Class 3: Learning phonological facts

(1) `italian3.pl` (relevant excerpts)

```
while ($current_number_correct != ($#inputs + 1)) {
    $iterations++;

    # We try to improve performance by swapping rules that might feed or bleed each other
    # There are various ways to explore the space of possible orderings,
    #   but one way is to start at the end of the grammar and try to "promote" rules
    #   by moving them up in the grammar
    $swap1 = -1;
    $swap2 = -1;
    FINDSWAPPAIR:
    for (my $r1 = $#rules; $r1 >= 1; $r1--) {
     for (my $r2 = $r1 - 1; $r2 >= 0; $r2--) {
         # Check if we already know that r2 crucially precedes r1
         next if ($known_orderings =~ /$ordering[$r2]<$ordering[$r1]/ or
              $explored_orderings =~ /$ordering[$r2]<$ordering[$r1]/ or
              $explored_orderings =~ /$ordering[$r1]<$ordering[$r2]/  );
         # Now check if moving r1 before r2 could possibly change things
         # There are three possibilities:
         #   (1) r1 and r2 have overlapping structural descriptions (bleeding/counterbleeding)
         #   (2) struc desc of r1 overlaps with output of r2 (currently feeding, try counterfeeding)
         #   (3) struc desc of r2 overlaps with output of r1 (currently counterfeeding, try feeding)
         # The struc descs are:  $rules[$ordering[$r1]][0]  and $rules[$ordering[$r2]][0]
         # "Overlapping" means that at least one of their "terms" could match the same segment
         # (a "term" in this simplified conception is an expression that refers to a single segment)
         if (overlap($rules[$ordering[$r1]][0], $rules[$ordering[$r2]][0])) {
             print "Rules $ordering[$r1] and $ordering[$r2] have overlapping struc descs (potential for bleeding/counter
             print "\t\t[$rules[$ordering[$r1]][0]] and [$rules[$ordering[$r2]][0]]\n";
             ($swap1, $swap2) = ($r1, $r2);
         } elsif (overlap($rules[$ordering[$r2]][0], $rules[$ordering[$r1]][1])) {
             print "Rule $ordering[$r2] operates on the output of $ordering[$r1] (currently counterfeeding; trying feedi
             print "\t\t[$rules[$ordering[$r2]][0]] and [$rules[$ordering[$r1]][1]]\n";
             ($swap1, $swap2) = ($r1, $r2);
         } elsif (overlap($rules[$ordering[$r1]][0], $rules[$ordering[$r2]][1])) {
             print "Rule $ordering[$r1] operates on the output of $ordering[$r2] (currently feeding; trying counterfeedi
             print "\t\t[$rules[$ordering[$r1]][0]] and [$rules[$ordering[$r2]][1]]\n";
             ($swap1, $swap2) = ($r1, $r2);
         }
         if ($swap1 >= 0) {
             last FINDSWAPPAIR;
         }
     }
     }
     # If we got through the FINDSWAPPAIR block and didn't find any eligible swaps, then we're stuck
     if ($swap1 == -1) {
         print "****It appears that there is no ordering of these rules that will correctly derive the data****\n";
         check_accuracy(@ordering);
         exit;
     } else { # Try this swap
         print "\tAttempting to move rule $ordering[$swap1] before $ordering[$swap2]\n";
         @proposed_ordering = (@ordering[0..$swap2-1],$ordering[$swap1],@ordering[$swap2..$swap1-1],@ordering[$swap1+1..
         print "Proposed ordering: @proposed_ordering\n";

