Learning alternations, cont.

24.964—Fall 2004
Modeling phonological learning

Class 12 (9 Dec, 2004)

Agenda items

- More on learning alternations
  - Albright and Hayes (2002)
  - Kruskal (1999)
- Course evals
- Guenther talk: 4:15,

Reminder: final projects

- Goal: lay out the issues, see where the problems lie
- Not intended to be polished, fully worked out proposals or programs
- Please get them to me by next Thursday (12/16)

What we saw last week

Bootstrapping: using knowledge of surface phonotactics to learn alternations (Tesar & Prince 2004)

- E.g., [rat] ∼ [radəs] in a language with final devoicing
- The intuition: given a choice between /rad/ and /rat/, the learner can use knowledge that FINDEVOI is high ranked to choose /rad/
  - The grammar already derives [rat] correctly from /rad/
  - There is no way to derive [radəs] from /rat-əs/
- Even when the grammar doesn't already work in the desired direction, it usually "works better" (desired output is a tied winner, rather than a loser)

What we saw last week

Bootstrapping, part 2: using knowledge of some alternations to infer other alternations (McCarthy, "Free rides")

- If you know /A/ $\rightarrow$ [B] in some words, try making attributing all surface [B] to underlying /A/
- Keep the results if it permits you to formulate a more restrictive grammar
- (Doesn't handle cases where you want /A/ but there's no restrictiveness payoff, or where you want to set up only SOME [B] as /A/)

Some issues with current OT approaches to alternations

Supervision: assumes that learner can

- Find pairs that exhibit alternations
- Apply morphology correctly, to test hypotheses about possible URs (Does /rat-əs/ yield [radəs]?)

Interdependence of phonology and morphology:

➢ Not necessarily safe to assume that morphology has been fully learned correctly prior to learning phonology of alternations!

Some issues with current OT approaches to alternations

Non-incremental:

- Learner learns new grammar from scratch with each datum or hypothesized modification to URs

Some issues with current OT approaches to alternations

Limitations

- Scalability to multiple variants/multiple feature values/multiple possible URs not yet explored
- No story (yet) for alternations *not* motivated by phonotactics
  - Derived environment phonology
  - Sychronically arbitrary (?) alternations
- Not equipped to handle alternations that change the segment count (insertion, deletion, etc.)

Goals for today

- Look at an approach that tries to take on the interdependence of morphology and phonology
- Brief intro to a procedure that can get past the segment count limitation (string edit distance)

Minimal Generalization Model

Recall Tesar & Prince:

- Learners are given pairs of forms that stand in (potentially) any morphological relation
- Morphology is known; learner's task is to make sure the phonological form can be derived using a single UR

Minimal Generalization Model

A different approach: Albright & Hayes (2002)

- Learn phonology as part of the process of learning morphology
- Learner's task is to develop a clean analysis of morphology; learning phonology helps improves accuracy of the analysis

Minimal Generalization Model

Input to the learner:

- Pairs of forms that in a particular morphological relation
- List of sequences known to be surface illegal

Minimal Generalization Model

E.g., German sg $\sim$ pl

- Pairs:

piːp	piːpə	'peep'
voRt	voRtə	'word'
ʃtRaɪt	ʃtRaɪtə	'fight'
vɛRk	vɛRkə	'work'
loːp	loːbə	'praise'
moRt	moRdə	'murder'
gRaːt	gRaːdə	'degree'
aɪt	aɪdə	'oath'
bɛRk	bɛRgə	'mountain'

- Illegal sequences:
  *b#, *d#, *g#, . . .

Minimal Generalization Model

E.g., Or, English pres $\sim$ past

- Pairs:

du	dɪd
se	sɛd
go	wɛnt
gɛt	gat
no	nu
mɪs	mɪst
prɛs	prɛst
læf	læft

- Illegal sequences:
  *pd, *td, *kd, *dd, *sd, *bt, *dt, *gt, …

Minimal Generalization Model

Step 1: Try to learn some morphology, by figuring out the changes

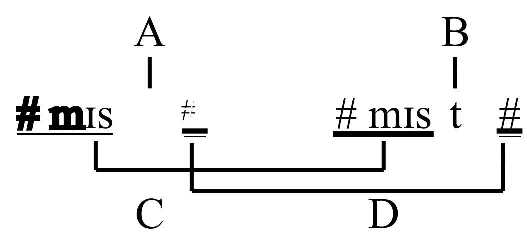
- Factor pairs into change (A → B) and context (C _ D)

- E.g.,

u → ɪd	/ d _
e → ɛd	/ s _
go → wɛnt	
ɛ → a	/ g _ t
o → u	/ n _
∅ → t	/ mɪs _
∅ → t	/ prɛs _
∅ → t	/ læf _

Minimal Generalization Model

Finding the change and the context for word-specific changes:



- Note that this is limited to a single contiguous change (A → B); can't handle two simultaneous changes

Minimal Generalization Model

Step 2: Generalization (but what to compare with what?)

