Natural Language Process Detection: From Conception to Implementation

Jason Gedge, Greg J. Hedlund, Yvan Rose, and Todd Wareham

***Abstract*— Advanced linguistic analyses of language acquisition data require the detection of various standard processes in the learner. Each such process is associated with specific systematic differences between forms of words produced by learners and speakers of a language; hence, process detection is equivalent to pattern detection. In this paper, we will describe efficient algorithms for detecting two important processes in language acquisition, consonant metathesis and consonant harmony, and the implementation of these algorithms (including user interface design) in Phon, a software program for analyzing language acquisition data (Rose *et al.* 2006).**

***Index Terms*— Algorithm Design and Analysis; Software Engineering; User Interface Design**

I. Introduction

Language acquisition is the subarea of linguistics concerned with how a person learns a language, where this language could be the first acquired by the learner as a child, a second language acquired after a first is already known, or a previously known language that has been partially or fully lost due to brain damage. Research in this area is done relative to language corpora, where a **corpora** is a set of actual utterances produced by a particular language learner at a particular time paired with the intended target versions of these utterances, *e.g.*, a child's utterance ("efut") and the corresponding adult utterance ("elephant"). Early research in this area was hampered by the available corpora, which consisted of small numbers of utterances by a very small number of learners (often a single learner) over a small number of time-periods.

Over the last 25 years, a number of projects have created comprehensive, electronically-accessible corpora which accurately reflect the variation in language acquisition by learners over time. Given the sizes of these corpora, analysis can no longer be done by hand, and must be computerized. Software packages implementing such analyses must not only be efficient but also be designed with linguist needs and methodology in mind, so that it is easy for linguists themselves to perform and understand the results of analyses. For example, given that there a number of standard processes that learners are known to invoke which systematically modify intended target utterances to result in the actual utterances, it is crucial that linguistic analysis software allow linguists to easily detect the utterance modifications characteristic of these processes.

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

This work has been supported by an NSERC USRA (JG), NIH grant XXX (GJH, YR) and NSERC grant 228104 (TW).

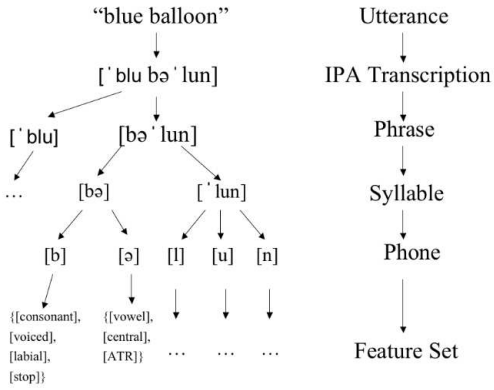

Fig. 1. Utterance Structure

In this paper, we will describe the derivation and implementation of detection algorithms for two important processes, consonant metathesis and consonant harmony, within Phon, a language acquisition analysis software system [6], [7]. This paper is organized as follows: Section II gives background on the Phon data format and defines some of the most important language-acquisition processes. Section III presents what we believe to be the first general algorithms for the detection of these processes. Section IV describes the implementation of these algorithms in Phon in terms of both software architecture and user interface design. Finally, Section V describes our conclusions and some promising directions for future research.

II. Background

A. Language Acquisition Data Format

Process detection is done relative to pairs of natural language utterances, where each pair consists of a learner (actual) and intended (target) utterance. Each utterance can be subdivided into smaller units. Phon uses a five-level utterance structure (see Figure 1). 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 [1, Appendix B]. Utterances transcribed within Phon are automatically syllabified using an algorithm based on a composition-cascade of deterministic finite-state transducers (See [6, Section 1.4] for details).

Process detection (as well as many other types of linguistic analyses) requires comparison of corresponding elements in the utterances of a target-actual pair. One way of computing

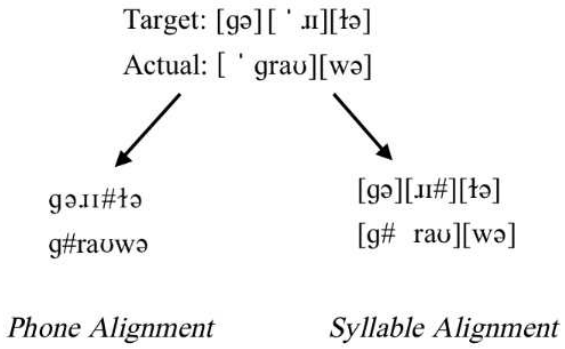

Fig. 2. Phone and Syllable Alignment

such correspondences is a target-actual utterance alignment, in which each symbol in each utterance is matched with either (i) a symbol in the other utterance 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), and is thus a succinct summary of possible processes relating a pair of utterances. Both phone and syllable alignments of actual-target pairs are required to ensure valid analyses of learner-produced speech errors (see Figure 2). A joint phone-syllabic alignment is automatically created for each target-actual pair in Phon using a modified pairwise sequence alignment dynamic programming algorithm (see [2], [3] and [6, Section 1.5] for details).