         $new_number_correct = check_accuracy(@proposed_ordering);
         if ($new_number_correct > $current_number_correct) {
             # Aha, this helped!  A new crucial ordering.  Save the new order
             @ordering = @proposed_ordering;
             $current_number_correct = $new_number_correct;
             $known_orderings .= "$ordering[$swap1]<$ordering[$swap2] ";
             $explored_orderings = undef;
```

```
                print "\tNew order is better: rule $proposed_ordering[$swap1] must come before rule $proposed_ordering[$swa
                print "\t\t$rules[$ordering[$swap1]][0] -> $rules[$ordering[$swap1]][1]   PRECEDES   ";
                print "$rules[$ordering[$swap2]][0] -> $rules[$ordering[$swap2]][1]\n";
                print "\tKnown orderings: $known_orderings\n";
                print "\tCurrent order: @ordering\n";
            } elsif ($new_number_correct < $current_number_correct) {
            # A detrimental move; make sure we don't do it again (at least until something else has changed)
            $explored_orderings .= "$ordering[$swap2]<$ordering[$swap1] ";
            print "\tNew order is worse: rule $ordering[$swap2] probably comes before rule $ordering[$swap1]\n";
            }
            # It's also possible that the move made no difference, and we shouldn't try it again.
            #  However, I think it may also be possible that something which seems not to make a difference at
            #  the moment might turn out to be crucial; So, we should keep track of the irrelevant_orderings at the moment,
            #  but keep in mind that if we change any other ordering, then we might as well re-check orderings
            #  that were previously irrelevant (by forgetting that they were irrelevant)
            else {
                print "\tNew order makes no difference; leaving things as they were\n\n";
                $explored_orderings .= "$ordering[$swap2]<$ordering[$swap1] ";
            }
        }
    }


sub check_accuracy {
    my @check_ordering = @_;
    my $correct = 0;
    for ($i = 0; $i <= $#inputs; $i++) {
        # We'll start with the current input, and transform it
        $output = $inputs[$i];
        for ($r = 0; $r <= $number_of_rules; $r++) {
            # Nab the backreferences
            if ($output =~ /$rules[$check_ordering[$r]][0]/) {
                @backrefs = ($output =~ /$rules[$check_ordering[$r]][0]/);
            }
            # Now do the replacement, putting in the "dummy" backref $1, etc.
            $output =~ s/$rules[$check_ordering[$r]][0]/$rules[$check_ordering[$r]][1]/g;
            while ($output =~ /\$(\d+)/) {
                $replacement = $backrefs['$1'];
                $output =~ s/\$$1/$replacement/;
            }
        }
        # Now check answer against the "real" answer in the checkfile
        if ($output eq $answers[$i]) {
            $correct++;
        } else {
            print "\tIncorrect on form $i  ($inputs[$i]: deriving [$output] instead of [$answers[$i]])\n";
        }
    }
    return $correct;
}

sub overlap {
    my $string1 = @_[0];
    my $string2 = @_[1];

    if (length($string1) > length($string2) ) {
        return ($string1 =~ /\Q$string2\E/);
    } else {
        return ($string2 =~ /\Q$string1\E/);
    }
}
```