- Restricting search space with a linguistic principle: locality

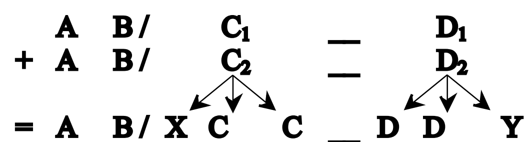
$$\begin{array}{llllll}
\varnothing \rightarrow t & / & m & \text{ɪ} & s & __ \\
\varnothing \rightarrow t & / & pr & \varepsilon & s & __ \\
\hline
\varnothing \rightarrow t & / & X & \begin{bmatrix} +\text{syl} \\ +\text{voi} \\ +\text{son} \\ -\text{low} \\ -\text{bk} \\ -\text{tns} \\ -\text{rnd} \end{bmatrix} & s & __
\end{array}$$

Minimal Generalization Model

Features of generalization scheme:

- "Myopic" description language: fully specified segments adjacent to the change, classes of segments farther out, free variables at edge

- Minimal generalization: retain *all* shared feature values in featural term



Minimal Generalization Model

Iterative generalization:

miss *press*

$\varnothing \rightarrow$ t / [ɪ,ɛ] s _ # *laugh*

$\varnothing \rightarrow$ t / vcls fric _ # *jump*

$\varnothing \rightarrow$ t / vcls cons _ #

Minimal Generalization Model

Rule evaluation:

- Scope of a rule = number of forms that meets its structural description
  - I.e., words containing CAD in input
- Hits, or positive examples = number of forms that it correctly derives
  - I.e., words containing CAD in input, and CBD in output
- Reliability = (hits / scope)

Minimal Generalization Model

Examples:

- Suffixing *-t* after voiceless consonants works quite well in general, but there are some exceptions
  - *think*, *take*, *eat*, *teach*, etc.
  - *want*, *start*, *wait*, etc.
  - Reliability $= \frac{\text{\# of vcls-final vbs} - (\text{[t]-final vbs} + \text{vcls-final irregs})}{\text{\# of vcls-final vbs}}$
- Suffixed *-t* after voiceless fricatives works exceptionlessly
  - *miss*, *press*, *laugh*, etc.
  - No irregs end in voiceless-final frics
  - Reliability $= \frac{\text{\# of vcls-fric final vbs}}{\text{\# of vcls-fric final vbs}} = 1$