B. Language Acquisition Processes and Patterns

A recurring modification of different features of language is known as a *language process*. Such modifications can affect the shape of produced utterances, and offer valuable insight into language development. Processes that act locally on target-actual pairs leave distinct patterns within the pairs. Hence, detecting a process is simply a matter of detecting its signature pattern.

There are many processes of interest in language acquisition. Two such patterns are:

- **Consonant Metathesis**: Consonant metathesis consists of a swapping of features between pairs of consonants. Sometimes, all features are exchanged (effectively resulting in the swapping of these consonants). The most common form of metathesis occurs between adjacent consonants, but is not limited to this case. Examples of consonant metathesis can be seen in Figure 3.
- **Consonant Harmony**: Consonant harmony involves the sharing of one or more features of a consonant to one or more adjacent consonants across intervening vowels; these features are referred to as the **shared** features of the harmony. When a feature is absorbed into one of the surrounding consonants, other features in the same family are inevitably lost (*e.g.*, a consonant cannot be articulated with the front and back of the tongue at the same time). These lost features are referred to as **neutralized** features. Each harmony also has an associated direction, since it can extend from left to right (progressive) or from right to left (regressive) in an utterance. Both of these types of harmony can be seen in Figure 4.

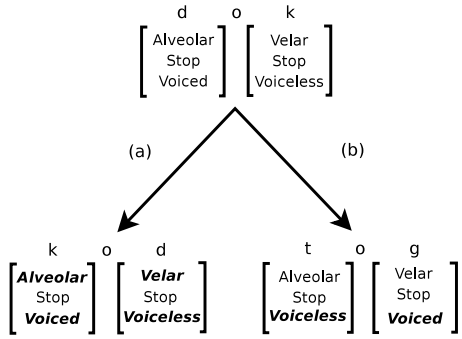

Fig. 3. Examples of Consonant Metathesis. Boldfaced features indicate those features involved in the metathesis. (a) A metathesis involving full swapping of consonants. (b) A metathesis involving only a single feature.

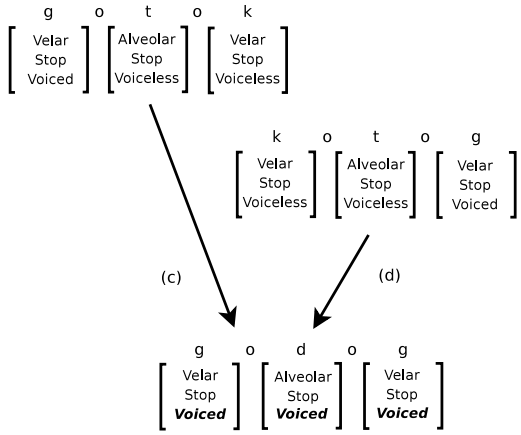

Fig. 4. Examples of Consonant Harmony. Only the voicing feature is involved in the harmony, with `voiced` being shared and `voiceless` being neutralized. (a) A progressive (left-to-right) harmony. (b) A regressive (right-to-left) harmony.

Even when dealing with two apparently simple processes such as these, complexities can arise, related to the interactions of processes and the precise definitions of the patterns associated with them. Multiple processes may occur in the same utterance and their resulting patterns may overlap; moreover, it is not clear what constitutes an occurrence of a process. For example, are several adjacent patterns indicative of several small occurrences of a process, or should they be joined to indicate a single large occurrence of that process? The latter is especially problematic in the detection of harmony processes, both in terms of features involved and the extent in the utterance over which harmony occurs. Such matters need to be carefully considered in the design of any algorithm for process detection, as will be done below in Section III.