(2) Timeline of acquisition for human learners
A couple overviews: Menn and Stoel-Gammon (1995), Hayes (2004)

1. 0 months
    - No lexicon or morphology

- Some knowledge of prosody
- Whatever biases/constraints/boundaries are innate

2. 6 months
   - Still no words or morphology
   - Showing sensitivity to L1 phonological categories ("perceptual magnet" effect; Kuhl)
3. 8-10 months
   - Words? (probably still little or no morphology)
   - Lose ability to distinguish non-native constrasts (Werker and colleagues)
   - Knowledge of native inventory, and also some sequencing constraints; work by Juszcyk and colleagues, including Jusczyk, Friderici, Wessels, Svenkerud, and Jusczyk (1993), Friederici and Wessels (1993), Jusczyk, Luce, and Charles-Luce (1994)
4. Beyond the first year
   - Lexicon expands rapidly
   - Morphology lags behind for quite some time
     - English-learning two-year olds don't necessarily have command of plural suffix (Smith 1973)
     - Even 4-year olds aren't always very good with it (Berko 1958; and subsequent work by Derwing and others)

(3) `findinventory.pl`

```
open (CORPUS, "little_corpus.txt") or die "Warning! Can't open corpus file
$!\n";

while ($line = <CORPUS>) {
    chomp($line);
    for ($i = 0; $i < length($line); $i++) {
        # The following stores the current character in a "hash table"
        $inventory{substr($line, $i, 1)} = 1 ;
    }
}
# Get a list of all the items in the inventory hash table
@phonemes = keys %inventory;
print "List of all phonemes in the file (" . scalar(@phonemes) . " total)\n";
foreach $phoneme (@phonemes) {
    print "$phoneme "
}
print "\n";
```

(4) `findpairs.pl`

```
open (CORPUS, "little_corpus.txt") or die "Warning! Can't open corpus file
$!\n";

while ($line = <CORPUS>) {
    chomp($line);

    for ($i = 0; $i < length($line) - 1; $i++) {
        # The following stores the current character in a "hash table"
        $sequences{substr($line, $i, 2)} = 1 ;
    }
}
# Get a list of all the items in the inventory hash table
@found_seqs = keys %sequences;
```

```
print "List of all two-char sequences in the file (" . scalar(@found_seqs) . " total)\n";
foreach $seq (@found_seqs) {
    print "$seq "
}
print "\n";

open (TEST, "test_words.txt") or die "Warning! Can't open file of test words: $!\n";
while ($line = <TEST>) {
    chomp($line);
    $legal = 1;
    for ($i = 0; $i < length($line) - 1; $i++) {
        if (not exists $sequences{substr($line, $i, 2)}) {
            print "Word $line is illegal (Sequence " . substr($line, $i, 2) ." not in corpus inve
            $legal = 0;
            last;
        }
    }
    if ($legal) {
        print "Word $line is legal\n";
    }
}
```

(5) Assignment 3, for next week (9/30)

- Readings:
  - Jusczyk, Luce, and Charles-Luce (1994) (ref below)
  - Kessler and Treiman (1997) Syllable Structure and the Distribution of Phonemes in English Syllables. *Journal of Memory and Language* 37: 295-311.
- Programming:
  The Jusczyk, Luce, and Charles-Luce study employed sets of monosyllables which were claimed to have high and low phonotactic probabilities in English. Your task is to check their claim, by computing the phonotactic probability of their test items. There is a file called CelexWordsInTranscription.txt, which contains a list of English words. Your task is to write a program that reads in this file, computes the probabilities of their items, by the criteria used in that study. (That is, by the "positional" probabilities). You will need to perform several sub-tasks:
  - You will need to figure out how to break the syllables up into onsets, nuclei, and rhymes (a key to the symbols that are used is provided on the web site along with the file)
  - You will need to calculate the probability of each phoneme in each position
  - You will then need to find a way to translate these individual probabilities to a single score for the entire word
  - I will provide a test file with the Jusczyk et al stimuli, that you can run your program on to see what their scores are.

References

Berko, J. (1958). The child's acquisition of English morphology. *Word 14*, 150–177.

Friederici, A. D. and J. E. Wessels (1993). Phonotactic knowledge of word boundaries and its use in infant speech perception. *Perception and Psychophysics 54*, 287–295.

Hayes, B. (2004). Phonological Acquisition in Optimality Theory: The early stages. In R. Kager, J. Pater, and W. Zonneveld (Eds.), *Fixing Priorities: Constraints in Phonological Acquisition*. Cambridge University Press. http://www.linguistics.ucla.edu/people/hayes/HayesEarlyAcquisition.pdf.

Jusczyk, P. W., A. Friderici, J. M. Wessels, V. Y. Svenkerud, and A. M. Jusczyk (1993). Infants' sensitivity to the sound patterns of native language words. *Journal of Memory and Language 32*, 402–420.

Jusczyk, P. W., P. A. Luce, and J. Charles-Luce (1994). Infants' sensitivity to phonotactic patterns in the native language. *Journal of Memory and Language 33*, 630–645.

Menn, L. and C. Stoel-Gammon (1995). Phonological development. In P. Fletcher and B. MacWhinney (Eds.), *The Handbook of Child Language*, pp. 335–359. Blackwell.