Sequence Alignment

Gerhard Jäger

ESSLLI 2016

Sequence alignment: Motivation

Motivation

- suppose we have no information except word lists
- goals:
  - estimate distances between languages
  - estimate cognate classes
  - track individual sound changes

Example

Meaning	Italian	English	cognate
few	ˈpɔko	fju:	1
rub	freˈgare	rʌb	0
dull	otˈtuzo	dʌl	0
hunt	katˈtʃare	hʊnt	0
year	ˈanno	jɪə	0
this	ˈkwesto	ðɪs	0
fish	ˈpeʃʃe	fɪʃ	1
rotten	ˈmartʃo	ˈrɒtən	0
right	ˈdʒusto	raɪt	0
when	ˈkwando	wɛn	1
drink	ˈbere	drɪŋk	0
heavy	peˈsante	ˈhɛvɪ	0
heavy	ˈgrɛve	ˈhɛvɪ	0
egg	ˈwɔvo	ɛg	1
earth	ˈtɛrra	ɜ:θ	0
dust	ˈpolvere	dʌst	0
laugh	ˈridere	lɑ:f	0
grass	ˈɛrba	grɑ:s	0
sharp	taʎˈʎɛnte	ʃɑ:p	0
wash	laˈvare	wɒʃ	0

Motivation

- suppose we have no information except word lists
- goals:
  - estimate distances between languages
  - estimate cognate classes
  - track individual sound changes

Example

Meaning	Italian	English	cognate
few	ˈpɔko	fju:	
rub	freˈgare	rʌb	
dull	otˈtuzo	dʌl	
hunt	katˈtʃare	hʊnt	
year	ˈanno	jɪə	
this	ˈkwesto	ðɪs	
fish	ˈpeʃʃe	fɪʃ	
rotten	ˈmartʃo	ˈrɒtən	
right	ˈdʒusto	raɪt	
when	ˈkwando	wɛn	
drink	ˈbere	drɪŋk	
heavy	peˈsante	ˈhɛvɪ	
heavy	ˈgrɛve	ˈhɛvɪ	
egg	ˈwɔvo	ɛg	
earth	ˈtɛrra	ɜ:θ	
dust	ˈpolvere	dʌst	
laugh	ˈridere	lɑ:f	
grass	ˈɛrba	grɑ:s	
sharp	taʎˈʎɛnte	ʃɑ:p	
wash	laˈvare	wɒʃ	

Preprocessing

- IPA is open-ended — 107 letters, 52 diacritics, 4 prosodic marks → 200,000 combinations
- good practice: map IPA strings to a uniform representation with fewer symbols
- common choices:
  - 10 Dolgopolsky sound classes (Dolgopolsky 1986; used i.a. in List 2014)
  - 41 ASJP sound classes
- this course: ASJP

No.	Class	Description	Example
1	P	labial obstruents	p,b,f
2	T	dental obstruents	d,t,θ,ð
3	S	sibilants	s,z,ʃ,ʒ
4	K	velar obstruents, dental and alveolar affricates	k,g,ʦ,ʧ
5	M	labial nasal	m
6	N	remaining nasals	n,ɲ,ŋ
7	R	liquids	r,l
8	W	voiced labial fricative and initial rounded vowels	v,u
9	J	palatal approximant	j
10	ø	laryngeals and initial velar nasal	h,ɦ,ŋ

Preprocessing

- IPA is open-ended — 107 letters, 52 diacritics, 4 prosodic marks → 200,000 combinations
- good practice: map IPA strings to a uniform representation with fewer symbols
- common choices:
  - 10 Dolgopolsky sound classes (Dolgopolsky 1986; used i.a. in List 2014)
  - 41 ASJP sound classes
- this course: ASJP

Meaning	Italian	English
few	poko	fyu
rub	fregare	rob
dull	ottuzo	dol
hunt	kattSare	hunt
year	anno	yi3
this	kwesto	8is
fish	peSSe	fiS
rotten	martSo	rot3n
right	dZusto	rait
when	kwando	wEn
drink	bere	driNk
heavy	pesante	hEvi
heavy	grEve	hEvi
egg	wovo	Eg
earth	tErra	38
dust	polvere	dost
laugh	ridere	lof
grass	Erba	gros
sharp	tallEnte	Sop
wash	lavare	woS

Pairwise alignment

Levenshtein alignment

- related to as *edit distance*
- defines the distance between two strings as the minimal number of *edit operations* to transform one string into the other
- edit operations:
  - deletion
  - insertion
  - replacemant
- example: grm. `mEnS` vs. Cimbrian `menEs`
  1. `mEnS` $\rightarrow$ `menS` (replace)
  2. `menS` $\rightarrow$ `menES` (insert)
  3. `menES` $\rightarrow$ `menEs` (insert)
- $d_L(\texttt{mEnS}, \texttt{menEs}) = 3$

Levenshtein alignment

- alternative presentation: alignment

```
m  E  n  —  S
|  |  |  |  |
m  e  n  E  s
```

- distance for a particular alignment is the number of non-identities
- Levenshtein distance is the number of mismatches for the optimal alignment

Computing the Levenshtein Distance

- recursive definition:
  1. $d_L(\epsilon, \alpha) = d_L(\alpha, \epsilon) = l(\alpha)$
  2. $$d_L(\alpha x, \beta y) = \min \begin{cases} d_L(\alpha, \beta) + \delta(x, y) \\ d_L(\alpha x, \beta) + 1 \\ d_L(\alpha, \beta y) + 1 \end{cases}$$
- apparently requires exponentially growing number of comparisons $\Rightarrow$ computationally not feasible
- but:
  - if $l(\alpha) = n$ and $l(\beta) = m$, there are $n + 1$ substrings of α and $m + 1$ substrings of β
  - hence there are only $(n + 1)(m + 1)$ many different comparisons to be performed
  - computational complexity is polynomial (quadratic in $l(\alpha) + l(\beta)$)

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1				
e	2				
n	3				
E	4				
s	5				

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1				
e	2				
n	3				
E	4				
s	5				

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0			
e	2				
n	3				
E	4				
s	5				

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0			
e	2				
n	3				
E	4				
s	5				

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0			
e	2				
n	3				
E	4				
s	5				

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1		
e	2				
n	3				
E	4				
s	5				

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	
e	2				
n	3				
E	4				
s	5				

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2				
n	3				
E	4				
s	5				

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1			
n	3				
E	4				
s	5				

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1		
n	3				
E	4				
s	5				

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1	2	
n	3				
E	4				
s	5				

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1	2	3
n	3				
E	4				
s	5				

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1	2	3
n	3	2			
E	4				
s	5				

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1	2	3
n	3	2	2		
E	4				
s	5				

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1	2	3
n	3	2	2	1	
E	4				
s	5				

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1	2	3
n	3	2	2	1	2
E	4				
s	5				

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1	2	3
n	3	2	2	1	2
E	4	3			
s	5				

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1	2	3
n	3	2	2	1	2
E	4	3	2		
s	5				

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1	2	3
n	3	2	2	1	2
E	4	3	2	2	
s	5				

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1	2	3
n	3	2	2	1	2
E	4	3	2	2	2
s	5				

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1	2	3
n	3	2	2	1	2
E	4	3	2	2	2
s	5	4			

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1	2	3
n	3	2	2	1	2
E	4	3	2	2	2
s	5	4	3		

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1	2	3
n	3	2	2	1	2
E	4	3	2	2	2
s	5	4	3	3	

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1	2	3
n	3	2	2	1	2
E	4	3	2	2	2
s	5	4	3	3	3

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1	2	3
n	3	2	2	1	2
E	4	3	2	2	2
s	5	4	3	3	**3**

- memorizing in each step which of the three cells to the left and above gave rise to the current entry lets us recover the corresponing optimal alignment

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1	2	3
n	3	2	2	1	2
E	4	3	2	2	2
s	5	4	3	3	3

- memorizing in each step which of the three cells to the left and above gave rise to the current entry lets us recover the corresponing optimal alignment

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1	2	3
n	3	2	2	1	2
E	4	3	2	2	2
s	5	4	3	3	3

- memorizing in each step which of the three cells to the left and above gave rise to the current entry lets us recover the corresponing optimal alignment

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1	2	3
n	3	2	2	1	2
E	4	3	2	2	2
s	5	4	3	3	3

- memorizing in each step which of the three cells to the left and above gave rise to the current entry lets us recover the corresponing optimal alignment

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1	2	3
n	3	2	2	1	2
E	4	3	2	2	2
s	5	4	3	3	3

- memorizing in each step which of the three cells to the left and above gave rise to the current entry lets us recover the corresponing optimal alignment

Computing the Levenshtein distance

- Dynamic Programming

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1	2	3
n	3	2	2	1	2
E	4	3	2	2	2
s	5	4	3	3	3

m	E	n	–	S
m	e	n	E	s

Computing the Levenshtein distance

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1	2	3
n	3	2	2	1	2
E	4	3	2	2	2
s	5	4	3	3	3

m	E	n	–	S
m	e	n	E	s

	–	m	E	n	S
–	0	1	2	3	4
m	1	0	1	2	3
e	2	1	1	2	3
n	3	2	2	1	2
E	4	3	2	2	2
s	5	4	3	3	3

m	E	n	S	–
m	e	n	E	s

Normalization for length

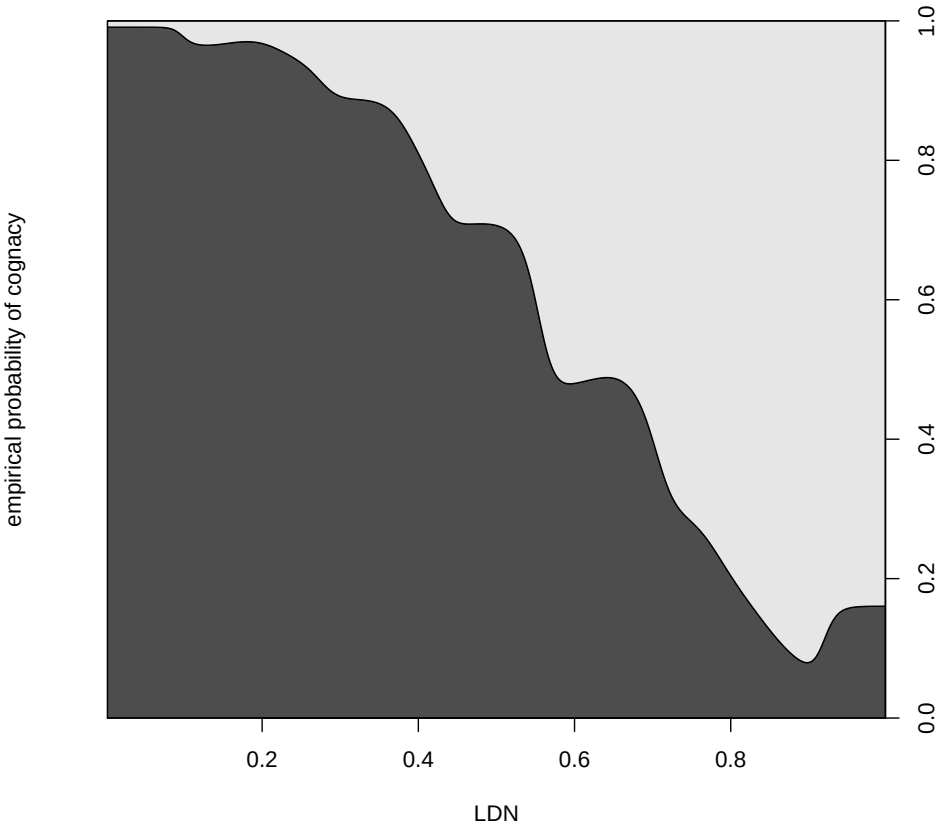
- grm. mEnS (*Mensch*, 'person') and Hindi manuSya are (partially) cognate
- grm. ze3n (*sehen*, 'see') and Hindi deg are not cognate
- still

