Issues in Analogical Learning over Sequences of Symbols: a Case Study with Named Entity Transliteration.

Philippe Langlais

Dept. I.R.O.
Université de Montréal, Québec, Canada
felipe@iro.umontreal.ca

SAMAI'12: August 27th, 2012

Analogy in Natural Language Processing

- parsing with a treebank
  [Lepage and Ando, 1996, Hassena, 2011]
- grapheme-to-phoneme conversion [Yvon, 1997]
- machine translation, pioneered by [Lepage and Denoual 2005]
- analogical puzzles in SAT [Turney and Littman, 2005]
- acquiring morphology [Lavallée and Langlais, 2011]
- query expansion in IR [Moreau et al., 2007]
- handwritten character recognition [Miclet et al., 2008].
- smoothing language models [Gosme et al., 2011]

Formal Analogy

Examples

- $[funny : funniest = lucky : luckiest]$
- $[suddenly, she\ appeared : she\ appeared\ suddenly = today, she\ appeared : she\ appeared\ today]$
- $[This\ guy\ drinks\ too\ much : This\ boat\ sinks = These\ guys\ drink\ too\ much : These\ boats\ sink]$
- $[Elle\ est\ émue : Il\ est\ ému = Elle\ est\ touchée : Il\ est\ touché]$

ALEPH [Lepage and Denoual 2005]

%bitext

translation(s_1,$\hat{s}_1$).

⋮

translation(s_n,$\hat{s}_n$).

% analogical learning

translation(D,$\hat{D}$) :-

translation(A,$\hat{A}$)
translation(B,$\hat{B}$)
translation(C,$\hat{C}$)
analogy(A,B,C,D)

analogy($\hat{A}$,$\hat{B}$,$\hat{C}$,$\hat{D}$)

assert(translation(D,$\hat{D}$)).

ALEPH [Lepage and Denoual 2005]

%bitext

translation(s_1,$\hat{s}_1$).
⋮
translation(s_n,$\hat{s}_n$).

% analogical learning
translation(D,$\hat{D}$) :-
translation(A,$\hat{A}$)
translation(B,$\hat{B}$)
translation(C,$\hat{C}$)
analogy(A,B,C,D)
analogy($\hat{A}$,$\hat{B}$,$\hat{C}$,$\hat{D}$)
assert(translation(D,$\hat{D}$)).

◁ Corpus of examples

in our case:

translation(Aposhian , 阿 波 希 安);
translation(Blagrove , 布格夫);
translation(Emission , 埃米申);
⋮

ALEPH [Lepage and Denoual 2005]

%bitext

translation(s_1,$\hat{s}_1$).

⋮

translation(s_n,$\hat{s}_n$).

% analogical learning

translation(D,$\hat{D}$) :-

translation(A,$\hat{A}$)
translation(B,$\hat{B}$)
translation(C,$\hat{C}$)
analogy(A,B,C,D)

analogy($\hat{A}$,$\hat{B}$,$\hat{C}$,$\hat{D}$)

assert(translation(D,$\hat{D}$)).

ALEPH [Lepage and Denoual 2005]

%bitext

translation(s_1,$\hat{s}_1$).

⋮

translation(s_n,$\hat{s}_n$).

% analogical learning

translation(D,$\hat{D}$) :-

translation(A,$\hat{A}$)
translation(B,$\hat{B}$)
translation(C,$\hat{C}$)
analogy(A,B,C,D)

analogy($\hat{A}$,$\hat{B}$,$\hat{C}$,$\hat{D}$)

assert(translation(D,$\hat{D}$)).

◁ Step1: Identifying input analogies

ALEPH [Lepage and Denoual 2005]

%bitext

translation(s_1,$\hat{s}_1$).

⋮

translation(s_n,$\hat{s}_n$).

% analogical learning

translation(D,$\hat{D}$) :-

translation(A,$\hat{A}$)
translation(B,$\hat{B}$)
translation(C,$\hat{C}$)
analogy(A,B,C,D)
analogy($\hat{A}$,$\hat{B}$,$\hat{C}$,$\hat{D}$)
assert(translation(D,$\hat{D}$)).

◁ Step2: Solving output equations

ALEPH [Lepage and Denoual 2005]

%bitext

translation(s_1,$\hat{s}_1$).

⋮

translation(s_n,$\hat{s}_n$).

% analogical learning

translation(D,$\hat{D}$) :-

translation(A,$\hat{A}$)
translation(B,$\hat{B}$)
translation(C,$\hat{C}$)
analogy(A,B,C,D)
analogy($\hat{A}$,$\hat{B}$,$\hat{C}$,$\hat{D}$)
assert(translation(D,$\hat{D}$)).

◁ Step3: Selector

Elegant but . . .

%bitext

translation(s_1,$\hat{s}_1$).

⋮

translation(s_n,$\hat{s}_n$).

% analogical learning

translation(D,$\hat{D}$) :-

translation(A,$\hat{A}$)
translation(B,$\hat{B}$)
translation(C,$\hat{C}$)
analogy(A,B,C,D)

analogy($\hat{A}$,$\hat{B}$,$\hat{C}$,$\hat{D}$)

assert(translation(D,$\hat{D}$)).

1. Searching for input analogies is a (very) time consuming stage
   - we need a definition of analogy
2. Solving an analogical equation is not easy
3. The overall process is highly noisy
   - we need a way to filter

Elegant but . . .

%bitext

translation(s_1,$\hat{s}_1$).

⋮

translation(s_n,$\hat{s}_n$).

% analogical learning

translation(D,$\hat{D}$) :-

translation(A,$\hat{A}$)
translation(B,$\hat{B}$)
translation(C,$\hat{C}$)
analogy(A,B,C,D)
analogy($\hat{A}$,$\hat{B}$,$\hat{C}$,$\hat{D}$)
assert(translation(D,$\hat{D}$)).

1. Searching for input analogies is a (very) time consuming stage
   - we need a definition of analogy
2. Solving an analogical equation is not easy
3. The overall process is highly noisy
   - we need a way to filter

Elegant but . . .

%bitext

translation(s_1,$\hat{s}_1$).

⋮

translation(s_n,$\hat{s}_n$).

% analogical learning

translation(D,$\hat{D}$) :-

translation(A,$\hat{A}$)
translation(B,$\hat{B}$)
translation(C,$\hat{C}$)
analogy(A,B,C,D)

analogy($\hat{A}$,$\hat{B}$,$\hat{C}$,$\hat{D}$)

assert(translation(D,$\hat{D}$)).

1. Searching for input analogies is a (very) time consuming stage
   - we need a definition of analogy
2. Solving an analogical equation is not easy
3. The overall process is highly noisy
   - we need a way to filter

Plan

Searching Input Analogies

Finding input triplets (x, y, z) that define with t an analogy

- brute-force: cubic in the size of the input space
  - ▷ *for toy problems only* ($|\mathcal{I}| \leq 100$)
- turned into a quadratic number of equation solving [Lepage and Denoual 2005]
  - instead of verifying $[x : y = z : t]$ $\forall x, y, z \in \mathcal{I}^3$
  - consider $[y : x = t : ?] \cap \mathcal{I}$ $\forall x, y \in \mathcal{I}^2$
    - ▷ *thanks to the symmetry:* $[x : y = z : t] \Leftrightarrow [y : x = t : z]$
    - ▷ *still impractical over a few thousands of input forms*
- sample pairs (x, y) [Langlais and Patry, 2007]
  - $\{ (x, y, z) \mid (x, y) \in \mathcal{N}(t)^2, [y : x = t : z] \}$

Searching Input Analogies

Finding input triplets (x, y, z) that define with t an analogy

- brute-force: cubic in the size of the input space
  - ▷ *for toy problems only* ($|\mathcal{I}| \leq 100$)
- turned into a quadratic number of equation solving [Lepage and Denoual 2005]
  - instead of verifying $[x : y = z : t] \ \forall x, y, z \in \mathcal{I}^3$
  - consider $[y : x = t : ?] \cap \mathcal{I} \ \forall x, y \in \mathcal{I}^2$
  - ▷ *thanks to the symmetry:* $[x : y = z : t] \Leftrightarrow [y : x = t : z]$
  - ▷ *still impractical over a few thousands of input forms*
- sample pairs (x, y) [Langlais and Patry, 2007]
  - $\{ (x, y, z) \mid (x, y) \in \mathcal{N}(t)^2, [y : x = t : z] \}$
  - ▷ *ex:* $\mathcal{N}(t) = \{f \in \mathcal{I} \mid \textit{edit-distance}(t, f) \leq \delta\}$
  - ▷ *manageable with tens of thousand forms*

Searching Input Analogies

Finding input triplets (x, y, z) that define with t an analogy

- brute-force: cubic in the size of the input space
  - ▷ *for toy problems only ($|\mathcal{I}| \leq 100$)*
- turned into a quadratic number of equation solving [Lepage and Denoual 2005]
  - instead of verifying $[x : y = z : t] \ \forall x, y, z \in \mathcal{I}^3$
  - consider $[y : x = t : ?] \cap \mathcal{I} \ \forall x, y \in \mathcal{I}^2$
  - ▷ *thanks to the symmetry: $[x : y = z : t] \Leftrightarrow [y : x = t : z]$*
  - ▷ *still impractical over a few thousands of input forms*
- sample pairs (x, y) [Langlais and Patry, 2007]
  - $\{ (x, y, z) \mid (x, y) \in \mathcal{N}(t)^2, [y : x = t : z] \}$
  - ▷ *ex: $\mathcal{N}(t) = \{f \in \mathcal{I} \mid \textit{edit-distance}(t, f) \leq \delta\}$*
  - ▷ *manageable with tens of thousand forms*

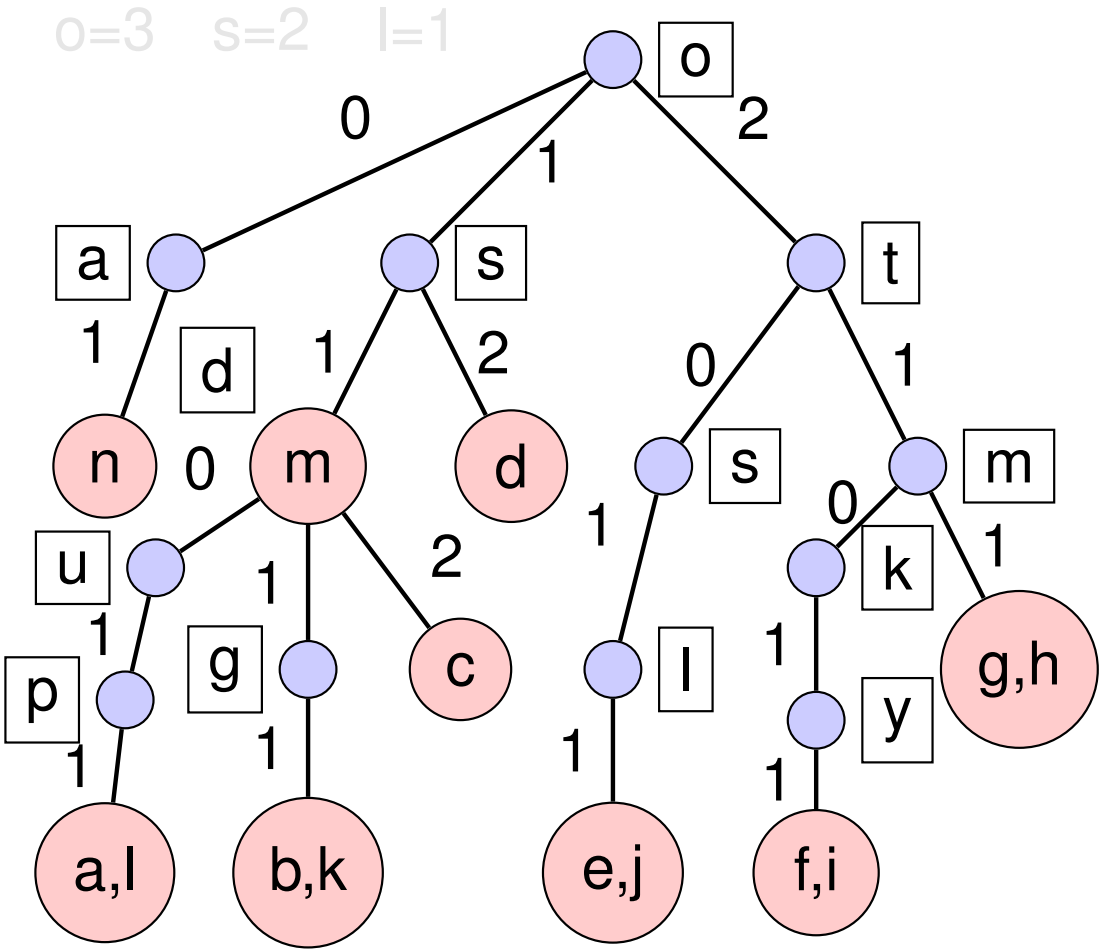
Searching Input Analogies

[Langlais and Yvon, 2008]

$$[x : y = z : t] \Rightarrow \overbrace{|x|_c + |t|_c = |y|_c + |z|_c \quad \forall c \in \mathcal{A}}^{p}$$

