On stochastic tree distances and their training via expectation-maximisation

Martin Emms

April 2, 2012

Simple edit distance

Consider transforming a sequence S into T, $S \Rightarrow T$

At any given moment an initial portion of S has been transformed into an initial portion of T, $S[0..(i-1)] \Rightarrow T[0..(j-1)]$.

Suppose the process is allowed to continue in one of 4 ways

- **delete**

Simple edit distance

Consider transforming a sequence S into T, $S \Rightarrow T$

At any given moment an initial portion of S has been transformed into an initial portion of T, $S[0..(i-1)] \Rightarrow T[0..(j-1)]$.

Suppose the process is allowed to continue in one of 4 ways

- **delete** the next symbol of S; denote this operation with $(S[i], \lambda)$, where $S[i]$ is the next symbol of S
- **insert**

Simple edit distance

Consider transforming a sequence S into T, $S \Rightarrow T$

At any given moment an initial portion of S has been transformed into an initial portion of T, $S[0..(i-1)] \Rightarrow T[0..(j-1)]$.

Suppose the process is allowed to continue in one of 4 ways

- **delete** the next symbol of S; denote this operation with $(S[i], \lambda)$, where $S[i]$ is the next symbol of S
- **insert** the next ungenerated symbol of T; denote this operation with $(\lambda, T[j])$, where $T[j]$ is the next symbol of T
- **swap**

Simple edit distance

Consider transforming a sequence S into T, $S \Rightarrow T$

At any given moment an initial portion of S has been transformed into an initial portion of T, $S[0..(i-1)] \Rightarrow T[0..(j-1)]$.

Suppose the process is allowed to continue in one of 4 ways

- **delete** the next symbol of S; denote this operation with $(S[i], \lambda)$, where $S[i]$ is the next symbol of S
- **insert** the next ungenerated symbol of T; denote this operation with $(\lambda, T[j])$, where $T[j]$ is the next symbol of T
- **swap** the next symbol of S for the next ungenerated symbol T, if these are different; denote this operation with $(S[i], T[j])$, where $S[i]$ is the next symbol of S, and $T[j]$ is the next symbol of T
- **match**

Simple edit distance

Consider transforming a sequence S into T, $S \Rightarrow T$

At any given moment an initial portion of S has been transformed into an initial portion of T, $S[0..(i-1)] \Rightarrow T[0..(j-1)]$.

Suppose the process is allowed to continue in one of 4 ways

- **delete** the next symbol of S; denote this operation with $(S[i], \lambda)$, where $S[i]$ is the next symbol of S
- **insert** the next ungenerated symbol of T; denote this operation with $(\lambda, T[j])$, where $T[j]$ is the next symbol of T
- **swap** the next symbol of S for the next ungenerated symbol T, if these are different; denote this operation with $(S[i], T[j])$, where $S[i]$ is the next symbol of S, and $T[j]$ is the next symbol of T
- **match** just skip past the next symbol of S as it is the same as the next ungenerated symbol of T; denote this also with $(S[i], T[j])$

Call the sequence of ops edit-script between S and T.

Scripts and Mappings

sold to *elder*

(s, λ)
(o, e)
(l, l)
(d, d)
(λ, e)
(λ, r)

Scripts and Mappings

sold to *elder*

(s, λ)	s
(o, e)	o
(l, l)	l
(d, d)	d
(λ, e)	
(λ, r)	

Scripts and Mappings

sold to *elder*

(s, λ)	s	
(o, e)	o	o
(l, l)	l	l
(d, d)	d	d
(λ, e)		
(λ, r)		

Scripts and Mappings

sold to *elder*

(s, λ)	s		
(o, e)	o	o	e
(l, l)	l	l	l
(d, d)	d	d	d
(λ, e)			
(λ, r)			

Scripts and Mappings

sold to *elder*

(s,λ)	s			
(o,e)	o	o	e	e
(l,l)	l	l	l	l
(d,d)	d	d	d	d
(λ,e)				
(λ,r)				

Scripts and Mappings

sold to *elder*

(s, λ)	s				
(o, e)	o	o	e	e	e
(l, l)	l	l	l	l	l
(d, d)	d	d	d	d	d
(λ, e)					
(λ, r)					

Scripts and Mappings

sold to *elder*

(s, λ)	s					
(o, e)	o	o	e	e	e	e
(l, l)	l	l	l	l	l	l
(d, d)	d	d	d	d	d	d
(λ, e)						e
(λ, r)						

Scripts and Mappings

sold to *elder*

(s, λ)	s						
(o, e)	o	o	e	e	e	e	e
(l, l)	l	l	l	l	l	l	l
(d, d)	d	d	d	d	d	d	d
(λ, e)						e	e
(λ, r)							r

Scripts and Mappings

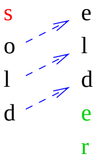
sold to *elder*

$$
\begin{array}{lccccccccl}
(s,\lambda) & s & & & & & & & & (\lambda,e) \\
(o,e) & o & o & e & e & e & e & e & & (\lambda,l) \\
(l,l) & l & l & l & l & l & l & l & & (s,d) \\
(d,d) & d & d & d & d & d & d & d & & (o,e) \\
(\lambda,e) & & & & & & e & e & & (l,\lambda) \\
(\lambda,r) & & & & & & & r & & (d,r)
\end{array}
$$

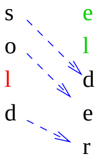
Scripts and Mappings

sold to *elder*

$$\begin{array}{l}(s,\lambda)\\(o,e)\\(l,l)\\(d,d)\\(\lambda,e)\\(\lambda,r)\end{array}$$



$$\begin{array}{l}(\lambda,e)\\(\lambda,l)\\(s,d)\\(o,e)\\(l,\lambda)\\(d,r)\end{array}$$



each script corresponds to an order preserving, partial mapping, and vice-versa

Costs for scripts or mappings

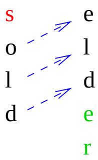
$$
\begin{array}{l}
(s, \lambda) \\
(o, e) \\
(l, l) \\
(d, d) \\
(\lambda, e) \\
(\lambda, r)
\end{array}
$$

s	e
o	l
l	d
d	e
	r

a cost table defines label-dependent costs

Costs for scripts or mappings

(s, λ)
(o, e)
(l, l)
(d, d)
(λ, e)
(λ, r)



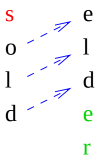
a cost table defines label-dependent costs

for example with table

	λ	a	$\cdots$	z
λ		1	$\cdots$	1
a	1	0	$\cdots$	1
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
z	1	1	$\cdots$	0

Costs for scripts or mappings

(s, λ)
(o, e)
(l, l)
(d, d)
(λ, e)
(λ, r)



a cost table defines label-dependent costs

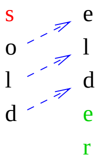
for example with table

	λ	a	$\cdots$	z
λ		1	$\cdots$	1
a	1	0	$\cdots$	1
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
z	1	1	$\cdots$	0

total cost of script or mapping is 4

Costs for scripts or mappings

(s, λ)
(o, e)
(l, l)
(d, d)
(λ, e)
(λ, r)



a cost table defines label-dependent costs

for example with table

	λ	a	......	z
λ		1	......	1
a	1	0	......	1
⋮	⋮	⋮	⋱	⋮
z	1	1	......	0

total cost of script or mapping is 4
this is also a least cost mapping/script for this table

Costs for scripts or mappings

(s, λ)
(o, e)
(l, l)
(d, d)
(λ, e)
(λ, r)

s	e
o	l
l	d
d	e
	r

a cost table defines label-dependent costs

for example with table

	λ	a	......	z
λ		1	......	1
a	1	0	......	1
⋮	⋮	⋮	⋱	⋮
z	1	1	......	0

total cost of script or mapping is 4
this is also a least cost mapping/script for this table

Definition

(*Sequence-distance*) between $\mathcal{S}$ and $\mathcal{T}$ is the cost of **the least-costly mapping/scirpt** from $\mathcal{S}$ to $\mathcal{T}$

Tree edits

a trees S can be transformed into a tree T, by delete, insert, swap/match operations

Tree edits

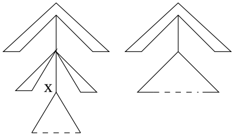
a trees S can be transformed into a tree T, by delete, insert, swap/match operations

delete

Tree edits

a trees S can be transformed into a tree T, by delete, insert, swap/match operations

delete



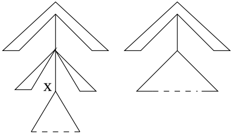
dtrs of x made dtrs of x's parent m

insert

Tree edits

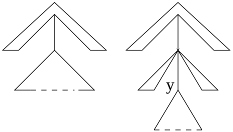
a trees S can be transformed into a tree T, by delete, insert, swap/match operations

delete



dtrs of x made dtrs of x's parent m

insert



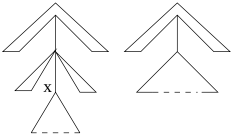
some dtrs of m made dtrs new daughter y of m

swap/match

Tree edits

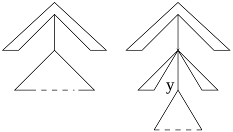
a trees S can be transformed into a tree T, by delete, insert, swap/match operations

delete



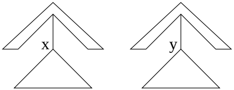
dtrs of x made dtrs of x's parent m

insert



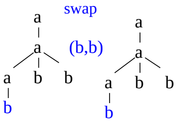
some dtrs of m made dtrs new daughter y of m

swap/match

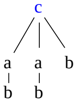


node x turned to node y

Example

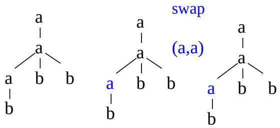


S

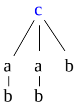


T

Example

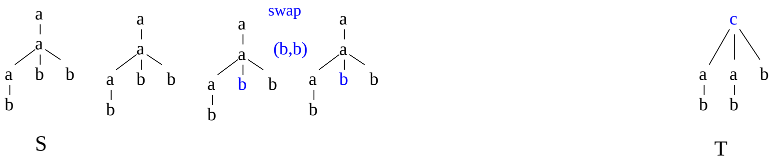


S

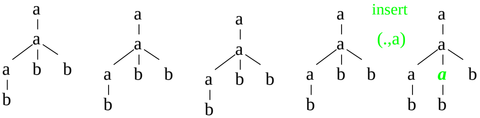


T

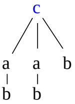
Example



Example

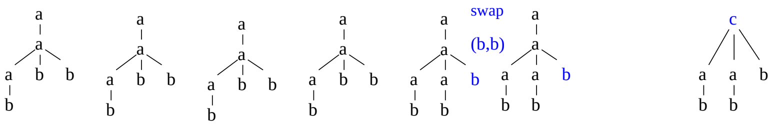


S



T

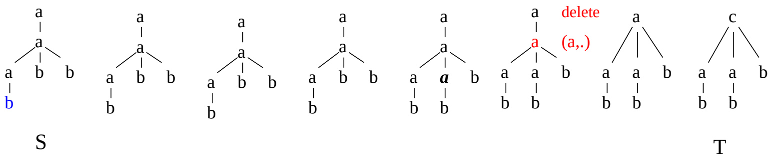
Example



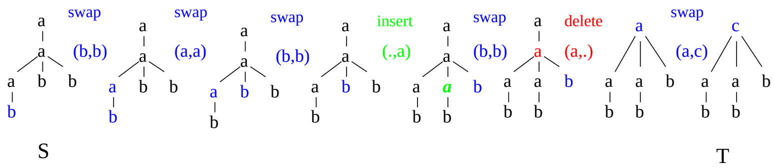
S

T

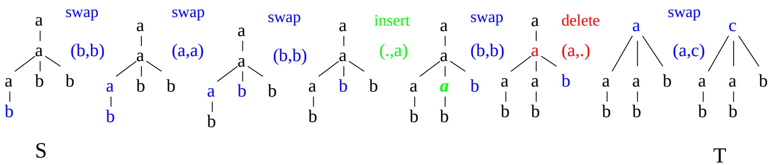
Example



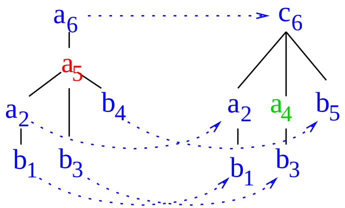
Example



Example

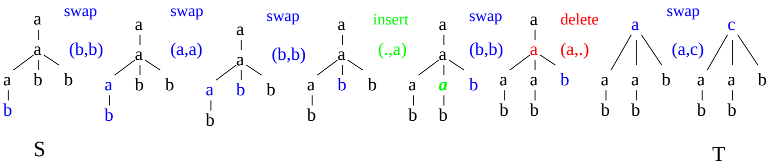


The script encodes a partial mapping $\sigma : \mathcal{S} \mapsto \mathcal{T}$

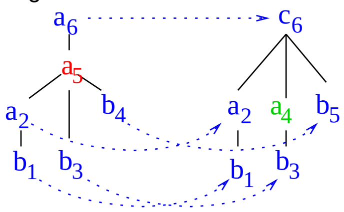


it is a mapping which respects left-to-right order and ancestry – call such mappings Tai mappings