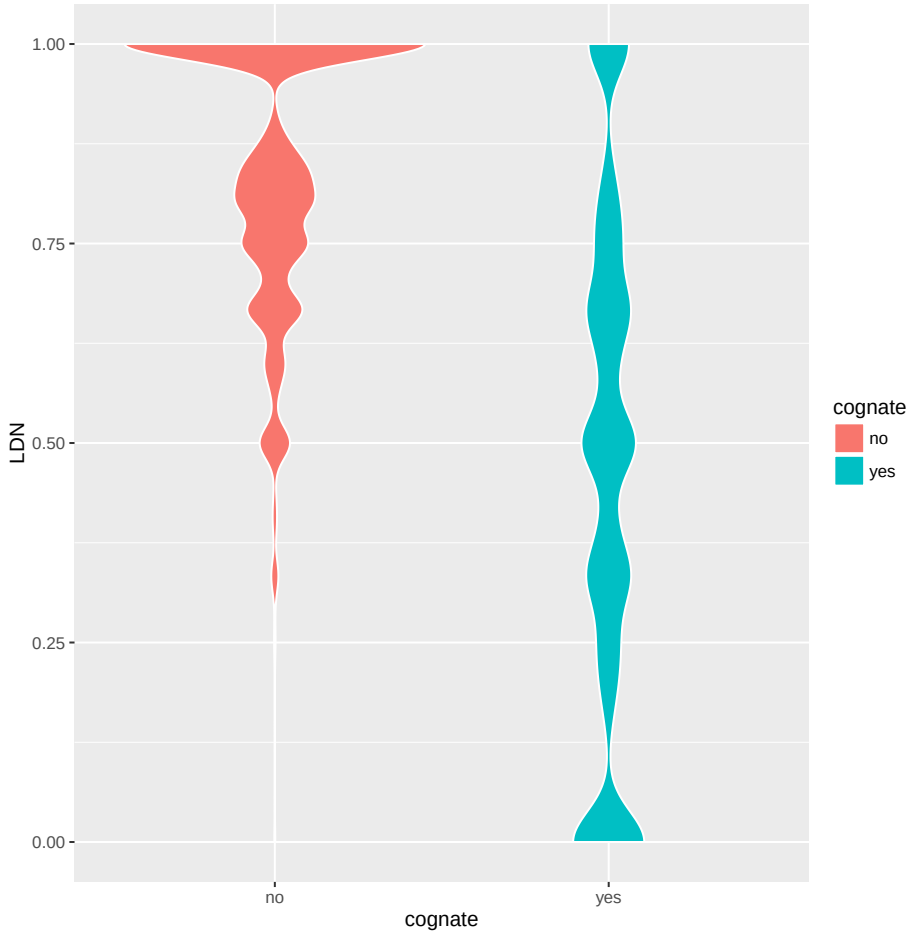
$$\begin{aligned} d_L(\texttt{mEnS}, \texttt{manuSya}) &= 4 \\ d_L(\texttt{ze3n}, \texttt{deg}) &= 3 \end{aligned}$$

- normalization: dividing Levenshtein distance by length of longer string:

$$\begin{aligned} d_{LD}(\texttt{mEnS}, \texttt{manuSya}) &= 4/7 \approx 0.57 \\ d_{LD}(\texttt{ze3n}, \texttt{deg}) &= 3/4 = 0.75 \end{aligned}$$

How well does normalized Levenshtein distance predict cognacy?





Problems

- binary distinction: match vs. non-match
- frequently genuin sound correspondences in cognates are missed:

c	v	a	i		n	a	**z**	3		-	-	-	**f**	i	S
-	-	**t**	u		n	-	o	**s**		**p**	i	s	k	i	s

- corresponding sounds count as mismatches even if they are aligend correctly

h	a	n	t		h	a	n	t
h	E	n	d		m	a	n	o

- substantial amount of chance similarities

Background: probability theory

- Given two sequences: How likely is it that they are aligned?
- More general question: Given some data, and two competing hypotheses, how likely is it that the first hypothesis is correct?

Bayesian Inference!!!

- given:
  - data: d
  - hypotheses: h_1, h_0
  - model: $P(d|h_1), P(d|h_0)$
- wanted:

$$P(h_1|d) : P(h_0|d)$$

Bayesian inference

- Bayes Theorem:

$$P(h|d) = \frac{P(d|h)P(h)}{\sum_{h'} P(d|h')P(h')}$$

- ergo:

$$\begin{aligned}
P(h_1|d) : P(h_0|d) &= P(d|h_1)P(h_1) : P(d|h_0)P(h_0) \\
P(h_1|d) : P(h_0|d) &= \frac{P(d|h_1)}{P(d|h_0)} \frac{P(h_1)}{P(h_0)} \\
\log(P(h_1|d) : P(h_0|d)) &= \log \frac{P(d|h_1)}{P(d|h_0)} + \log \frac{P(h_1)}{P(h_0)}
\end{aligned}$$

Bayesian inference

- suppose we have many independent data: $\vec{d} = d_1, \ldots, d_n$

$$
\begin{aligned}
P(\vec{d}|h) &= \prod_{i=1}^{n} P(d_i|h) \\
\log P(\vec{d}|h) &= \sum_{i=1}^{n} \log P(d_i|h) \\
\log \frac{P(\vec{d}|h_1)}{P(\vec{d}|h_0)} &= \sum_{i=1}^{n} \log \frac{P(d_i|h_1)}{P(d_i|h_0)} \\
\log(P(h_1|\vec{d}) : P(h_0|\vec{d})) &= \sum_{i=1}^{n} \log \frac{P(d_i|h_1)}{P(d_i|h_0)} + \log \frac{P(h_1)}{P(h_0)}
\end{aligned}
$$

Bayesian inference

- mein argument against using Bayes' rule: the **prior probabilities** $P(h_1), P(h_0)$ are not known
- there are various heuristics, but no generally accepted way to obtain them
- if n is large though, $\log P(h_1)/P(h_0)$ doesn't matter very much:[1]

$$\log(P(h_1|\vec{d}) : P(h_0|\vec{d})) \approx \sum_{i=1}^{n} \log \frac{P(d_i|h_1)}{P(d_i|h_0)} = \log(P(\vec{d}|h_1) : P(\vec{d}|h_0))$$

- the quantity $\log(P(\vec{d}|h_1) : P(\vec{d}|h_0))$ is called **log-odds**

[1] Also, if we choose an *uninformative prior* with $P(h_1) = P(h_0)$, we have $\log P(h_1)/P(h_0) = 0$ anyway.

Log-odds

- log-odds can take any real value
- a positive value indicates evidence for h_1 and a negative value evidence for h_0
- the higher the absolute value, the stronger is the evidence

Weighted alignment

- suppose our data are two aligned sequences $\vec{x}$, $\vec{y}$
- for the time being, we assume there are no gaps in the alignment
  - h_1: they developed from a common ancestor via substitions
  - h_0: they are unrelated
- additional assumptions (rough approximation in biology, pretty much off the mark in linguistics): substitions in different positions occur independently

The null model

- if $\vec{x}$ and $\vec{y}$ are unrelated, their joint probability equals the product of their individual probabilities
- as a start (quite wrong both in biology and in linguistics): let us assume the strings have no "grammar"; each position is independent from all other positions
- then

$$\begin{aligned} P(\vec{x}, \vec{y}|h_0) &= P(\vec{x}|h_0)P(\vec{y}|h_0) \\ &= \prod_i P(x_i|h_0)P(y_i|h_0) \\ \log P(\vec{x}, \vec{y}|h_0) &= \sum_i \log(P(x_i|h_0) + \log P(y_i|h_0)) \end{aligned}$$

The null model

- suppose $\vec{x}$ and $\vec{y}$ are generated by the same process (reasonable for DNA and protein comparison, false for cross-linguistic word comparison)
- then $P(x_i|h), P(y_i|h)$ are simply the probabilities of occurrence
- q_a: probability that symbol a occurs in a sequence

$$\log P(\vec{x}, \vec{y}|h_0) = \sum_i \log q_{x_i} + \sum_j \log q_{y_j}$$

- q can be estimated from relative frequencies

The alignment model

- suppose $\vec{x}$ and $\vec{y}$ evolved from a common ancestor via independent substitution mutations
- independence between positions:

$$P(\vec{x}, \vec{y}|h_1) = \prod_i P(x_i, y_i|h_2)$$

- $p_{a,b}$: probability that a position in the latest common ancestor of x and y evolved into an a in sequence $\vec{x}$ and into a b in sequence $\vec{y}$

$$P(\vec{x}, \vec{y}|h_1) = \prod_i p_{x_i,y_i}$$

$$\log P(\vec{x}, \vec{y}|h_1) = \sum_i \log p_{x_i,y_i}$$

The log-odds score

- taking things together, we have

$$\log(P(\vec{x},\vec{y}|h_1) : P(\vec{x},\vec{y}|h_0)) \quad = \quad \sum_i \log \frac{p_{x_i,y_i}}{q_{x_i} q_{y_i}}$$

- $\log \frac{p_{ab}}{q_a q_b}$: **score** of the alignment of a with b
- also goes by the name of **Pointwise Mutual Information** (PMI)
- assembled in a **PMI substitution matrix**

Substitution matrices

- in bioinformatics, several commonly used substitution matrices for nucleotids and proteins
- based on explicit models of evolution and careful empirical testing
- for nucleotids:

	A	G	T	C
A	2	-5	-7	-7
G	-5	2	-7	-7
T	-7	-7	2	-5
C	-7	-7	-5	2

Substitution matrices

- for proteins: different matrices for different evolutionary distances
- for instance: BLOSUM50

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	**5**	−2	−1	−2	−1	−1	−1	0	−2	−1	−2	−1	−1	−3	−1	1	0	−3	−2	0
R	−2	**7**	−1	−2	−4	1	0	−3	0	−4	−3	3	−2	−3	−3	−1	−1	−3	−1	−3
N	−1	−1	**7**	2	−2	0	0	0	1	−3	−4	0	−2	−4	−2	1	0	−4	−2	−3
D	−2	−2	2	**8**	−4	0	2	−1	−1	−4	−4	−1	−4	−5	−1	0	−1	−5	−3	−4
C	−1	−4	−2	−4	**13**	−3	−3	−3	−3	−2	−2	−3	−2	−2	−4	−1	−1	−5	−3	−1
Q	−1	1	0	0	−3	**7**	2	−2	1	−3	−2	2	0	−4	−1	0	−1	−1	−1	−3
E	−1	0	0	2	−3	2	**6**	−3	0	−4	−3	1	−2	−3	−1	−1	−1	−3	−2	−3
G	0	−3	0	−1	−3	−2	−3	**8**	−2	−4	−4	−2	−3	−4	−2	0	−2	−3	−3	−4
H	−2	0	1	−1	−3	1	0	−2	**10**	−4	−3	0	−1	−1	−2	−1	−2	−3	2	−4
I	−1	−4	−3	−4	−2	−3	−4	−4	−4	**5**	2	−3	2	0	−3	−3	−1	−3	−1	4
L	−2	−3	−4	−4	−2	−2	−3	−4	−3	2	**5**	−3	3	1	−4	−3	−1	−2	−1	1
K	−1	3	0	−1	−3	2	1	−2	0	−3	−3	**6**	−2	−4	−1	0	−1	−3	−2	−3
M	−1	−2	−2	−4	−2	0	−2	−3	−1	2	3	−2	**7**	0	−3	−2	−1	−1	0	1
F	−3	−3	−4	−5	−2	−4	−3	−4	−1	0	1	−4	0	**8**	−4	−3	−2	1	4	−1
P	−1	−3	−2	−1	−4	−1	−1	−2	−2	−3	−4	−1	−3	−4	**10**	−1	−1	−4	−3	−3
S	1	−1	1	0	−1	0	−1	0	−1	−3	−3	0	−2	−3	−1	**5**	2	−4	−2	−2
T	0	−1	0	−1	−1	−1	−1	−2	−2	−1	−1	−1	−1	−2	−1	2	**5**	−3	−2	0
W	−3	−3	−4	−5	−5	−1	−3	−3	−3	−3	−2	−3	−1	1	−4	−4	−3	**15**	2	−3
Y	−2	−1	−2	−3	−3	−1	−2	−3	2	−1	−1	−2	0	4	−3	−2	−2	2	**8**	−1
V	0	−3	−3	−4	−1	−3	−3	−4	−4	4	1	−3	1	−1	−3	−2	0	−3	−1	**5**

