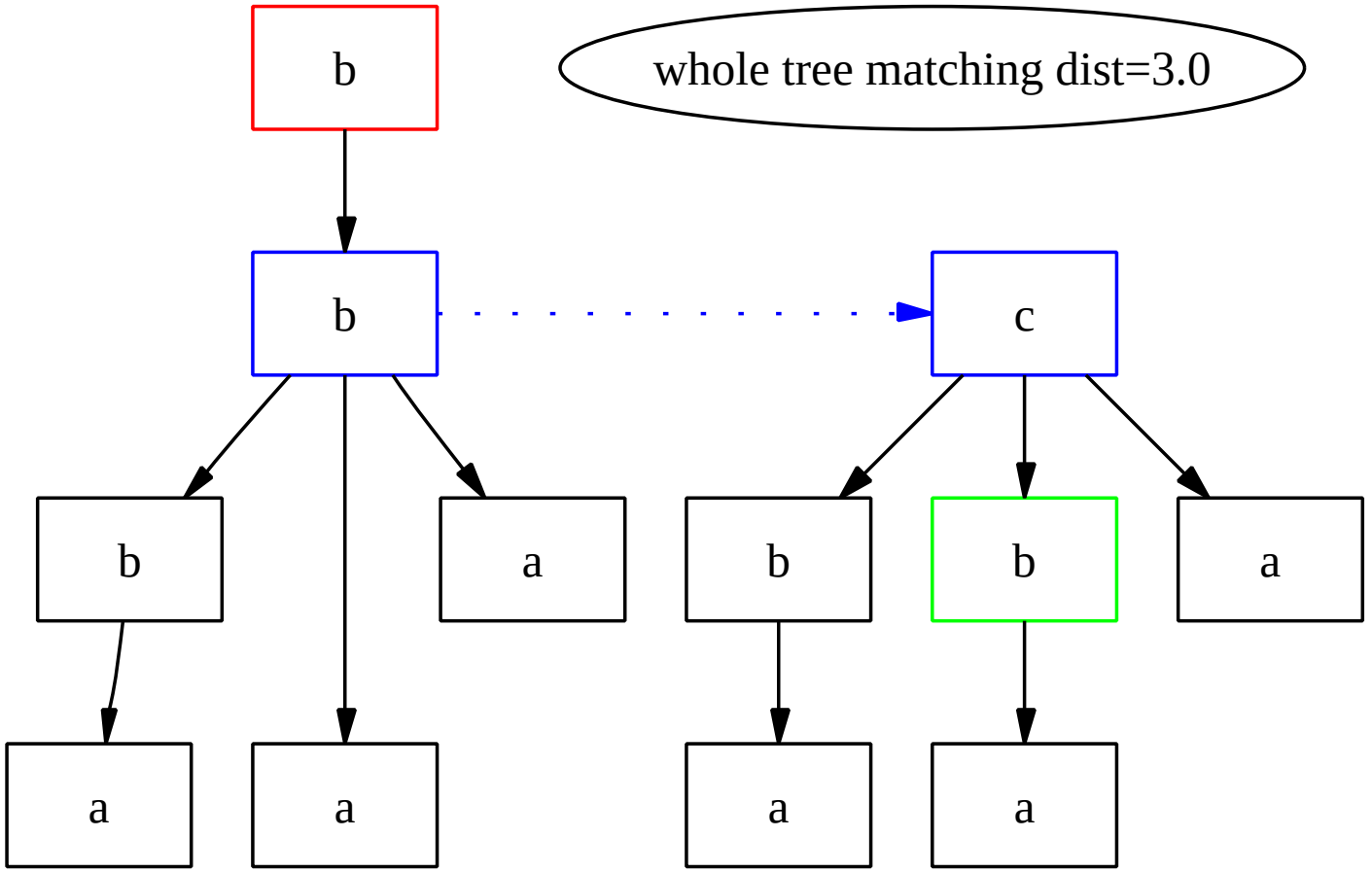
The potential of QATD: Question Answering by Tree Distance

Martin Emms
Dept. of Computer Science, Trinity College, Dublin, Ireland



Plan

- tree distance
- tree distance in answer retrieval
- retrieval performance as function of parser performance
- comparison with Collins parser
- comparison of tree-distance variants

Question Answering

Given questions such as eg.