Example



The script encodes a partial mapping $\sigma : \mathcal{S} \mapsto \mathcal{T}$



it is a mapping which respects left-to-right order and ancestry – call such mappings Tai mappings

costs can be assigned to scripts or mappings

Definition

(*Tree- or Tai-distance*) between $\mathcal{S}$ and $\mathcal{T}$ is the cost of **the least-costly Tai mapping (or script)** from $\mathcal{S}$ to $\mathcal{T}$

Stochastic string distances

- for the case of strings (linear trees), a stochastic variant was first proposed by Ristad and Yianilos (98)

Stochastic string distances

- for the case of strings (linear trees), a stochastic variant was first proposed by Ristad and Yianilos (98)
- where Σ is an alphabet, let *edit operation identifiers*, *EdOp*, be:

$$EdOp = ((\Sigma \cup \{\lambda\}) \times (\Sigma \cup \{\lambda\})) \setminus \langle \lambda, \lambda \rangle$$

  and represent a script with $op_1 \ldots op_n \#$, with each $op_i \in EdOp$.
- assuming a prob distribution p on $EdOp \cup \{\#\}$, define a script probability as

$$P(e_1 \ldots e_n) = \prod_i p(e_i)$$

Stochastic string distances

- for the case of strings (linear trees), a stochastic variant was first proposed by Ristad and Yianilos (98)
- where Σ is an alphabet, let *edit operation identifiers*, *EdOp*, be:

$$EdOp = ((\Sigma \cup \{\lambda\}) \times (\Sigma \cup \{\lambda\})) \backslash \langle \lambda, \lambda \rangle$$

  and represent a script with $op_1 \dots op_n \#$, with each $op_i \in EdOp$.
- assuming a prob distribution p on $EdOp \cup \{\#\}$, define a script probability as

$$P(e_1 \dots e_n) = \prod_i p(e_i)$$

- Can think of a script as *yielding* a pair of strings (s, t). If $E(s, t)$ is all scripts which yield (s, t), they defined

  all-paths stochastic edit distance:
  the sum of the probabilities of all scripts $e \in E(s, t)$

  viterbi stochastic edit distance:
  prob. of the most probable $e \in E(s, t)$

Stochastic tree distances

this can be adapted to the case of trees (first proposed by Boyer et al 2007)

Stochastic tree distances

this can be adapted to the case of trees (first proposed by Boyer et al 2007)

Definition (All-scripts stochastic Tai similarity/distance)

The all-scripts stochastic Tai similarity, $\Theta^A_s(S,T)$, is the sum of the probabilities of all edit-scripts which represent a *Tai*-mapping from S to T. The all-scripts stochastic Tai distance, $\Delta^A_s(S,T)$, is its negated logarithm, ie.

$$2^{-\Delta^A_s(S,T)} = \Theta^A_s(S,T)$$

Stochastic tree distances

this can be adapted to the case of trees (first proposed by Boyer et al 2007)

Definition (All-scripts stochastic Tai similarity/distance)

The all-scripts stochastic Tai similarity, $\Theta^A_s(S,T)$, is the sum of the probabilities of all edit-scripts which represent a *Tai*-mapping from S to T. The all-scripts stochastic Tai distance, $\Delta^A_s(S,T)$, is its negated logarithm, ie.

$$2^{-\Delta^A_s(S,T)} = \Theta^A_s(S,T)$$

Definition (Viterbi-script stochastic Tai similarity/distance)

The Viterbi-script stochastic Tai similarity, $\Theta^V_s(S,T)$, is the probability of the most probable edit-script which represents a *Tai*-mapping from S to T. The Viterbi-script stochastic Tai distance, $\Delta^V_s(S,T)$, is its negated logarithm, ie.

$$2^{-\Delta^V_s(S,T)} = \Theta^V_s(S,T)$$

EM Cost adaptation

EM Cost adaptation

- a possible use of a distance is k-NN classifier:

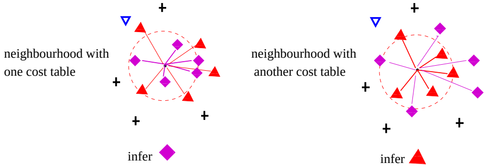
  cat(S) = *VOTE*($\{$*categories of k* **nearest neighbours** *of S* $\}$)

EM Cost adaptation

- a possible use of a distance is k-NN classifier:

  $$\mathit{cat}(S) = \mathit{VOTE}(\{\mathit{categories\ of\ k}\ \textbf{nearest neighbours}\ \mathit{of\ S}\ \})$$

- change cost table $\Rightarrow$ change nearest neighbours $\Rightarrow$ change categorisation:

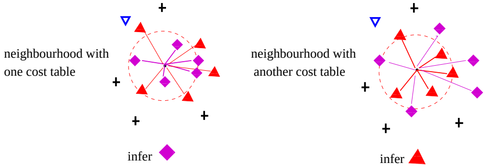


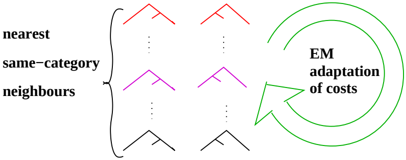
EM Cost adaptation

- a possible use of a distance is k-NN classifier:

  $$cat(S) = VOTE(\{categories\ of\ k\ \textbf{nearest neighbours}\ of\ S\ \})$$

- change cost table $\Rightarrow$ change nearest neighbours $\Rightarrow$ change categorisation:





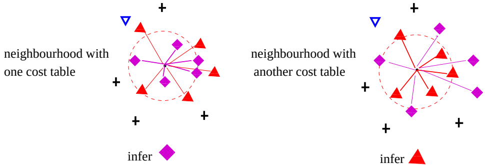
scripts between between *same-category neighbours should have distinctive probs*

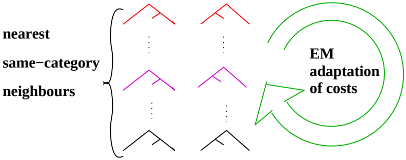
EM Cost adaptation

- a possible use of a distance is k-NN classifier:

  $cat(S)$ = *VOTE({categories of k* **nearest neighbours** *of S })*

- change cost table ⇒ change nearest neighbours ⇒ change categorisation:





scripts between between *same-category neighbours should have distinctive probs* ⇒ perhaps can use *Expectation-Maximisation techniques to adapt edit-probs from a corpus of same-category nearest neighbours*

Outline

Brute force All-paths EM (infeasible)

Given a corpus of trainings pairs $\mathcal{TP} = \ldots(S, T)\ldots$, let the *brute-force all-scripts EM algorithm*, EM^{A}_{bf}, be iterations of pair of steps

Brute force All-paths EM (infeasible)

Given a corpus of trainings pairs $\mathcal{TP} = \ldots(S, T)\ldots$, let the *brute-force all-scripts EM algorithm*, EM^A_{bf}, be iterations of pair of steps

(Exp)$_A$ *generate a virtual corpus of scripts by treating each training pair* (S, T) *as standing for* all *the edit-scripts* σ*, which can relate* S *to* T*, weighting each by its conditional probability* $P(\sigma/\Theta^A_s(S, T)$*, under current probalities* C^Θ

Brute force All-paths EM (infeasible)

Given a corpus of trainings pairs $\mathcal{TP} = \ldots (S, T) \ldots$, let the *brute-force all-scripts EM algorithm*, EM^A_{bf}, be iterations of pair of steps

(Exp)$_A$ *generate a virtual corpus of scripts by treating each training pair* (S, T) *as standing for* all *the edit-scripts* σ*, which can relate* S *to* T*, weighting each by its conditional probability* $P(\sigma / \Theta^A_s(S, T)$*, under current probabilities* C^Θ

(Max) *apply maximum likelihood estimation to the virtual corpus to derive a new probability table.*

Brute force All-paths EM (infeasible)

Given a corpus of trainings pairs $\mathcal{TP} = \ldots (S, T) \ldots$, let the *brute-force all-scripts EM algorithm*, EM^A_{bf}, be iterations of pair of steps

(Exp)$_A$ *generate a virtual corpus of scripts by treating each training pair (S, T) as standing for* all *the edit-scripts σ, which can relate S to T, weighting each by its conditional probability $P(\sigma/\Theta^A_s(S, T))$, under current probalities C^Θ*

(Max) *apply maximum likelihood estimation to the virtual corpus to derive a new probability table.*

A virtual count or expectation $\gamma_{S,T}(op)$ contributed by S, T for an operaton *op* can be defined by

$$\gamma_{S,T}(op) = \sum_{\sigma : S \mapsto T} [\frac{P(\sigma)}{\Theta^A_s(S, T)} \times freq(op \in \sigma)]$$

Brute force All-paths EM (infeasible)

Given a corpus of trainings pairs $\mathcal{TP} = \ldots(S,T)\ldots$, let the *brute-force all-scripts EM algorithm*, EM^{A}_{bf}, be iterations of pair of steps

(Exp)$_A$ *generate a virtual corpus of scripts by treating each training pair* (S,T) *as standing for* all *the edit-scripts* σ*, which can relate* S *to* T*, weighting each by its conditional probability* $P(\sigma/\Theta^{A}_{s}(S,T)$*, under current probalities* C^{Θ}

(Max) *apply maximum likelihood estimation to the virtual corpus to derive a new probability table.*

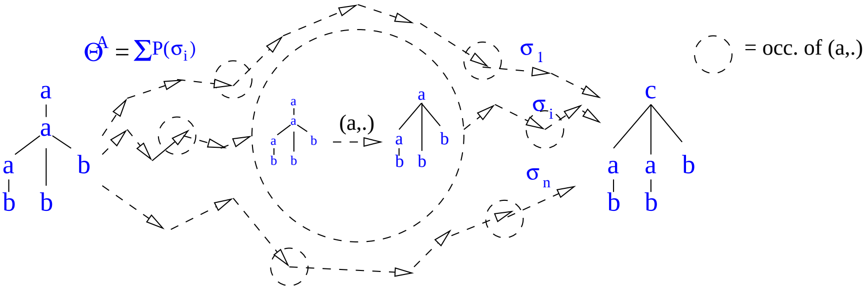
A virtual count or expectation $\gamma_{S,T}(op)$ contributed by S,T for an operaton *op* can be defined by

$$\gamma_{S,T}(op) = \sum_{\sigma:S\mapsto T}[\frac{P(\sigma)}{\Theta^{A}_{s}(S,T)} \times freq(op \in \sigma)]$$