Substitution matrix for the ASJP data

1. identify large sample of pairs of closely related languages (using expert information or heuristics based on aggregated Levenshtein distance)

```
An.NORTHERN_PHILIPPINES.CENTRAL_BONTOC
An.MESO-PHILIPPINE.NORTHERN_SORSOGON

WF.WESTERN_FLY.IAMEGA
WF.WESTERN_FLY.GAMAEWE

Pan.PANOAN.KASHIBO_BAJO_AGUAYTIA
Pan.PANOAN.KASHIBO_SAN_ALEJANDRO

AA.EASTERN_CUSHITIC.KAMBAATA_2
AA.EASTERN_CUSHITIC.HADIYYA_2

ST.BAI.QILIQIAO_BAI_2
ST.BAI.YUNLONG_BAI

An.SULAWESI.MANDAR
An.OCEANIC.RAGA

An.SULAWESI.TANETE
An.SAMA-BAJAW.BOEPINANG_BAJAU

UA.AZTECAN.NAHUATL_HUEYAPAN_TETELA_DEL_VOLCAN
UA.AZTECAN.NAHUATL_CUENTEPEC_TEMIXCO
```

```
An.SOUTHERN_PHILIPPINES.KAGAYANEN
An.NORTHERN_PHILIPPINES.LIMOS_KALINGA

An.MESO-PHILIPPINE.CANIPAAN_PALAWAN
An.NORTHWEST_MALAYO-POLYNESIAN.LAHANAN

NC.BANTOID.LIFONGA
NC.BANTOID.BOMBOMA_2

IE.INDIC.WAD_PAGGA
IE.INDIC.TALAGANG_HINDKO

NC.BANTOID.LINGALA
NC.BANTOID.LIFONGA

An.CENTRAL_MALAYO-POLYNESIAN.BALILEDO
An.CENTRAL_MALAYO-POLYNESIAN.PALUE

AuA.MUNDA.HO
AuA.MUNDA.KORKU

MGe.GE-KAINGANG.KAYAPO
MGe.GE-KAINGANG.APINAYE
```

Substitution matrix for the ASJP data

2. pick a concept and a pair of related languages at random
   - languages: `Pen.MAIDUAN.MAIDU_KONKAU`, `Pen.MAIDUAN.NE_MAIDU`
   - concept: *one*
3. find corresponding words from the two languages:
   - `nisam`, `niSem`
4. do Levenshtein alignment

n	i	s	a	m
n	i	S	e	m

5. for each sound pair, count number of correspondences
   - nn: 1; ii: 1; sS; 1; ae: 1; mm: 1

Substitution matrix for the ASJP data

- steps 2-5 are repeated 100,000 times

```
klem   S3--v   ligini   kulox   Naltir---i   ...
klom   S37on   ji---p   Gulox   Naltirtiri   ...
```

a	a	56,047	⋮	⋮	⋮
i	i	33,955	4	8	2
u	u	23,731	4	a	2
n	n	21,363	G	t	2
o	o	19,619	i	!	2
m	m	18,263	G	y	2
t	t	16,975	d	!	2
k	k	16,773	s	G	2
e	e	12,745	Z	5	2
r	r	11,601	G	s	2
l	l	11,377	X	z	2
b	b	8,965	!	k	2
s	s	8,245	q	8	2
d	d	6,829	a	!	2
p	p	6,681	a	!	2
w	w	6,613	!	y	2
N	N	6,275	!	E	2
h	h	5,331	j	G	2
y	y	5,321	G	i	2
3	3	5.255	E	!	2

Substitution matrix for the ASJP data

6. determine relative frequency of occurrence of each sound within the entire database

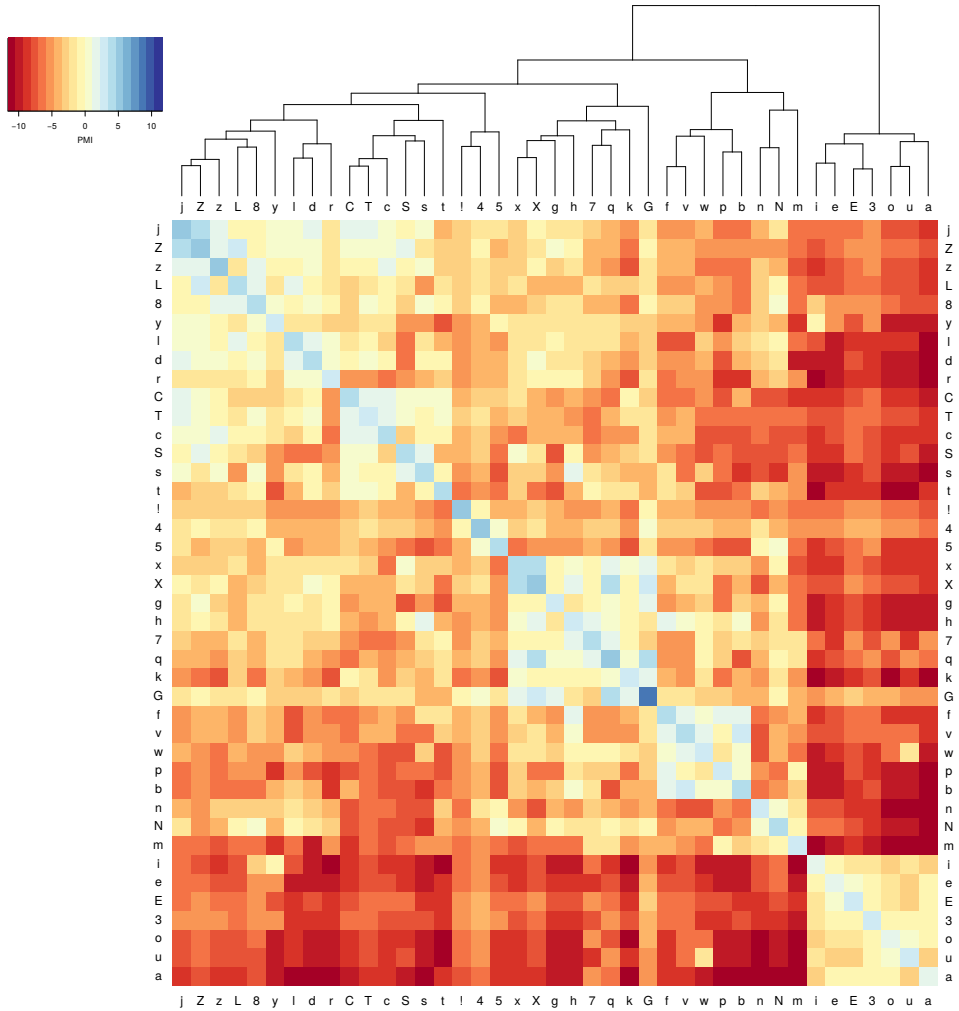
a	0.1479	E	0.0134
i	0.0969	7	0.0124
u	0.0696	C	0.0073
o	0.0626	S	0.0064
n	0.0614	x	0.0062
e	0.0478	c	0.0056
k	0.0478	f	0.0052
m	0.0465	5	0.0049
t	0.0449	v	0.0045
r	0.0346	q	0.0041
l	0.0331	z	0.0035
b	0.0248	j	0.0035
s	0.0243	T	0.0029
w	0.0232	L	0.0027
3	0.0228	X	0.0022
y	0.0222	8	0.0014
d	0.0214	Z	0.0011
h	0.0213	!	0.0009
p	0.0202	4	0.0002
N	0.0201	G	0.0001
g	0.0178		

Substitution matrix for the ASJP data

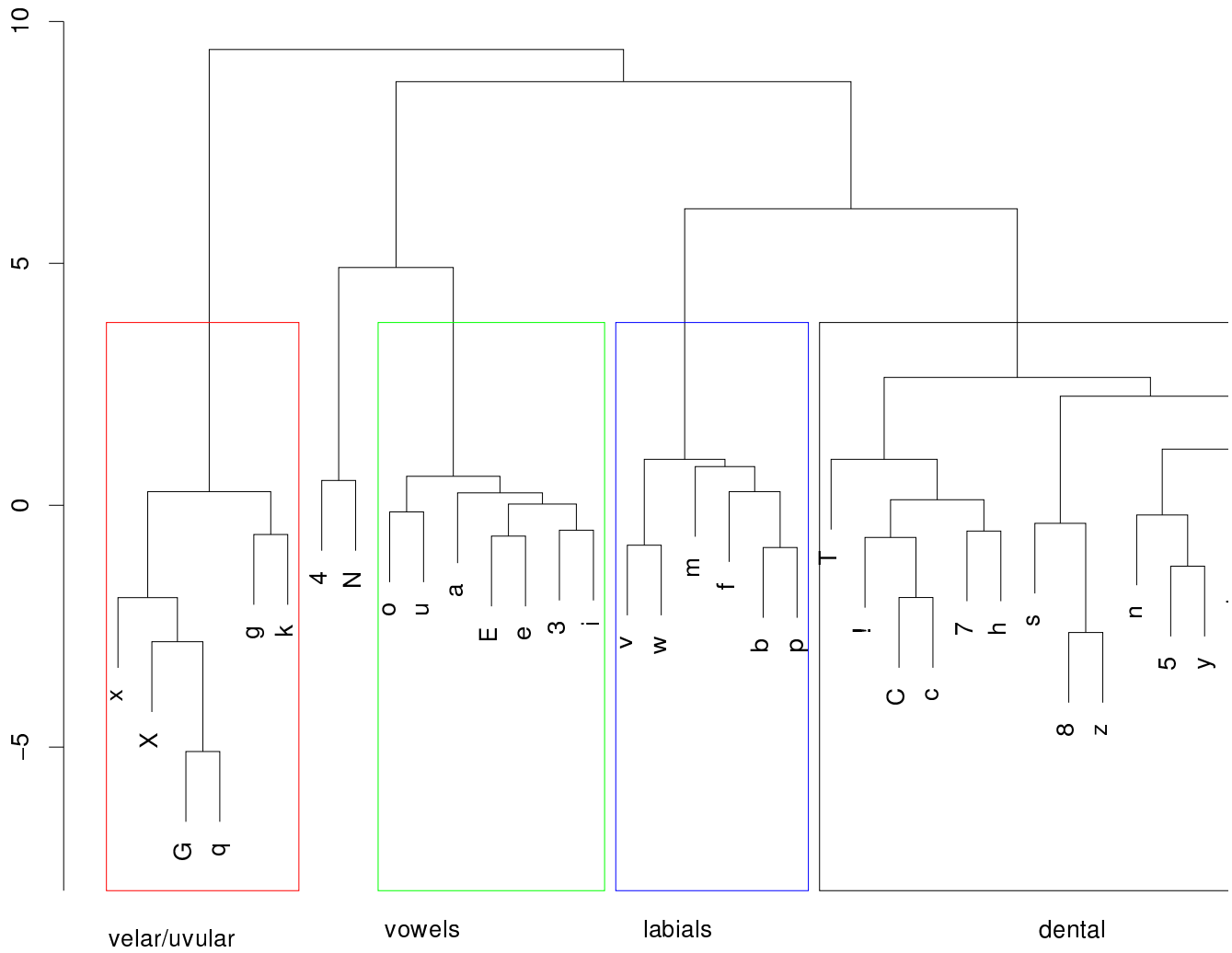
7. estimate p_{ab} as relative frequency of co-occurrence of a with b, q_a, q_b as individual relative frequencies, and determine PMI scores $\log_2 \frac{p_{ab}}{q_a q_b}$