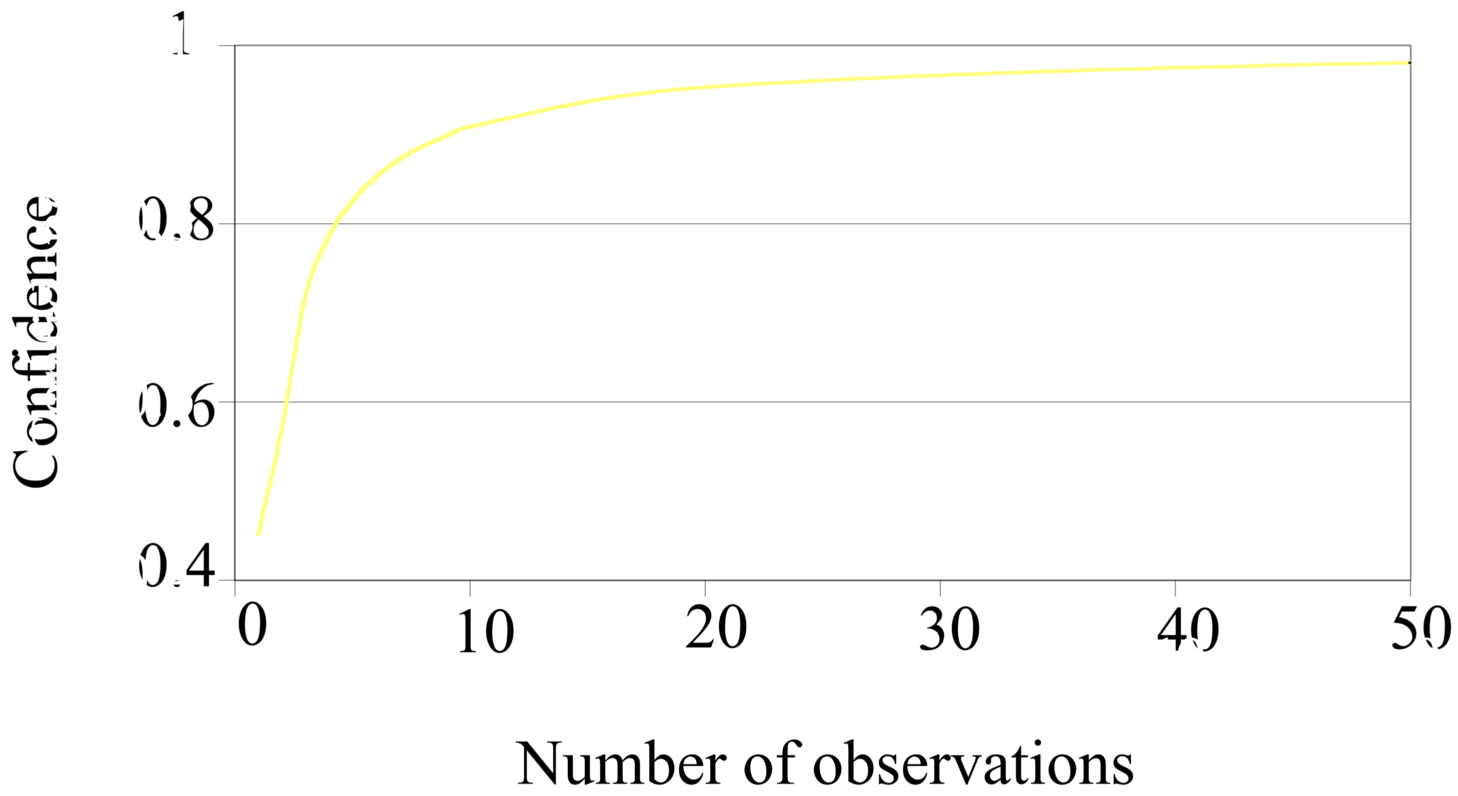
Minimal Generalization Model

Comparing generalizations of different sizes:

- Affix -*t* after [s], after [s, ʃ], and after [f, θ, s, ʃ] all work perfectly
  - No irregulars among any subset of voiceless frics
- Intuitively, the more striking fact is lack of irregs after [f, θ, s, ʃ], since it's more general
- *Confidence adjustments*;
  - Reliability ratios are adjusted downwards, using statistical adjustment that compensates for small numbers
  - E.g., 2/2 = .57, 5/5 = .83, 20/20 = .95, 100/100 = .99

Minimal Generalization Model

Confidence limits



Minimal Generalization Model

Learning phonology to improve confidence

- Exceptions to suffixing *-d* after vcd segments:
  - *come*, *give*, *find*, *leave*, etc.
  - *need*, *decide*, *avoid*, etc.
  - Reliability = $\frac{\text{\# of vcd-final vbs} - (\text{[d]-final vbs} + \text{vcd-final irregs})}{\text{\# of vcd-final vbs}}$
- The latter batch has a principled explanation—namely, phonology

Minimal Generalization Model

Path to phonological rules:

- After comparing (*hug*, *hugged*) and (*rub*, *rubbed*), the learner knows -*d* can be affixed after voiced stops
- When the learner encounters (*need*, *needed*), it treats the pair as a $\varnothing \rightarrow$ əd rule

Minimal Generalization Model

Path to phonological rules:

- However, *need* also meets the structural description of $\varnothing \rightarrow$ d / vcd stop _ #, so its reliability must be updated
- Try applying $\varnothing \rightarrow$ d / vcd stop _ # to *need*, yielding incorrect *[nidd]
- Scan *[nidd] for surface illegal sequences (here, *[dd]
  - Could also just run /nid+d/ through grammar and see if faithful candidate is eliminated
- Posit phonological rule: /dd/ $\rightarrow$ [dəd]

Minimal Generalization Model

Phonological rules can improve morphological confidence

- Exceptions to suffixing *-d* after vcd segments:
  - *come*, *give*, *find*, *leave*, etc.
  - *need*, *decide*, *avoid*, etc.
  - $\text{Reliability} = \frac{\text{\# of vcd-final vbs} - (\text{vcd-final irregs} + \text{[d]-final vbs})}{\text{\# of vcd-final vbs}}$

Minimal Generalization Model

Phonological rules can improve morphological confidence

- Exceptions to suffixing *-d* after vcd segments:
  - *come*, *give*, *find*, *leave*, etc.
  - *need*, *decide*, *avoid*, etc.
  - Reliability = $\frac{\text{\# of vcd-final vbs} - \text{(vcd-final irregs)}}{\text{\# of vcd-final vbs}}$

Minimal Generalization Model

Error-driven learning

- In this case, errors are generated in the course of evaluating morphological generalizations
- (What generates the errors in Tesar & Prince's model?)