(Exp)$_A$ accumulates the $\gamma_{S,T}(op)$ for all *op*'s, for all (S,T)

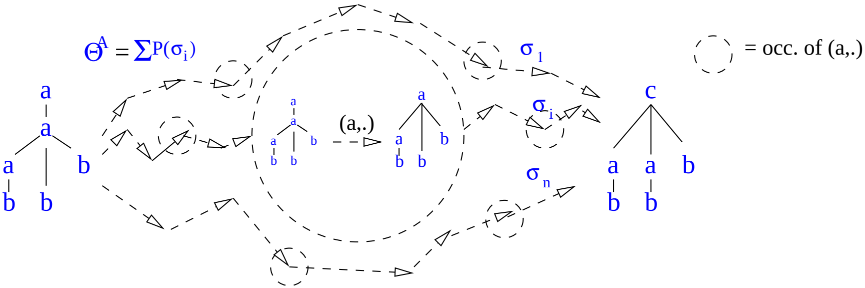
Brute force All-paths EM (infeasible)

Brute force All-paths EM (infeasible)



$$\gamma_{S,T}(op) = \sum_{\sigma:S\mapsto T} [\frac{P(\sigma)}{\Theta^A_s(S,T)} \times freq(op \in \sigma)]$$

Brute force All-paths EM (infeasible)



$$\gamma_{S,T}(op) = \sum_{\sigma:S\mapsto T}\left[\frac{P(\sigma)}{\Theta^A_S(S,T)} \times freq(op \in \sigma)\right]$$

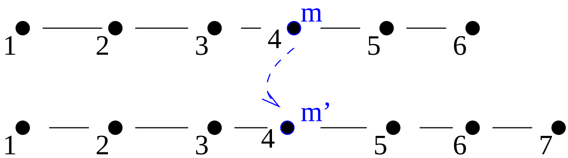
- **infeasible**

try to split exp. $\gamma_{(S,T)}(op)$ into position specific versions $\gamma_{(S,T)}[i,j](op)$ and then sum

$$\gamma_{(S,T)}(op) = \sum_{i,j} \gamma_{(S,T)}[i][j](op)$$

try to split exp. $\gamma_{(S,T)}(op)$ into position specific versions $\gamma_{(S,T)}[i,j](op)$ and then sum

$$\gamma_{(S,T)}(op) = \sum_{i,j} \gamma_{(S,T)}[i][j](op)$$

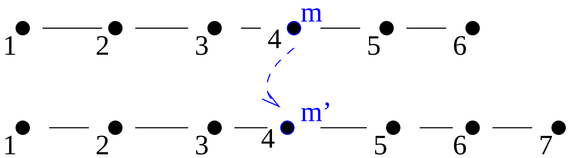


Define $(m_i, m'_j) \in \sigma$, occurrence of posn-specific subst (m, m') in σ as

$$(m_i, m'_j) \in \sigma \text{ if } \sigma = pre \circ (m, m') \circ suff \quad \begin{cases} \text{some } pre \in E(S_{1:i-1}, T_{1:j-1}) \\ \text{some } suff \in E(S_{i+1:I}, T_{j+1:J}) \end{cases}$$

try to split exp. $\gamma_{(S,T)}(op)$ into position specific versions $\gamma_{(S,T)}[i,j](op)$ and then sum

$$\gamma_{(S,T)}(op) = \sum_{i,j} \gamma_{(S,T)}[i][j](op)$$



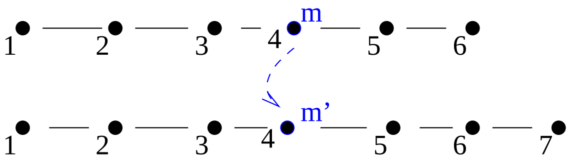
Define $(m_i, m'_j) \in \sigma$, occurrence of posn-specific subst (m, m') in σ as

$$(m_i, m'_j) \in \sigma \text{ if } \sigma = pre \circ (m, m') \circ suff \quad \begin{cases} \text{some } pre \in E(S_{1:i-1}, T_{1:j-1}) \\ \text{some } suff \in E(S_{i+1:I}, T_{j+1:J}) \end{cases}$$

then define $\gamma_{(S,T)}[4][4](m, m')$, the expectation for a swap (m, m') at $(4, 4)$ as

try to split exp. $\gamma_{(S,T)}(op)$ into position specific versions $\gamma_{(S,T)}[i,j](op)$ and then sum

$$\gamma_{(S,T)}(op) = \sum_{i,j} \gamma_{(S,T)}[i][j](op)$$



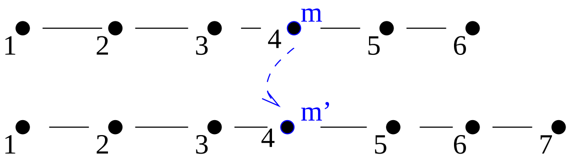
Define $(m_i, m'_j) \in \sigma$, occurrence of posn-specific subst (m, m') in σ as

$$(m_i, m'_j) \in \sigma \text{ if } \sigma = pre \circ (m, m') \circ suff \quad \begin{cases} \text{some } pre \in E(S_{1:i-1}, T_{1:j-1}) \\ \text{some } suff \in E(S_{i+1:I}, T_{j+1:J}) \end{cases}$$

then define $\gamma_{(S,T)}[4][4](m, m')$, the expectation for a swap (m, m') at $(4, 4)$ as

try to split exp. $\gamma_{(S,T)}(op)$ into position specific versions $\gamma_{(S,T)}[i,j](op)$ and then sum

$$\gamma_{(S,T)}(op) = \sum_{i,j} \gamma_{(S,T)}[i][j](op)$$



Define $(m_i, m'_j) \in \sigma$, occurrence of posn-specific subst (m, m') in σ as

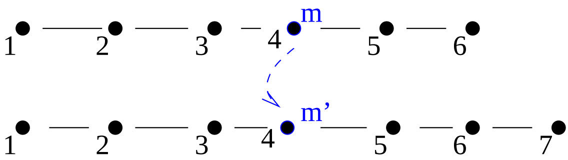
$$(m_i, m'_j) \in \sigma \text{ if } \sigma = pre \circ (m, m') \circ suff \quad \begin{cases} \text{some } pre \in E(S_{1:i-1}, T_{1:j-1}) \\ \text{some } suff \in E(S_{i+1:I}, T_{j+1:J}) \end{cases}$$

then define $\gamma_{(S,T)}[4][4](m, m')$, the expectation for a swap (m, m') at $(4, 4)$ as in words,

> the sum over the conditional probabilities of any script σ containing a m_4, m'_4 substitution, given that it is a script between S and T

try to split exp. $\gamma_{(S,T)}(op)$ into position specific versions $\gamma_{(S,T)}[i,j](op)$ and then sum

$$\gamma_{(S,T)}(op) = \sum_{i,j} \gamma_{(S,T)}[i][j](op)$$

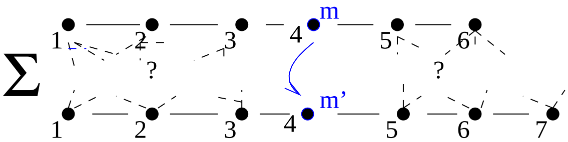


Define $(m_i, m'_j) \in \sigma$, occurrence of posn-specific subst (m, m') in σ as

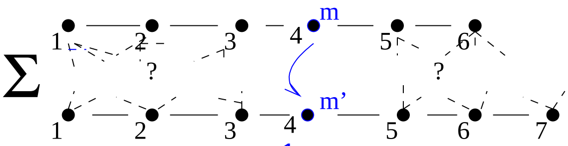
$$(m_i, m'_j) \in \sigma \text{ if } \sigma = pre \circ (m, m') \circ suff \quad \begin{cases} \text{some } pre \in E(S_{1:i-1}, T_{1:j-1}) \\ \text{some } suff \in E(S_{i+1:I}, T_{j+1:J}) \end{cases}$$

then define $\gamma_{(S,T)}[4][4](m, m')$, the expectation for a swap (m, m') at $(4, 4)$ as

$$\begin{aligned} \gamma_{(S,T)}[4,4](m,m') &= \sum_{\sigma \in E(S,T), (m_4, m'_4) \in \sigma} \left[\frac{p(\sigma)}{\Theta^A_s(S,T)}\right] \\ &= \frac{1}{\Theta^A_s(S,T)} \times \sum_{\sigma \in E(S,T), (m_4, m'_4) \in \sigma} [p(\sigma)] \end{aligned}$$

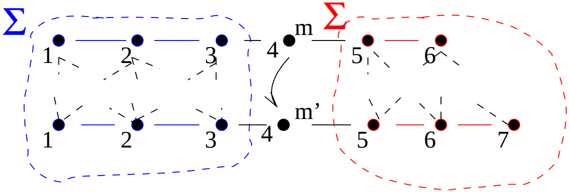


$$\gamma_{(S,T)}[4,4](m,m') = \frac{1}{\Theta_s^A(S,T)} \times \sum_{\sigma\in E(S,T),(m_4,m'_4)\in\sigma} [p(\sigma)]$$



$$\gamma_{(S,T)}[4,4](m,m') = \frac{1}{\Theta_s^A(S,T)} \times \sum_{\sigma \in E(S,T),(m_4,m'_4)\in\sigma} [p(\sigma)]$$

Ristad observes the sum can be factorised into a product of 3 terms

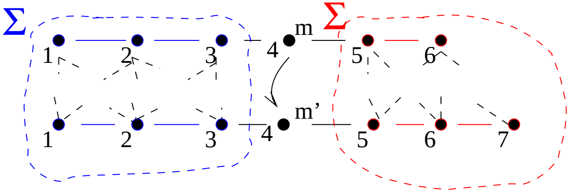


$$\gamma_{(S,T)}[4,4](m,m') = \frac{1}{\Theta_s^A(S,T)} \times \sum_{\sigma \in E(S,T),(m_4,m'_4)\in\sigma} [p(\sigma)]$$

Ristad observes the sum can be factorised into a product of 3 terms

$$\gamma_{(S,T)}[4,4](m,m') = \frac{1}{\Theta_s^A(S,T)} \times \left[\begin{array}{lcl} & \sum_{pre \in E(S_{1:3},T_{1:3})} [p(pre)] & [i] \\ \times & p(m,m') & [ii] \\ \times & \sum_{suff \in E(S_{4:6},T_{4:7})} [p(suff)] & [iii] \end{array} \right]$$

[i] values of the sum over $p(pre)$ can efficiently tabulated – this is the all-scripts algorithm



$$\gamma_{(S,T)}[4,4](m,m') = \frac{1}{\Theta^A_s(S,T)} \times \sum_{\sigma \in E(S,T),(m_4,m'_4)\in\sigma} [p(\sigma)]$$

Ristad observes the sum can be factorised into a product of 3 terms

$$\gamma_{(S,T)}[4,4](m,m') = \frac{1}{\Theta^A_s(S,T)} \times \left[\begin{array}{lccl} & \sum_{pre \in E(S_{1:3},T_{1:3})} & [p(pre)] & [i] \\ \times & p(m,m') & & [ii] \\ \times & \sum_{suff \in E(S_{4:6},T_{4:7})} & [p(suff)] & [iii] \end{array} \right]$$

[i] values of the sum over $p(pre)$ can efficiently tabulated – this is the all-scripts algorithm
[iii] values of sum over $p(suff)$ can be efficiently tabulated by an easily formulated 'backwards' variant.