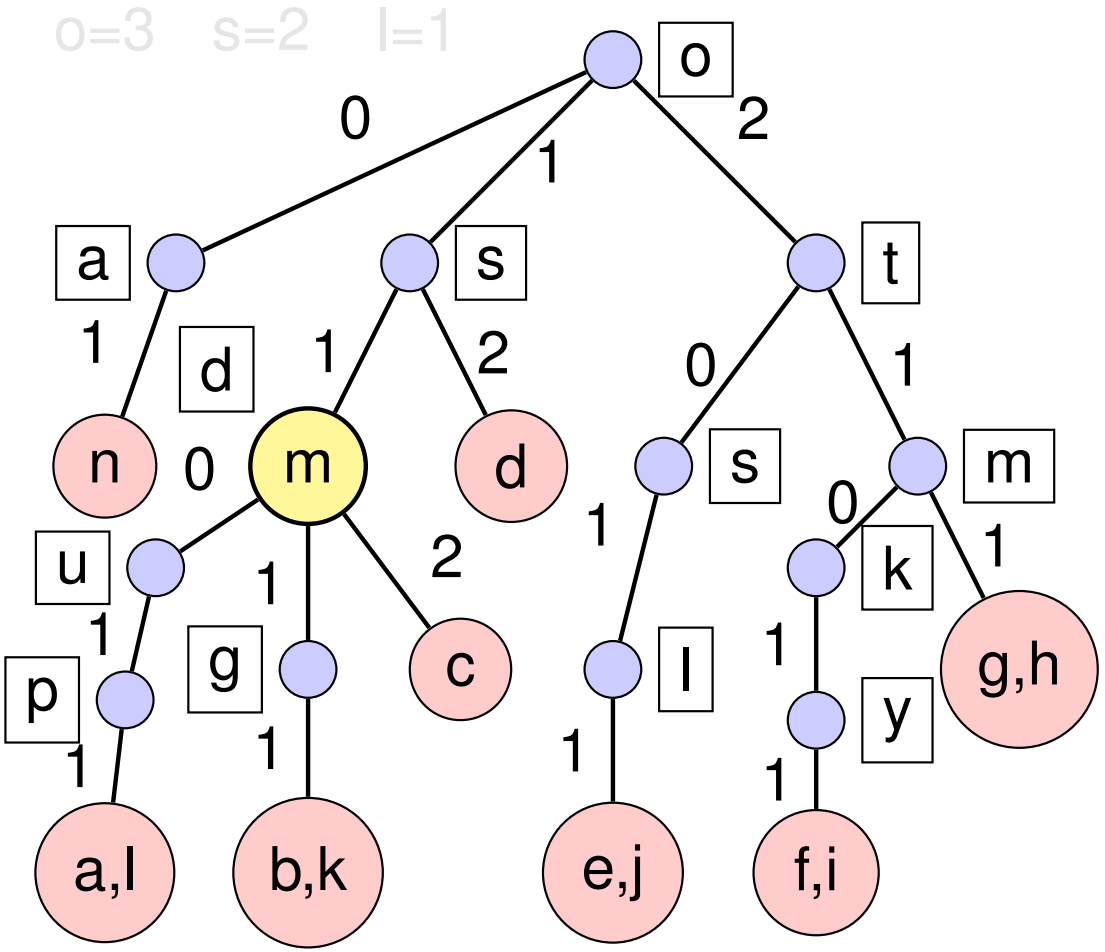
1. $\forall x \in \mathcal{I}$, search (y, z) such that p is true
   - ▷ *linear number of searches*
2. check $[x : y = z : t]$
   - ▷ *efficient if (y, z) can be retrieved efficiently given (x, t)*
   - ▷ *and if the number of checkings is small*
   - ▷ *manageable up to a few hundreds of thousand forms*

Efficiently Retrieving (y, z) pairs given (x, t)



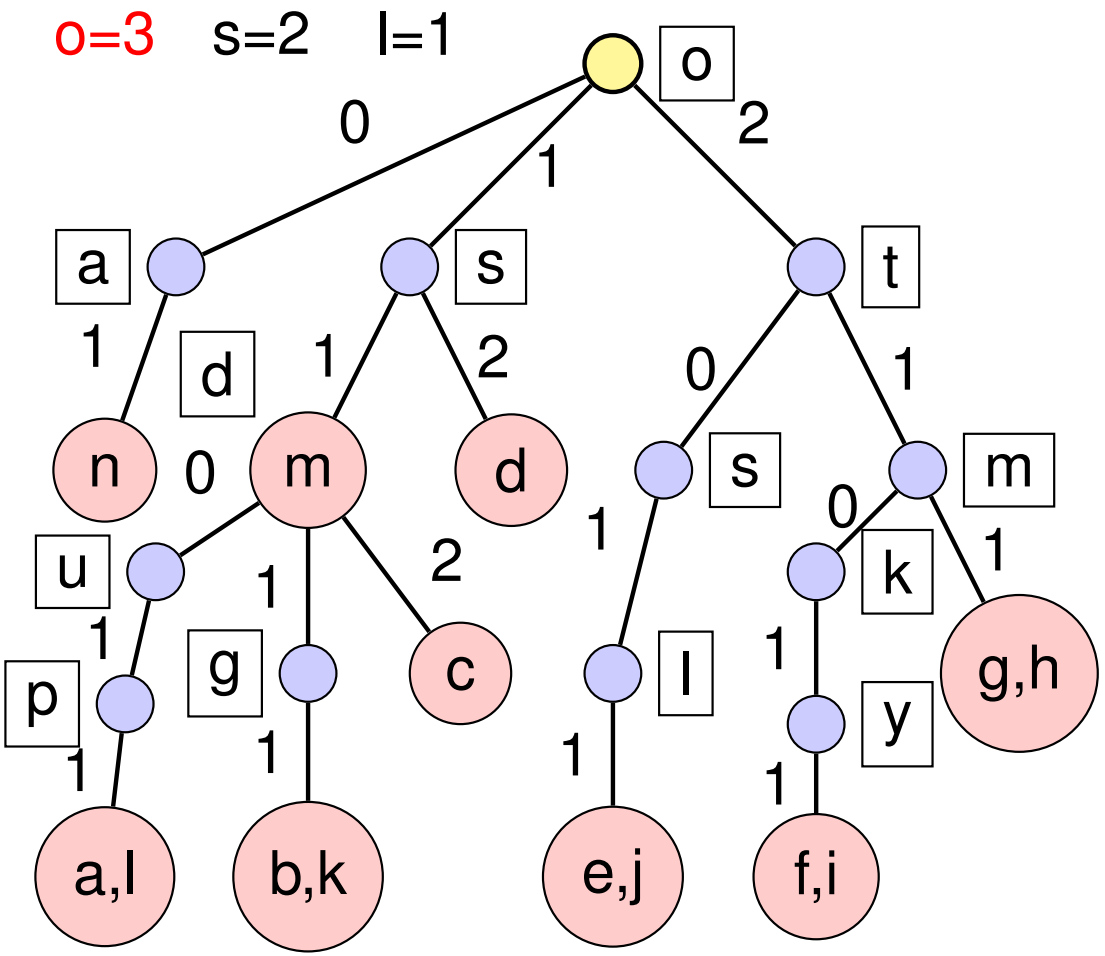
a	soup
b	gods
c	odds
d	sos
e	solo
f	tokyo
g	moot
h	moto
i	kyoto
j	oslo
k	dogs
l	opus
m	os
n	a

Efficiently Retrieving (y, z) pairs given (x, t)



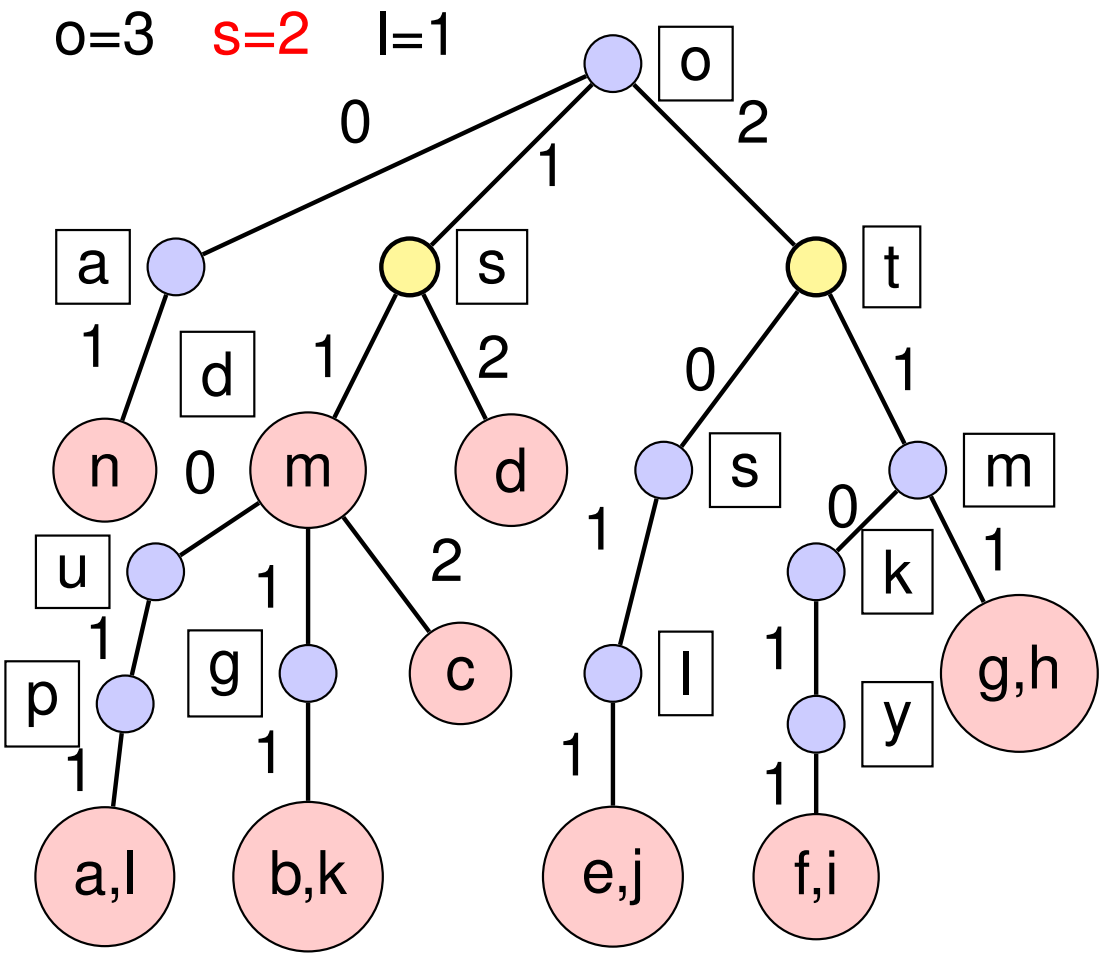
a	**soup**
b	**gods**
c	**odds**
d	sos
e	solo
f	tokyo
g	moot
h	moto
i	kyoto
j	oslo
k	**dogs**
l	**opus**
m	**os**
n	a

Efficiently Retrieving (y, z) pairs given (x, t)