III. Algorithm Design

Though some analytical work has been done on consonant harmony [4], [5], [8], there is to our knowledge no published research on detection algorithms for consonant metathesis or consonant harmony. In this section, we describe efficient algorithms for detecting both processes. Relative to the issues mentioned at the end of Section II-B, the following decisions have been made in the design of these algorithms:

- Consonant metathesis detection will be limited to two types: (1) metathesis involving adjacent consonants (ignoring intervening vowels) and (2) metathesis involving consonants at the ends of an utterance when there are indels.
- All potential candidates for occurrences of processes are returned by the algorithms.
- Both processes will only consider aligned consonants and ignore everything else.

In the algorithm descriptions below, let $\Gamma(C)$ be the features associated with consonant C. Note that $\Gamma(\text{indel}) = \phi$.

The consonant metathesis algorithm is fairly intuitive and straight-forward. We simply iterate through each aligned pair in the utterance, collecting adjacent consonants and checking to see if features were swapped. Suppose C_1^a (C_2^a) and C_1^t (C_2^t) are the first (second) aligned consonants in the actual and target pair, respectively. We first calculate

$$\alpha_1 = \Gamma(C_1^a) \cup \Gamma(C_1^t),$$

$$\alpha_2 = \Gamma(C_2^a) \cup \Gamma(C_2^t).$$

The set of features involved in the metathesis is then

$$\alpha = \alpha_1 \oplus \alpha_2.$$

where $\oplus$ is the exclusive-or operation. This applies to both of the situations we consider in metathesis, as described in the design decisions above, since Γ was also defined for indels. The complete algorithm can be seen in Figure 5.

```
 1. results <- []
 2. for i from 2 to the number of aligned consonants do
 3.     actual1, target1 = (i-1)-th aligned consonant pair
 4.     actual2, target2 = i-th aligned consonant pair
 5.     features1 = intersection(G(actual1), G(target2))
 6.     features2 = intersection(G(actual2), G(target1))
 7.     mfeatures = exclusiveor(features1, features2)
 8.     if mfeatures is not empty then
 9.         add (i-1, i, mfeatures) to results
10. return results
```

Fig. 5. Metathesis Detection Algorithm. In this pseudocode, $\Gamma(C)$ is written as `G(C)`.

To detect harmony, we need to know not only the sets of shared and neutralized features but also the extent and direction of the harmony in the utterance (as harmony can involve more than two consonants). Let us consider first extent and direction, in terms of finding the start and end points of a harmony. A starting point exists whenever a feature is found in both consonants of an aligned target-actual pair. As long as any features shared between shared between those consonants then continues through the target utterance, the harmony continues. To determine when this harmony ends we check for one of two conditions:

1) the feature is no longer found in the feature set of the current consonant of the target utterance.
2) the same feature is again found to be a starting point (*i.e.*, it is found in both the target and actual utterance)

Let us now consider finding the sets of shared and neutralized features. Let C_1^a (C_2^a) and C_1^t (C_2^t) be the aligned consonants in the actual and target pair at the start (end) position of the harmony, respectively. The set α of shared features is then

$$\alpha = [\Gamma(C_1^t) \cap \Gamma(C_1^a) \cap \Gamma(C_2^a)] - \Gamma(C_2^t).$$

Unfortunately, such a formula does not seem to exist for the set of neutralized features. Instead, we exploit the fact that features can be grouped into logical families called **natural classes**. We first find the natural classes of all features in α and then look at the consonant at the end of the harmony in the actual utterance to see what features from the same class were neutralized. The complete algorithm can be seen in Figure 6. Note that the calculation of shared and neutralized feature-sets as described above is done in the `add * to results` statements on lines 10, 16, and 21.

```
 1. results <- []
 2. pStarts <- dictionary() ; previous start positions
 3. N <- the number of aligned consonants
 4. for i from 1 to N do
 5.     actual, target = i-th aligned consonant pair
 7.     for each feature F in pStarts do
 8.         if F is not in G(target) then
 9.             if pStarts[F] is not i-1 then
10.                 add (pStarts[F], i-1, F) to results
11.
12.     features = intersection(G(actual), G(target))
13.     for each feature F in features do
14.         if F is in pStarts then
15.             if pStarts[F] is not i-1 then
16.                 add (pStarts[F], i-1, F) to results
17.         pStarts[F] <- i
18.
19. for each feature F in pStarts do
20.     if pStarts[F] is not N then
21.         add (pStarts[F], N, F) to results
22.
22. return results
```

Fig. 6. Consonant Harmony Detection Algorithm. In this pseudocode, $\Gamma(C)$ is written as `G(C)`.