All-paths EM for linear trees

procedure for determining expectations $\gamma_{S,T}(m, m')$ is then:

- compute table of 'forward' probs: $\alpha[i][j] = \sum_{pre \in E(S_{1:1-1}, T_{1:j-1})} [p(pre)]$
- compute table of 'backward' probs: $\beta[i][j] = \sum_{suff \in E(S_{i+1:I}, T_{j+1:J})} [p(suff)]$
- use to calculate pos.-dept exp:
  $\gamma_{S,T}(m_i, m'_j) = \alpha[i-1][j-1] \times p(m_i, m', j) \times \beta[i+1, j+1]$
- use to calculate pos-indpt exp: $\gamma_{S,T}(m, m') = \sum_{i,j} [\gamma_{S,T}[i][j](m, m')]$

first is essentially the algorithm proposed by Ristad and Yianilos (98)

All-paths EM for linear trees

procedure for determining expectations $\gamma_{S,T}(m, m')$ is then:

- compute table of 'forward' probs: $\alpha[i][j] = \sum_{pre \in E(S_{1:1-1}, T_{1:j-1})} [p(pre)]$
- compute table of 'backward' probs: $\beta[i][j] = \sum_{suff \in E(S_{i+1:I}, T_{j+1:J})} [p(suff)]$
- use to calculate pos.-dept exp:
  $\gamma_{S,T}(m_i, m'_j) = \alpha[i-1][j-1] \times p(m_i, m', j) \times \beta[i+1, j+1]$
- use to calculate pos-indpt exp: $\gamma_{S,T}(m, m') = \sum_{i,j} [\gamma_{S,T}[i][j](m, m')]$

first is essentially the algorithm proposed by Ristad and Yianilos (98)

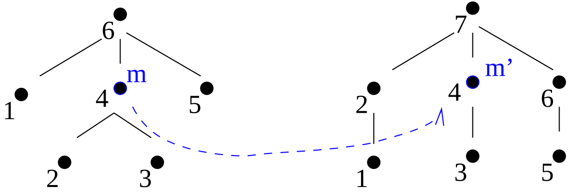
this has seen widely used to train a string distance measure (ie. linear trees) from a corpus of pairs

Position dept exp. for trees

lets try to apply similar reasoning to stochastic tree distance

Position dept exp. for trees

lets try to apply similar reasoning to stochastic tree distance

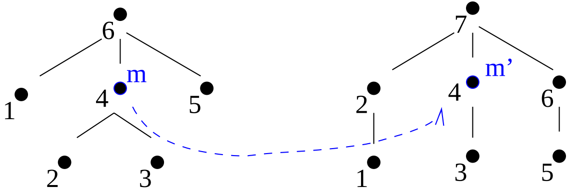


again define $(m_i, m'_j) \in \sigma$, occurrence of posn-specific subst (m, m') in σ as

$$(m_i, m'_j) \in \sigma \text{ if } \sigma = pre \circ (m, m') \circ suff \quad \begin{cases} \text{some } pre \in E(S_{1:i-1}, T_{1:j-1}) \\ \text{some } suff \in E(S_{i+1:I}, T_{j+1:J}) \end{cases}$$

Position dept exp. for trees

lets try to apply similar reasoning to stochastic tree distance



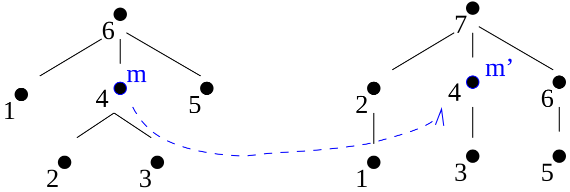
again define $(m_i, m'_j) \in \sigma$, occurrence of posn-specific subst (m, m') in σ as

$$(m_i, m'_j) \in \sigma \text{ if } \sigma = pre \circ (m, m') \circ suff \quad \begin{cases} \text{some } pre \in E(S_{1:i-1}, T_{1:j-1}) \\ \text{some } suff \in E(S_{i+1:I}, T_{j+1:J}) \end{cases}$$

and define $\gamma_{(S,T)}[4][4](m, m')$, the expectation for a swap (m, m') at $(4, 4)$ as

Position dept exp. for trees

lets try to apply similar reasoning to stochastic tree distance



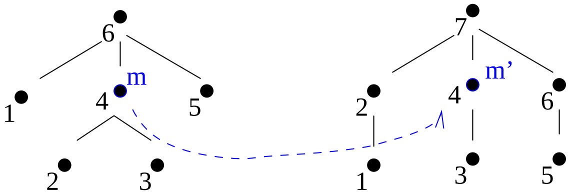
again define $(m_i, m'_j) \in \sigma$, occurrence of posn-specific subst (m, m') in σ as

$$(m_i, m'_j) \in \sigma \text{ if } \sigma = pre \circ (m, m') \circ suff \quad \begin{cases} \text{some } pre \in E(S_{1:i-1}, T_{1:j-1}) \\ \text{some } suff \in E(S_{i+1:I}, T_{j+1:J}) \end{cases}$$

and define $\gamma_{(S,T)}[4][4](m, m')$, the expectation for a swap (m, m') at $(4, 4)$ as

Position dept exp. for trees

lets try to apply similar reasoning to stochastic tree distance



again define $(m_i, m'_j) \in \sigma$, occurrence of posn-specific subst (m, m') in σ as

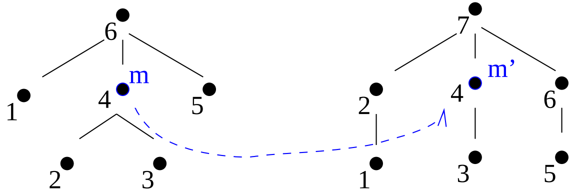
$$(m_i, m'_j) \in \sigma \text{ if } \sigma = pre \circ (m, m') \circ suff \quad \begin{cases} \text{some } pre \in E(S_{1:i-1}, T_{1:j-1}) \\ \text{some } suff \in E(S_{i+1:I}, T_{j+1:J}) \end{cases}$$

and define $\gamma_{(S,T)}[4][4](m, m')$, the expectation for a swap (m, m') at $(4, 4)$ as in words,

> the sum over the conditional probabilities of any script σ containing a m_4, m'_4 substitution, given that it is a script between S and T

Position dept exp. for trees

lets try to apply similar reasoning to stochastic tree distance



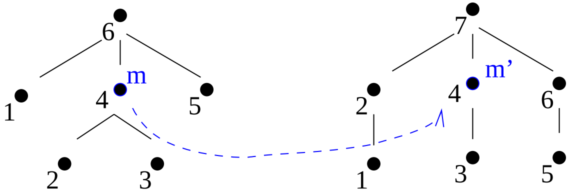
again define $(m_i, m'_j) \in \sigma$, occurrence of posn-specific subst (m, m') in σ as

$$(m_i, m'_j) \in \sigma \text{ if } \sigma = pre \circ (m, m') \circ suff \quad \begin{cases} \text{some } pre \in E(S_{1:i-1}, T_{1:j-1}) \\ \text{some } suff \in E(S_{i+1:I}, T_{j+1:J}) \end{cases}$$

and define $\gamma_{(S,T)}[4][4](m, m')$, the expectation for a swap (m, m') at $(4, 4)$ as

$$\begin{aligned} \gamma_{(S,T)}[4,4](m,m') &= \sum_{\sigma \in E(S,T), (m_4, m'_4) \in \sigma} \left[\frac{p(\sigma)}{\Theta^A_s(S,T)}\right] \\ &= \frac{1}{\Theta^A_s(S,T)} \times \sum_{\sigma \in E(S,T), (m_4, m'_4) \in \sigma} [p(\sigma)] \end{aligned}$$

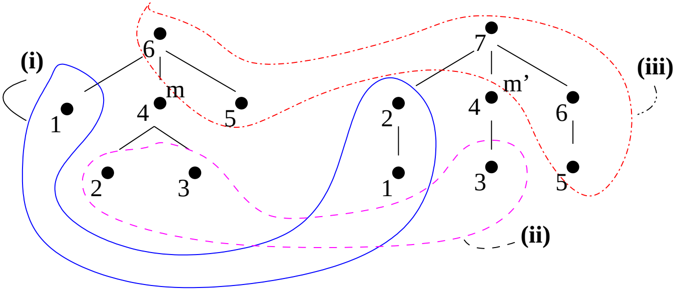
Efficient calculation of $\gamma_{(S,T)}[i][j](op)$?



So how to efficiently calculate:

$$\frac{1}{\Theta_s^A(S,T)} \times \sum_{\sigma \in E(S,T), (m_4, m_4') \in \sigma} [p(\sigma)]$$

Efficient calculation of $\gamma_{(S,T)}[i][j](op)$?



Boyer et al (2007) suggest the fectorisation

$$\sum_{e_1 \in E([\cdot_1],[\cdot_2(\cdot_1)])} [p(e_1)] \times \sum_{e_2 \in E([\cdot_2 \cdot_3],[\cdot_3])} [p(e_2)] \times p(m,m') \times \sum_{e_3 \in E([\cdot_6(\cdot_5)],[\cdot_7(\cdot_6(\cdot_5))])} [p(e_3)]$$

but we can show that this is not a sound factorisation

Unsoundness

$$\sum_{\sigma \in E(S,T),(m_4,m_4') \in \sigma} p(\sigma)$$

Unsoundness

$\sum_{\sigma \in E(S,T), (m_4, m'_4) \in \sigma} p(\sigma)$ means sum $p(\sigma)$ for scripts which represent a mapping containing (m_4, m'_4)

Unsoundness

$\sum_{\sigma \in E(S,T),(m_4,m'_4) \in \sigma} p(\sigma)$ means sum $p(\sigma)$ for scripts which represent a mapping containing (m_4,m'_4)

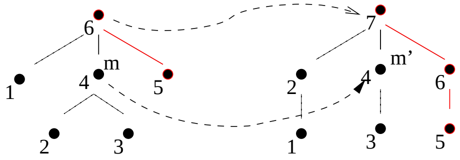
$\Rightarrow$ if an ancestor of m_4 is in the mapping (ie. not deleted) then its image under the mapping must be an ancestor of m'_4 also

Unsoundness

$\sum_{\sigma \in E(S,T),(m_4,m'_4)\in\sigma} p(\sigma)$ means sum $p(\sigma)$ for scripts which represent a mapping containing (m_4,m'_4)

$\Rightarrow$ if an ancestor of m_4 is in the mapping (ie. not deleted)
then its image under the mapping must be an ancestor of m'_4 also

so $\cdot_6$ of S being mapped to $\cdot_7$ of T is consistent with (m_4, m'_4)

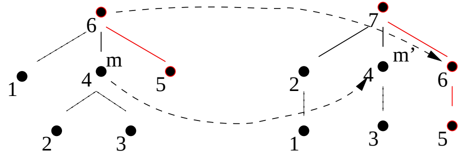


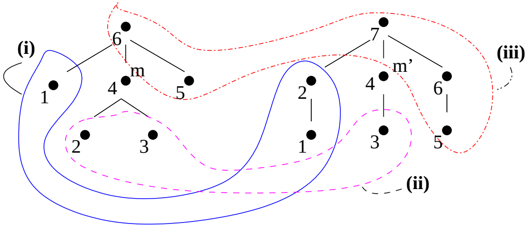
Unsoundness

$$\sum_{\sigma \in E(S,T),(m_4,m_4') \in \sigma} p(\sigma)$$ means sum $p(\sigma)$ for scripts which represent a mapping containing (m_4, m_4')