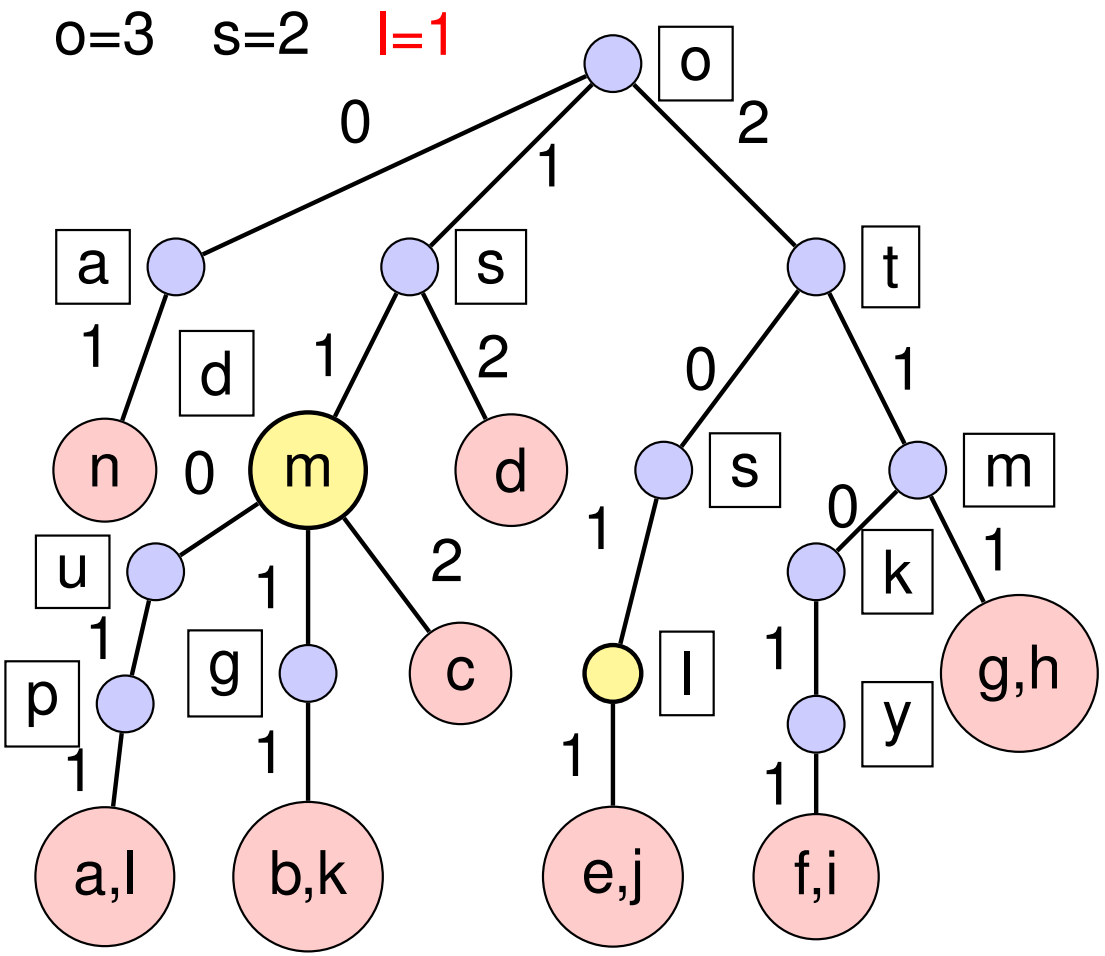
a	soup
b	gods
c	odds
d	sos
e	solo
f	tokyo
g	moot
h	moto
i	kyoto
j	oslo
k	dogs
l	opus
m	os
n	a

Efficiently Retrieving (y, z) pairs given (x, t)



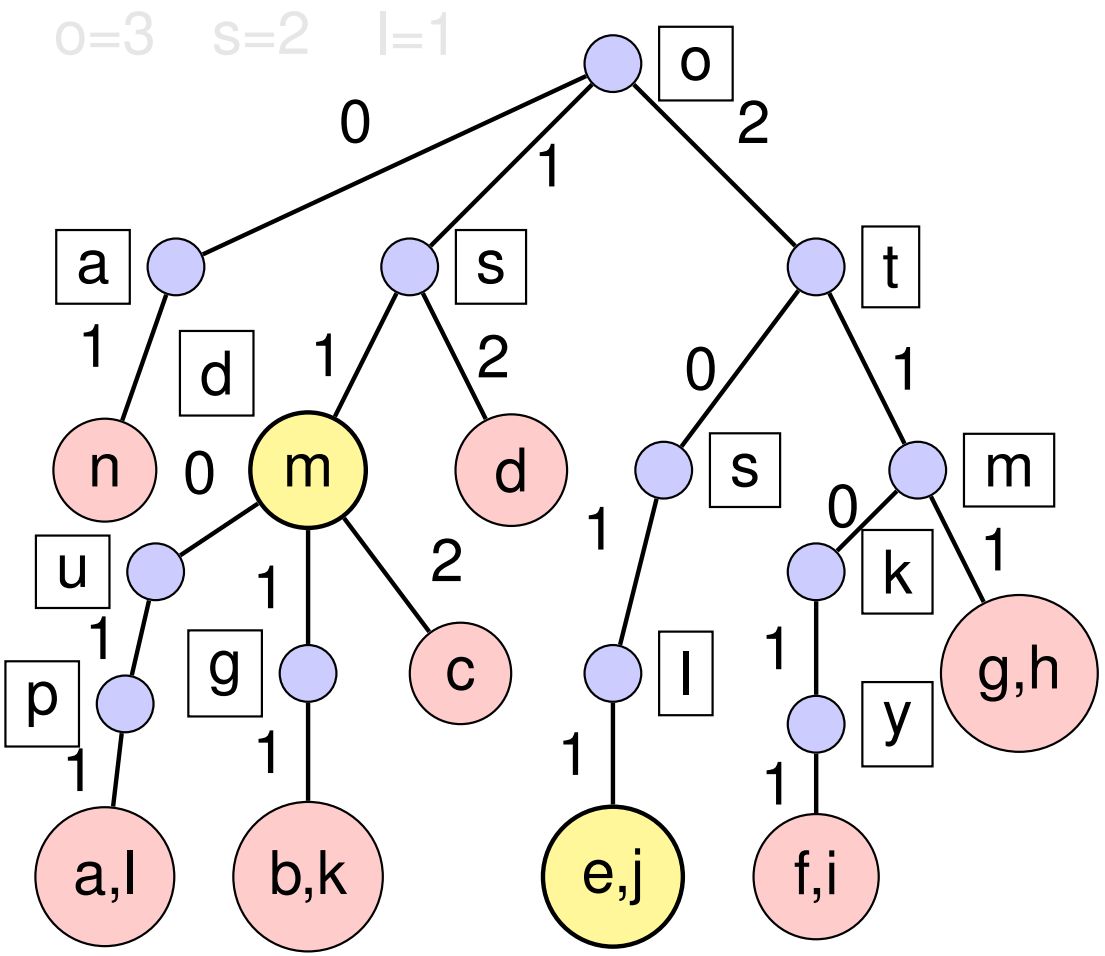
a	soup
b	gods
c	odds
d	sos
e	solo
f	tokyo
g	moot
h	moto
i	kyoto
j	oslo
k	dogs
l	opus
m	os
n	a

Efficiently Retrieving (y, z) pairs given (x, t)



a	soup
b	gods
c	odds
d	sos
e	solo
f	tokyo
g	moot
h	moto
i	kyoto
j	oslo
k	dogs
l	opus
m	os
n	a

Efficiently Retrieving (y, z) pairs given (x, t)



a	soup
b	gods
c	odds
d	sos
e	**solo**
f	tokyo
g	moot
h	moto
i	kyoto
j	**oslo**
k	dogs
l	opus
m	**os**
n	a

Exhaustive Retrieval Time

avg. time to retrieve all (y, z) that verify p knowing (x, t):

$\lvert\mathcal{I}\rvert$	time (ms)	|frontier|	|nodes| per form
10k	5.5e-05	38	6.8
100k	0.0003	150	6.3
1M	0.003	1082	6.6
2M	0.0055	1655	6.5
10M	0.02	3921	5.8

- memory and retrieval roughly linear with $|\mathcal{I}|$
- 0.02 ms on average to find all (y, z) verifying p knowing (x, t) for an input space of over 10 million forms

Exhaustive Retrieval Time

avg. time to retrieve all (y, z) that verify p knowing (x, t):

$\lvert\mathcal{I}\rvert$	time (ms)	|frontier|	|nodes| per form
10k	5.5e-05	38	6.8
100k	0.0003	150	6.3
1M	0.003	1082	6.6
2M	0.0055	1655	6.5
10M	0.02	3921	5.8

- memory and retrieval roughly linear with $|\mathcal{I}|$
- 0.02 ms on average to find all (y, z) verifying p knowing (x, t) for an input space of over 10 million forms

Exhaustive Retrieval Time

avg. time to retrieve all (y, z) that verify p knowing (x, t):

$\lvert\mathcal{I}\rvert$	time (ms)	|frontier|	|nodes| per form
10k	5.5e-05	38	6.8
100k	0.0003	150	6.3
1M	0.003	1082	6.6
2M	0.0055	1655	6.5
10M	0.02	3921	5.8

- memory and retrieval roughly linear with $\lvert\mathcal{I}\rvert$
- 0.02 ms on average to find all (y, z) verifying p knowing (x, t) for an input space of over 10 million forms

Exhaustive Retrieval Time

avg. time to retrieve all (y, z) that verify p knowing (x, t):

$\lvert\mathcal{I}\rvert$	time (ms)	|frontier|	|nodes| per form
10k	5.5e-05	38	6.8
100k	0.0003	150	6.3
1M	0.003	1082	6.6
2M	0.0055	1655	6.5
10M	0.02	3921	5.8

- memory and retrieval roughly linear with $|\mathcal{I}|$
- 0.02 ms on average to find all (y, z) verifying p knowing (x, t) for an input space of over 10 million forms

Exhaustive Retrieval Time

ratio	time (ms)	|frontier|	|nodes| per form
100k	0.0003	150	6.3
1M	0.003	1082	6.6
10M	0.02	3921	5.8

- $|\mathcal{I}| \simeq 100k : 0.0003 \times 100k \simeq 0.3s$ *:-)*
- $|\mathcal{I}| \simeq 1M : 0.003 \times 1M \simeq 3s$ *:-0*
- $|\mathcal{I}| \simeq 10M : 0.02 \times 10M \simeq 200s$ $(> 3$ min.$)$ *:-(*

(not counting analogy checking)

Definition of [Stroppa and Yvon, 2005]

- $[x : y = z : t]$ **iff** we can find **factorizations** f_x, f_y, f_z and f_t such that, $\forall i \in [1, d]$:

$$(f_y^{(i)}, f_z^{(i)}) \in \left\{(f_x^{(i)}, f_t^{(i)}), (f_t^{(i)}, f_x^{(i)})\right\}$$

  - $f_x^{(i)}, f_y^{(i)}, f_z^{(i)}$ and $f_t^{(i)}$ are called the **factors**
  - the smallest d for which this holds is called the **degree**

- $[$*this guy drinks too much* : *this boat sinks* $=$ *these guys drank too much* : *these boats sank*$]$ because:

x	$\equiv$	this	guy	ϵ	dr	inks	too much
y	$\equiv$	this	boat	ϵ	s	inks	ϵ
z	$\equiv$	these	guy	s	dr	ank	too much
t	$\equiv$	these	boat	s	s	ank	ϵ

  - the degree of this analogy is 6

Definition of [Stroppa and Yvon, 2005]

- $[x : y = z : t]$ **iff** we can find **factorizations** f_x, f_y, f_z and f_t such that, $\forall i \in [1, d]$:

$$(f_y^{(i)}, f_z^{(i)}) \in \left\{(f_x^{(i)}, f_t^{(i)}), (f_t^{(i)}, f_x^{(i)})\right\}$$

  - $f_x^{(i)}, f_y^{(i)}, f_z^{(i)}$ and $f_t^{(i)}$ are called the **factors**
  - the smallest d for which this holds is called the **degree**

