Segmental Similarity, cont.

(1) Summary of last time:
- Reviewed how to calculate the edit distance between two strings
- Our default values (indel = .5, substitution = 0 or 1) yielded dozens of possible alignments; the “most intuitive” one was only one among many
- Started looking at a way to reward substitutions that involve more similar segments/penalize substitutions between less similar segments

(2) Correction/clarification from last time’s handout: def. of the similarity of two segments, \( x \) and \( y \)

\[
\text{Similarity}(x,y) = \frac{\text{Number of natural classes shared by both } x \text{ and } y}{\text{Number of shared natural classes + number of unshared natural classes}}
\]

(3) Example using vowel features from last time:
- List of all natural classes:
  - \([i]\): +high, -back
  - \([e]\): -high, -back
  - \([o]\): -high, +round
  - \([a]\): +low
  - \([u]\): +high, +back
  - \([e,i]\): -back
  - \([e,o]\): -high, -low
  - \([o,u]\): +round
  - \([a,o]\): -high, +back
  - \([a,e]\): -high, -round
  - \([i,u]\): +high
  - \([a,o,u]\): +back
  - \([a,e,o]\): -high
  - \([a,e,i]\): -round
  - \([e,i,o,u]\): -low
  - \([a,e,i,o,u]\): (empty root node)
- Similarity of \([e]\) and \([i]\):
  - Shared natural classes: \([e,i]\), \([a,e,i]\), \([e,i,o,u]\), \([a,e,i,o,u]\)
  - Unshared natural classes: \([e]\), \([i]\), \([e,o]\), \([a,e]\), \([i,u]\), \([a,e,o]\)
  - Similarity = \(\frac{4}{4+6} = .4\)
- Similarity of \([a]\) and \([i]\):
  - Shared natural classes: \([a,e,i]\), \([a,e,i,o,u]\)
  - Unshared natural classes: \([a]\), \([i]\), \([e,i]\), \([a,o]\), \([a,e]\), \([i,u]\), \([a,o,u]\), \([a,e,o]\), \([e,i,o,u]\)
  - Similarity = \(\frac{2}{2+9} = .182\)
- Details in the Frisch reading; listed on your syllabus, but not required

(4) More generally:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>i</th>
<th>u</th>
<th>e</th>
<th>o</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>0.182</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>u</td>
<td>0.182</td>
<td>0.333</td>
<td>1.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>0.364</td>
<td>0.400</td>
<td>0.167</td>
<td>1.000</td>
<td></td>
</tr>
<tr>
<td>o</td>
<td>0.364</td>
<td>0.167</td>
<td>0.400</td>
<td>0.333</td>
<td>1.000</td>
</tr>
</tbody>
</table>
(5) Using similarity values to define the substitution cost in string alignment:
- Substitution Cost\((x,y) = 1 - \text{Similarity}\((x,y)\)
- Result: if \(x = y\), the cost is still 0 (since \(\text{Similarity}(a,a) = 1\)); otherwise, the cost is between 0 and 1, depending on how similar the segments are

(6) The effect this has on our computer/compare example
- Calculated similarity values of all English segments, using a larger feature chart that includes all segments
- Only one optimal alignment (irrelevant arrows are not shown)

<table>
<thead>
<tr>
<th></th>
<th>c</th>
<th>o</th>
<th>m</th>
<th>p</th>
<th>u</th>
<th>t</th>
<th>e</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
<td>0.5</td>
<td>1.0</td>
<td>0.5</td>
<td>1.0</td>
<td>0.5</td>
<td>1.0</td>
<td>0.5</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>1.0</td>
<td>0.5</td>
<td>2.0</td>
</tr>
<tr>
<td>1.0</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.5</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.5</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2.0</td>
<td>0.5</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>2.5</td>
<td>0.5</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>3.0</td>
<td>0.5</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>3.5</td>
<td>0.5</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

- Yielding the following (rather intuitive) alignment:

```
  compu	
er
  compa
er
```

cost: 0.00 0.00 0.00 0.00 0.73 0.90 0.00 0.50

(7) Some other uses of string edit distance in linguistics
- We already talked about word finding, spell-checking, fuzzy searching, etc.
- Dialectology: how different is dialect X from dialect Y?
- Analogy: how do similar words influence one another?