Minimal Generalization Model

What this procedure won't get you:

- /pd/ → [pt], etc. (progressive devoicing)
- Reason: in order to learn this, we would need to generate errors like *[d͡ʒʌmpd]
- In order to generate *[d͡ʒʌmpd], we need a rule suffixing *-d* after voiceless consonants
- However, *-d* only occurs after voiced consonants (for precisely this reason). Minimal generalization will only yield $\varnothing \rightarrow$ d / [+voi] _ #
  - All *-d* examples share [+voi])

The problem: *complementary distribution*

Minimal Generalization Model

Overcoming complementary distribution

- Try to identify "competing" changes (A $\rightarrow$ B, A $\rightarrow$ B$'$, ...)
- When two changes share the same input (A), clone their contexts and see whether any phonological rules can be found
- Example: given both $\varnothing \rightarrow t$ and $\varnothing \rightarrow d$, try creating $\varnothing \rightarrow d$ rules in the voiceless contexts (and vice versa)
  - E.g., $\varnothing \rightarrow$ d / vcls fric _ #
  - Generates errors *[mɪsd], *[prɛsd], *[læfd], etc.
  - Yields rules devoicing after [s], [f], ...

Minimal Generalization Model

Another problem that one often encounters

- Exceptional words that behave as if they have the opposite value of one of their features
- Kenstowicz and Kisseberth (1977): "input exceptions"
- Examples in English: *burnt, dwelt*
  - These could lead the learner to conclude the *-t* occurs after any consonant, even though most examples are after voiceless consonants
- Solution (details omitted): compare the reliability of bigger generalizations against the reliability of their subsets; if most of the positive examples (hits) are from a particular subset, then you must penalize the broader generalization

Minimal Generalization Model

Summary

- Similar in spirit to Tesar & Prince (2004), in that previous knowledge of phonotactics is employed to identify errors that might be attributed to phonology
- Unlike Tesar & Prince's proposal, it is embedded in a more general model of learning morphological relations
  - Errors are generated in the course of trying to find cleaner morphological generalizations (fewer exceptions)
  - Contains mechanisms for handling pairs that cannot be explained phonotactically
    - Rule format allows any alternation to be expressed (not just those provided by universal constraint inventory)
    - Word-specific rules provides mechanism for handling idiosyncratic exceptions

Minimal Generalization Model

This can get the phonological rules, but what about deciding URs of individual lexical items?

- Same intuition as Tesar & Prince (2004): derivations work in one direction, but not the opposite direction
- E.g., /bɛʁg/ $\rightarrow$ [bɛʁk] can be derived by devoicing (since *[bɛʁg] would be surface illegal); /bɛʁk+ə/ $\rightarrow$ [bɛʁgə] can't be derived phonologically, since *[bɛʁkə] is incorrect, but legal

Minimal Generalization Model

Some problems with the model

- Representation of phonological "rules" is clunky
- No a priori assumptions about fixes (is this good or bad?)
- Environments are limited be generalization scheme to local contexts
  - More recent work attempting to relax this, and integrate resulting generalizations into an OT-based grammar, using the GLA
  - Albright & Hayes (2004) Modeling productivity with the Gradual Learning Algorithm

Minimal Generalization Model

Some problems with the model

- No proofs concerning algorithmic difficulty
- Can't handle morphological relations involving multiple changes

String alignment

The problem: how do you know what stays the same, and what changes?

- Example: Spanish

v	e	ŋ	g	o			‘I come’
v	je	n		e			‘he comes’
v	e	n		i	r		‘to come’
v	e	n	d		r	e	‘I will come’

- Before you can even begin to generalize about or explain a change, you have to figure out what the change actually is

(How are correspondences usually calculated within OT?)