- $[$*this guy drinks too much* : *this boat sinks* $=$ *these guys drank too much* : *these boats sank*$]$ because:

x	$\equiv$	this	guy	ϵ	dr	inks	too much
y	$\equiv$	this	boat	ϵ	s	inks	ϵ
z	$\equiv$	these	guy	s	dr	ank	too much
t	$\equiv$	these	boat	s	s	ank	ϵ

  - the degree of this analogy is 6

Checking for an analogy

[Stroppa, 2005]

a(i,j,k,l) $\leftarrow$ **false** , **if** i, j, k or l < 0
for i $\leftarrow$ 0 **to** $|x|$ **do**
  for j $\leftarrow$ 0 **to** $|y|$ **do**
    for k $\leftarrow$ 0 **to** $|z|$ **do**
      for l $\leftarrow$ 0 **to** $|t|$ **do**
        if i = j = k = l **then**
          a(i,j,k,l) $\leftarrow$ **true**
        else
          a(i,j,k,l) $\leftarrow$
          or
          $\begin{cases} \text{a(i-1,j-1,k,l)} \wedge x[i] = y[j] \\ \text{a(i-1,j,k-1,l)} \wedge x[i] = z[k] \\ \text{a(i,j-1,k,l-1)} \wedge t[l] = y[j] \\ \text{a(i,j,k-1,l-1)} \wedge t[l] = z[k] \end{cases}$
return a($|x|, |y|, |z|, |t|$)

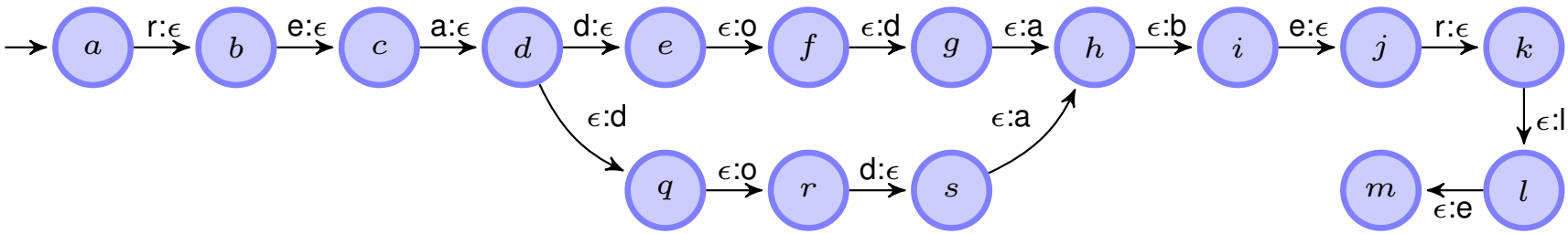
- too slow . . .
- $\sim$ 35% savings when first checking for:

$$[x : y = z : t] \Rightarrow$$
$$\{x[1], t[1]\} \cap \{y[1], z[1]\} \neq \phi$$
$$\{x[\$], t[\$]\} \cap \{y[\$], z[\$]\} \neq \phi$$

- timeout if too many candidate analogies to check

Solving an Analogical Equation

- We can built a finite-state **transducer** which produces the solutions to $[x : y = z : ?]$ while recognizing the form x [Yvon, 2003]

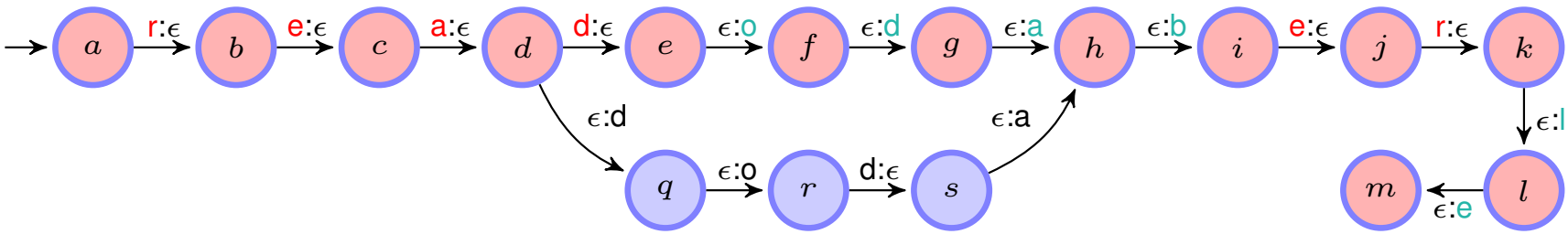


$[\textit{reader} : \textit{readable} = \textit{doer} : ?] \triangleright$ odable, **doable**

- **problem**: highly non deterministic automaton
  - ▷ *we may face combinatorial problems when building it*
- we sample the automaton without building it

Solving an Analogical Equation

- We can built a finite-state **transducer** which produces the solutions to $[x : y = z : ?]$ while recognizing the form x [Yvon, 2003]

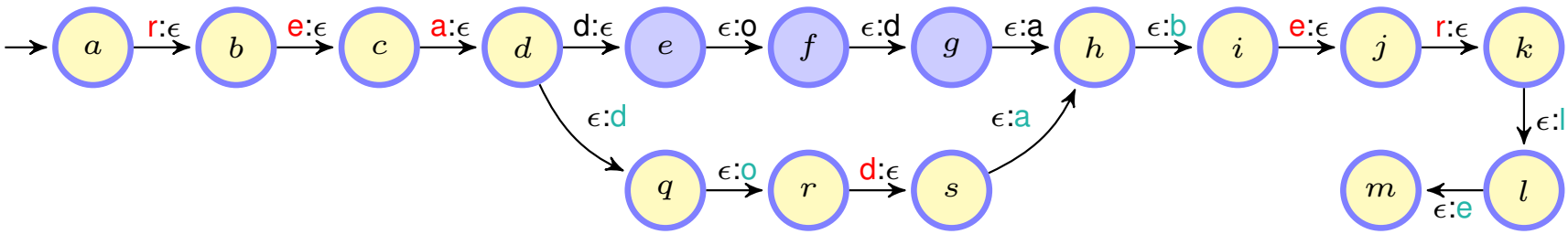


$[\textit{reader} : \textit{readable} = \textit{doer} : ?] \triangleright$ odable, doable

- **problem**: highly non deterministic automaton
  - ▷ *we may face combinatorial problems when building it*
- we sample the automaton without building it

Solving an Analogical Equation

- We can built a finite-state **transducer** which produces the solutions to $[x : y = z : ?]$ while recognizing the form x [Yvon, 2003]

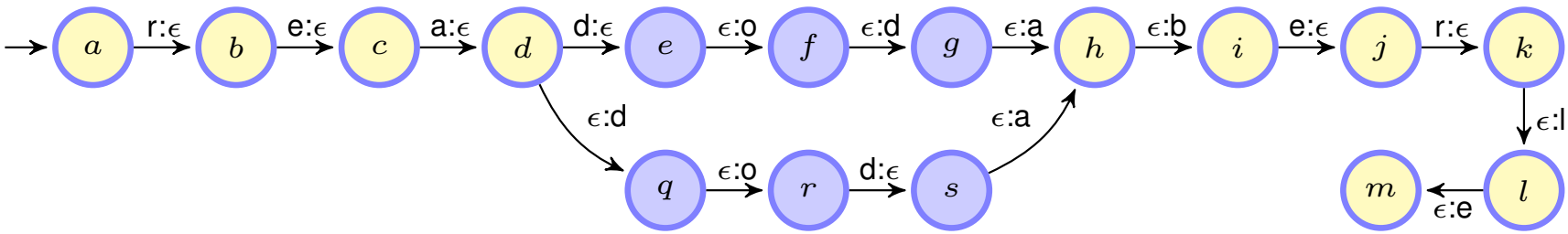


$[\textit{reader} : \textit{readable} = \textit{doer} : ?] \triangleright$ odable, **doable**

- **problem**: highly non deterministic automaton
  - ▷ *we may face combinatorial problems when building it*
- we sample the automaton without building it

Solving an Analogical Equation

- We can built a finite-state **transducer** which produces the solutions to $[x : y = z : ?]$ while recognizing the form x [Yvon, 2003]

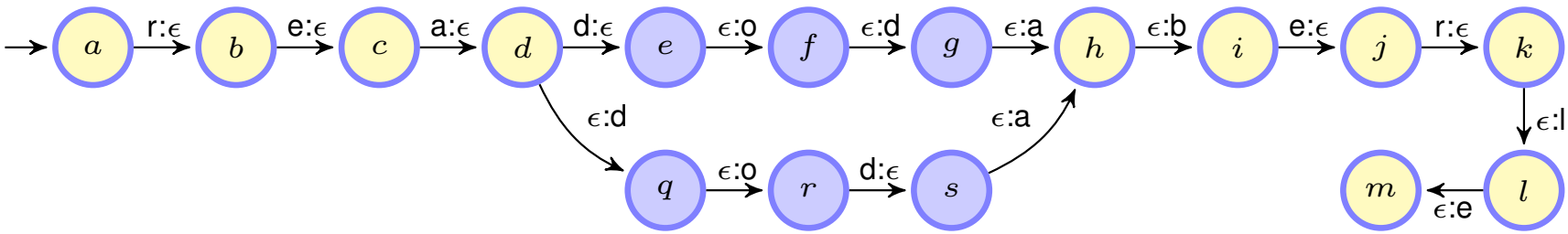


$[\textit{reader} : \textit{readable} = \textit{doer} : ?] \triangleright$ odable, **doable**

- **problem**: highly non deterministic automaton
  - ▷ *we may face combinatorial problems when building it*
- we sample the automaton without building it

Solving an Analogical Equation

- We can built a finite-state **transducer** which produces the solutions to $[x : y = z : ?]$ while recognizing the form x [Yvon, 2003]



$[reader : readable = doer : ?] \triangleright$ odable, **doable**

- **problem**: highly non deterministic automaton
  - $\triangleright$ *we may face combinatorial problems when building it*
- we sample the automaton without building it

$$[\textit{even} : \textit{usual} = \textit{unevenly} : ?]$$

ρ	nb	solutions		
20	12	*usuaunlly* (3)	***unusually*** (2)	*usunually* (2)
100	34	***unusually*** (6)	*usuaunlly* (6)	*uunsually* (4)
1000	67	***unusually*** (57)	*uunsually* (23)	*usuunally* (19)
2000	72	***unusually*** (130)	*uunsually* (77)	*usunually* (43)

[*this guy drinks too much* : *this boat sinks* = *those guys drink too much* : *?*]

$\rho = 20$	$nb = 8$
$t = 0.0003$	$rank = \phi$
thos_boate_sinks (2)	
tho_boatse_sinks (2)	
thoboatse__sinks (2)	

$\rho = 100$	$nb = 28$
$t = 0.001$	$rank = 13$
thoboatse__sinks (2)	
tho_boatse_sinks (2)	
those_sboat_sink (2)	

$\rho = 1000$	$nb = 28$
$t = 0.009$	$rank = 2$
those_boat_ssink (5)	
those_boats_sink (5)	
thoes_tboa_sinks (5)	

$\rho = 10^6$	$nb = 19\,796$
$t = 3.82$	$rank = 10$
thoes_boat_sinks (2550)	
thoses_boat_sink (1037)	
those_boat_ssink (999)	

[*this guy drinks too much* : *this boat sinks* = *those guys drink too much* : *?*]

$\rho = 20$	$nb = 8$
$t = 0.0003$	$rank = \phi$
thos_boate_sinks (2)	
tho_boatse_sinks (2)	
thoboatse__sinks (2)	

$\rho = 100$	$nb = 28$
$t = 0.001$	$rank = 13$
thoboatse__sinks (2)	
tho_boatse_sinks (2)	
those_sboat_sink (2)	

$\rho = 1000$	$nb = 28$
$t = 0.009$	$rank = 2$
those_boat_ssink (5)	
those_boats_sink (5)	
thoes_tboa_sinks (5)	

$\rho = 10^6$	$nb = 19\,796$
$t = 3.82$	$rank = 10$
thoes_boat_sinks (2550)	
thoses_boat_sink (1037)	
those_boat_ssink (999)	

[*this guy drinks too much* : *this boat sinks* = *those guys drink too much* : *?*]