G	G	11.2348	**Z**	**j**	**4.9386**	o	q	-3.2842
!	!	10.0202	d	d	4.9263	C	a	-3.2893
4	4	9.1480	g	g	4.8958	j	o	-3.2914
8	8	8.0650	b	b	4.8906	a	m	-3.2915
Z	Z	7.9575	s	s	4.8277	E	v	-3.3035
X	X	7.9375	**4**	**5**	**4.7508**	!	w	-3.3079
L	L	7.6276	E	E	4.7143	!	u	-3.3087
z	z	7.2624	w	w	4.6512	5	q	-3.3116
q	q	7.2542	h	h	4.5819	T	o	-3.3158
f	f	6.9117	**G**	**x**	**4.5573**	!	k	-3.3526
v	v	6.8418	**Z**	**z**	**4.4943**	e	z	-3.3763
5	5	6.7731	y	y	4.4637	!	s	-3.3788
j	j	6.7587	l	l	4.4037	...		
T	T	6.6580	**!**	**G**	**4.3760**	f	q	-3.3942
S	S	6.6054	3	3	4.3692	N	S	-3.3954
c	c	6.5989	r	r	4.3061	!	b	-3.4077
C	C	6.2439	**X**	**q**	**4.1200**	L	b	-3.4558
4	**G**	**6.1943**	m	m	4.1087	T	u	-3.4690
x	x	6.1210	t	t	4.1021	4	i	-3.5529
G	**X**	**5.3342**	**G**	**Z**	**4.0429**	5	a	-3.8294
G	**q**	**5.3017**	k	k	3.9046	C	N	-3.8451
7	7	5.2111	**X**	**x**	**3.8116**	!	t	-4.2625
p	p	5.0693	**T**	**Z**	**3.7380**	!	e	-4.3534
N	N	4.9821	**8**	**G**	**3.6993**	!	i	-4.3712
						!	a	-4.9817

Evaluation


−10
−5
0
5
10
PMI
j Z z L 8 y l d r C T c S s t ! 4 5 x X g h 7 q k G f v w p b n N m i e E 3 o u a

Evaluation


10
5
0
−5
x
X
G
q
g
k
4
N
o
u
a
E
e
3
i
v
w
m
f
b
p
T
!
C
c
7
h
s
8
z
n
5
y
velar/uvular
vowels
labials
dental

Gap penalties

- gaps in an alignment correspond either to an insertion or a deletion
- simplified assumption: insertions and deletions are equally likely at all positions; symbols are inserted according to their general frequency of occurrence
- Suppose an item x_i is aligned to a gap. Let α be the probability that an insertion occured since the latest common ancestor, and β the probability of a deletion

$$\begin{aligned} P(x_i, -|h_1) &= \alpha q_{x_i} + \beta q_{x_i} \\ P(x_i, -|h_0) &= q_{x_i} \\ \log(P(x_i, -|h_1) : P(x_i, -|h_0)) &= \log(\alpha + \beta) \\ &= -d \end{aligned}$$

- i.e., there is a constant term for each gap
- as $\alpha + \beta < 1$, this term is negative, i.e. there a constant **penalty** for each gap

Affine gap penalties

- deletions/insertions frequently apply to entire blocks of symbols (both in biology and linguistics)
- probability of a gap of length n are higher than the product of probabilities of n individual gaps
- penalty e for **extending** a gap is lower than penalty d for **opening** a gap
- g: length of a gap

$$\gamma(g) \quad = \quad -d - (g-1)e$$

- no principled way to derive the values of d and e; have to be fixed via trial and error
- $d = 2.5$ and $e = 1.6$ work quite well for the ASJP data

Weighted alignment

- so far, we assumed that the alignment between $\vec{x}$ and $\vec{y}$ is known
- to assess strength of evidence for h_1 given $\vec{x}, \vec{y}$, we need to consider all alignments between $\vec{x}$ and $\vec{y}$
- enumeration is infeasible, because the number of alignments between two sequences of length n is

$$\left(\begin{array}{c} 2n \\ n \end{array} \right) = \frac{(2n)!}{(n!)^2} \approx \frac{2^{2^n}}{\sqrt{\pi n}}$$

- computation is nonetheless possible using *Pair Hidden Markov Models*
- simpler task: find the most likely alignment and determine its log-odds!

The Needleman-Wunsch algorithm

- almost identical to Levenshtein algorithm, except:
  - matches/mismatches are counted not as 1 and 0, but as log-odds scores of the corresponding symbol pair
  - insertions/deletions are counted as gap penalties
  - by convention, the similarity rather than the distance is counted, i.e. we try to find the alignment that maximizes the score
- let $\vec{x}$ have length n, $\vec{y}$ lenth m, s_{ab} be the log-odds score of a and b, and d/e the gap penalties

The Needleman-Wunsch algorithm

$$
\begin{aligned}
F(0,0) &= 0 \\
G(0,0) &= 0 \\
\forall i: & \quad 0 < i \leq n \\
F(i,0) &= F(i-1,0) + G(i-1,0)e + (1 - G(i-1,0))d \\
G(i,0) &= 1 \\
\forall j: & \quad 0 < j \leq m: \\
F(0,j) &= F(0,j-1) + G(0,j-1)e + (1 - G(0,j-1))d \\
G(0,j) &= 1 \\
\forall i,j: & \quad 0 < i \leq n, 0 < j \leq m \\
F(i,j) &= \max \left\{ \begin{array}{l} F(i-1,j) + G(i-1,j)e + (1 - G(i-1,j))d \\ F(i,j-1) + G(i,j-1) + (1 - G(i,j-1))d \\ F(i-1,j-1) + s_{x_i y_j} \end{array} \right. \\
G(i,j) &= 0 \text{ if } \arg\max \left\{ \begin{array}{l} F(i-1,j) + G(i-1,j)e + (1 - G(i-1,j))d \\ F(i,j-1) + G(i,j-1)e + (1 - G(i,j-1))d \\ F(i-1,j-1) + s_{x_i y_j} \end{array} \right\} = 3 \\
& \quad 1 \text{ else}
\end{aligned}
$$

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5				
e	−4.1				
n	−5.7				
E	−7.3				
s	−8.9				

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5				
e	−4.1				
n	−5.7				
E	−7.3				
s	−8.9				

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5				
e	−4.1				
n	−5.7				
E	−7.3				
s	−8.9				

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13			
e	−4.1				
n	−5.7				
E	−7.3				
s	−8.9				

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13			
e	−4.1				
n	−5.7				
E	−7.3				
s	−8.9				

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13			
e	−4.1				
n	−5.7				
E	−7.3				
s	−8.9				

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53		
e	−4.1				
n	−5.7				
E	−7.3				
s	−8.9				

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	
e	−4.1				
n	−5.7				
E	−7.3				
s	−8.9				

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	−1.47
e	−4.1				
n	−5.7				
E	−7.3				
s	−8.9				

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	−1.47
e	−4.1	1.53			
n	−5.7				
E	−7.3				
s	−8.9				

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	−1.47
e	−4.1	1.53	5.65		
n	−5.7				
E	−7.3				
s	−8.9				

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	−1.47
e	−4.1	1.53	5.65	3.05	
n	−5.7				
E	−7.3				
s	−8.9				

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	−1.47
e	−4.1	1.53	5.65	3.05	1.55
n	−5.7				
E	−7.3				
s	−8.9				

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	−1.47
e	−4.1	1.53	5.65	3.05	1.55
n	−5.7	0.03			
E	−7.3				
s	−8.9				

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	−1.47
e	−4.1	1.53	5.65	3.05	1.55
n	−5.7	0.03	3.05		
E	−7.3				
s	−8.9				

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	−1.47
e	−4.1	1.53	5.65	3.05	1.55
n	−5.7	0.03	3.05	9.2	
E	−7.3				
s	−8.9				

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	−1.47
e	−4.1	1.53	5.65	3.05	1.55
n	−5.7	0.03	3.05	9.2	6.6
E	−7.3				
s	−8.9				

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	−1.47
e	−4.1	1.53	5.65	3.05	1.55
n	−5.7	0.03	3.05	9.2	6.6
E	−7.3	−1.47			
s	−8.9				

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	−1.47
e	−4.1	1.53	5.65	3.05	1.55
n	−5.7	0.03	3.05	9.2	6.6
E	−7.3	−1.47	4.75		
s	−8.9				

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	−1.47
e	−4.1	1.53	5.65	3.05	1.55
n	−5.7	0.03	3.05	9.2	6.6
E	−7.3	−1.47	4.75	6.6	
s	−8.9				

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	−1.47
e	−4.1	1.53	5.65	3.05	1.55
n	−5.7	0.03	3.05	9.2	6.6
E	−7.3	−1.47	4.75	6.6	7.62
s	−8.9				

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	−1.47
e	−4.1	1.53	5.65	3.05	1.55
n	−5.7	0.03	3.05	9.2	6.6
E	−7.3	−1.47	4.75	6.6	7.62
s	−8.9	−2.97			

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	−1.47
e	−4.1	1.53	5.65	3.05	1.55
n	−5.7	0.03	3.05	9.2	6.6
E	−7.3	−1.47	4.75	6.6	7.62
s	−8.9	−2.97	2.15		

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	−1.47
e	−4.1	1.53	5.65	3.05	1.55
n	−5.7	0.03	3.05	9.2	6.6
E	−7.3	−1.47	4.75	6.6	7.62
s	−8.9	−2.97	2.15	5.1	

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	−1.47
e	−4.1	1.53	5.65	3.05	1.55
n	−5.7	0.03	3.05	9.2	6.6
E	−7.3	−1.47	4.75	6.6	7.62
s	−8.9	−2.97	2.15	5.1	8.84

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	−1.47
e	−4.1	1.53	5.65	3.05	1.55
n	−5.7	0.03	3.05	9.2	6.6
E	−7.3	−1.47	4.75	6.6	7.62
s	−8.9	−2.97	2.15	5.1	8.84

- memorizing in each step which of the three cells to the left and above gave rise to the current entry lets us recover the corresponing optimal alignment

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	−1.47
e	−4.1	1.53	5.65	3.05	1.55
n	−5.7	0.03	3.05	9.2	6.6
E	−7.3	−1.47	4.75	6.6	7.62
s	−8.9	−2.97	2.15	5.1	8.84

- memorizing in each step which of the three cells to the left and above gave rise to the current entry lets us recover the corresponing optimal alignment

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	−1.47
e	−4.1	1.53	5.65	3.05	1.55
n	−5.7	0.03	3.05	9.2	6.6
E	−7.3	−1.47	4.75	6.6	7.62
s	−8.9	−2.97	2.15	5.1	8.84

- memorizing in each step which of the three cells to the left and above gave rise to the current entry lets us recover the corresponing optimal alignment

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	−1.47
e	−4.1	1.53	5.65	3.05	1.55
n	−5.7	0.03	3.05	9.2	6.6
E	−7.3	−1.47	4.75	6.6	7.62
s	−8.9	−2.97	2.15	5.1	8.84

- memorizing in each step which of the three cells to the left and above gave rise to the current entry lets us recover the corresponing optimal alignment

Finding the best alignment

- Dynamic Programming

	–	m	E	n	S
–	0	−2.5	−4.1	−5.7	−7.3
m	−2.5	4.13	1.53	0.03	−1.47
e	−4.1	1.53	5.65	3.05	1.55
n	−5.7	0.03	3.05	9.2	6.6
E	−7.3	−1.47	4.75	6.6	7.62
s	−8.9	−2.97	2.15	5.1	8.84

- memorizing in each step which of the three cells to the left and above gave rise to the current entry lets us recover the corresponing optimal alignment