String alignment

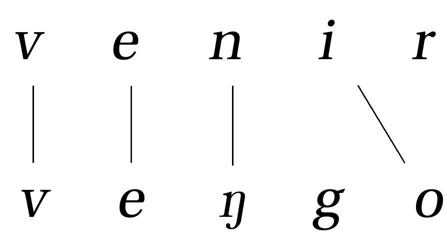
A useful technique: string alignment by string edit/levenshtein distance

- Intuition: alignment can be calculated by figuring out the smallest number of changes needed to change one string to another
  - If two strings share material, don't need to change it
  - Unshared material must be deleted, inserted, or substituted

String alignment

Equivalence of alignments and operations

v	*e*	*n*	—	*i*	*r*
v	*e*	*ŋ*	*g*	*o*	—



- Leave *v* unchanged
- Leave *e* unchanged
- Substitute *n* by *ŋ*
- Insert *g*
- Substitute *i* by *o*
- Delete *r*

String alignment

The task: analyze correspondence as a sequence of substitutions, insertions, and deletions

- In practice, we usually want the *shortest* sequence of alignments/changes
- That is, the *best* alignment

String alignment

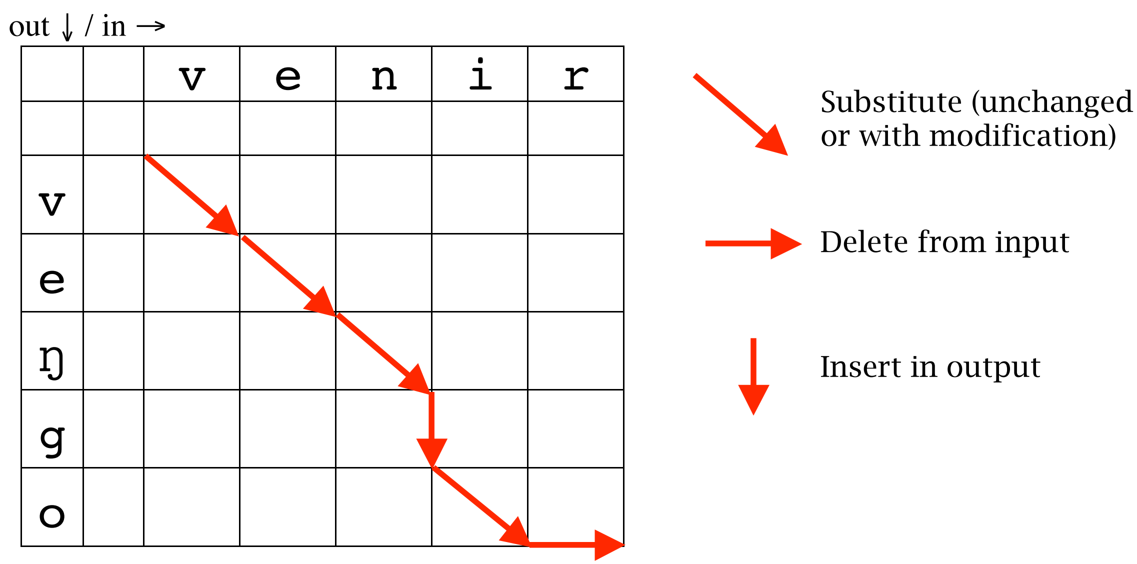
Chart to calculate alignment

out ↓ / in →

		v	e	n	i	r
v						
e						
ŋ						
g						
o						

String alignment

Ideal path (one of many)

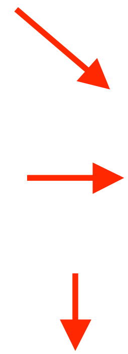


String alignment

Using corners to calculate substitution and indel costs

out ↓ / in →

		v	e	n	i	r
	0	0.5	1.0	1.5	2.0	2.5
v	0.5					
e	1.0					
ŋ	1.5					
g	2.0					
o	2.5					



Substitute (unchanged or with modification)

Delete from input

Insert in output

subst cost	del cost
insert cost	

String alignment

Using corners to calculate substitution and indel costs

out ↓ / in →

		v	e	n	i	r
	0	0.5	1.0	1.5	2.0	2.5
v	0.5	**0 .5** **.5**				
e	1.0					
ŋ	1.5					
g	2.0					
o	2.5					

↘ Substitute (unchanged or with modification)

→ Delete from input

↓ Insert in output

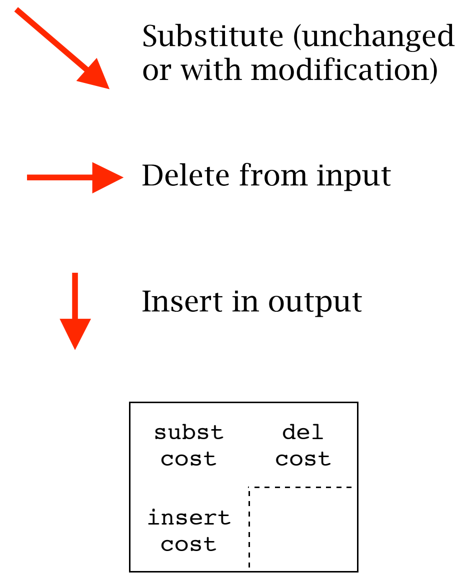
subst cost	del cost
insert cost	

String alignment

Using corners to calculate substitution and indel costs

out ↓ / in →

		v	e	n	i	r
	0	0.5	1.0	1.5	2.0	2.5
v	0.5	0 .5 .5	1 .5 .5			
e	1.0					
ŋ	1.5					
g	2.0					
o	2.5					