$\rho = 20$	$nb = 8$
$t = 0.0003$	$rank = \phi$
thos_boate_sinks (2)	
tho_boatse_sinks (2)	
thoboatse__sinks (2)	

$\rho = 100$	$nb = 28$
$t = 0.001$	$rank = 13$
thoboatse__sinks (2)	
tho_boatse_sinks (2)	
those_sboat_sink (2)	

$\rho = 1000$	$nb = 28$
$t = 0.009$	$rank = 2$
those_boat_ssink (5)	
those_boats_sink (5)	
thoes_tboa_sinks (5)	

$\rho = 10^6$	$nb = 19\,796$
$t = 3.82$	$rank = 10$
thoes_boat_sinks (2550)	
thoses_boat_sink (1037)	
those_boat_ssink (999)	

[*this guy drinks too much* : *this boat sinks* = *those guys drink too much* : *?*]

$\rho = 20$	$nb = 8$
$t = 0.0003$	$rank = \phi$
thos_boate_sinks (2)	
tho_boatse_sinks (2)	
thoboatse__sinks (2)	

$\rho = 100$	$nb = 28$
$t = 0.001$	$rank = 13$
thoboatse__sinks (2)	
tho_boatse_sinks (2)	
those_sboat_sink (2)	

$\rho = 1000$	$nb = 28$
$t = 0.009$	$rank = 2$
those_boat_ssink (5)	
those_boats_sink (5)	
thoes_tboa_sinks (5)	

$\rho = 10^6$	$nb = 19\,796$
$t = 3.82$	$rank = 10$
thoes_boat_sinks (2550)	
thoses_boat_sink (1037)	
those_boat_ssink (999)	

Dealing with over-generation

- filtering by frequency
  [Lepage and Denoual 2005, Stroppa and Yvon, 2005, Denoual, 2007]
- filtering forms unseen in a (large) set of (output) forms
  [Langlais and Patry, 2007]
- filtering forms containing character-ngrams unseen in the training material
  [Lepage and Lardilleux]
- learning to recognize meaningful **examples** from bad ones
  [Langlais et al. 2009]

Supervised Classification of Good Examples

$$[\textit{androlologie} : \textit{pathologie} = \textit{androgène} : \textit{pathogène}]$$
$$[\textit{andrology} : \textit{pathology} = \textit{androgen} : \textit{paogthen}] \quad ☹$$
$$\vdots$$
$$[\textit{otologiste} : \textit{pathologiste} = \textit{otogène} : \textit{pathogène}]$$
$$[\textit{otologist} : \textit{pathologist} = \textit{otogenic} : \textit{pathogenic}] \quad ☺$$

- Examples of features:
  1. degree of the input and output analogies,
  2. frequency of a candidate translation,
  3. character-based ngram probabilities given to a candidate translation,
  4. code-books of factors involved, etc.

Plan

NEWS Evaluation Campaign
[Li et al., 2009]

	train	*dev*	*test*
examples	31 961	2 896	2 896
EN symbols	26	26	26
CH symbols	370	275	283

examples	
Emission	埃米申
Blagrove	布格夫
Aposhian	阿 波 希 安

- 2 configurations:
  1. FULL-TC: exhaustive tree-count search
  2. SAMP-TC: 1 000 closest forms to each t in the vector space represented by the 1 000 most frequent character 3-grams

  - timeout (1 min.) in step 1
  - sampling rate $\rho = 200$

Calibrating

	FULL-TC	SAMP-TC
avg. time (step-1)	17s	2s
avg. time (step-2)	0.22s	0.01s
number of timeouts	327	1
avg. input analogies	4517	158
avg. output equations	487	18
avg. number of solutions	405	37.5
silence (step-1)	18	76
silence (step-2)	50	249

Calibrating

	FULL-TC	SAMP-TC
avg. time (step-1)	17s	2s
avg. time (step-2)	0.22s	0.01s
number of timeouts	327	1
avg. input analogies	4517	158
avg. output equations	487	18
avg. number of solutions	405	37.5
silence (step-1)	18	76
silence (step-2)	50	249

Calibrating

	FULL-TC	SAMP-TC
avg. time (step-1)	17s	2s
avg. time (step-2)	0.22s	0.01s
number of timeouts	327	1
avg. input analogies	4517	158
avg. output equations	487	18
avg. number of solutions	405	37.5
silence (step-1)	18	76
silence (step-2)	50	249

Calibrating

	FULL-TC	SAMP-TC
avg. time (step-1)	17s	2s
avg. time (step-2)	0.22s	0.01s
number of timeouts	327	1
avg. input analogies	4517	158
avg. output equations	487	18
avg. number of solutions	405	37.5
silence (step-1)	18	76
silence (step-2)	50	249

Calibrating

	FULL-TC	SAMP-TC
avg. time (step-1)	17s	2s
avg. time (step-2)	0.22s	0.01s
number of timeouts	327	1
avg. input analogies	4517	158
avg. output equations	487	18
avg. number of solutions	405	37.5
silence (step-1)	18	76
silence (step-2)	50	249

Performance

rank	nb	$r_{2374}\%$	$r_{all}\%$	nb	$r_{1659}\%$	$r_{all}\%$
1	1093	46.0	37.7	1410	85.0	48.7
2	1418	59.7	48.9	1627	98.1	56.2
3	1582	66.6	54.6	1657	99.9	57.2
4	1699	71.6	58.7	1659	100.0	57.3
5	1796	75.7	62.0	.	.	.
⋮	⋮	⋮	⋮	⋮	⋮	⋮
100	2374	100.0	82.0	1659	100.0	57.3
	generator			generator+selector		

- 2374 of the 2896 (82%) test forms are found in the 100-first solutions proposed by the generator

Performance

rank	nb	$r_{2374}\%$	$r_{all}\%$	nb	$r_{1659}\%$	$r_{all}\%$
1	1093	46.0	37.7	1410	85.0	48.7
2	1418	59.7	48.9	1627	98.1	56.2
3	1582	66.6	54.6	1657	99.9	57.2
4	1699	71.6	58.7	1659	100.0	57.3
5	1796	75.7	62.0	.	.	.
⋮	⋮	⋮	⋮	⋮	⋮	⋮
100	2374	100.0	82.0	1659	100.0	57.3
	generator			generator+selector		

- for 1093 test forms (46% / 37.7%), the first solution proposed by the generator is the good one

Performance

rank	nb	$r_{2374}\%$	$r_{all}\%$	nb	$r_{1659}\%$	$r_{all}\%$
1	1093	46.0	37.7	1410	85.0	48.7
2	1418	59.7	48.9	1627	98.1	56.2
3	1582	66.6	54.6	1657	99.9	57.2
4	1699	71.6	58.7	1659	100.0	57.3
5	1796	75.7	62.0	.	.	.
⋮	⋮	⋮	⋮	⋮	⋮	⋮
100	2374	100.0	82.0	1659	100.0	57.3
	generator			generator+selector		

- only 1659 test forms solved at rank 100 (or less)

Performance

rank	nb	$r_{2374}\%$	$r_{all}\%$	nb	$r_{1659}\%$	$r_{all}\%$
1	1093	46.0	37.7	1410	85.0	48.7
2	1418	59.7	48.9	1627	98.1	56.2
3	1582	66.6	54.6	1657	99.9	57.2
4	1699	71.6	58.7	1659	100.0	57.3
5	1796	75.7	62.0	.	.	.
⋮	⋮	⋮	⋮	⋮	⋮	⋮
100	2374	100.0	82.0	1659	100.0	57.3
	generator			generator+selector		

- generator + selector $>>$ generator at low ranks

  (especially at rank 1)

Official metrics

- configuration: generator + selector

metric	FULL-TC	SAMP-TC	$1st$	$last$
ACC:	0.486	0.308	0.731	0.199
F-score	0.772	0.612	0.895	0.606
MRR	0.527	0.330	0.812	0.229
MAP_{ref}	0.486	0.308	0.652	0.199

- 14th out of 18 participating systems at NEWS'09
- without lowerizing the English forms, the generator alone is performing better (ACC: 0.493 versus 0.378) ...

Examples

EN forms	reference	solutions	nb	r_g	r_{g+s}
auchter	x 克 特	*x* 克 特 (218)	380	1	1
sundell	森 德 y	森 德 *y* (692)	664	5	1
fannin	范 宁	范 妮 恩 (54)	104	5	ϕ
frere	弗 里 y	弗 里 *y* (6113)	630	1	1
shurkin	舒 金	舒 *y* 金 (237) 舒 金 (208)	386	3	2

- $nb \equiv$ number of solutions produced
- $r_g \equiv$ rank of the correct solution by the generator only

  (ϕ means no correct solution)
- $r_{g+s} \equiv$ rank of the correct solution by the generator+selector

Discussion

- practical solutions to scale analogical learning
- case study on transliteration with encouraging results (deserves further investigations)
- many issues await solutions:

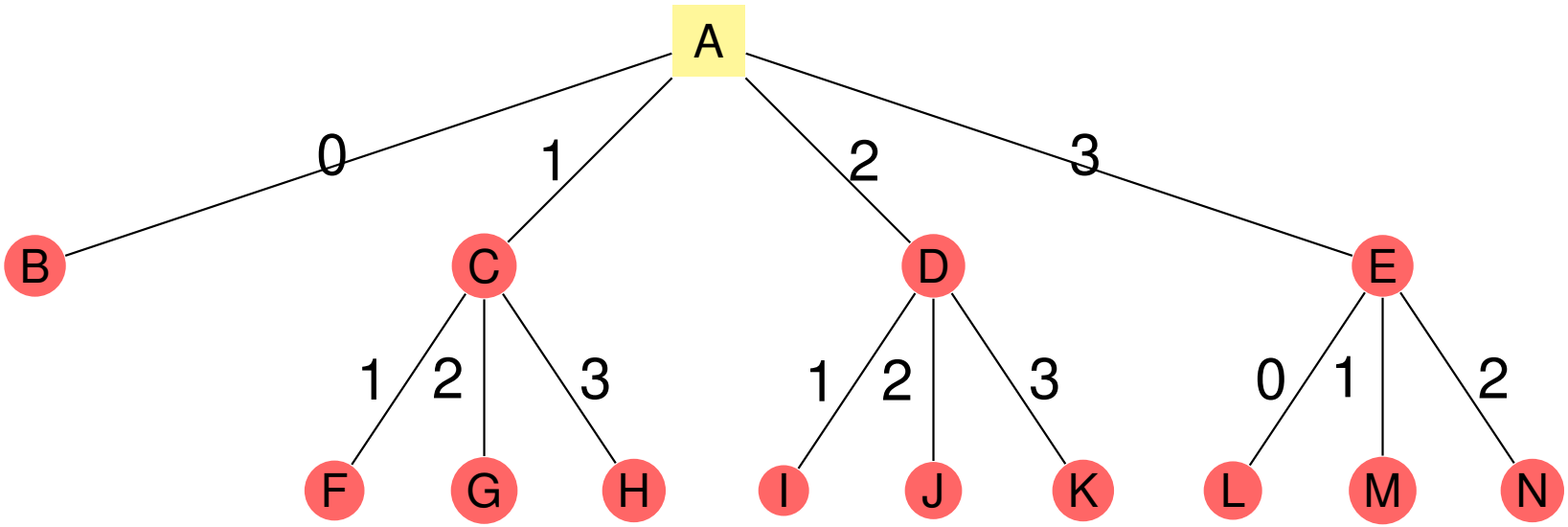
1. silence (more acute for spaces with large forms)
2. noise (must learn to weight the solutions)
3. better aggregation of solutions (ranking instead of classifying)
4. reducing the training set to its minimum **analogical support**

$$\underbrace{\{[x : y = z : ?] : (x, y, z) \in A^3\}}_{\text{analogical support of E}} \supseteq E$$

5. learning over tree structures
   [Stroppa and Yvon, 2007, Hassena, 2011]