- Intro to Kruskal string edits book on reserve, also online at: [http://odur.let.rug.nl/~nerbonne/papers/timewarp.pdf](http://odur.let.rug.nl/~nerbonne/papers/timewarp.pdf)
- Took phonetic transcriptions of 100 words, as they were pronounced in 104 dialects of Dutch
- For each pair of dialects (D1, D2), calculated string edit distance between each pair of words, to get an estimate of how similar the dialects were
- Results: distinct regions of similar dialects

(Nerbonne et al, p. xi)
Another use of string edit distance: quantifying analogy

A plausible assumption that many people subscribe to:

- Similar words can influence one another in various ways
- The more similar two words are, the more they influence each other
- Example using string edit distance to measure similarity: (Albright and Hayes 2001)
  - What is the past tense of the made-up verb scride?
    * "John likes to scride. Yesterday he ________"
  - Several possibilities, of varying degrees of likelihood: scrided, scrode, scrid
  - Some of the existing verbs that would support scrode:

\[
\begin{array}{c}
\text{dive} \\
\text{shine} \\
\text{rise} \\
\text{write} \\
\text{ride} \\
\text{strive} \\
\end{array}
\]

\[
\begin{array}{c}
\text{drive} \\
\text{smite} \\
\end{array}
\]

Summary of string edit distance:

- String edit (or Levenshtein) distance is a broadly useful tool
- Overview of how it is calculated, at least in principle (we did not look at an implementation of it, though perhaps we will use one later in the quarter)
- Some parameters that influence how strings are aligned: indel values and substitution costs

Discussion of the MDL assignment

Reminder of why we got into string alignment in the first place:

- The task was to find recurring elements to add to a lexicon
- Minimum description length (length of lexicon + length of describing length of encoding utterances using that lexicon) favors an analysis in which utterances are broken into word- and morpheme-like units
- Finding the best string alignment using string edit distance is just one approach
  - There are many others, ranging from brute force trial and error to fancy techniques using information theory, probability, and statistics

Some of your findings in doing the MDL assignment

- The text had 619 characters in it; any lexicon with a total description length greater than this would be pointless (we could just memorize the whole text for cheaper)
- Your solutions tended to achieve description lengths in the 450 to 480 range
- The shortest I’ve seen for this text is 439
- Often beneficial to group together units bigger than the word (gotmyslipper, meover, etc.)
- Units smaller than the word are also sometimes advantageous: ing (do+ing, play+ing, knock+ing)
• More surprising: better rather than bett
  – It can be more efficient to keep it together even if er is already an independent word in
    the lexicon! Why?
  – Another example: wash and washoff; slightly better to list both than to list just wash and
    off. (DL savings of approx. 2 chars for this text). Why?
  – Interaction with frequency: the more frequently a complex word occurs in its inflected/derived
    form, the more advantageous it is to list the complex form as a chunk rather than breaking
    it apart. This is probably a good result.

• Neutral changes:
  – In this text, need appears only after we (and it occurs twice); but listing weneed yields the
    same DL as listing we and need separately. Why?
  – Similarity imalmostdone vs. malmostdone, alrightlet vs. alrightlets, etc.

• Extreme deconstruction:
  – a, you, re, s, i, m all seem like reasonable entries for words or suffixes
  – But once we have them, it’s possible to get carried away: What a re you doing s i lly? Oh
    re a lly? (And in fact doing so achieves a shorter description length for this text!)
  – Is there anything that would eventually lead us to reject this analysis?