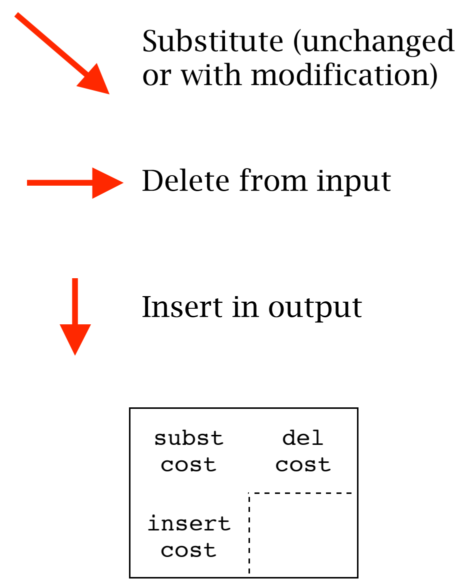


String alignment

Using corners to calculate substitution and indel costs

out ↓ / in →

		v	e	n	i	r
	0	0.5	1.0	1.5	2.0	2.5
v	0.5	0 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5
e	1.0					
ŋ	1.5					
g	2.0					
o	2.5					

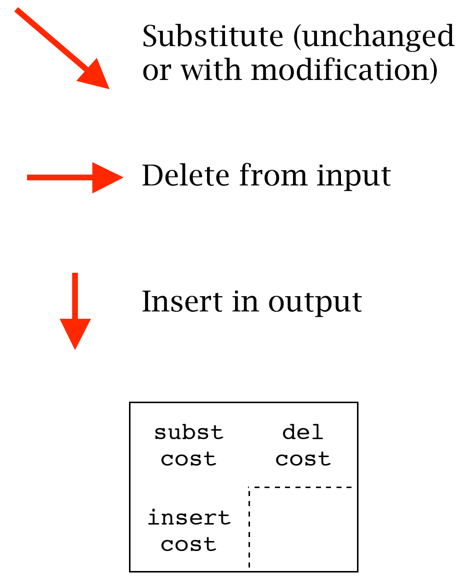


String alignment

Using corners to calculate substitution and indel costs

out ↓ / in →

		v	e	n	i	r
	0	0.5	1.0	1.5	2.0	2.5
v	0.5	0 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5
e	1.0	1 .5 .5	0 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5
ŋ	1.5					
ɡ	2.0					
o	2.5					

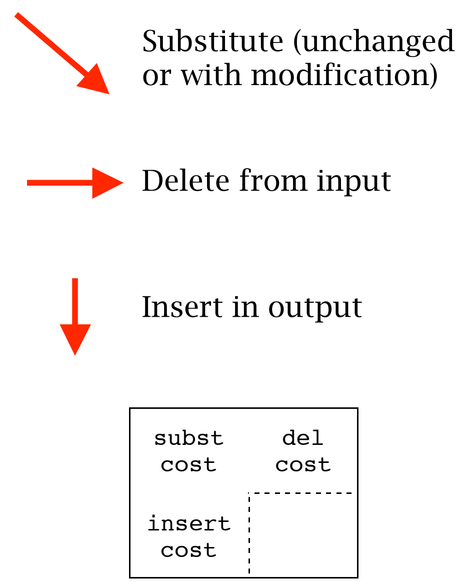


String alignment

Using corners to calculate substitution and indel costs

out ↓ / in →

		v	e	n	i	r
	0	0.5	1.0	1.5	2.0	2.5
v	0.5	0 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5
e	1.0	1 .5 .5	0 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5
ŋ	1.5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5
g	2.0	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5
o	2.5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5