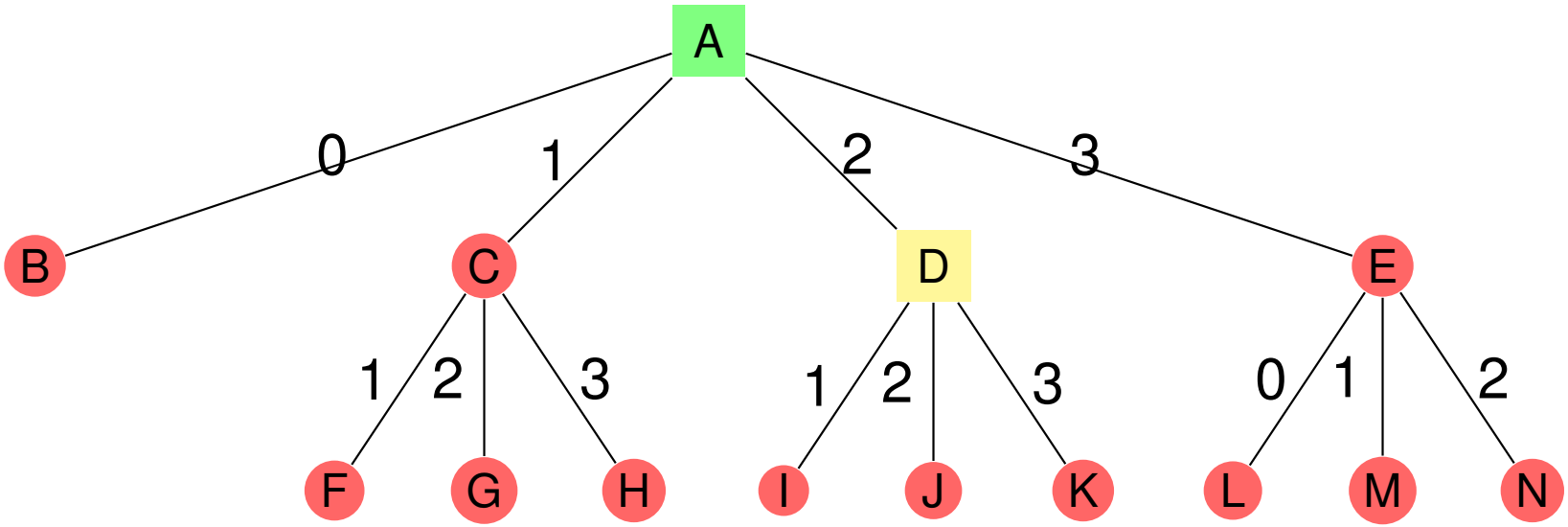
Questions / Suggestions ?

Retrieval controlled by the frontier size



level 0		(A,A)
level 1	4	(D,D) (C,E)
level 2	3	(I,J) (G,M) (H,L) (F,N)
...		

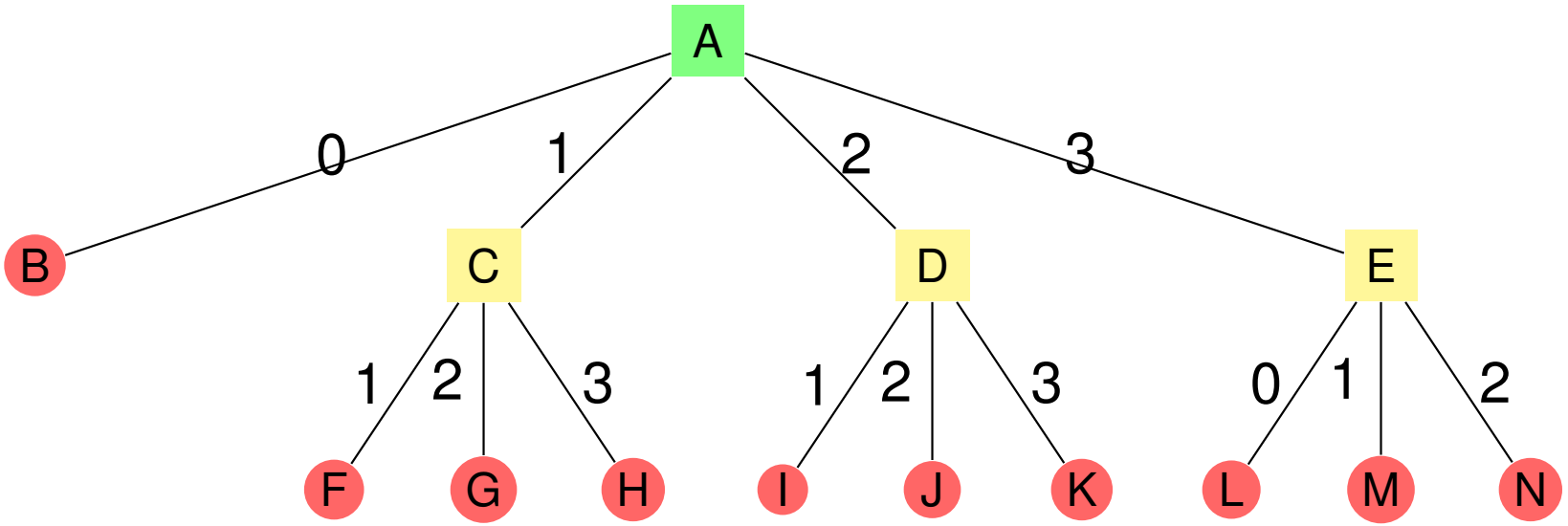
Retrieval controlled by the frontier size



level 0		(A,A)
level 1	4	(D,D) (C,E)
level 2	3	(I,J) (G,M) (H,L) (F,N)

. . .

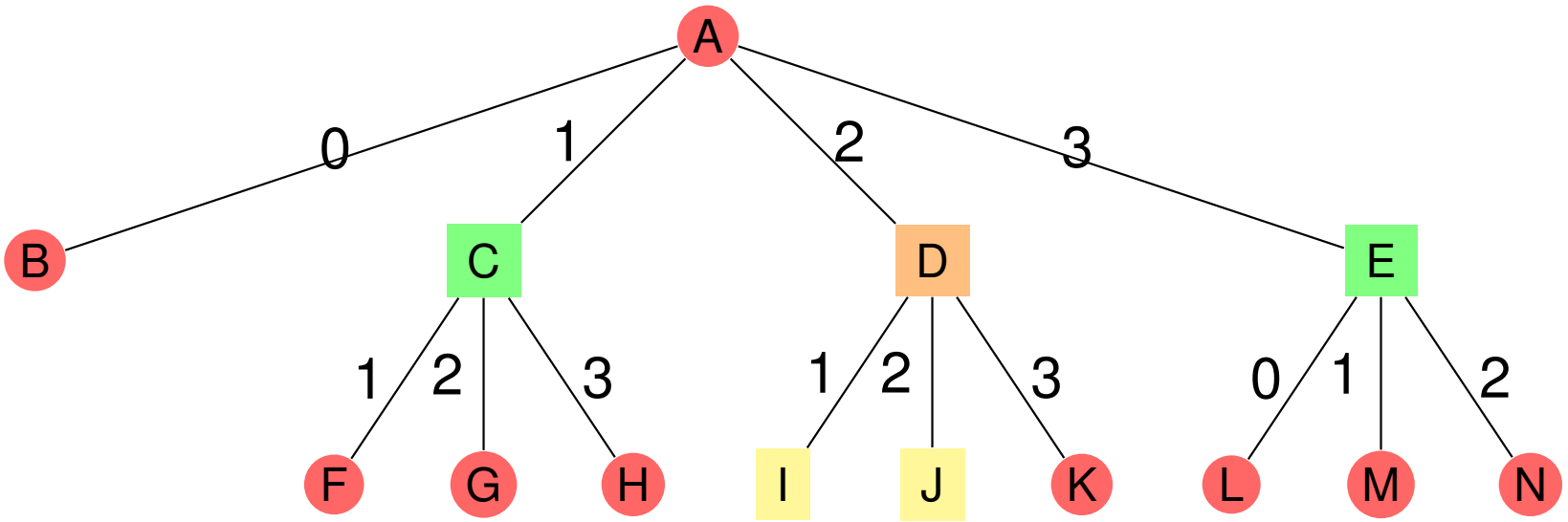
Retrieval controlled by the frontier size



level 0		(A,A)
level 1	4	(D,D) (C,E)
level 2	3	(I,J) (G,M) (H,L) (F,N)

. . .

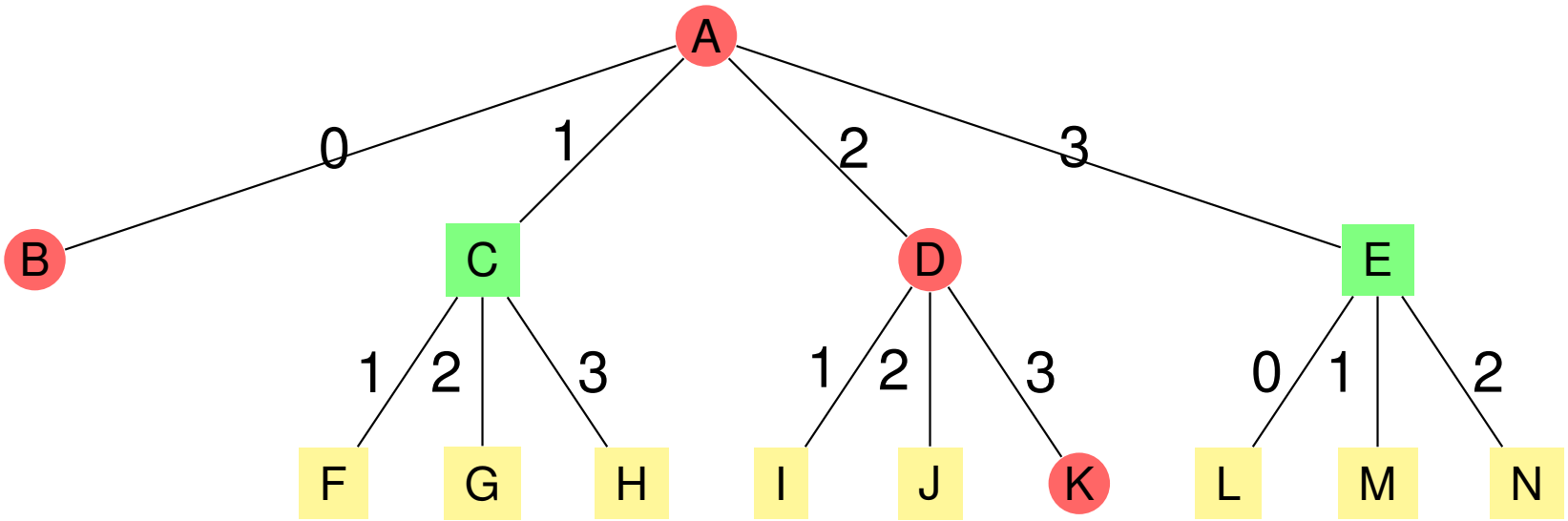
Retrieval controlled by the frontier size



level 0		(A,A)
level 1	4	(D,D) (C,E)
level 2	3	(I,J) (G,M) (H,L) (F,N)

. . .

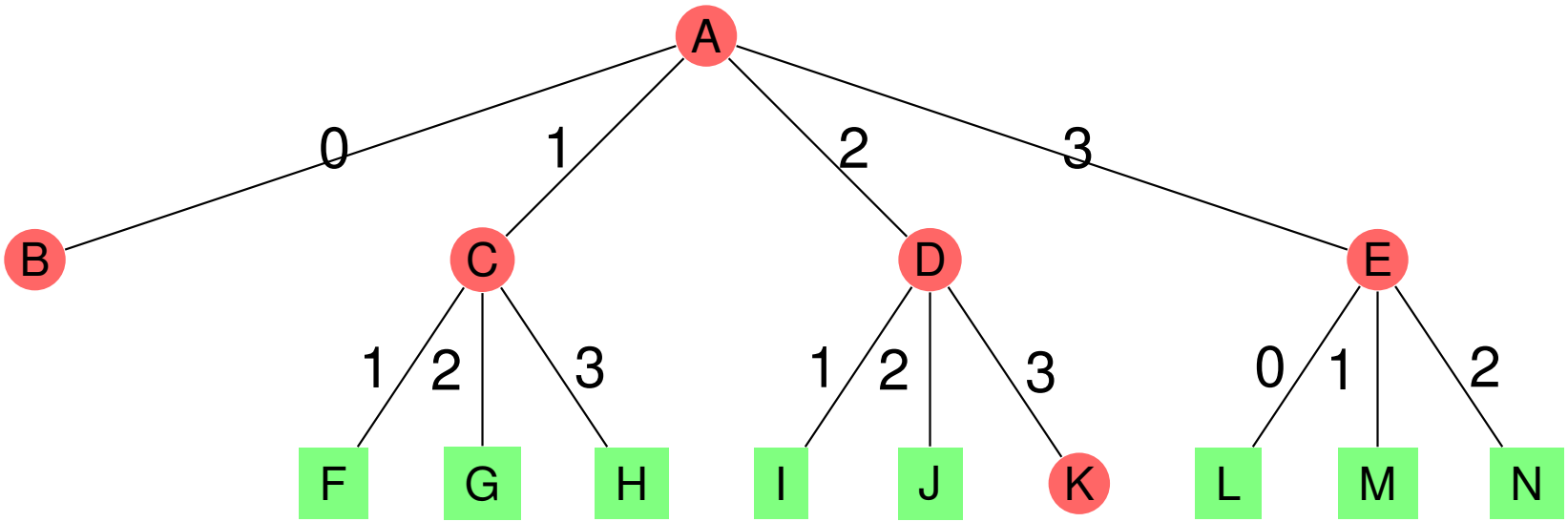
Retrieval controlled by the frontier size



level 0		(A,A)
level 1	4	(D,D) (C,E)
level 2	3	(I,J) (G,M) (H,L) (F,N)

. . .