Evaluation

left: Levenshtein alignment; right: Needleman-Wunsch alignment

```
-iX      iX-
ego      ego

du       du
tu       tu

vir      vir
nos      nos

ains     ain-s
unus     -unus

cvai     cvai
-duo     duo-

---mEnS  mEnS---
persona  persona

---fiS   fiS---
piskis   piskis

hun-t    hun-t
kanis    kanis

-----laus ------laus
pedikulus pedikul-us
```

```
-blat    b-lat
folyu    folyu

haut--   haut--
-kutis   k-utis

---blut  ---blut
saNgwis  saNgwis

knoX3n   knoX3n
--o--s   --os--

horn-    horn-
kornu    kornu

-au-g3   a-ug3-
okulus   okulus

na-z3    naz3-
nasus    nasus

chan     chan-
dens     d-ens

-chuN3   chuN--3
liNgwE   -liNgwE
```

```
han-t    han-t
manus    manus

--brust  b--rust
pektus-  pektus-

leb3r    leb3r
yekur    yekur

triNk3n  triNk3n-
-bibere  -bi-bere

--ze3n   --ze3n
widere   widere

-her3n   --her3n
audire   audire-

Sterb3n  Sterb3n
-mor--i  -mor-i-

khom3n   khom3n---
wenire   w---enire

zon3     zon3
so-l     sol-
```

Evaluation

```
vas3r          --vas3r
-akwa          akwa---

Stain          Sta-in
lapis          -lapis

-foia          fo-ia
iNnis          iNnis

pfat           p-fat
viya           viya-

bErk           bErk
mons           mons

n-at           na-t
noks           noks

---fol         fol----
plenus         p-lenus

no--i          no-i-
nowus          nowus

nam-3          nam3-
nomen          nomen
```

German — Swabian

```
'I':      0.3
iX
i

'you':    8.26
du
du

'we':    -1.09
vir
mia

'one':    4.63
ains
ois

'two':    16.0
cvai
cvoi

'person': 12.61
mEnS
mEnZE

'fish':   16.35
fiS
fiS

'louse':  15.01
laus
laus

'tree':   6.57
baum
bom

'leaf':   11.92
blat
blad

'skin':   14.42
haut
haut

'blood':  12.88
blut
blud

'bone':   16.88
knoX3n
knoXE

'horn':   8.75
horn
hoan

'tongue': 9.8
chuN3
cuN

'knee':   7.77
kni
knui

'hand':   8.6
hant
hEnd

'breast': 14.81
brust
bXuSt

'liver':  10.01
leb3r
leba

'drink':  4.99
triNk3n
dXiNg

'see':    0.63
ze3n
se

'die':    10.16
Sterb3n
StEab

'come':   11.84
khom3n
khom

'sun':    8.79
zon3
sonE

'star':   16.16
StErn
StEan

'water':  7.8
vas3r
vaza

'stone':  10.36
Stain
Stoi

'fire':   12.43
foia
fuia
```

German — English

'I': -2.3
iX
Ei

'you': 2.34
du
yu

'we': 2.21
vir
wi

'one': -2.3
ains
w3n

'two': -5.25
cvai
tu

'fish': 16.35
fiS
fiS

'dog': -7.46
hunt
dag

'tree': -7.83
baum
tri

'leaf': -0.47
blat
lif

'blood': 9.46
blut
bl3d

'bone': -1.36
knoX3n
bon

'horn': 15.73
horn
horn

'eye': -4.1
aug3
Ei

'nose': 1.63
naz3
nos

'tongue': -0.63
chuN3
t3N

'knee': 3.86
kni
ni

'hand': 8.6
hant
hEnd

'breast': 16.93
brust
brest

'liver': 14.65
leb3r
liv3r

'drink': 7.48
triNk3n
drink

'see': -3.04
ze3n
si

'die': -7.7
Sterb3n
dEi

'come': 1.22
khom3n
k3m

'sun': 1.95
zon3
s3n

'star': 8.2
StErn
star

'water': 12.06
vas3r
wat3r

'stone': 6.75
Stain
ston

'fire': 6.79
foia
fEir

German — Latin

'I': -3.87
iX
ego

'you': 3.62
du
tu

'we': -5.06
vir
nos

'one': 2.39
ains
unus

'two': -5.51
cvai
duo

'person': -4.66
mEnS
persona

'fish': 0.29
fiS
piskis

'louse': -0.08
laus
pedikulus

'tree': -3.85
baum
arbor

'leaf': -3.57
blat
folyu

'skin': -0.25
haut
kutis

'blood': -9.18
blut
saNgwis

'bone': -5.72
knoX3n
os

'horn': 7.55
horn
kornu

'nose': 4.49
naz3
nasus

'tooth': -2.78
chan
dens

'tongue': -3.4
chuN3
liNgwE

'knee': 0.8
kni
genu

'hand': 0.73
hant
manus

'breast': 1.39
brust
pektus

'liver': 5.37
leb3r
yekur

'see': -4.15
ze3n
widere

'hear': -4.24
her3n
audire

'die': -6.12
Sterb3n
mori

'come': -9.25
khom3n
wenire

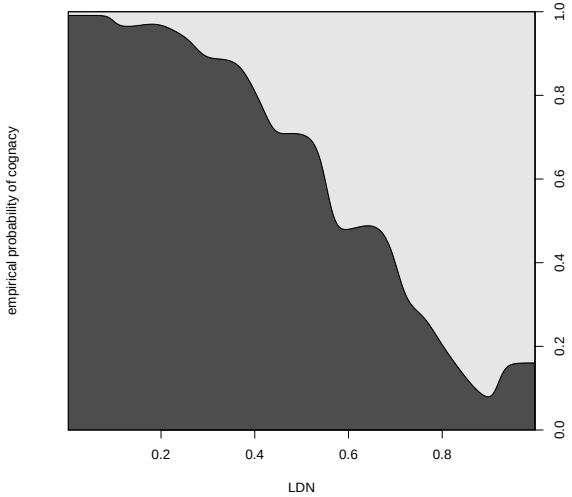
'sun': 0.97
zon3
sol

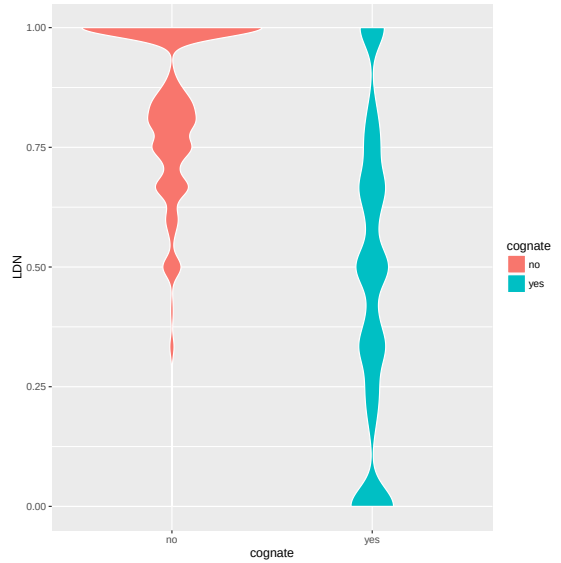
'star': 5.72
StErn
stela

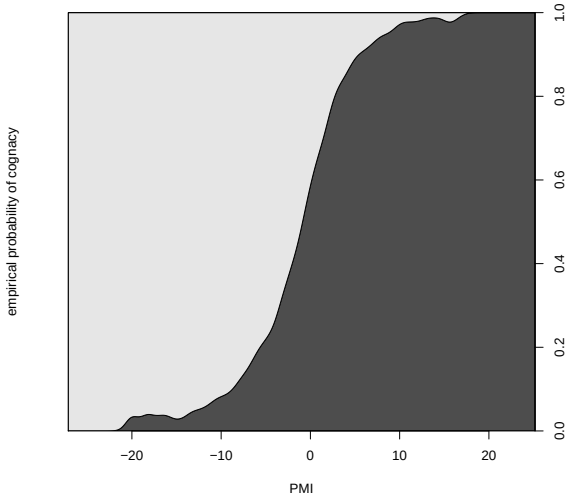
'water': -5.4
vas3r
akwa

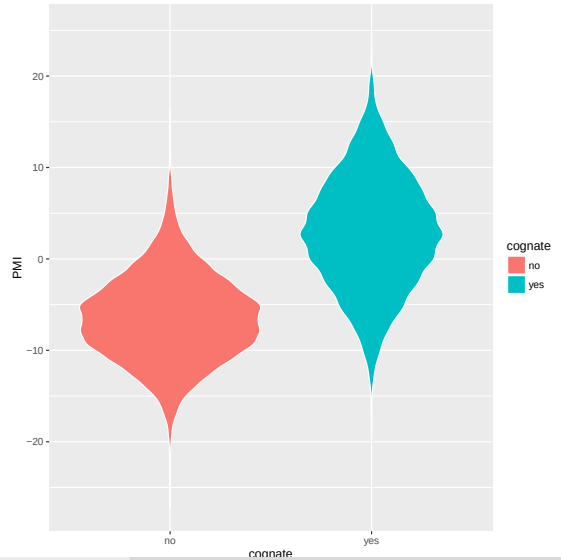
How well does PMI similarity predict cognacy?

expert cognacy judgments used as gold standard





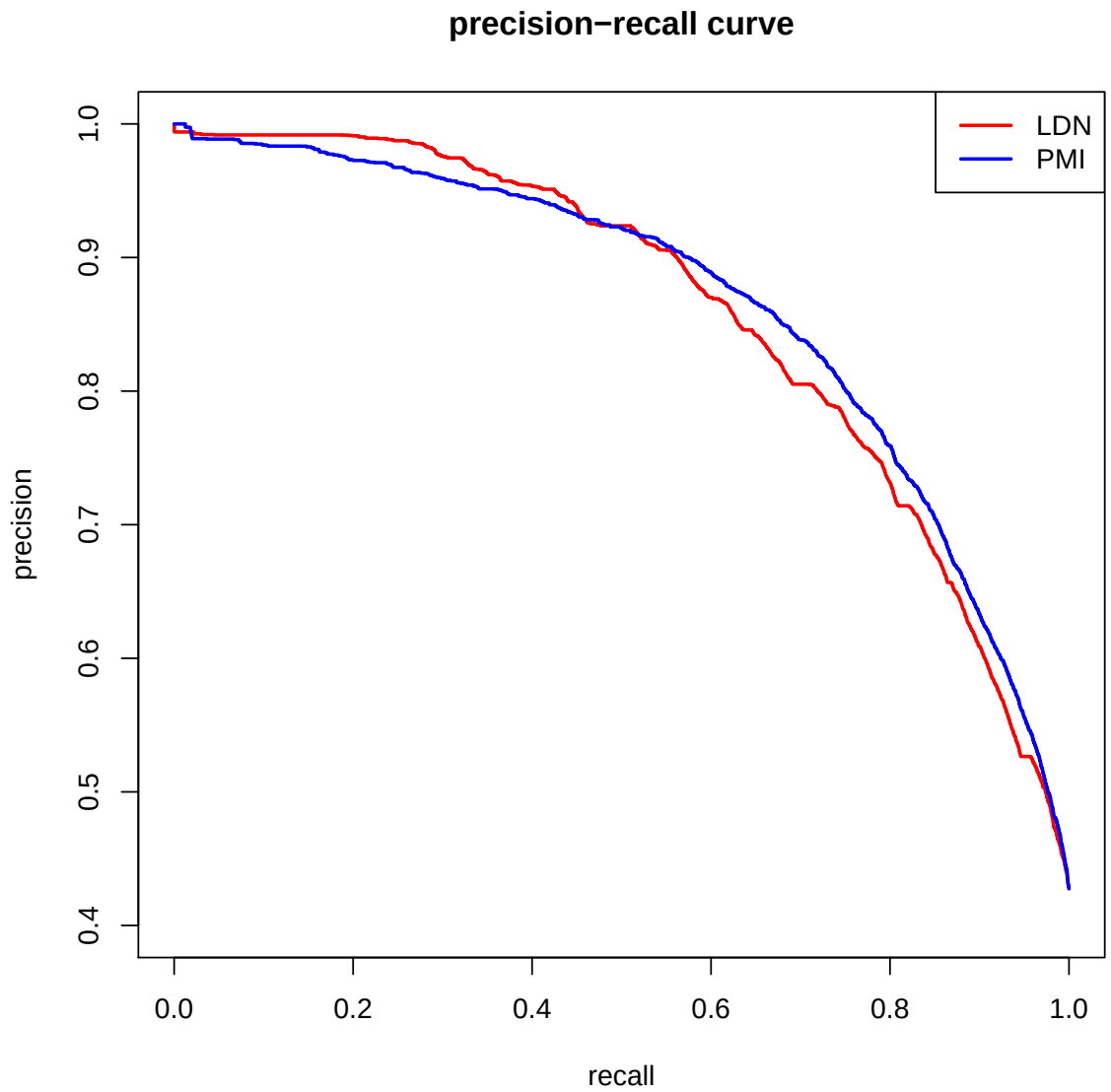




How well does PMI similarity predict cognacy?