$\Rightarrow$ if an ancestor of m_4 is in the mapping (ie. not deleted) then its image under the mapping must be an ancestor of m_4' also

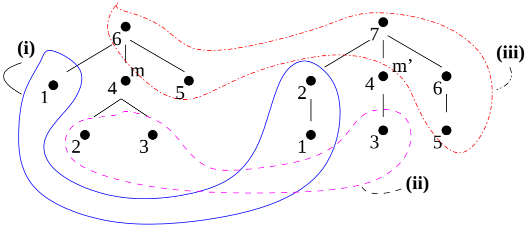
but $\cdot_6$ of S being mapped to $\cdot_6$ of T is not consistent with (m_4, m_4')





so the problem with the factorisation

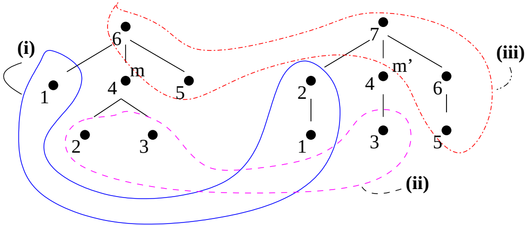
$$\sum_{e_1 \in E([\cdot_1],[\cdot_2(\cdot_1)])} [p(e_1)] \times \sum_{e_2 \in E([\cdot_2 \cdot_3],[\cdot_3])} [p(e_2)] \times p(m, m') \times \sum_{e_3 \in E([\cdot_6(\cdot_5)],[\cdot_7(\cdot_6(\cdot_5))])} [p(e_3)]$$



so the problem with the factorisation

$$\sum_{e_1 \in E([\cdot_1],[\cdot_2(\cdot_1)])} [p(e_1)] \times \sum_{e_2 \in E([\cdot_2 \cdot_3],[\cdot_3])} [p(e_2)] \times p(m, m') \times \sum_{e_3 \in E([\cdot_6(\cdot_5)],[\cdot_7(\cdot_6(\cdot_5))])} [p(e_3)]$$

is the third term sums both things consistent and inconsistent with (m_4, m'_4)

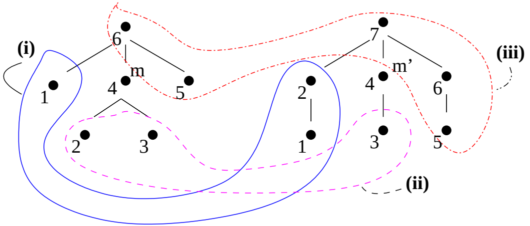


so the problem with the factorisation

$$\sum_{e_1 \in E([\cdot_1],[\cdot_2(\cdot_1)])} [p(e_1)] \times \sum_{e_2 \in E([\cdot_2 \cdot_3],[\cdot_3])} [p(e_2)] \times p(m,m') \times \sum_{e_3 \in E([\cdot_6(\cdot_5)],[\cdot_7(\cdot_6(\cdot_5))])} [p(e_3)]$$

is the third term sums both things consistent and inconsistent with (m_4, m_4')

$\sum_{e_3 \in E([\cdot_6(\cdot_5)],[\cdot_7(\cdot_6(\cdot_5))])} [p(e_3)] =$

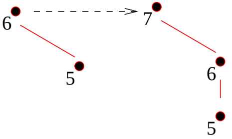


so the problem with the factorisation

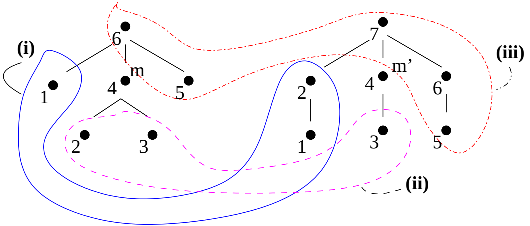
$$\sum_{e_1 \in E([\cdot_1],[\cdot_2(\cdot_1)])} [p(e_1)] \times \sum_{e_2 \in E([\cdot_2 \cdot_3],[\cdot_3])} [p(e_2)] \times p(m, m') \times \sum_{e_3 \in E([\cdot_6(\cdot_5)],[\cdot_7(\cdot_6(\cdot_5))])} [p(e_3)]$$

is the third term sums both things consistent and inconsistent with (m_4, m'_4)

$\sum_{e_3 \in E([\cdot_6(\cdot_5)],[\cdot_7(\cdot_6(\cdot_5))])} [p(e_3)] =$



$+$

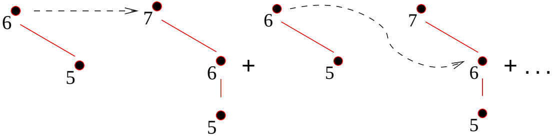


so the problem with the factorisation

$$\sum_{e_1\in E([\cdot_1],[\cdot_2(\cdot_1)])}[p(e_1)]\times\sum_{e_2\in E([\cdot_2\cdot_3],[\cdot_3])}[p(e_2)]\times p(m,m')\times\sum_{e_3\in E([\cdot_6(\cdot_5)],[\cdot_7(\cdot_6(\cdot_5))])}[p(e_3)]$$

is the third term sums both things consistent and inconsistent with (m_4, m'_4)

$\sum_{e_3\in E([\cdot_6(\cdot_5)],[\cdot_7(\cdot_6(\cdot_5))])}[p(e_3)] =$



For general trees, a feasible equivalent to the brute-force EM_A^{bf} remains an unsolved problem.

Outline

Let *the Viterbi EM algorithm* EM^V, be iterations of pair of steps

(Exp)$_V$ *generate a virtual corpus of scripts by treating each training pair* (S, T) *as standing for the* best *edit-script* σ, *which can relate* S *to* T, *weighting it by its conditional probability* $P(\sigma)/\Theta^A_s(S, T)$, *under current costs* $\mathcal{C}$

Let *the Viterbi EM algorithm* EM^V, be iterations of pair of steps

(Exp)$_V$ *generate a virtual corpus of scripts by treating each training pair* (S, T) *as standing for the* best *edit-script* σ*, which can relate* S *to* T*, weighting it by its conditional probability* $P(\sigma)/\Theta^A_s(S, T)$*, under current costs* $\mathcal{C}$

(Max) *apply maximum likelihood estimation to the virtual corpus to derive a new probability table.*

Where $\mathcal{V}$ is the best-script, the virtual count or expectation $\gamma_{S,T}(op)$ contributed by S, T for the operation *op* is defined by

$$\gamma_{(S,T)}(op) = \frac{\Theta^V_s(S, T)}{\Theta^A_s(S, T)} \times freq(op \in \mathcal{V})$$

Let *the Viterbi EM algorithm* EM^V, be iterations of pair of steps

(Exp)$_V$ *generate a virtual corpus of scripts by treating each training pair* (S, T) *as standing for the* best *edit-script* σ, *which can relate* S *to* T, *weighting it by its conditional probability* $P(\sigma)/\Theta^A_s(S, T)$, *under current costs* $\mathcal{C}$

(Max) *apply maximum likelihood estimation to the virtual corpus to derive a new probability table.*

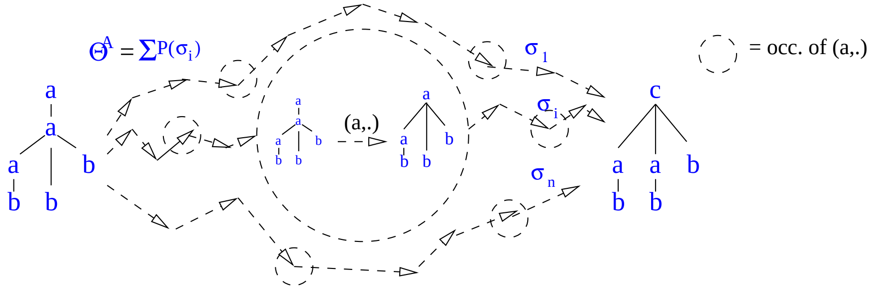
Where $\mathcal{V}$ is the best-script, the virtual count or expectation $\gamma_{S,T}(op)$ contributed by S, T for the operation *op* is defined by

$$\gamma_{(S,T)}(op) = \frac{\Theta^V_s(S, T)}{\Theta^A_s(S, T)} \times freq(op \in \mathcal{V})$$

(Exp)$_V$ accumulates the $\gamma_{S,T}(op)$ for all *op*'s, for all (S, T)

Viterbi approximation EM^V (feasible)

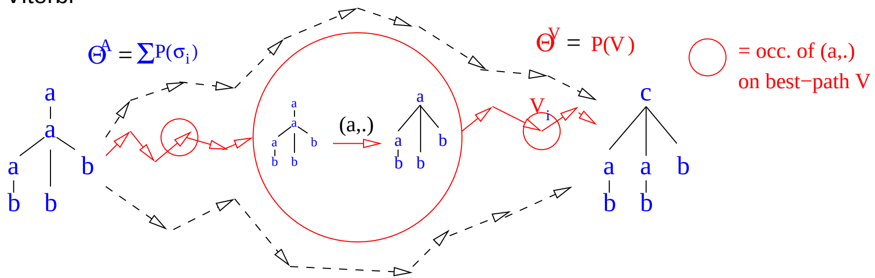
All paths



$$\gamma_{S,T}(op) = \sum_{\sigma:S\mapsto T}\left[\frac{P(\sigma)}{\Theta^A_s(S,T)} \times freq(op \in \sigma)\right]$$

Viterbi approximation EM^V (feasible)

Viterbi



$$\gamma_{(S,T)}(op) = \frac{\Theta^V_s(S,T)}{\Theta^A_s(S,T)} \times freq(op \in \mathcal{V})$$

Outline

The methodology in outline is

1. choose a set of parameters C ie. probs for all ops

The methodology in outline is

1. choose a set of parameters C ie. probs for all ops
2. derive a corpus of edit scripts in accordance with $C^{\ominus}$

The methodology in outline is

1. choose a set of parameters C ie. probs for all ops
2. derive a corpus of edit scripts in accordance with $C^{\ominus}$
3. generate a corpus $\mathcal{TP}$ of tree pairs consistent with these edit scripts
4. apply learning algorithm to tree-pair corpus $\mathcal{TP}$ to learn parameters C' and compare to see if C' is close to original C.