(13) In a longer text:
• A lot of the words that accidentally appear together in this text would end up being separated
  (got my slipper, etc.)
• Some elements that occur together a lot would actually be much more advantageous to keep
  together: e.g., its rather than it+s

(14) Overall result, broadly speaking:
• A lexicon file with a list of words and morphemes
• Utterances can be broken into lexical items; even stands a good chance at being able to ana-
  lyze new utterances that have never been seen before
• But what we need is a more exact procedure for breaking utterances and words apart, given
  a list of lexical items that they might be composed of

Introduction to morpheme parsing

(15) The task: given a lexicon of known items, break a new word up into its component parts
• Lexicon: maybe something like what you developed in the MDL assignment, maybe some-
  thing a bit more sophisticated (syntactic information, semantic information, etc.)
• Parse: an indication of what the parts are, and what the entire word/phrase means

(16) An example from Quechua [Weber, Black, and McConnel 1988]

<table>
<thead>
<tr>
<th>Quechua</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>rikan</td>
<td>‘s/he sees him/her’</td>
</tr>
<tr>
<td>rikaykan</td>
<td>‘s/he is looking at him/her’</td>
</tr>
<tr>
<td>nan</td>
<td>‘s/he does something’</td>
</tr>
<tr>
<td>naykan</td>
<td>‘s/he is doing something’</td>
</tr>
<tr>
<td>mikun</td>
<td>‘s/he eats it’</td>
</tr>
<tr>
<td>mikuykan</td>
<td>‘s/he is eating something’</td>
</tr>
</tbody>
</table>

Recurring elements:

| rika  | see’                        |
| na    | ‘do’                        |
| miku  | ‘eat’                       |
| yka   | imperfective (is Xing)      |
| n     | 3sg object                  |

1Warning: I’ve had to make up a couple forms here to fill out the paradigm; it’s possible that a couple of these may not be exactly
right in Quechua!
(17) Some valid parses:

\[
\begin{align*}
\text{rika-n} & \quad \text{see-3sg.} \\
\text{rika-yka-n} & \quad \text{see-impf.-3sg.} \\
\text{miku-yka-n} & \quad \text{eat-impf.-3sg.}
\end{align*}
\]

(18) An invalid parse:

\[
\begin{align*}
\text{n-a-yka} & \quad \text{3sg.-??-impf.-3sg.}
\end{align*}
\]

• What's wrong with this parse?

(19) Exhaustivity: parses should aim to make sense of everything in the word

• When possible, avoid leaving some material unaccounted for

• Similar to the \text{\textit{i want toast on ish\texttt{h\texttt{im}}} example from a couple weeks ago}

(20) Some more Quechua words:

\[
\begin{align*}
\text{maymanta} & \quad \text{‘from where’} \\
\text{maymantan} & \quad \text{‘from where?’} \\
\text{Qusquta} & \quad \text{‘to Cusco’} \\
\text{Qusqu\texttt{t}an} & \quad \text{‘to Cusco?’}
\end{align*}
\]

Another morpheme:

\[
\begin{align*}
n & \quad \text{interrogative (X?) (can also be used for affirmative answers)}
\end{align*}
\]

(21) Some more invalid parses:

\[
\text{Qusqu\texttt{t}an} \quad \text{Cusco-3sg. (‘s/he Cusco’s’)}
\]

(22) Category restrictions: parses should make sure they are combining elements that are compatible with one another

• Don’t put together morphemes that require different categories (parts of speech)

(23) Yet more Quechua verb forms:

\[
\begin{align*}
\text{mikuykanna} & \quad \text{‘s/he is eating something now’} \\
\text{mikunaykan} & \quad \text{‘s/he is wanting to eat something’}
\end{align*}
\]

Need some more lexical items:

\[
\begin{align*}
\text{na} & \quad \text{desiderative (want to X)} \\
\text{na} & \quad \text{‘now’}
\end{align*}
\]

(24) Yet more invalid parses:

\[
\begin{align*}
\text{miku-yka-n-na} & \quad \text{eat-impf.-3sg.-give} \\
\text{miku-yka-n-na} & \quad \text{eat-impf.-3sg.-des.}
\end{align*}
\]

(25) Ordering constraints: parses should make sure they aren’t combining elements in the wrong order

• Don’t posit morphemes in inappropriate places

(26) Over the next week, we’ll look at one technique for parsing that accomplishes these goals

References