Average Precision

- LDN: 0.847
- PMI: 0.864



Estimating distances from pairwise alignments

Probability of cognacy

- logistic regression to predict probability of cognacy from PMI similarity

concept	Italian	English	predicted prob.	expert judgment
sharp	tallEnte	Sop	0.004	0
float	galleddZare	fl3ut	0.004	0
Kill	ammattsare	kil	0.007	0
bark	skordza	bok	0.009	0
husband	marito	hozb3nd	0.010	0
walk	kamminare	wok	0.011	0
eat	mandZare	it	0.011	0
bark	kortettSa	bok	0.013	0
know	sapere	n3u	0.015	0
come	venire	kom	0.016	1
swim	nwotare	swim	0.016	0
back	dosso	bEk	0.018	0
burn	ardere	b3n	0.018	0
think	pensare	8iNk	0.019	0
dust	polvere	dost	0.019	0
wife	molle	waif	0.020	0
swell	gonfyare	swEl	0.021	0
sing	kantare	siN	0.022	0
knee	rotElla	ni	0.022	0
dry	aSSutto	drai	0.022	0
five	tSinkwe	faiv	0.023	1
skin	pElle	skin	0.024	0
hand	mano	hEnd	0.025	0
blood	sangwe	blod	0.025	0
flow	skorrere	fl3u	0.026	0
wipe	aSSugare	waip	0.026	0
turn	dZirare	t3n	0.026	0

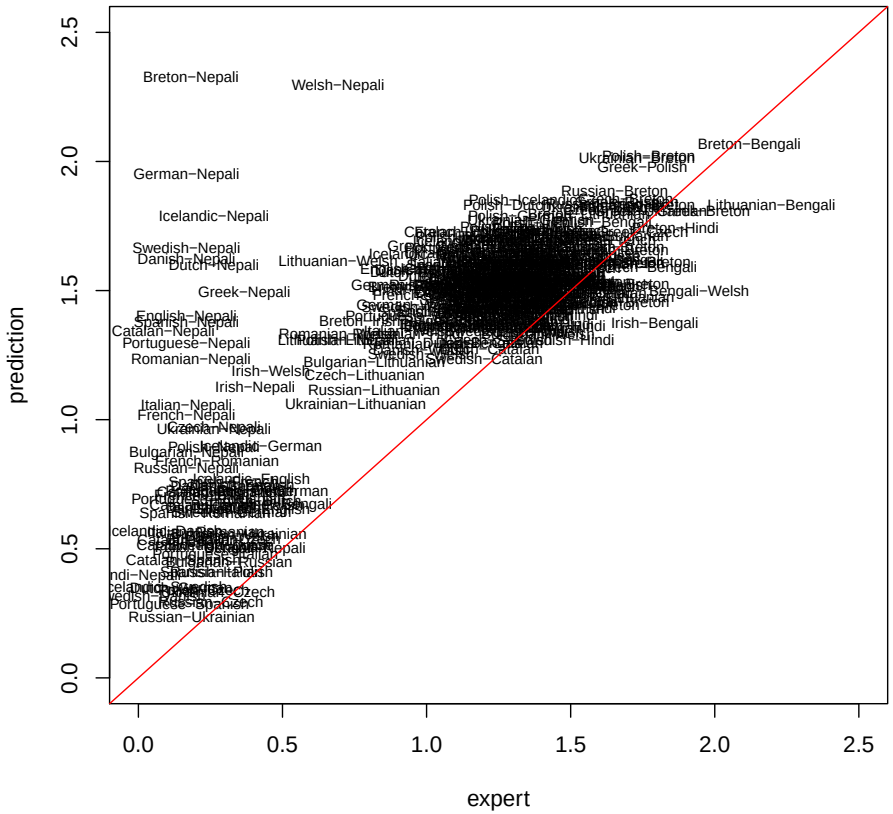
concept	Italian	English	predicted prob.	expert judgment
father	padre	fo83	0.480	1
when	kwando	wEn	0.483	1
night	notte	nait	0.508	1
and	eed	End	0.518	0
name	nome	neim	0.519	1
worm	vErme	w3m	0.521	1
round	tondo	raund	0.526	1
many	molti	mEni	0.569	0
wind	vEnto	wind	0.573	1
two	due	tu	0.600	1
mother	madre	mo83	0.624	1
thou	tu	8au	0.629	1
child	fantSullo	tSaild	0.638	0
long	lungo	loN	0.651	1
fish	peSSe	fiS	0.659	1
count	kontare	kaunt	0.660	1
star	stella	sto	0.664	1
belly	vEntre	bEli	0.679	0
sun	sole	son	0.692	1
fly	volare	flai	0.742	0
three	tre	8ri	0.744	1
flow	fluire	fl3u	0.759	0
heavy	grEve	hEvi	0.769	0
person	persona	p3s3n	0.799	1
animal	animale	Enim3l	0.947	1
vomit	vomitare	vomit	0.960	1
fruit	frutto	frut	0.966	1

Estimating distances

- average or maximal predicted probability of cognacy per concept = expected relative frequency of cognate pairs
- expected relative frequency of cognate pairs $= e^{-t}$

$\Rightarrow$ distance estimation from raw data

$\Rightarrow$ applicable across language families

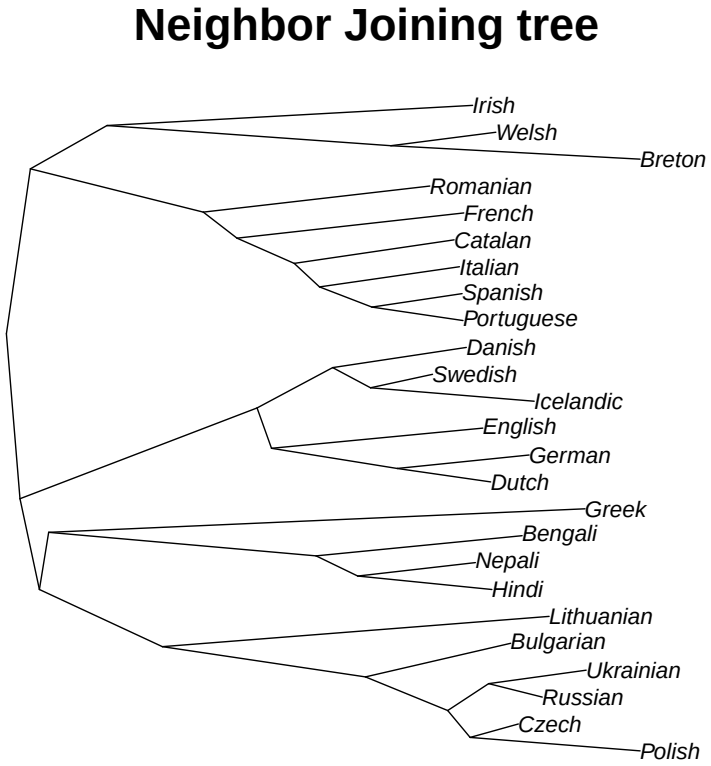


Estimating distances

- average or maximal predicted probability of cognacy per concept = expected relative frequency of cognate pairs
- expected relative frequency of cognate pairs $= e^{-t}$

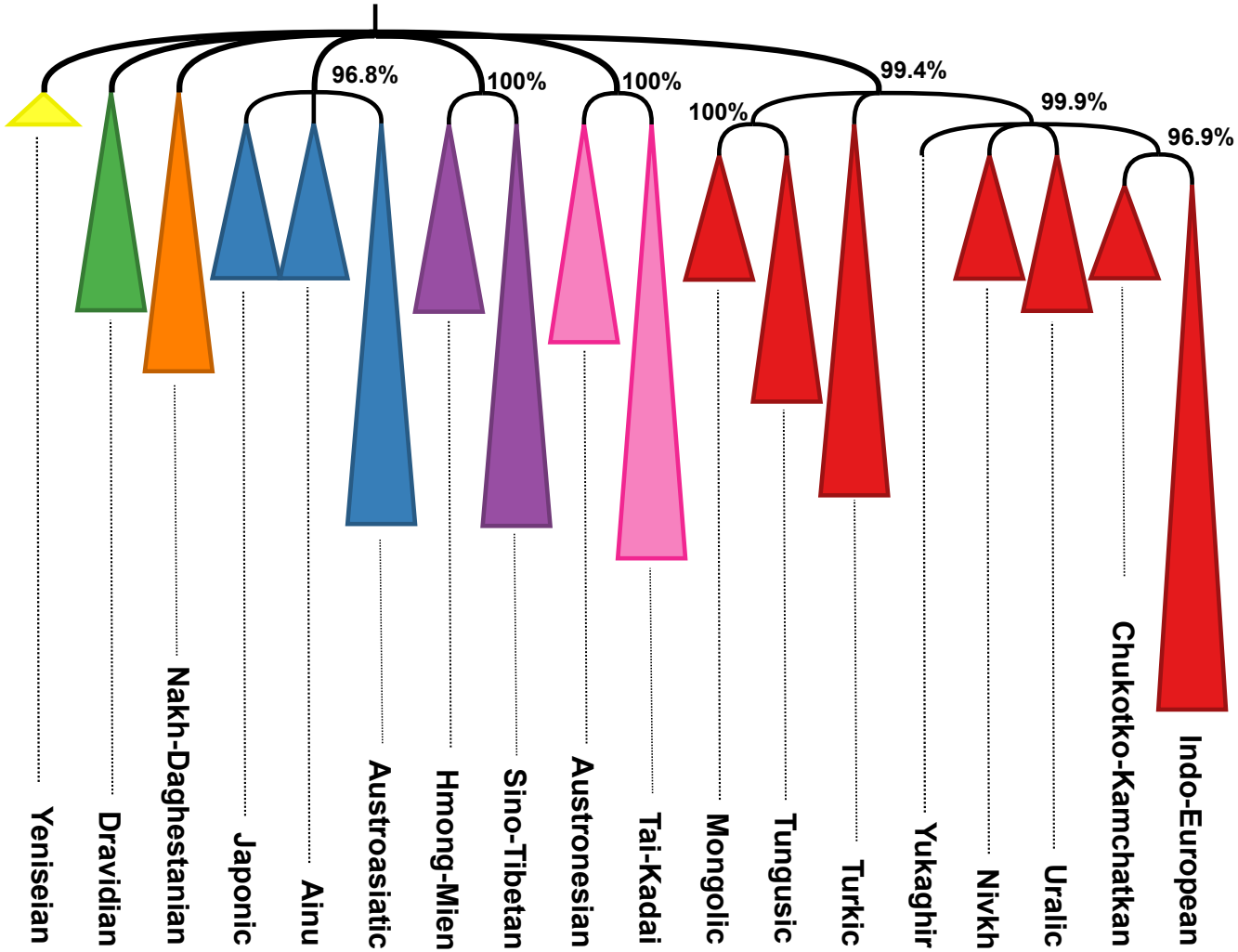
$\Rightarrow$ distance estimation from raw data