Retrieval controlled by the frontier size



level 0		(A,A)
level 1	4	(D,D) (C,E)
level 2	3	(I,J) (G,M) (H,L) (F,N)
...		

Approximate Retrieval Time

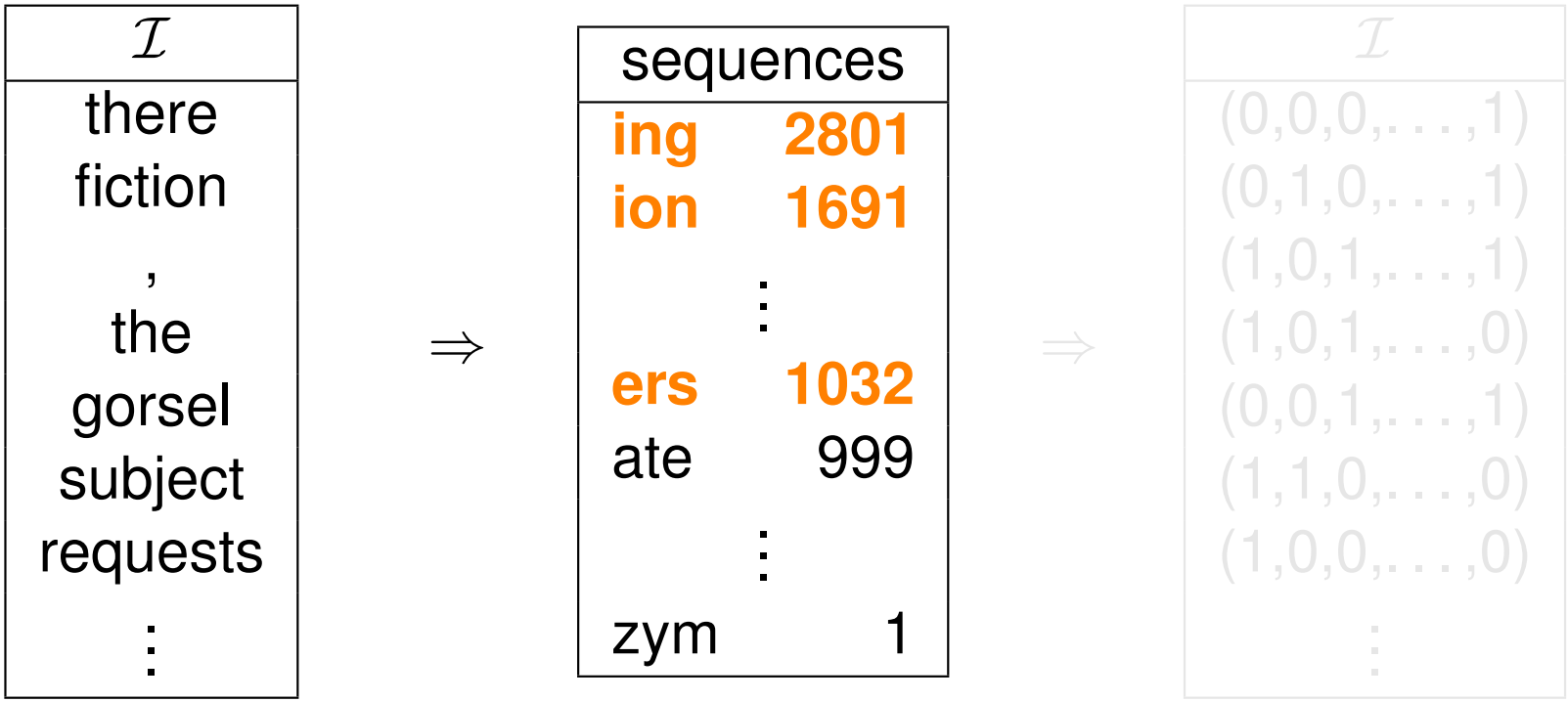
$\mathcal{I}$
there
fiction
,
the
gorsel
subject
requests
⋮

sequences	
ing	2801
ion	1691
⋮	
ers	1032
ate	999
⋮	
zym	1

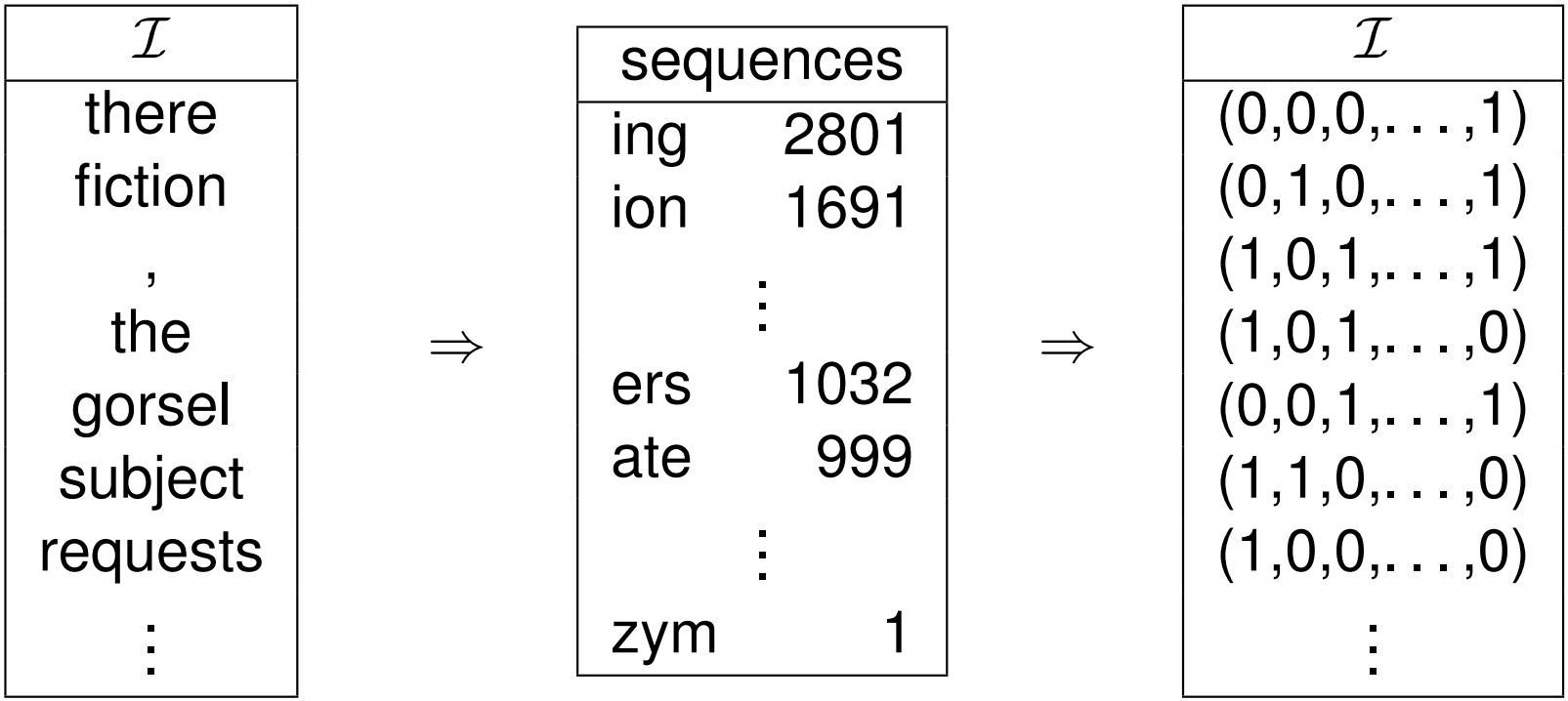
$\mathcal{I}$
(0,0,0,...,1)
(0,1,0,...,1)
(1,0,1,...,1)
(1,0,1,...,0)
(0,0,1,...,1)
(1,1,0,...,0)
(1,0,0,...,0)
⋮

$t \equiv (0,1,0,\ldots,0) \Rightarrow$ request,requests, query, quest, fiction, ...