Choosing a set of (target) parameters

- label alphabet $\Sigma = \{A, B, C, D, E\}$
- define subst. prob to be:
  max for letters one apart in ASCII code (eg A/B
  falling as you get further from this (eg $A/C < A/B$)

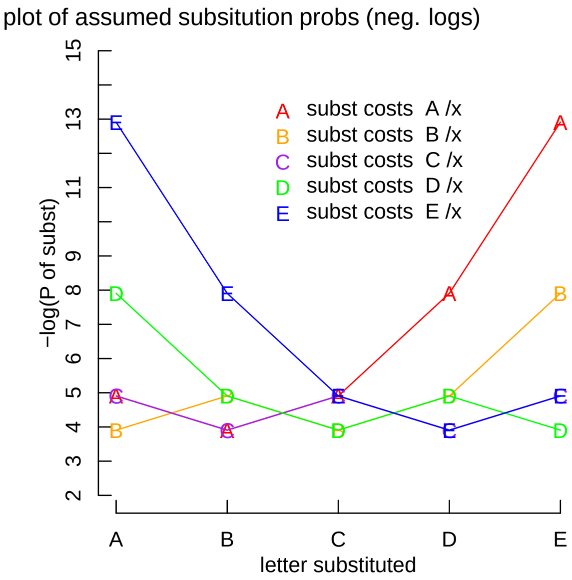
$$p(x, y) \;\; \alpha \;\; (|(|ASCII(x) - ASCII(y)| - 1)|)^2$$

- del and ins uniform, and such that ins+del ist just more than the worst swap

table (as neg. logs):

	λ	A	B	C	D	E
λ		6.907	6.907	6.907	6.907	6.907
A	6.907	4.907	3.907	4.907	7.907	12.91
B	6.907	3.907	4.907	3.907	4.907	7.907
C	6.907	4.907	3.907	4.907	3.907	4.907
D	6.907	7.907	4.907	3.907	4.907	3.907
E	6.907	12.91	7.907	4.907	3.907	4.907

The target parameters



Deriving a set of edit-scripts

generated 5k scripts in accordance with these parameters

it starts like this:

0 $[(A,D),(E,D),(D,\lambda),(C,D),(D,C),(A,B),(E,C),(D,C),(C,B),(C,D),(A,B),(C,A),($

1 $[(B,D)]$

2 $[(C,C),(D,B),(E,D),(B,D)]$

3 $[(D,B),(C,E),(A,A),(D,B),(C,B),(E,E),(C,D),(D,B),(\lambda,A),(E,C),(E,D)]$

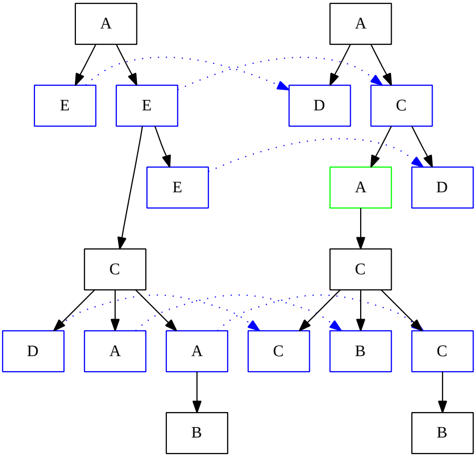
:

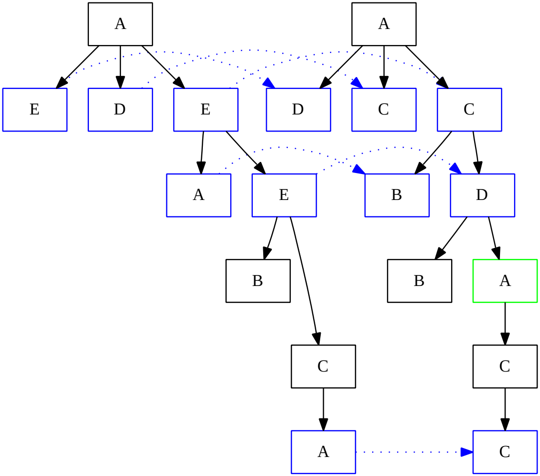
Generating tree pairs

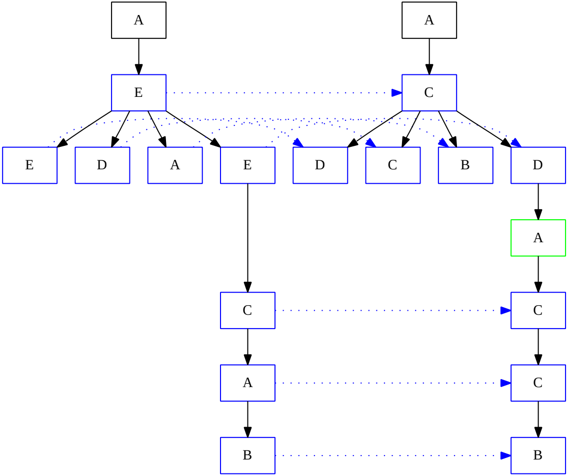
- from these scripts a corpus $\mathcal{TP}$ of consistent tree-pairs is generated
- for each script, a random 5 are chosen from all pairs consistent
- following pages for the script:

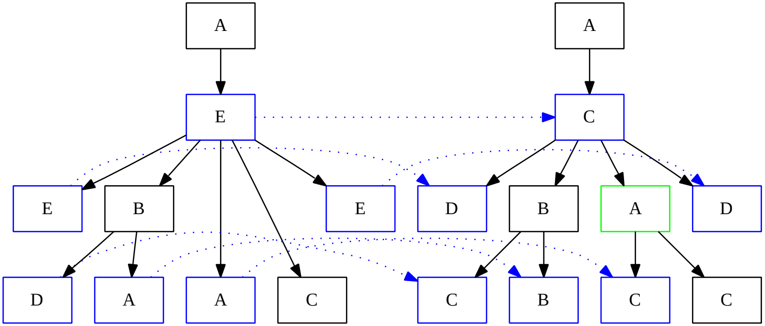
$$[(E,D)(D,C)(A,B)(B,B)(A,C)(C,C)([],A)(E,D)(E,C)(A,A)]$$

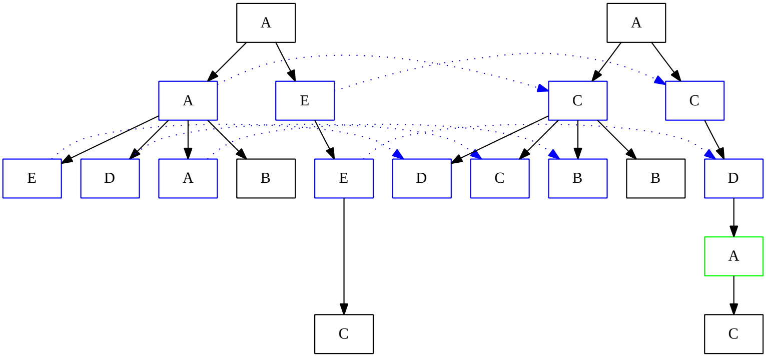
show the consistent tree pairs


A
A
E
E
D
C
E
A
D
C
C
D
A
A
C
B
C
B
B


A
A
E
D
E
D
C
C
A
E
B
D
B
B
A
C
C
A
C

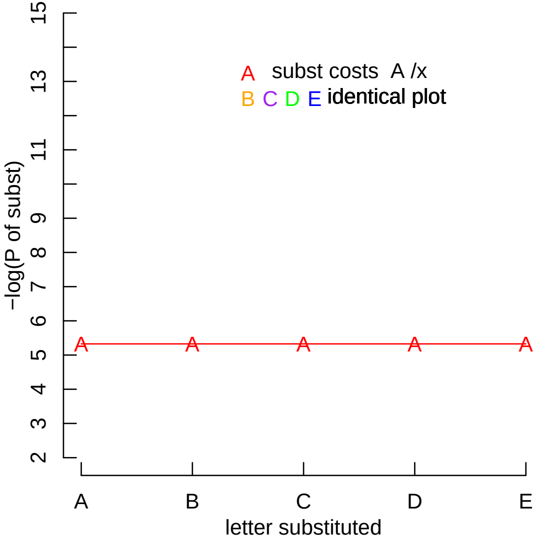

A
A
E
C
E
D
A
E
D
C
B
D
A
C
C
A
C
B
B


A
A
E
C
E
B
E
D
B
A
D
D
A
A
C
C
B
C
C


A
A
A
E
C
C
E
D
A
B
E
D
C
B
B
D
A
C
C

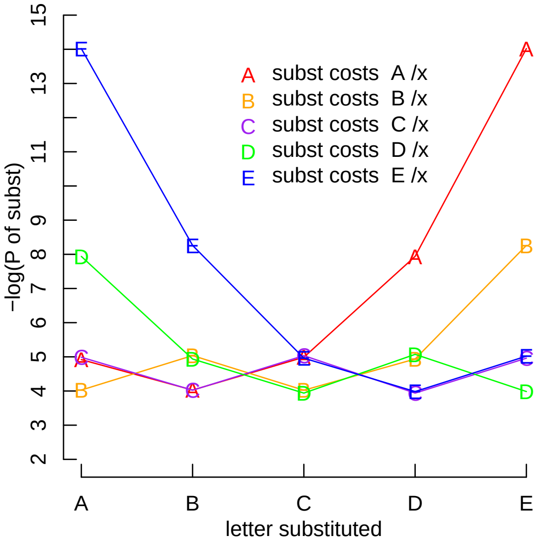
Applying training algorithm

Viterbi EM applied to corpus of tree pairs $\mathcal{TP}$
starting from initial uniform costs:



Applying training algorithm

Viterbi EM applied to corpus of tree pairs $\mathcal{TP}$ learns costs:



Outline

Adapting a k-NN classifier

Adapting a k-NN classifier

- a possible use of a distance is k-NN classifier:

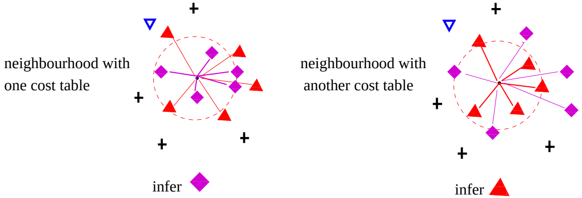
  $cat(S) = VOTE(\{\textit{categories of } k \textbf{ nearest neighbours} \textit{ of } S\})$

Adapting a k-NN classifier

- a possible use of a distance is k-NN classifier:

  $$cat(S) = VOTE(\{categories\ of\ k\ \textbf{nearest neighbours}\ of\ S\ \})$$

- change cost table $\Rightarrow$ change nearest neighbours $\Rightarrow$ change categorisation:

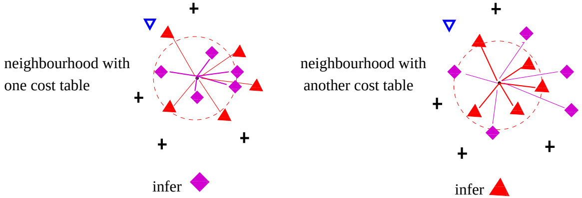


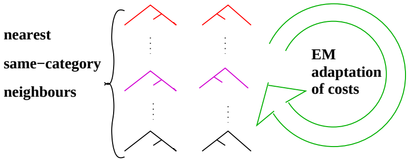
Adapting a k-NN classifier

- a possible use of a distance is k-NN classifier:

  $$\mathit{cat}(S) = \mathit{VOTE}(\{\mathit{categories\ of\ } k\ \textbf{nearest neighbours}\ \mathit{of}\ S\ \})$$

- change cost table ⇒ change nearest neighbours ⇒ change categorisation:





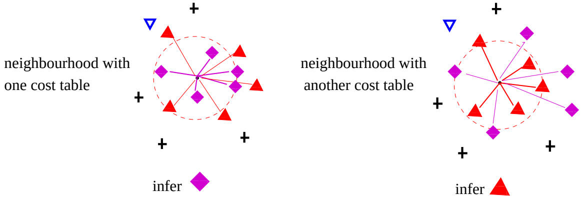
scripts between between *same-category neighbours should have distinctive probs*

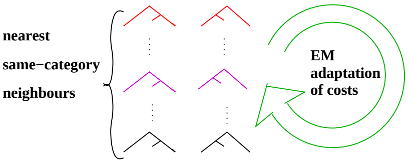
Adapting a k-NN classifier

- a possible use of a distance is k-NN classifier:

  $$cat(S) = VOTE(\{categories\ of\ k\ \textbf{nearest neighbours}\ of\ S\ \})$$

- change cost table ⇒ change nearest neighbours ⇒ change categorisation:





scripts between between *same-category neighbours should have distinctive probs* ⇒ perhaps can use *Expectation-Maximisation techniques to adapt edit-probs from a corpus of same-category nearest neighbours*