Both of these algorithms run in $O(n)$ time and space, where n is the maximum of the lengths of the given target and actual utterances, and are hence not only optimal but practical for real-world use. As one can see from the single loop found on line 2 of Figure 5, the time complexity of the consonant metathesis detection algorithm is $O(n)$. The space required is also $O(n)$, since we may have to store a result at every pair of adjacent aligned consonants (this can be reduced to $O(1)$ if results are generated as needed rather than stored). Though the harmony algorithm in Figure 6 contains nested loops, it still runs in $O(n)$ time. This is so because the outermost loop on line 4 of Figure 6 executes $O(n)$ times over all aligned pairs and the inner loops execute $O(1)$ times over the fixed set of features. The space required by this algorithm is similarly only $O(n)$ because the `pStarts`

dictionary only contains as many entries as features and the `results` list could have an entry between every two adjacent aligned target-actual pairs.

IV. Algorithm Implementation

A. Initial Implementation and Testing

Initial implementations of the algorithms were done in the Java programming language to facilitate easy integration into Phon, which is itself written in Java. Initial testing was done relative a simple test harness that accepted target-actual pairs in plaintext format. The plaintext output produced by this harness was subsequently verified by a linguist.

The first round of test results were described in a very low-level format, indicating a result by the target-actual pair, the internal alignment indices of any pertinent locations and the features involved. It was found that the use of internal values – the alignment indices – caused major confusion in analyzing the results. A second attempt at a more linguist-friendly output format involved printing the target-actual pair as they are aligned, printing aligned characters next to each other and using spaces for indels. Indicators were then printed below the pertinent locations and, following this, the set of features involved in the consonant metathesis/harmony. This format was found to be very easy to read and check.

Several "test and re-design" cycles were required to obtain satisfactory results. During initial testing of the consonant metathesis algorithm, it was realized that the algorithm did not detect metathesis involving consonants at the ends of the target-actual pair. As this algorithm could not be generalized to handle this special case, a separate detector for this case was created. Similarly, the initial implementation of consonant harmony detection simply merged results on every possible feature, and did not detect neutralized features. This led to the development of the more complex algorithm as described in Section III.

B. Integration into Phon: Software Architecture

The process detection system has minimal dependency on Phon. Since Phon is always evolving, the low coupling between the two means that Phon can continue to evolve with minimal effort in keeping its process detection component up-to-date with architecture changes. The two components within the Phon architecture that are coupled to process detection are those encoding feature matrices and words, because feature matrices allow a detector to obtain feature information pertaining to a phone and words contain the target-actual pairs that we analyze. Process detection itself consists of two key components:

- A *result* that encapsulates all of the information pertinent to describing a consonant metathesis or harmony detection by the algorithms.
- A *detector* that encapsulates a detection mechanism.

The architecture developed around these two key components is shown in Figure 7.

The final stage of process detection involves storing results within Phon itself. This is done through a class called `SearchResultsManager`. This class can unify any form

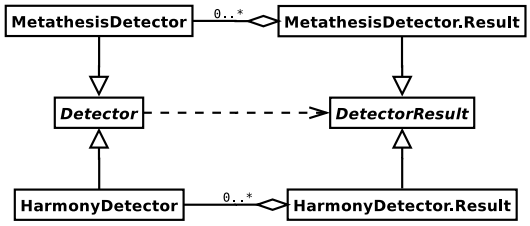

Fig. 7. Process Detection Architecture Class Diagram.

of a result from searching such that they are all stored in a single location and can use a single user interface.

C. Integration into Phon: User Interface Design

Phon's philosophy on user interface stresses two major points for new modules. First, the user interface for a new module should be consistent with the existing design. This allows users of Phon to approach new modules with familiarity, providing a comfortable foundation to begin using the new module. Second, the interface itself should be easy for linguists to use.

When a linguist is viewing a session, he or she has the option of searching for consonant metathesis or consonant harmony within that session by means of a context menu. The corresponding search dialog then appears with possible result-filtering options. For example, with the harmony interface displayed in Figure 8, the filtering options include the direction of the harmony (progressive and/or regressive) and features required to be in the set of neutralized features. Similarly, with the metathesis interface, one can choose features that are required to be in the metathesis.