Approximate Retrieval Time

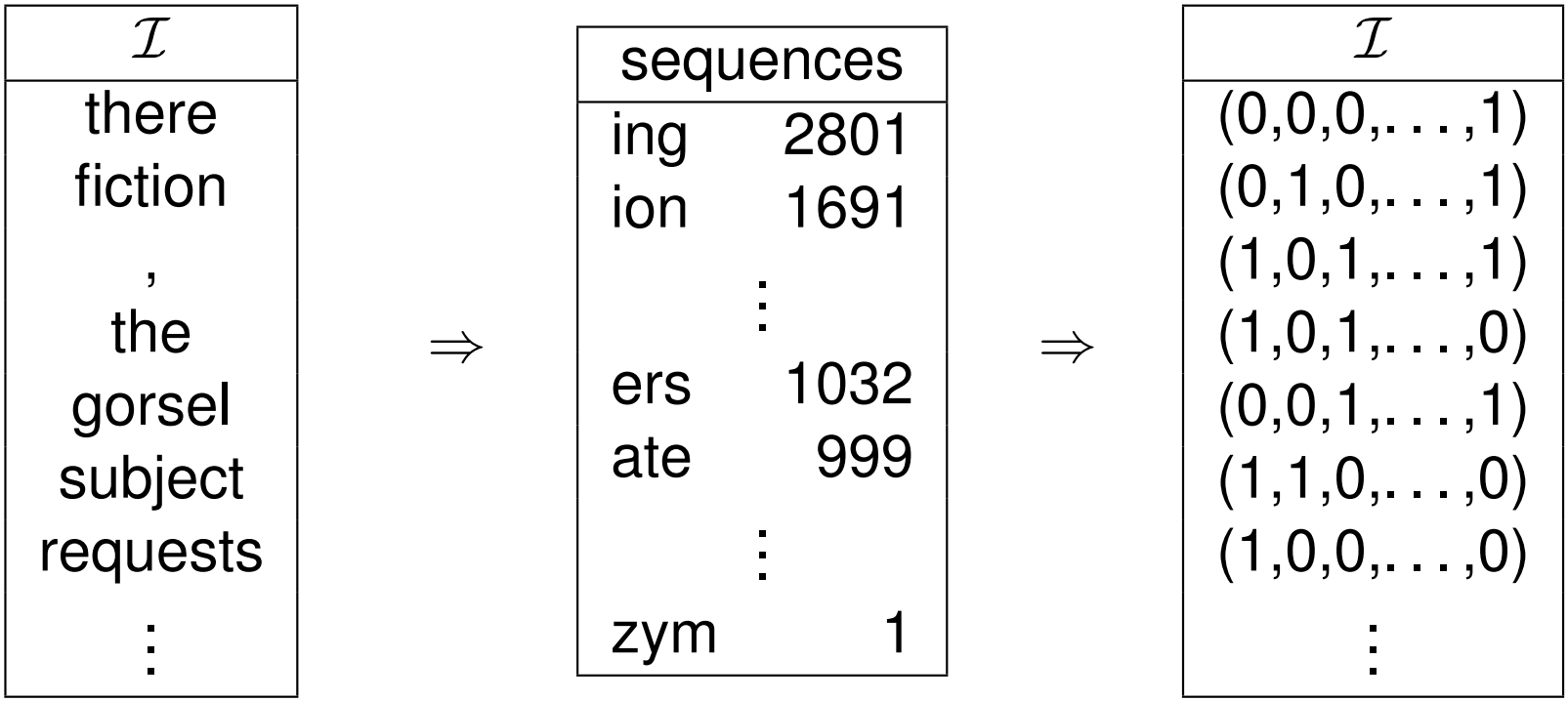


Approximate Retrieval Time



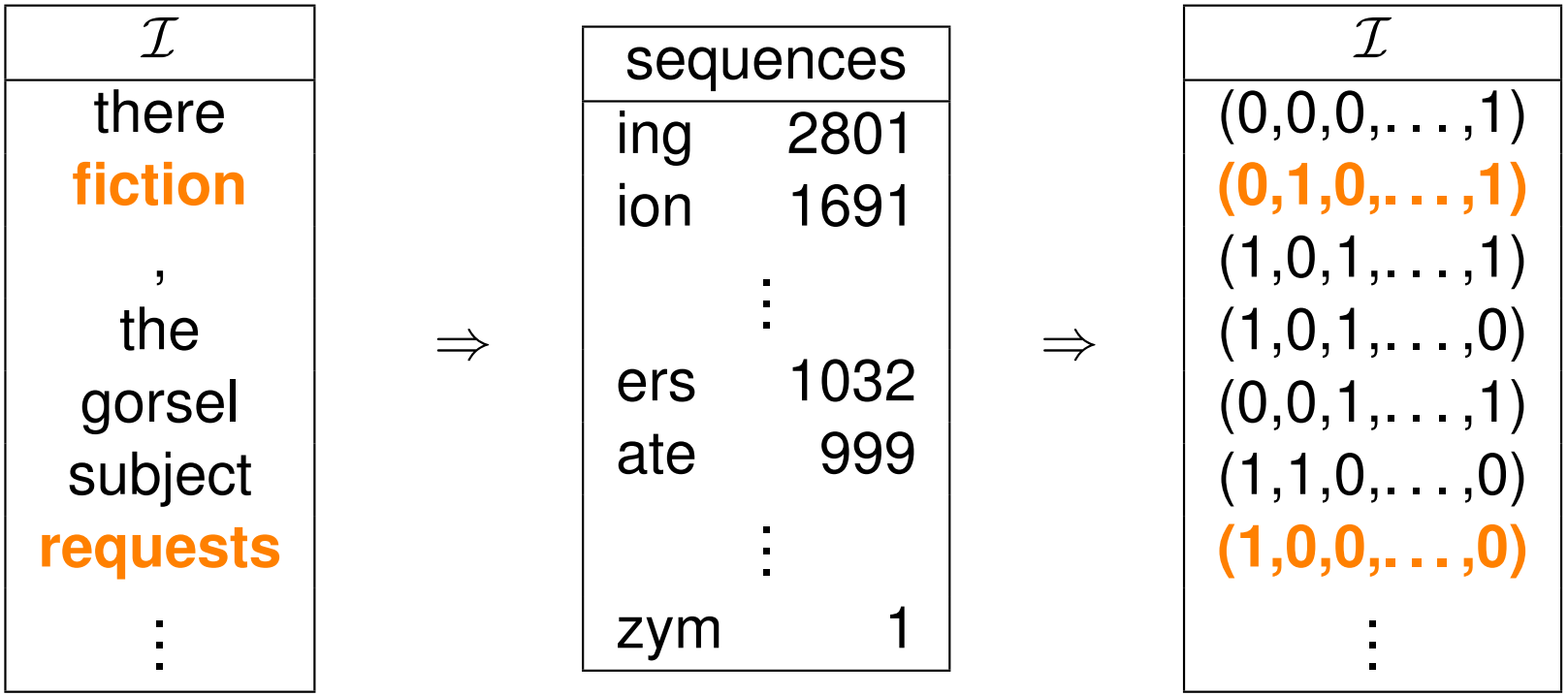
$t \equiv (0,1,0,\ldots,0) \Rightarrow$ request,requests, query, quest, fiction, ...

Approximate Retrieval Time



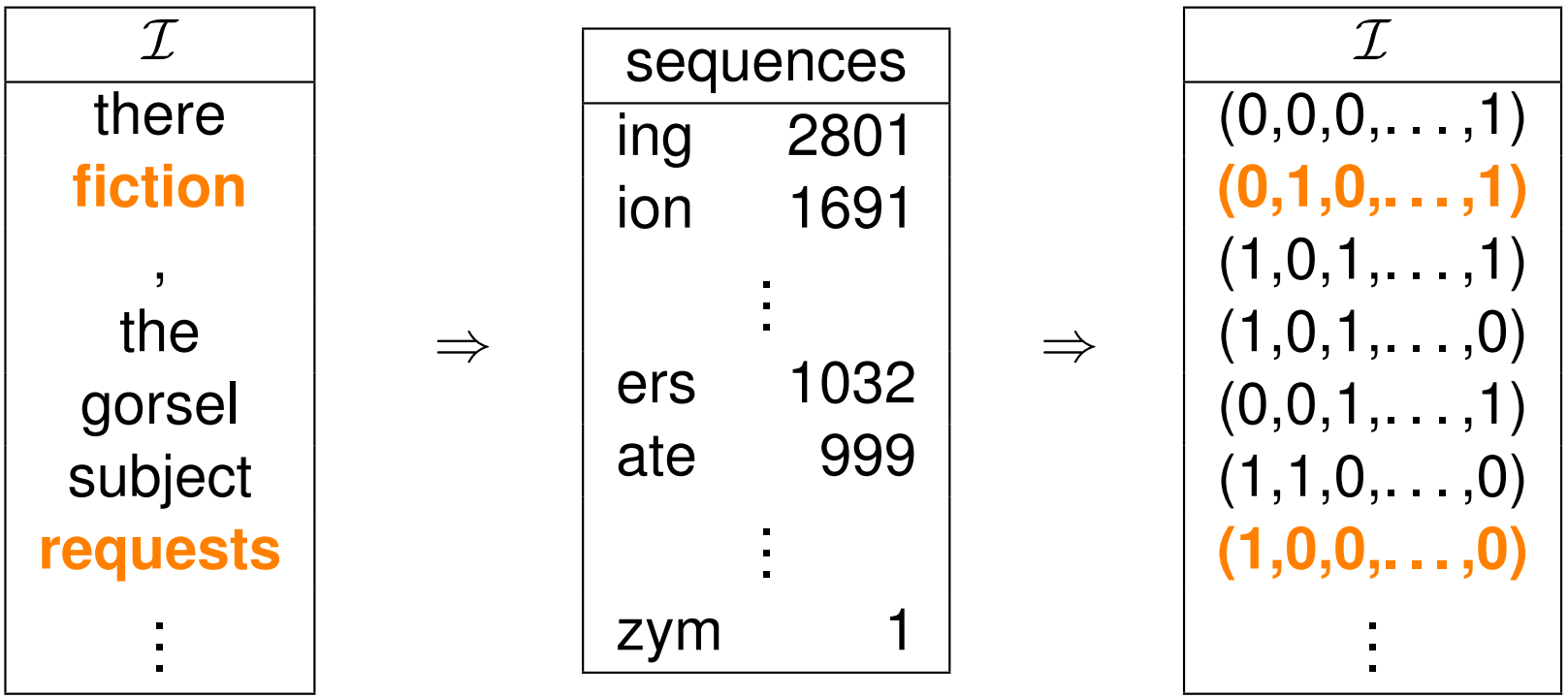
$t \equiv$ question $\Rightarrow$ request,requests, query, quest, fiction, . . .

Approximate Retrieval Time



$t \equiv (0,1,0,\ldots,0) \Rightarrow$ request,requests, query, quest, fiction, ...

Approximate Retrieval Time



$t \equiv (0,1,0,\ldots,0) \Rightarrow$ request,requests, query, quest, fiction, ...

Bibliography I

- Anouar Ben Hassena, *Apprentissage analogique par analogie de structures d'arbres*, Ph.D. dissertation, Univ. de Rennes I, France, 2011.
- Étienne Denoual, 'Analogical translation of unknown words in a statistical machine translation framework', in *MT Summit XI*, pp. 135–141, Copenhagen, Denmark, (2007).
- Julien Gosme and Yves Lepage, 'Structure des trigrammes inconnus et lissage par analogie', in *18e TALN*, Montpellier, France, (2011).

Bibliography II

- Philippe Langlais and Alexandre Patry, 'Translating Unknown Words by Analogical Learning', in *EMNLP*, pp. 877–886, Prague, Czech Republic, (2007).
- Philippe Langlais and François Yvon, 'Scaling up analogical learning', Technical report, Paritech, INFRES, IC2, Paris, France, (Oct. 2008).
- Philippe Langlais, François Yvon, and Pierre Zweigenbaum, 'Improvements in Analogical Learning: Application to Translating multi-Terms of the Medical Domain', in *12th EACL*, pp. 487–495, Athens, (2009).

Bibliography III

- Jean-François Lavallée and Philippe Langlais, 'Moranapho: un système multilingue d'analyse morphologique basé sur l'analogie formelle', *TAL*, 52(2), 17–44, (2011).
- Yves Lepage and Étienne Denoual, 'Purest ever example-based machine translation: Detailed presentation and assesment', *Mach. Translat*, 19, 25–252, (2005).
- Yves Lepage and Adrien Lardilleux, 'The greyc translation memory for the iwslt 2007 evaluation campaign', in *4th IWSLT*, pp. 49–54, Trento, Italy, (2008).

Bibliography IV

- Yves Lepage and Ando Shin-ichi, 'Saussurian analogy: A theoretical account and its application', in *7th COLING*, pp. 717–722, (1996).
- Haizhou Li, A. Kumaran, Vladimir Pervouchine, and Min Zhang, 'Report of news 2009 machine transliteration shared task', in *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration*, NEWS '09, pp. 1–18, (2009).
- Laurent Miclet, Sabri Bayroudh, and Arnaud Delhay, 'Analogical dissimilarity: Definitions, algorithms and two experiments in machine learning', *Journal of Artificial Intelligence Research*, 793–824, (2008).

Bibliography V

- Fabienne Moreau, Vincent Claveau, and Pascale Sébillot, 'Automatic morphological query expansion using analogy-based machine learning', in *29th European conference on IR research (ECIR'07)*, pp. 222–233, Berlin, Heidelberg, (2007).
- Harold Somers, Sandipan Sandapat, and Sudip Kumar Naskar, 'A review of ebmt using proportional analogies', in *3rd Workshop on Example-Based Machine Translation*, pp. 53–60, Dublin, Ireland, (2009).

Bibliography VI

- **Nicolas Stroppa, *Définitions et caractérisations de modèles à base d'analogies pour l'apprentissage automatique des langues naturelles*, Ph.D. dissertation, Telecom Paris, ENST, Paris, France, 2005.**
- **Nicolas Stroppa and François Yvon, 'Formal Models of Analogical Proportions'.**
  Available on HAL Portal, 2007.
- **Nicolas Stroppa and François Yvon, 'An analogical learner for morphological analysis', in *9th Conf. on Computational Natural Language Learning (CoNLL)*, pp. 120–127, Ann Arbor, USA, (2005).**

Bibliography VII

- P.D. Turney and M.L. Littman, 'Corpus-based learning of analogies and semantic relations', in *Machine Learning*, volume 60, pp. 251–278, (2005).
- François Yvon, 'Paradigmatic cascades: a linguistically sound model of pronunciation by analogy', in *In Proceedings of 35th ACL*, pp. 429–435, (1997).
- François Yvon, 'Finite-state machines solving analogies on words', Technical Report D008, École Nationale Supérieure des Télécommunications, (2003).

Bibliography VIII

François Yvon, Nicolas Stroppa, Arnaud Delhay, and Laurent Miclet, 'Solving analogies on words', Technical Report D005, École Nationale Supérieure des Télécommuncations, Paris, France, (2004).