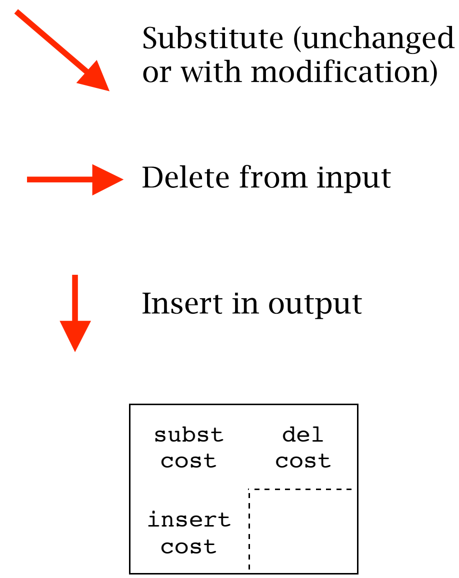


String alignment

Center value is minimum of corners

out ↓ / in →

		v	e	n	i	r
	0	0.5	1.0	1.5	2.0	2.5
v	0.5	0 .5 .5 0	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5
e	1.0	1 .5 .5	0 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5
ŋ	1.5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5
g	2.0	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5
o	2.5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5

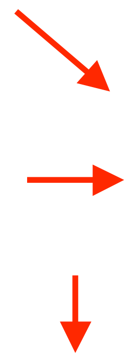


String alignment

Center value is minimum of corners

out ↓ / in →

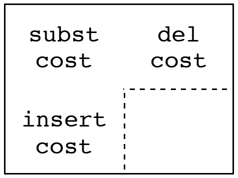
		v	e	n	i	r
	0	0.5	1.0	1.5	2.0	2.5
v	0.5	0 .5 .5 0	1 .5 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5
e	1.0	1 .5 .5	0 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5
ŋ	1.5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5
g	2.0	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5
o	2.5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5



Substitute (unchanged or with modification)

Delete from input

Insert in output

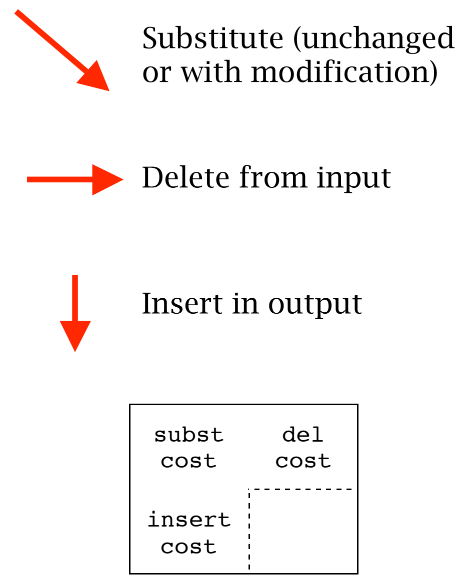


String alignment

Center value is minimum of corners

out ↓ / in →

		v	e	n	i	r
	0	0.5	1.0	1.5	2.0	2.5
v	0.5	0 .5 .5 0	1 .5 .5 .5	1 .5 .5 1.0	1 .5 .5 1.5	1 .5 .5 2.0
e	1.0	1 .5 .5	0 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5
ŋ	1.5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5
g	2.0	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5
o	2.5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5	1 .5 .5