Data set: QuestionBank

Data set: QuestionBank

2755 syntactically analysed and semantically categorised questions

Data set: QuestionBank

2755 syntactically analysed and semantically categorised questions

Cat	Example
NUM	When was London 's Docklands Light Railway constructed ? (SBARQ (WHADVP (WRB When))(SQ (VBD was)(NP (NP (NNP London)(POS 's))(NNPS Docklands) (JJ Light)(NN Railway))(VP (VBN constructed)))(. ?))
LOC	What country is the biggest producer of tungsten ? (SBARQ (WHNP (WDT What)(NN country))(SQ (VBZ is)(NP (NP (DT the)(JJS biggest)(NN producer)) (PP (IN of)(NP (NN tungsten)))))(. ?))
HUM	What is the name of the managing director of Apricot Computer ? (WHNP (WP What))(SQ (VBZ is)(NP (NP (DT the)(NN name))(PP (IN of)(NP (NP (DT the)(JJ managing)(NN director))

Data set: QuestionBank

2755 syntactically analysed and semantically categorised questions

Cat	Example
NUM	When was London 's Docklands Light Railway constructed ? (SBARQ (WHADVP (WRB When))(SQ (VBD was)(NP (NP (NNP London)(POS 's))(NNPS Docklands) (JJ Light)(NN Railway))(VP (VBN constructed)))(. ?))
LOC	What country is the biggest producer of tungsten ? (SBARQ (WHNP (WDT What)(NN country))(SQ (VBZ is)(NP (NP (DT the)(JJS biggest)(NN producer)) (PP (IN of)(NP (NN tungsten)))))(. ?))
HUM	What is the name of the managing director of Apricot Computer ? (WHNP (WP What))(SQ (VBZ is)(NP (NP (DT the)(NN name))(PP (IN of)(NP (NP (DT the)(JJ managing)(NN director))

Intuitiion: in scripts between between same-category neighbours *should* have distinctive probs eg. . $P(\textit{who}/\textit{when}) << P(\textit{state}/\textit{country})$.

k-NN categorisation

- experiments make 9:1 split into Examples vs Testing and evaluate a distance measure in k-NN classifier

k-NN categorisation

- experiments make 9:1 split into Examples vs Testing and evaluate a distance measure in k-NN classifier

  so for each T in Testing, assign a category based on the categories of its k nearest neighbours in Examples

k-NN categorisation

- experiments make 9:1 split into Examples vs Testing and evaluate a distance measure in k-NN classifier

  so for each T in Testing, assign a category based on the categories of its k nearest neighbours in Examples

  $$cat(T) = VOTE(\{categories\ of\ k\ \textbf{nearest neighbours}\ of\ T\})$$

- compare

k-NN categorisation

- experiments make 9:1 split into Examples vs Testing and evaluate a distance measure in k-NN classifier

  so for each T in Testing, assign a category based on the categories of its k nearest neighbours in Examples

  $$cat(T) = VOTE(\{categories\ of\ k\ \textbf{nearest neighbours}\ of\ T\})$$

- compare

  tree-distance with standard unit costs

k-NN categorisation

- experiments make 9:1 split into Examples vs Testing and evaluate a distance measure in k-NN classifier

  so for each T in Testing, assign a category based on the categories of its k nearest neighbours in Examples

  $$cat(T) = VOTE(\{\text{categories of } k \textbf{ nearest neighbours} \text{ of } T\})$$

- compare

  tree-distance with standard unit costs

  stochastic tree-distance with untrained costs

k-NN categorisation

- experiments make 9:1 split into Examples vs Testing and evaluate a distance measure in k-NN classifier

  so for each T in Testing, assign a category based on the categories of its k nearest neighbours in Examples

  $$cat(T) = VOTE(\{categories\ of\ k\ \textbf{nearest neighbours}\ of\ T\})$$

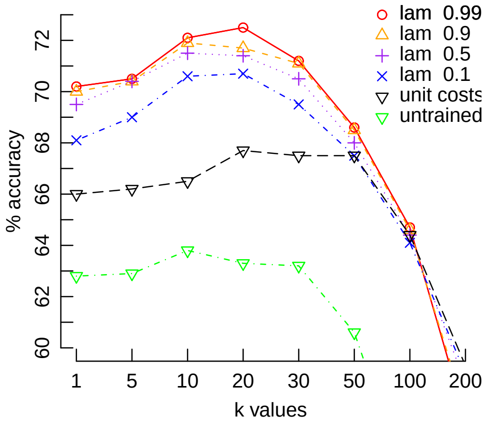
- compare

  tree-distance with standard unit costs

  stochastic tree-distance with untrained costs

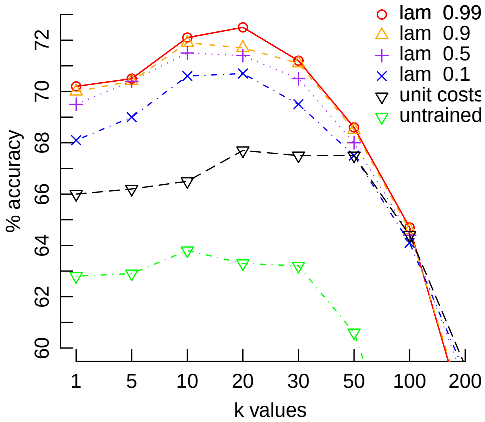
  stochastic tree distance with trained costs
  training by EM^V on same-category neighbours from the Example set

Experimental outcome (brief)



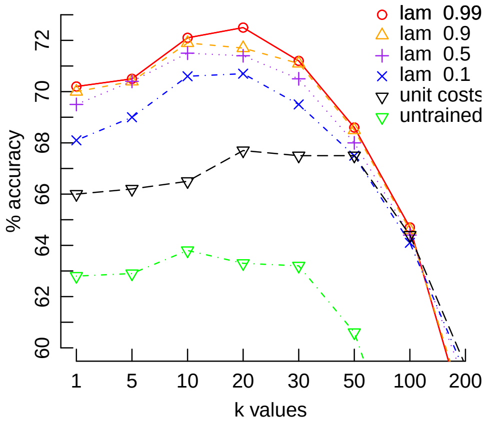
Experimental outcome (brief)

- standard unit-costs
  ▽, max. 67.7%



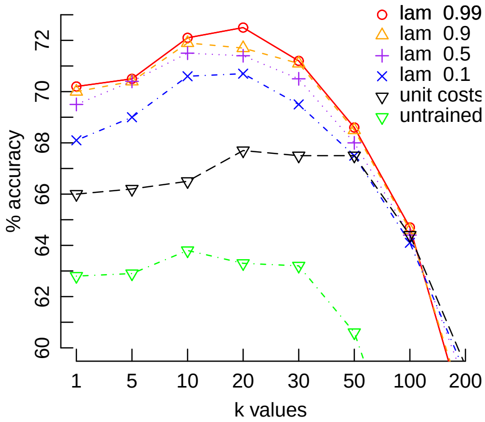
Experimental outcome (brief)

- standard unit-costs
  ▽, max. 67.7%
- initial stochastic costs
  ▽ max. 63.8%
  worse than unit costs



Experimental outcome (brief)

- standard unit-costs
  ▽, max. 67.7%
- initial stochastic costs
  ▽ max. 63.8%
  worse than unit costs
- best EM^V-adapted costs
  ○, max. 72.5%
  about 5% better than unit-costs
  (▽, max. 67.7%)



Outline

Stochastic cost initialisation

EM^V needs an initialisation of its parameters.

we used a basically uniform initialisation except

Stochastic cost initialisation

EM^V needs an initialisation of its parameters.

we used a basically uniform initialisation except

diagonal entries are d times more probable than non-diagonal.

Stochastic cost initialisation

EM^V needs an initialisation of its parameters.

we used a basically uniform initialisation except

diagonal entries are d times more probable than non-diagonal.

examples for d = 3, 10, 100, and 1000 are:

3	λ	a	b
λ	3.7	3.7	3.7
a	3.7	2.115	3.7
b	3.7	3.7	2.115

10	λ	a	b
λ	4.755	4.755	4.755
a	4.755	1.433	4.755
b	4.755	4.755	1.433

100	λ	a	b
λ	7.693	7.693	7.693
a	7.693	1.05	7.693
b	7.693	7.693	1.05

1000	λ	a	b
λ	10.97	10.97	10.97
a	10.97	1.005	10.97
b	10.97	10.97	1.005

NOTE: diagonal entries are not insignificant

Smoothing

We used a *smoothing* option on a table C^{Δ} derived by EM^{V}, interpolating it with the stochastic initialisation ${C^{\Delta}}_{u}(d)$ as follows:

Smoothing

We used a *smoothing* option on a table C^{Δ} derived by EM^{V}, interpolating it with the stochastic initialisation $C^{\Delta}{}_{u}(d)$ as follows:

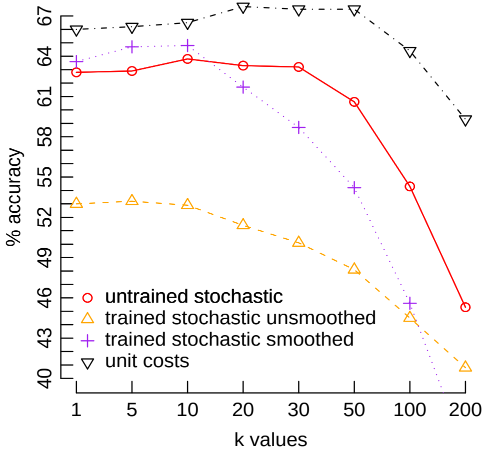
$$2^{-C^{\Delta}{}_{\lambda}[x][y]} = \lambda(2^{-C^{\Delta}[x][y]}) + (1-\lambda)(2^{-C^{\Delta}{}_{u}(d)[x][y]})$$

with $0 \leq \lambda \leq 1$

$\lambda = 1$ gives all the weight to the derived table

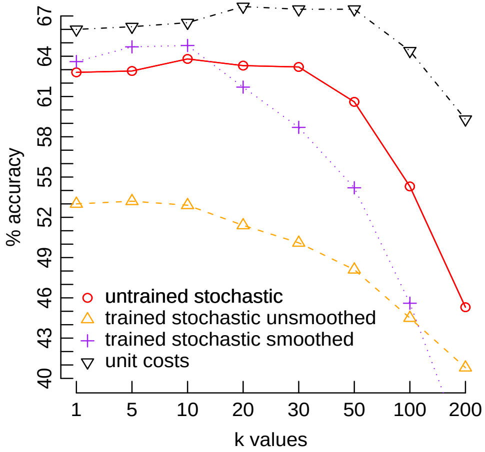
$\lambda = 0$ gives all the weight to the initial table

Experiment One



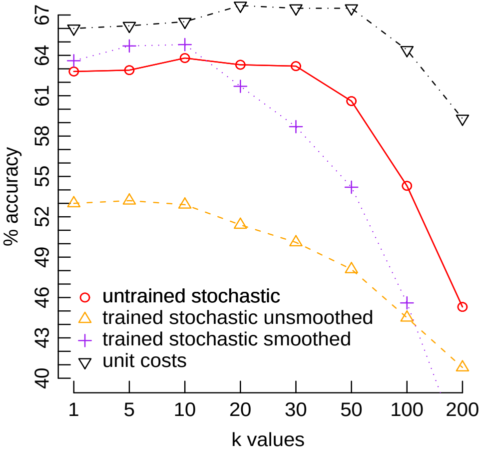
Experiment One

- unit-costs (▽, max. 67.7%) exceeds non-adapted $C^{\Delta}{}_{u}(3)$ costs (○, max. 63.8%)