Once the result-filtering options are set to the user's satisfaction, he/she can then begin the detection process. After detection is complete, a panel of results is displayed to the user, such as the results for metathesis detection shown in Figure 9. In this step of the detection process, linguists verify which results are actual instances of consonant metathesis and harmony by selecting them. Metathesis and harmony results are displayed so as to take up the minimal amount of

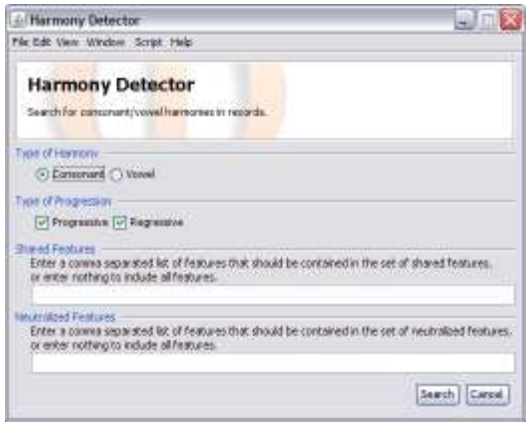

Fig. 8. Consonant Harmony Detection Dialog.

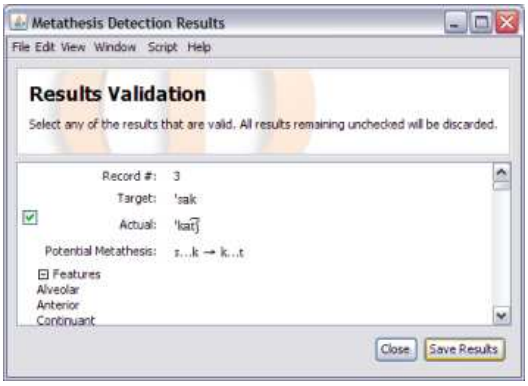

Fig. 9. Consonant Metathesis Search Results Dialog.

screen-space while still showing all of the information vital to both processes. This allows the verification process to flow much more smoothly. Once finished, the user can then save the results to the search results module of Phon. From there it is possible to save the verified results to a comma-separated (CSV) file for archival purposes.

V. Future Work

In this paper, we have described what we believe to be the first published detection algorithms for consonant harmony and consonant harmony processes in natural language data, as well as the implementation of these algorithms in the Phon system. Two directions for future research are (1) extend the algorithms to detect consonant metathesis and harmony over whole corpora instead of isolated target-actual pairs, and (2) develop and implement algorithms to detect other processes of interest, *e.g.*, vowel harmony and consonant / vowel epenthesis.

Acknowledgments

The authors would like to thank Brian MacWhinney (Department of Linguistics, Carnegie Mellon University) for providing funding to support the 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] 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.

[2] Hedlund, G.J., Maddocks, K., Rose, Y., and Wareham, T. (2005) Natural Language Syllable Alignment: From Conception to Implementation. In *Proceedings of the Fifteenth Annual Newfoundland Electrical and Computer Engineering Conference (NECEC 2005)*.

[3] 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.

[4] Pater, J. and Werle, A. (2001) Typology and Variation in Child Consonant Harmony. In C. Féry, A.D. Green, and R. van de Vijver (eds.) *Proceedings of the Holland Institute of Linguistics – Phonology 5*. University of Potsdam. 119–139.

[5] Pater, J. and Werle, A. (2003) Direction of Assimilation in Child Consonant Harmony. *Canadian Journal of Linguistics* 48:385–408.

[6] Rose, Y., Hedlund, G.J., Byrne, R., Wareham, T. and MacWhinney, B. (2007) Phon 1.2: A Computational Basis for Phonological Database Elaboration and Model Testing. In P. Buttery, A. Villavicencio, and A. Korhonen (eds.) *Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition: 45th Annual Meeting of the Association for Computational Linguistics*. ACL; Stroudsburg, PA. 17–24.

[7] Rose, Y., MacWhinney, B., Byrne, R., Hedlund, G., Maddocks, K., O'Brien, P., and Wareham, T. (2006) Introducing Phon: A Software Solution for the Study of Phonological Acquisition. In D. Bamman, T. Magnitskaia, and C. Zaller (eds.) *Proceedings of the 30th Annual Boston University Conference on Language Development*. Cascadilla Press; Somerville, MA. 489–500.

[8] Werle, A. (2001) Variation in child consonant harmony. In G. Horwood and S.-K. Kim (eds.) *RuLing Papers (Rutgers Linguistics Working Papers), vol. 2.*. 197–205.