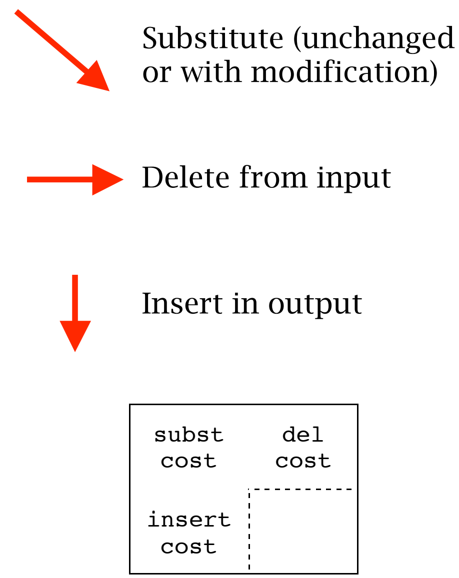


String alignment

Center value is minimum of corners

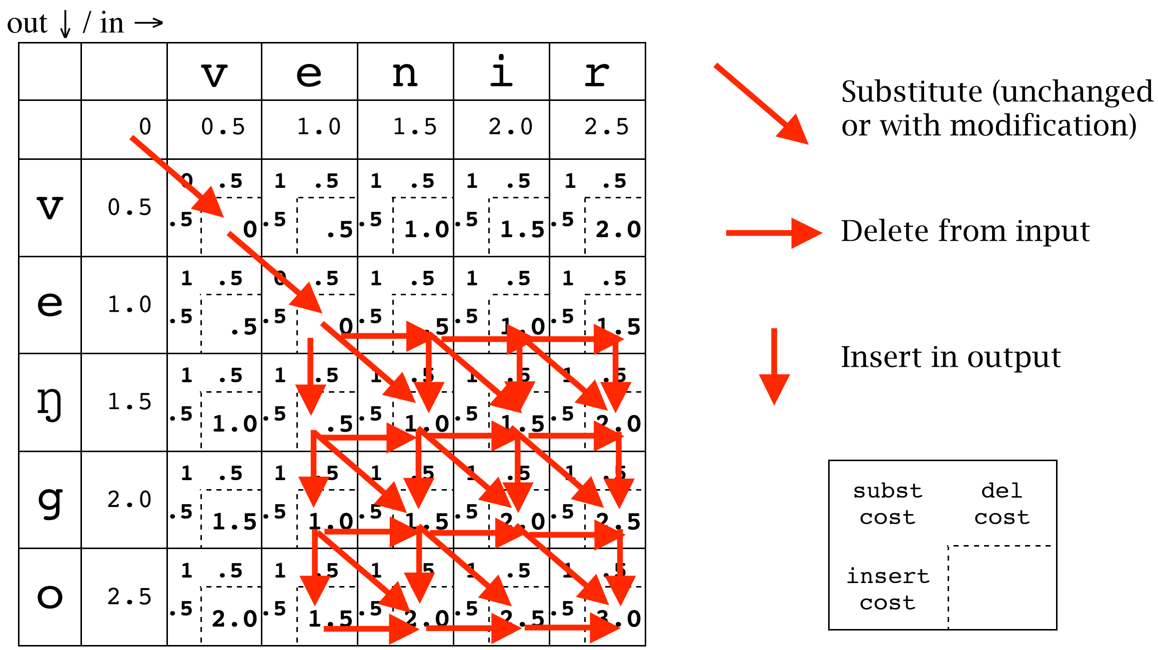
out ↓ / in →

		v	e	n	i	r
	0	0.5	1.0	1.5	2.0	2.5
v	0.5	0 .5 .5 0	1 .5 .5 .5	1 .5 .5 1.0	1 .5 .5 1.5	1 .5 .5 2.0
e	1.0	1 .5 .5 .5	0 .5 .5 0	1 .5 .5 .5	1 .5 .5 1.0	1 .5 .5 1.5
ŋ	1.5	1 .5 .5 1.0	1 .5 .5 .5	1 .5 .5 1.0	1 .5 .5 1.5	1 .5 .5 2.0
g	2.0	1 .5 .5 1.5	1 .5 .5 1.0	1 .5 .5 1.5	1 .5 .5 2.0	1 .5 .5 2.5
o	2.5	1 .5 .5 2.0	1 .5 .5 1.5	1 .5 .5 2.0	1 .5 .5 2.5	1 .5 .5 3.0



String alignment

Paths with smallest costs

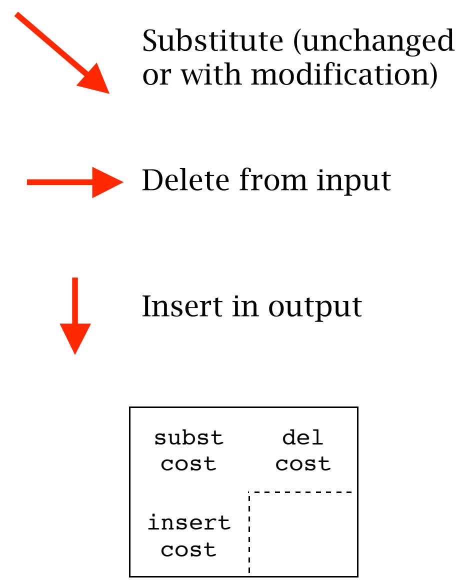


String alignment

Using more sensible substitution costs, based on phonetic similarity

out ↓ / in →

		v	e	n	i	r
	0	0.5	1.0	1.5	2.0	2.5
v	0.5	0 .5 / .5 0	.95 .5 / .5 .5	.80 .5 / .5 1.0	.94 .5 / .5 1.5	.86 .5 / .5 2.0
e	1.0	.95 .5 / .5 .5	0 .5 / .5 0	.97 .5 / .5 .5	.47 .5 / .5 1.0	.91 .5 / .5 1.5
ŋ	1.5	.71 .5 / .5 1.0	.95 .5 / .5 .5	.61 .5 / .5 1.0	.94 .5 / .5 1.5	.85 .5 / .5 2.0
g	2.0	.5 / .5 1.5	.5 / .5 1.0	.5 / .5 1.5	.5 / .5 2.0	.5 / .5 2.5
o	2.5	.5 / .5 2.0	.5 / .5 1.5	.5 / .5 2.0	.5 / .5 2.5	.5 / .5 3.0



(Tedious to count by hand; remaining values left to your imagination. . .)