population
```

Fig. 7. Generic Genetic Algorithm Structure

problem of interest, and a fitness function (which is often just the solution cost or value) determines a solution's fitness and hence the likelihood that this solution will be used to produce the solutions in the next generation. In both cases, over time, the average fitness of the population increases until optimal or near-optimal fitness is achieved, resulting in healthy organisms and good solutions, respectively.

The process described above can be implemented algorithmically using the generic framework given in Figure 7. Given this framework and a problem of interest, three things have to specified in order to produce a GA for that problem:
1. How are solutions encoded in the population?
2. How are solutions in one generation combined to produce solutions in the next generation?
3. What is the solution fitness function?

Part (2) above is typically done by mutation (construct a new solution by randomly changing portions of a parent solution) and/or recombination (construct one or more new solutions by stitching together fragments of two parent solutions). One of the more popular forms of recombination is crossover, which involves swapping elements in one parent solution with the corresponding elements in a second parent solution. Note that all of these operations are based on actual biological mechanisms by which parental DNA is combined and changed to produce child DNA in organisms.

How can we specify the elements listed above to create a GA for optimizing alignment parameters? The obvious solution-encoding of a set of alignment parameters is as an integer array. This encoding allows efficient crossover and mutation operations; moreover, the fitness of a solution is simply the accuracy of the alignment algorithm operating under the parameter-values encoded in that solution relative to a given corpus of correct alignments.

The resulting GA has been implemented and tested against several corpora from English and Dutch. Though our results are still preliminary, it appears that this algorithm can find parameter-settings with 98%+ accuracies using small populations of solutions in relatively few iterations. For example, given an

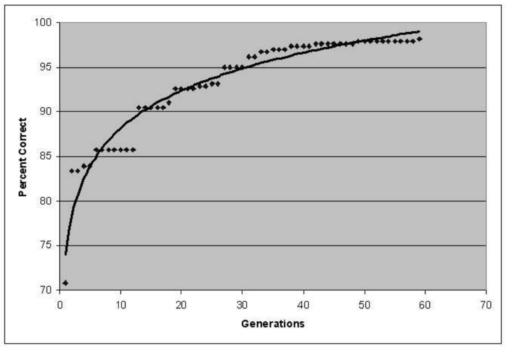

Fig. 8. Average GA Population Fitness Increases Over Time

initial population of random parameter-settings, an accuracy of 98.5% was achieved relative to the 168-pair corpus mentioned in Section III-A in sixty generations using a population consisting of 20 solutions (see Figure 8); when populations of even moderately accurate seed parameter-settings are used initially, the algorithm converges even faster.

V. Future Work

The major direction for future research on the syllable alignment module will be to integrate the stand-alone genetic algorithm for alignment parameter optimization described in Section IV into Phon. We envision its operation as follows:

1. The user will associate a particular group of alignment parameter settings with a corpus. This group can be selected from a set comprised of a generic parameter-setting and parameter-settings generated in previous research.
2. The user will validate (and thus select) the alignments in what is hopefully a representative subset of the records in the corpus. For example, in a corpus of 1000 records, the user might validate a subset of 100 records.
3. Upon activation by the user, the GA will revise the parameter settings to attain the most accurate results possible based on the validated records. As certain parameter settings may be set to 0, this would also correspond to revising the set of relevant reward and penalty conditions.
4. The new settings will then be used to align subsequent records in the corpus.

Note that the parameter / optimizer framework described above for utterance alignment is fairly general and could be applied to any automated activity within Phon; hence, another possible direction for future work would be to develop a more accurate utterance syllabification algorithm using this framework.

Acknowledgments

The authors would like to thank Brian MacWhinney (Department of Linguistics, Carnegie Mellon University) for providing funding to support the initial and ongoing development of Phon. The authors would also like to thank the Deans of Arts and Science as well as the Heads of the Departments of Computer Science and Linguistics of Memorial University of Newfoundland for bureaucratically enabling our collaboration.

References

[1] Connolly, J.H. 1997. Quantifying target-realization differences. *Clinical Linguistics & Phonetics*, 11:267–298.

[2] Covington, M.A. 1996. An Algorithm to Align Words for Historical Comparison. *Computational Linguistics*, 22(4): 481–496

[3] Hedlund, G.J. 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.

[4] Kondrak, G. 2003. Phonetic alignment and similarity. *Computers and the Humanities*, 37(3): 273–291.

[5] Maddocks, K. 2005. *An Effective Algorithm for the Alignment of Target and Actual Syllables for the Study of Language Acquisition.* B.Sc.h. dissertation, Department of Computer Science, Memorial University of Newfoundland.

[6] Reeves, C.R. and Rowe, J.E. 2003. *Genetic Algorithms: Principles and Perspectives – A Guide to GA Theory.* Kluwer Academic Publishers, Boston, MA.

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

[8] Rose, Y., MacWhinney, B., Byrne, R., Hedlund, G.J., Maddocks, K., O'Brien, P., and Wareham, T. 2005. Introducing Phon: A Software Solution for the Study of Phonological Acquisition. To appear, *Proceedings of the 30th Annual Boston University Conference on Language Development.*

[9] Sankoff, D. and Kruskal, J.B. (eds.) 1983. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of String Comparison.* Addison-Wesley, Reading, MA.

[10] Somers, H.L. 1999. Aligning Phonetic Segments for Children's Articulation Assessment *Computational Linguistics*, 25(2): 267–275.