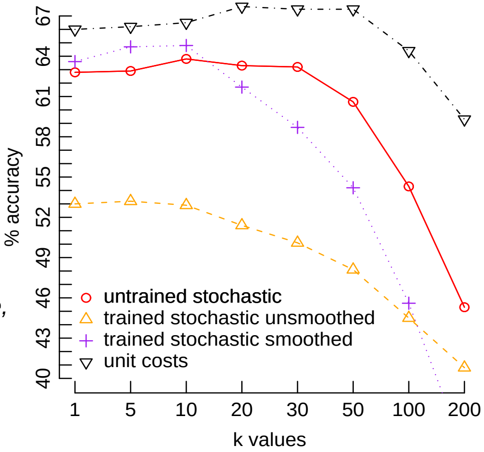
Experiment One

- unit-costs (▽, max. 67.7%) exceeds non-adapted $C^{\Delta}{}_{u}(3)$ costs (○, max. 63.8%)
- unsmoothed EM^{V}-adapted costs (△, max. 53.2%) worse than initial, stochastic costs (○, max. 63.8%)



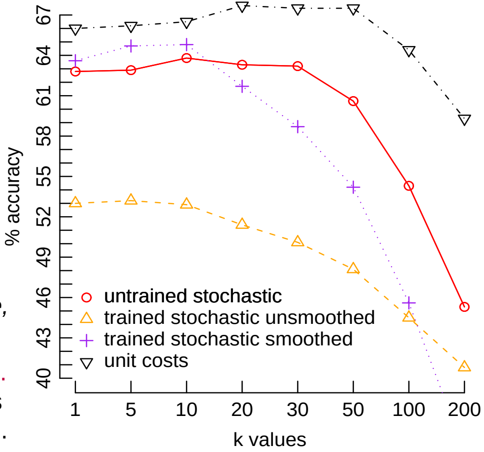
Experiment One

- unit-costs (▽, max. 67.7%) exceeds non-adapted $C^{\Delta}{}_{u}(3)$ costs (○, max. 63.8%)
- unsmoothed EM^V-adapted costs (△,max. 53.2%) worse than initial, stochastic costs (○, max. 63.8%)
- EM^V-adapted costs on training set gives 95% accuracy: $\Rightarrow EM^V$ makes training pairs *too probable, and over-fits*.



Experiment One

- unit-costs (▽, max. 67.7%) exceeds non-adapted $C^{\Delta}{}_{u}(3)$ costs (○, max. 63.8%)
- unsmoothed EM^V-adapted costs (△,max. 53.2%) worse than initial, stochastic costs (○, max. 63.8%)
- EM^V-adapted costs on training set gives 95% accuracy: $\Rightarrow EM^V$ makes training pairs *too probable, and over-fits*.
- *smoothing* adapted costs (+,max. 64.8%) improves over initial costs (○) but is still below unit costs (▽).



Despite poor performace of the EM^V-adapted costs, some of the adapted costs seem intuitive. Here is a sample from top 1% of adapted swap costs, which are plausibly discounted relative to others:

8.50	?	.	12.31	The	the
8.93	NNP	NN	12.65	you	I
9.47	VBD	VBZ	13.60	can	do
9.51	NNS	NN	13.83	many	much
9.78	a	the	13.92	city	state
11.03	was	is	13.93	city	country
11.03	's	is			

Outline

- Recall: For the stochastic distance Δ_s^V cost-table entries represent probabilities via

$$2^{-C^{\Delta}(x,y)} = p(x,y)$$

- Recall: For the stochastic distance Δ_s^V cost-table entries represent probabilities via

$$2^{-C^{\Delta}(x,y)} = p(x,y)$$

- a single 0 entry in C^{Δ} implies *infinite* cost entries everywhere else.

- Recall: For the stochastic distance Δ_s^V cost-table entries represent probabilities via

$$2^{-C^{\Delta}(x,y)} = p(x,y)$$

- a single 0 entry in C^{Δ} implies *infinite* cost entries everywhere else. $\Rightarrow$ a stochastically valid cost table cannot have zero costs on the diagonal

- Recall: For the stochastic distance Δ^V_s cost-table entries represent probabilities via

$$2^{-C^{\Delta}(x,y)} = p(x,y)$$

- a single 0 entry in C^{Δ} implies *infinite* cost entries everywhere else.
  $\Rightarrow$ a stochastically valid cost table cannot have zero costs on the diagonal
- perhaps this impedes good categorisation; note also the unit-cost setting, which is clearly 'uniform' in a sense, out-performs the 'uniform' stochastic initialisations

- Recall: For the stochastic distance Δ_s^V cost-table entries represent probabilities via

$$2^{-C^{\Delta}(x,y)} = p(x,y)$$

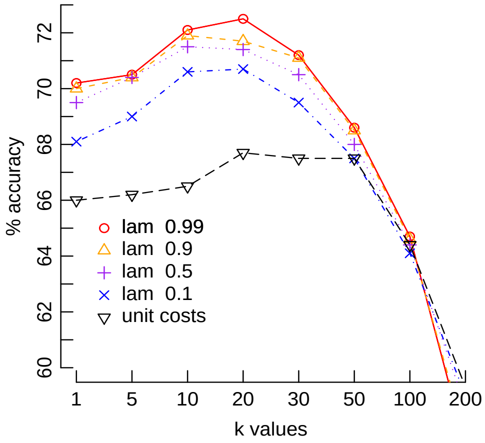
- a single 0 entry in C^{Δ} implies *infinite* cost entries everywhere else. $\Rightarrow$ a stochastically valid cost table cannot have zero costs on the diagonal
- perhaps this impedes good categorisation; note also the unit-cost setting, which is clearly 'uniform' in a sense, out-performs the 'uniform' stochastic initialisations
- suggests final step in which all the entries on the cost-table's diagonal are zeroed.

- Recall: For the stochastic distance Δ_s^V cost-table entries represent probabilities via

$$2^{-C^{\Delta}(x,y)} = p(x,y)$$

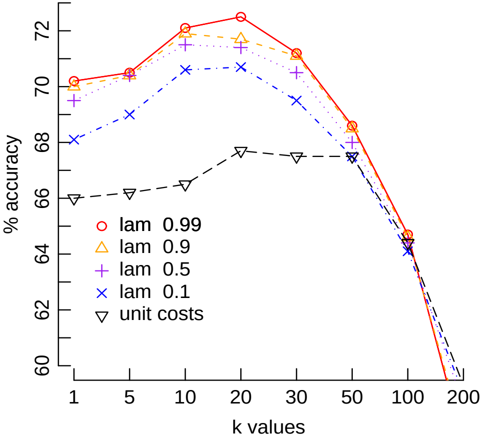
- a single 0 entry in C^{Δ} implies *infinite* cost entries everywhere else.
  $\Rightarrow$ a stochastically valid cost table cannot have zero costs on the diagonal
- perhaps this impedes good categorisation; note also the unit-cost setting, which is clearly 'uniform' in a sense, out-performs the 'uniform' stochastic initialisations
- suggests final step in which all the entries on the cost-table's diagonal are zeroed.
- Bilenko et al 2003 does essentially this in work on stochastic string distance

Experiment Two



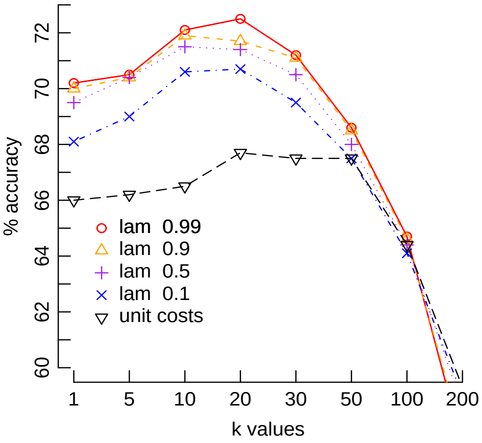
Experiment Two

- now with smoothing at varius levels of interpolation ($\lambda \in \{0.99, 0.9, 0.5, 0.1\}$) and with the diagonal zeroed, the *EM^V-adapted costs clearly out-perform* the unit-costs case ($\triangledown$).



Experiment Two

- now with smoothing at varius levels of interpolation ($\lambda \in \{0.99, 0.9, 0.5, 0.1\}$) and with the diagonal zeroed, the EM^V-adapted costs clearly out-perform the unit-costs case ($\bigtriangledown$).
- the best result being 72.5% ($k = 20$, $\lambda = 0.99$), as compared to 67.5% for unit-costs ($k = 20$)



Conclusions

Conclusions

- evidence to show that Viterbi EM cost-adaptation can increase the performance of a tree-distance based classifier, and improve it to above that attained in the unit-cost setting,

Conclusions

- evidence to show that Viterbi EM cost-adaptation can increase the performance of a tree-distance based classifier, and improve it to above that attained in the unit-cost setting,
- experiments on further data-sets is required: one possibility is the NLP-related tasks of question-answering, where the need is to assess pairs of sentences for their likelihood to be a question-answer pairs. A training set of such pairs could also serve as potential input to the cost adaptation algorithm.

Literature

- *The paper this talks is mainly based on is* Emms (2011)
- *Background on string-distance and tree-distance*: Tai (1979) Zhang and Shasha (1989) Ristad and Yianilos (1998) Bilenko and Mooney (2003) Boyer et al. (2007)
- *Background on EM*: Prescher (2004)
- *Some work using similar approximation to all-paths EM*: Benedí and Sánchez (2005)
- *Question-bank*: Judge et al. (2006); Judge (2006a), Judge (2006b)
- *Some work using related models of stochastic tree-distance*: Takasu et al. (2007), Dalvi et al. (2009) Wang and Manning (2010)

J.-M. Benedí and J.-A. Sánchez. Estimation of stochastic context-free grammars and their use as language models. *Computer Speech and Language*, 19(3):249–274, July 2005.

M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003)*, pages 39–48, 2003.

L. Boyer, A. Habrard, and M. Sebban. Learning metrics between tree structured data: Application to image recognition. In *Proceedings of the 18th European Conference on Machine Learning (ECML 2007)*, pages 54–66, 2007.

N. Dalvi, P. Bohannon, and F. Sha. Robust web extraction: an approach based on a probabilistic tree-edit model. In *SIGMOD '09: Proceedings of the 35th SIGMOD international conference on Management of data*, pages 335–348, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-551-2. doi: http://doi.acm.org/10.1145/1559845.1559882.

M. Emms. On stochastic tree distances and their training via expectation-maximisation. In *Proceedings of ICPRAM 2012 International Conference on Pattern Recognition Application and Methods*, 2011.

J. Judge. *Adapting and Developing Linguistic Resources for Question Answering*. PhD thesis, Dublin City University, 2006a.

J. Judge, 2006b. Corpus of syntactically annotated questions http://www.computing.dcu.ie/ jjudge/qtreebank/.

J. Judge, A. Cahill, and J. van Genabith. Questionbank: creating a corpus of parse-annotated questions. In *ACL '06: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 497–504, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

D. Prescher. A tutorial on the expectation-maximization algorithm including maximum-likelihood estimation and em training of probabilistic context-free grammars. *Computing Research Repository*, 2004.

E. S. Ristad and P. N. Yianilos. Learning string edit distance. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 20(5): 522–532, May 1998.

K.-C. Tai. The tree-to-tree correction problem. *Journal of the ACM (JACM)*, 26(3):433, 1979.

A. Takasu, D. Fukagawa, and T. Akutsu. Statistical learning algorithm for tree similarity. In *ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, pages 667–672, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-3018-4. doi: http://dx.doi.org/10.1109/ICDM.2007.38.

M. Wang and C. D. Manning. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1164–1172, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing*, 18: 1245–1262, 1989.