$\Rightarrow$ applicable across language families



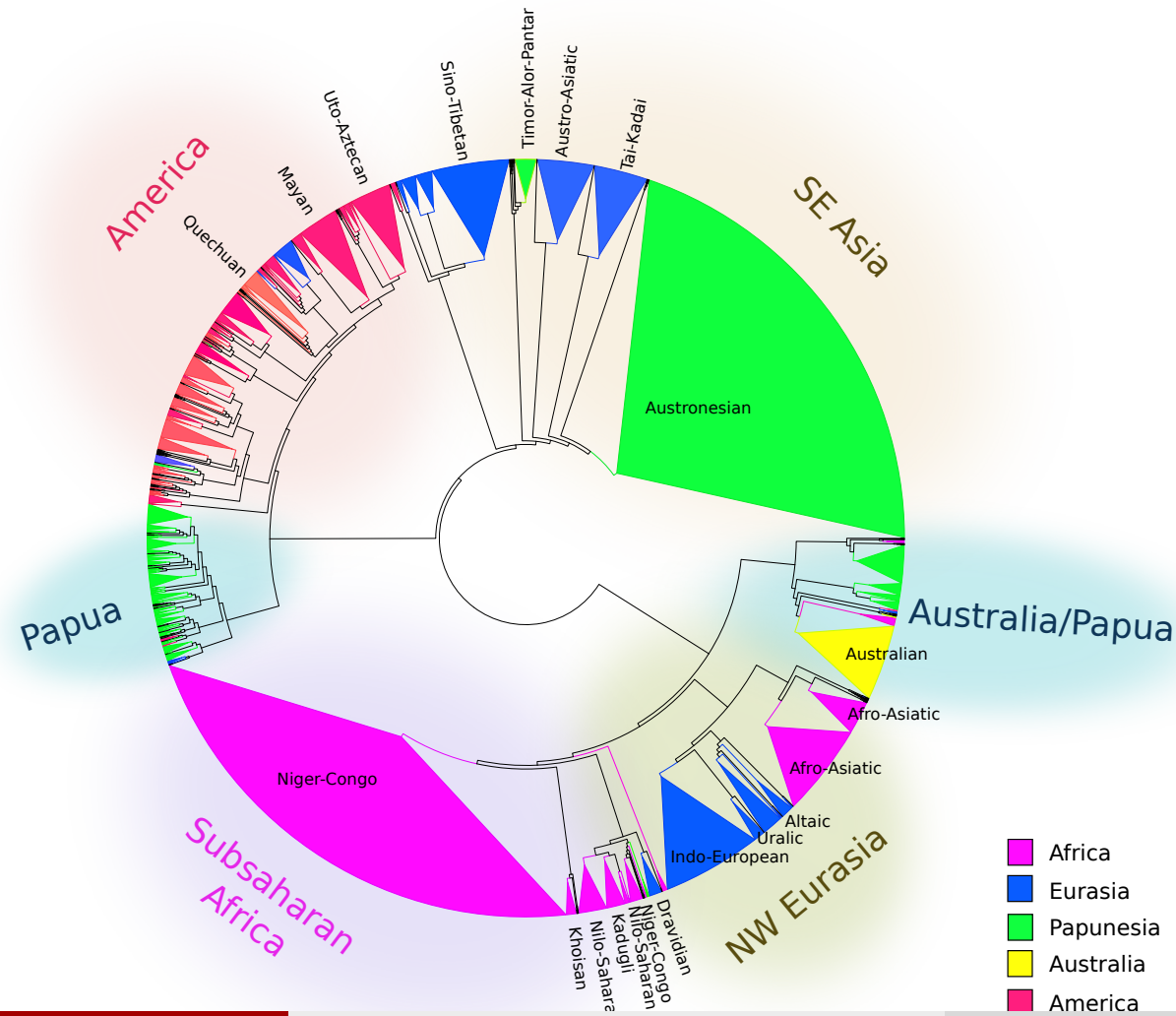
Languages of Eurasia/ ASJP data

cf. Jäger (2015); full tree can be inspected here



Languages of the World/ ASJP data

cf. Jäger and Wichmann (2016); full tree can be inspected here.



Multiple sequence alignment

Multiple sequence alignment

- Needleman-Wunsch only does pairwise alignment
- desirable: aligning all sequences of a taxon into one matrix
  - necessary for character-based phylogenetic inference
  - improves the quality of the alignment

Multiple sequence alignment

- example: 'one'
  - PIE: oinos
  - Bosian: yedan
  - Kashubian: yEdEn
  - optimal pairwise alignments:

o	i	n	o	s		o	i	n	o	s		y	e	d	a	n
y	e	d	a	n		y	E	d	E	n		y	E	d	E	n

- optimal multiple alignment (maximizing sum of pairwise similarities per column):

y	E	d	E	n	-	-
-	o	-	i	n	o	s
y	e	d	a	n	-	-

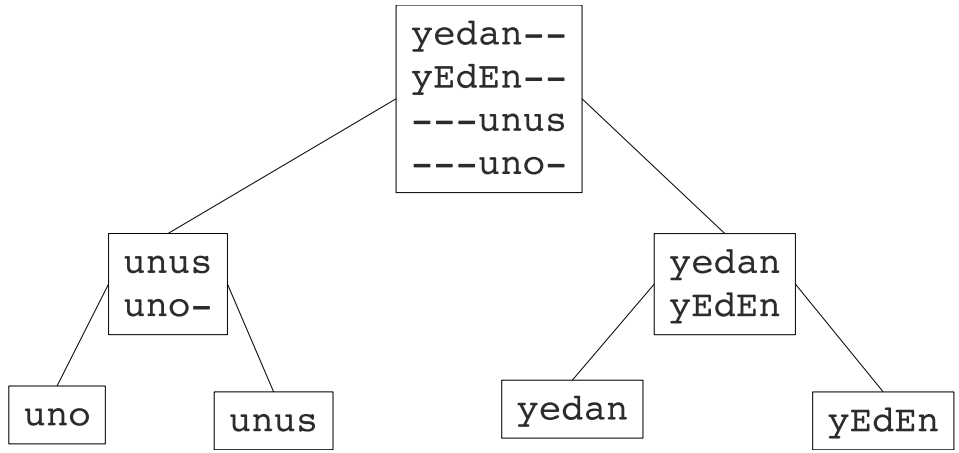
- alignment of all 'n's is etymologically correct

Multiple sequence alignment

- in principle, the Needleman-Wunsch algorithm can be generalized to aligning k sequences
- however, aligning k sequences of length n has complexity $\mathcal{O}(n^{k^2}) \Rightarrow$ computationally intractable
- two strategies
  - heuristic search
  - progressive alignment

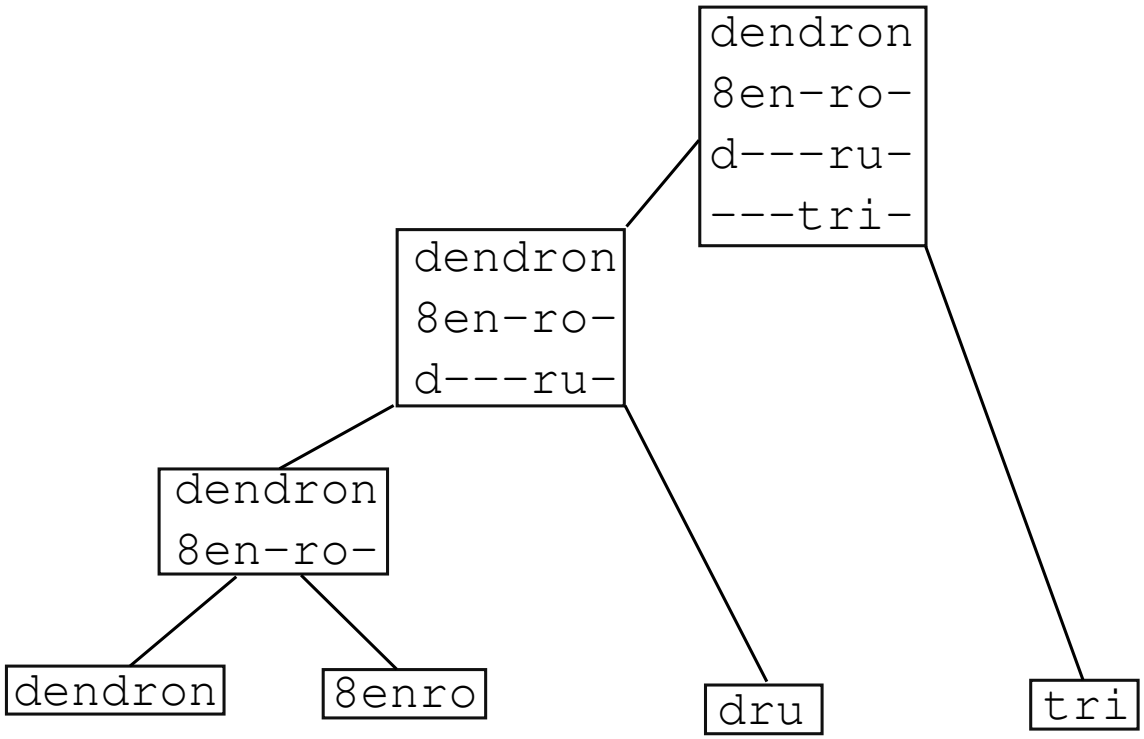
Progressive sequence alignment

- start with a **guide tree** (using some heuristics like pairwise alignment + Neighbor Joining)
- working bottom-up, at each internal node, do pairwise alignment of the block alignments at the daugher node
- complexity is $\mathcal{O}(n^2k^3) \Rightarrow$ computationally feasible



T-Coffee

- progressive alignment only uses (phylogenetically) local information
- erroneous decisions cannot be corrected later

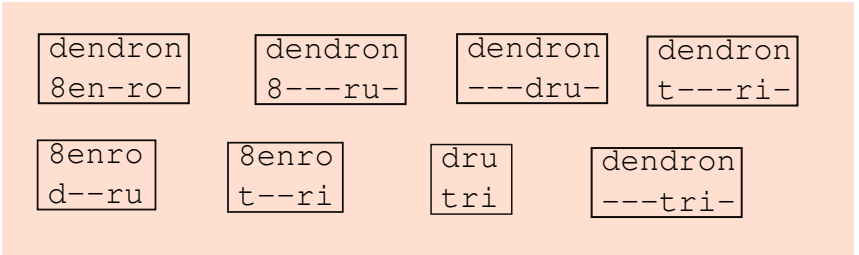


T-Coffee

Tree-based Consistency Objective Function for alignment Evaluation (Notredame et al., 2000)

(slightly adapted for linguistic application)

1. pairwise alignment for all word pairs, using PMI scores

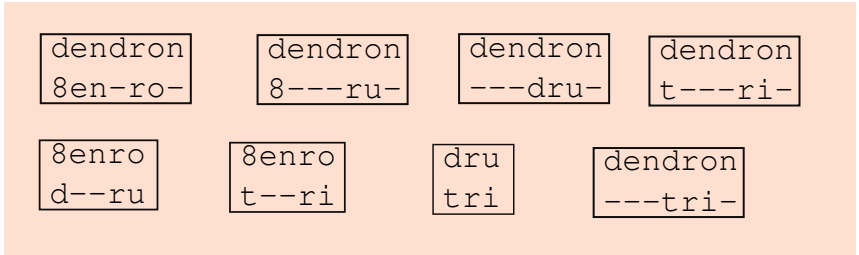


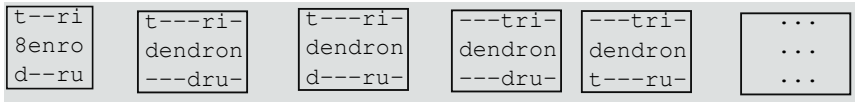
T-Coffee

Tree-based Consistency Objective Function for alignment Evaluation (Notredame et al., 2000)

(slightly adapted for linguistic application)

1. pairwise alignment for all word pairs, using PMI scores
2. ternary alignments via relation composition



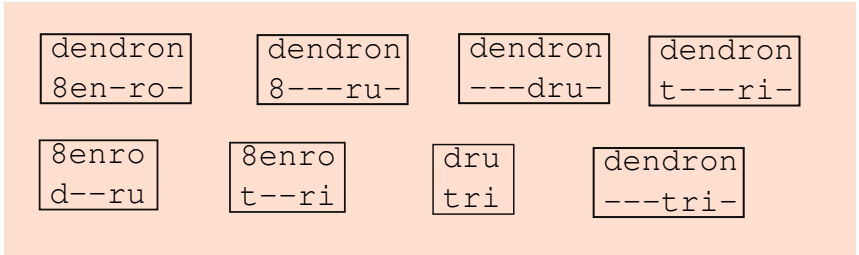


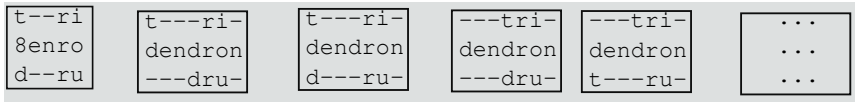
T-Coffee

Tree-based Consistency Objective Function for alignment Evaluation (Notredame et al., 2000)

(slightly adapted for linguistic application)

1. pairwise alignment for all word pairs, using PMI scores
2. ternary alignments via relation composition
3. indirect alignment scores between sound **occurrences**



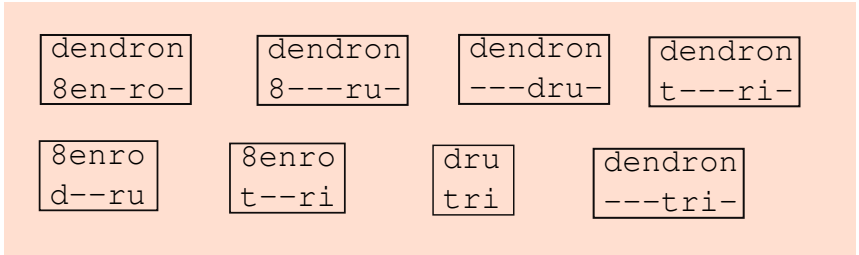


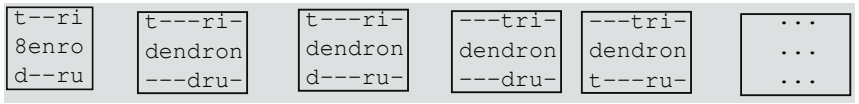
T-Coffee

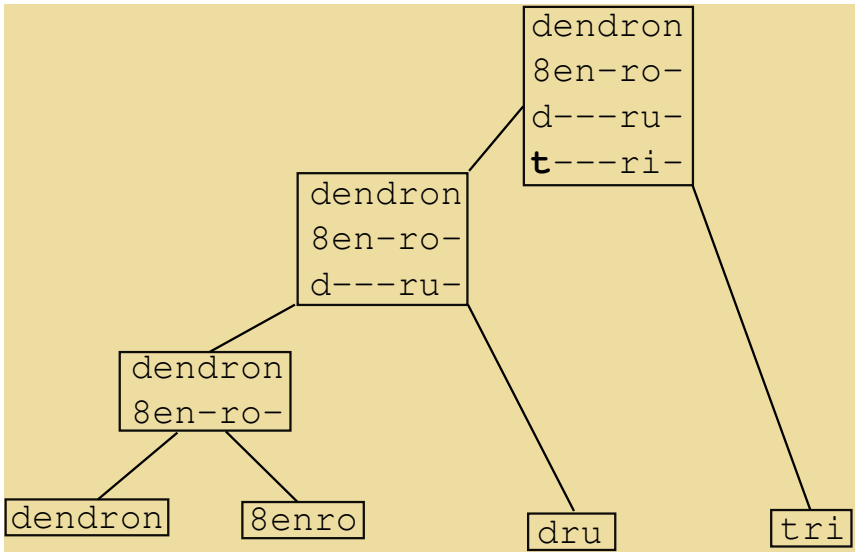
Tree-based Consistency Objective Function for alignment Evaluation (Notredame et al., 2000)

(slightly adapted for linguistic application)

1. pairwise alignment for all word pairs, using PMI scores
2. ternary alignments via relation composition
3. indirect alignment scores between sound **occurrences**
4. progressive alignment using those scores







Examples

cognate class	language	word
one:A	German	-a-i--n-
one:A	Dutch	---e--n-
one:A	English	-w-o--n-
one:A	Danish	---e--n-
one:A	Swedish	---E--n-
one:A	Icelandic	---eidn-
one:A	Irish	---e--n-
one:A	Breton	---i--n-
one:A	French	---E----
one:A	Catalan	---u--n-
one:A	Spanish	---u--no
one:A	Portuguese	---u----
one:A	Italian	---u--no
one:A	Romanian	---u--nu
one:A	Bengali	---E--k-
one:A	Nepali	---e--k-
one:A	Czech	yEdE--n-
one:A	Polish	yEdE--n-
one:A	Ukrainian	-odi--n-
one:A	Russian	-adi--n-
one:A	Bulgarian	-3di--n-

cognate class	language	word
heart:J	German	h-Er-t--s-
heart:J	Dutch	h-or-t----
heart:J	English	h-o--t----
heart:J	Danish	y-Ea-d--3-
heart:J	Swedish	y-E--t--a-
heart:J	Icelandic	S-ar-t--a-
heart:J	French	k-Er------
heart:J	Catalan	k-or------
heart:J	Spanish	k-ora8--on
heart:J	Portuguese	k-uras--aw
heart:J	Italian	kwor----e-
heart:J	Hindi	h--r-d--ai
heart:J	Lithuanian	S-ir-dis--
heart:J	Czech	s--r-t-sE-
heart:J	Polish	s-Er-t-sE-
heart:J	Ukrainian	s-Er-t-sE-
heart:J	Russian	s-Erdt-sE-
heart:J	Bulgarian	s-3r-t-sE-
heart:J	Greek	k-ar-8-Sa-

Examples

cognate class	language	word
two:A	German	tsvai-
two:A	Dutch	t-we--
two:A	English	t--u--
two:A	Danish	d--o--
two:A	Swedish	t-vo--
two:A	Icelandic	t-veir
two:A	French	d--e--
two:A	Catalan	d--o-s
two:A	Spanish	d--o-s
two:A	Portuguese	d--oiS
two:A	Italian	d--ue-
two:A	Romanian	d--o-y
two:A	Nepali	d--ui-
two:A	Czech	d-va--
two:A	Polish	d-va--
two:A	Ukrainian	d-wa--
two:A	Russian	d-va--
two:A	Bulgarian	d-va--
two:A	Greek	8-io--

cognate class	language	word
mother:A	German	mu-t--a-
mother:A	Dutch	mu-d--3r
mother:A	English	mo-8--3-
mother:A	Danish	mo----a-
mother:A	Swedish	mu-d--3r
mother:A	Icelandic	mou8--ir
mother:A	French	mE---r--
mother:A	Catalan	ma---r3-
mother:A	Spanish	ma-8-re-
mother:A	Portuguese	ma----i-
mother:A	Italian	ma-d-re-
mother:A	Czech	ma-t-ka-
mother:A	Polish	ma-t-ka-
mother:A	Ukrainian	ma-t--i-
mother:A	Russian	ma-t----
mother:A	Bulgarian	ma-y-k3-
mother:A	Greek	mi-tera-

Examples

cognate class	language	word
tongue:W	German	`---tsuN--3`
tongue:W	Dutch	`---t-oN---`
tongue:W	English	`---t-oN---`
tongue:W	Danish	`---d-oN--3`
tongue:W	Swedish	`---t-3N--a`
tongue:W	Icelandic	`---t-uNg-a`
tongue:W	French	`---l-o-g--`
tongue:W	Catalan	`---l-ENgw3`
tongue:W	Spanish	`---l-eNgwa`
tongue:W	Portuguese	`---l-i-gua`
tongue:W	Italian	`---l-ingwa`
tongue:W	Romanian	`---l-im-b3`
tongue:W	Hindi	`---dZi--b-`
tongue:W	Czech	`ya-z-ik---`
tongue:W	Polish	`yEwz-3k---`
tongue:W	Ukrainian	`ya-z-ik---`
tongue:W	Russian	`yi-z-3k---`
tongue:W	Bulgarian	`-3-z-ik---`

cognate class	language	word
tooth:B	Greek	`8-ondi`
tooth:B	German	`tsan--`
tooth:B	Dutch	`t-ont-`
tooth:B	English	`t-u-8-`
tooth:B	Danish	`d-an--`
tooth:B	Swedish	`t-and-`
tooth:B	Icelandic	`t-En--`
tooth:B	French	`d-o---`
tooth:B	Catalan	`d-en--`
tooth:B	Spanish	`dyente`
tooth:B	Portuguese	`d-e-t3`
tooth:B	Italian	`d-Ente`
tooth:B	Romanian	`d-inte`
tooth:B	Bengali	`d-o-t-`
tooth:B	Hindi	`d-a-t-`

Examples

cognate class	language	word
dog:A	Lithuanian	S-u---o-
dog:A	Ukrainian	s-obaka-
dog:A	Russian	s-abaka-
dog:A	Danish	h-u-n---
dog:A	Swedish	h-3-nd--
dog:A	Icelandic	h-i-ndir
dog:A	German	h-u-nt--
dog:A	Dutch	h-o-nt--
dog:A	Welsh	k-----i-
dog:A	Breton	k-----i-
dog:A	Irish	k-----u-
dog:A	French	S-i---E-
dog:A	Italian	k-a-n-e-
dog:A	Portuguese	k-a---u-
dog:A	Romanian	k-3yn-e-
dog:A	Greek	Tio-n---

cognate class	language	word
tree:C	Danish	d---G-E--
tree:C	Swedish	t---r-Ed-
tree:C	Icelandic	t---ryE--
tree:C	English	t---r-i--
tree:C	Ukrainian	dE--r-Ewo
tree:C	Russian	dE--r-Evo
tree:C	Polish	d---Z-Evo
tree:C	Bulgarian	d3--r--vo
tree:C	Greek	8endr---o

Wrapping up

Important topics not covered

- Bayesian tree estimation ⇒ no good introductory texts so far (that I would be aware of); best starting point might be Chen et al. (2014), esp. chapters 1 and 7
- estimation of time depths in years (rather than in "expected number of mutations") ⇒ Chang et al. (2015)
- automatic cognate detection ⇒ first part of the Jäger/List-manuscript on the course homepage

Hot research topics

- automatic discovery of regular sound correspondences and sound laws
- automatic reconstruction of proto-forms
- factoring vertical descent from language contact
- integrated probabilistic inference of sequence alignment and phylogenies

Chang, W., C. Cathcart, D. Hall, and A. Garrett (2015). Ancestry-constrained phylogenetic analysis supports the Indo-European steppe hypothesis. *Language*, **91**(1):194–244.

Chen, M.-H., L. Kuo, and P. O. Lewis (2014). *Bayesian Phylogenetics. Methods, Algorithms and Applications*. CRC Press, Abingdon.

Dolgopolsky, A. B. (1986). A probabilistic hypothesis concerning the oldest relationships among the language families of northern eurasia. In V. V. Shevoroshkin, ed., *Typology, Relationship and Time: A collection of papers on language change and relationship by Soviet linguists*, pp. 27–50. Karoma Publisher, Ann Arbor.

Jäger, G. (2015). Support for linguistic macrofamilies from weighted sequence alignment. *Proceedings of the National Academy of Sciences*, **112**(41):12752–12757. Doi: 10.1073/pnas.1500331112.

Jäger, G. and S. Wichmann (2016). Inferring the world tree of languages from word lists. In S. G. Roberts, C. Cuskley, L. McCrohon, L. Barceló-Coblijn, O. Feher, and T. Verhoef, eds., *The Evolution of Language: Proceedings of the 11th International Conference*

(EVOLANG11). Available online: http://evolang.org/neworleans/papers/147.html.

List, J.-M. (2014). *Sequence Comparison in Historical Linguistics*. Düsseldorf University Press, Düsseldorf.