Q1 *what does malloc return ?*

Q2 *What year did poet Emily Dickinson die?*

and a collection of sentences (eg. a computer manual, a corpus of newspaper articles), the task is to retrieve the sentences that answer the question, eg.

A1 *the malloc function returns a null pointer*

A2 *In 1886 , poet Emily Dickinson died in Amherst , Mass*

Question Answering by Tree Distance

- *strategy*: assume answers are *similar* to questions
- *how to measure similarity*: **tree-distance**
- measures how much *editing* of the answer's structure to derive question's structure, using

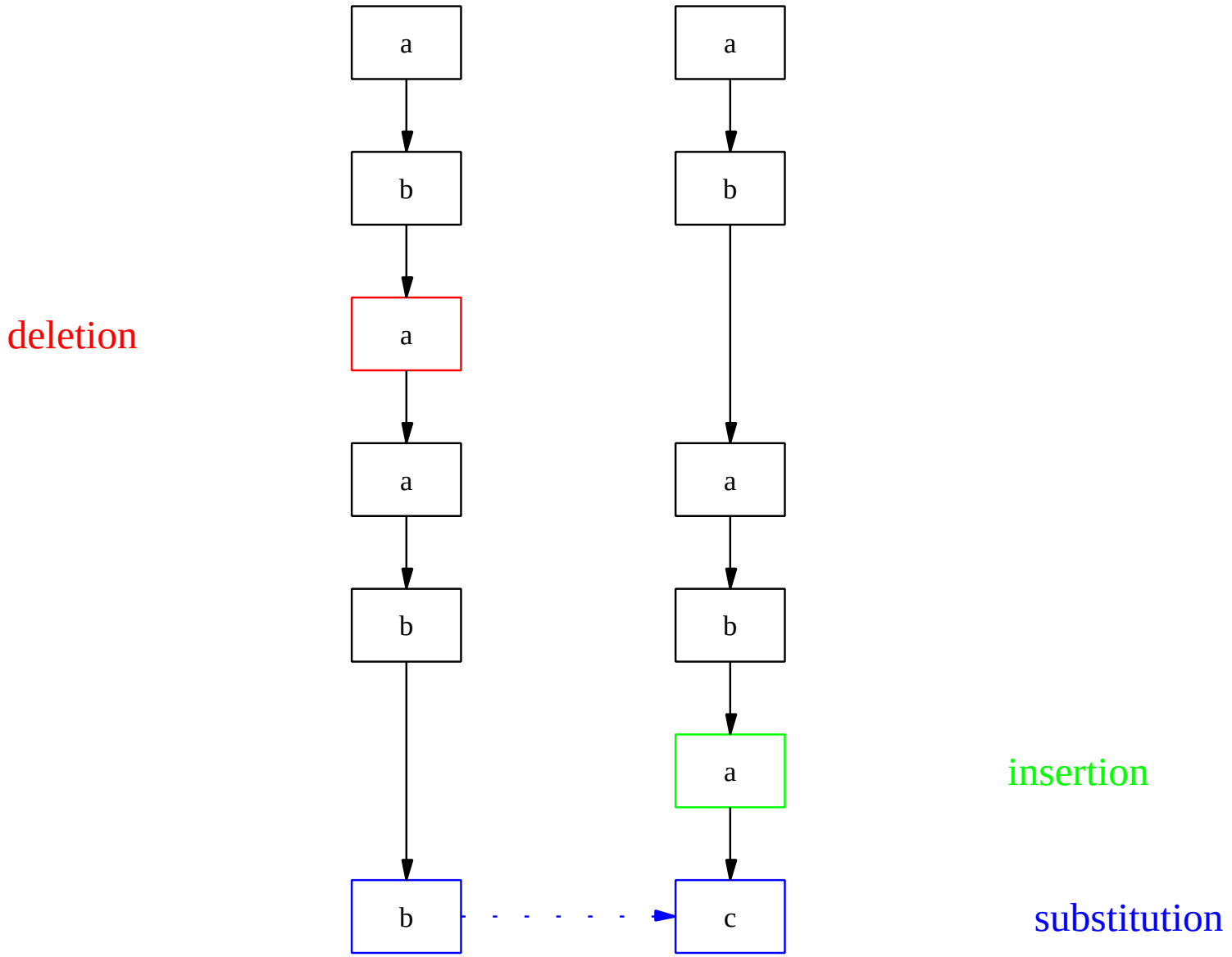
  deletion

  insertion

  substitution

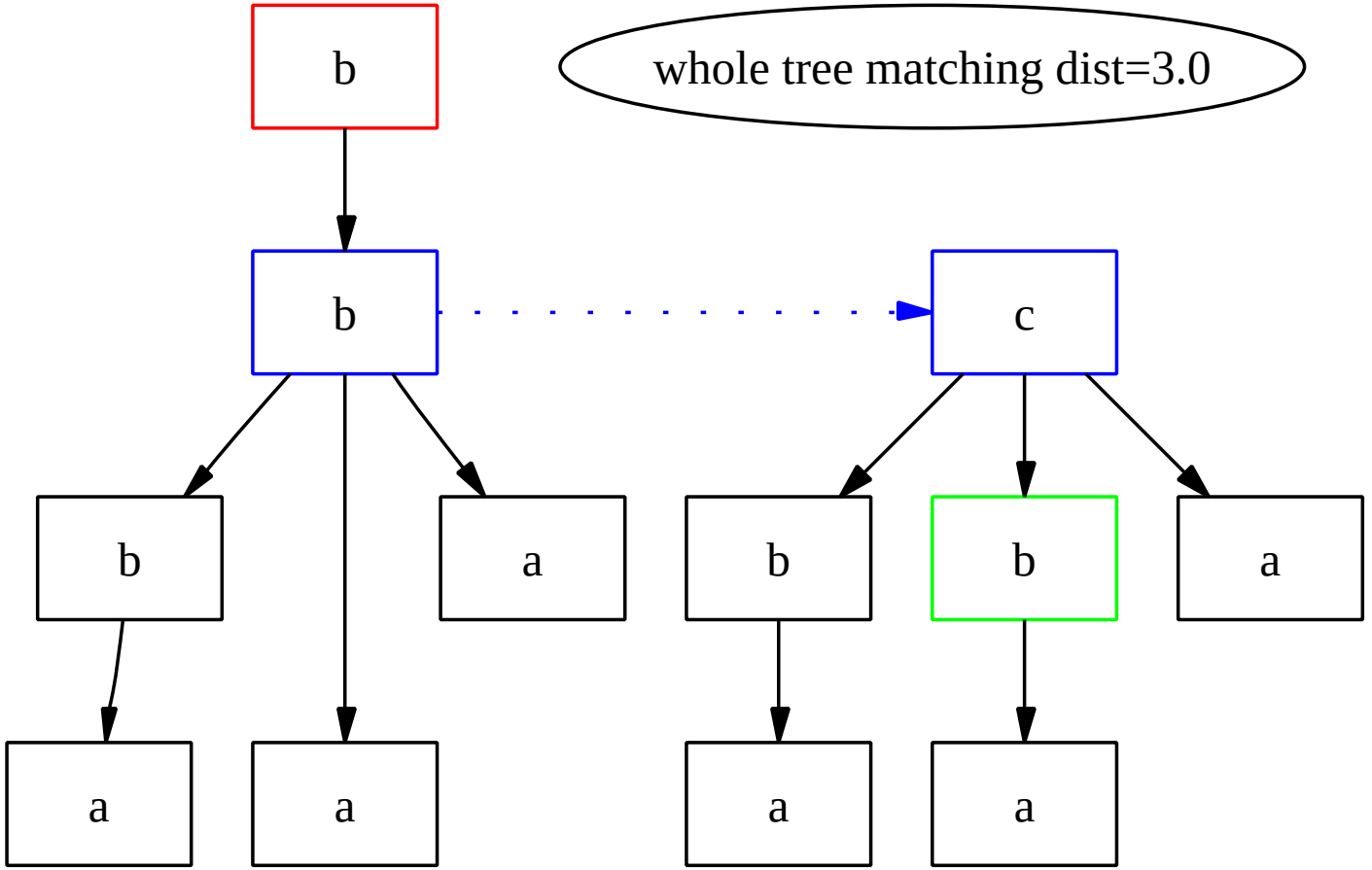
String Distance

- defined by 'best' partial map $\sigma : s \mapsto t$ (s and t are sequences)
- deletion: item in s not in domain of σ **cost 1**
- insertion: item in t not in range of σ **cost 1**
- substitution: non-identical items in s and t mapped **cost1**


deletion
a
b
a
a
b
b
a
b
a
b
c
insertion
substitution

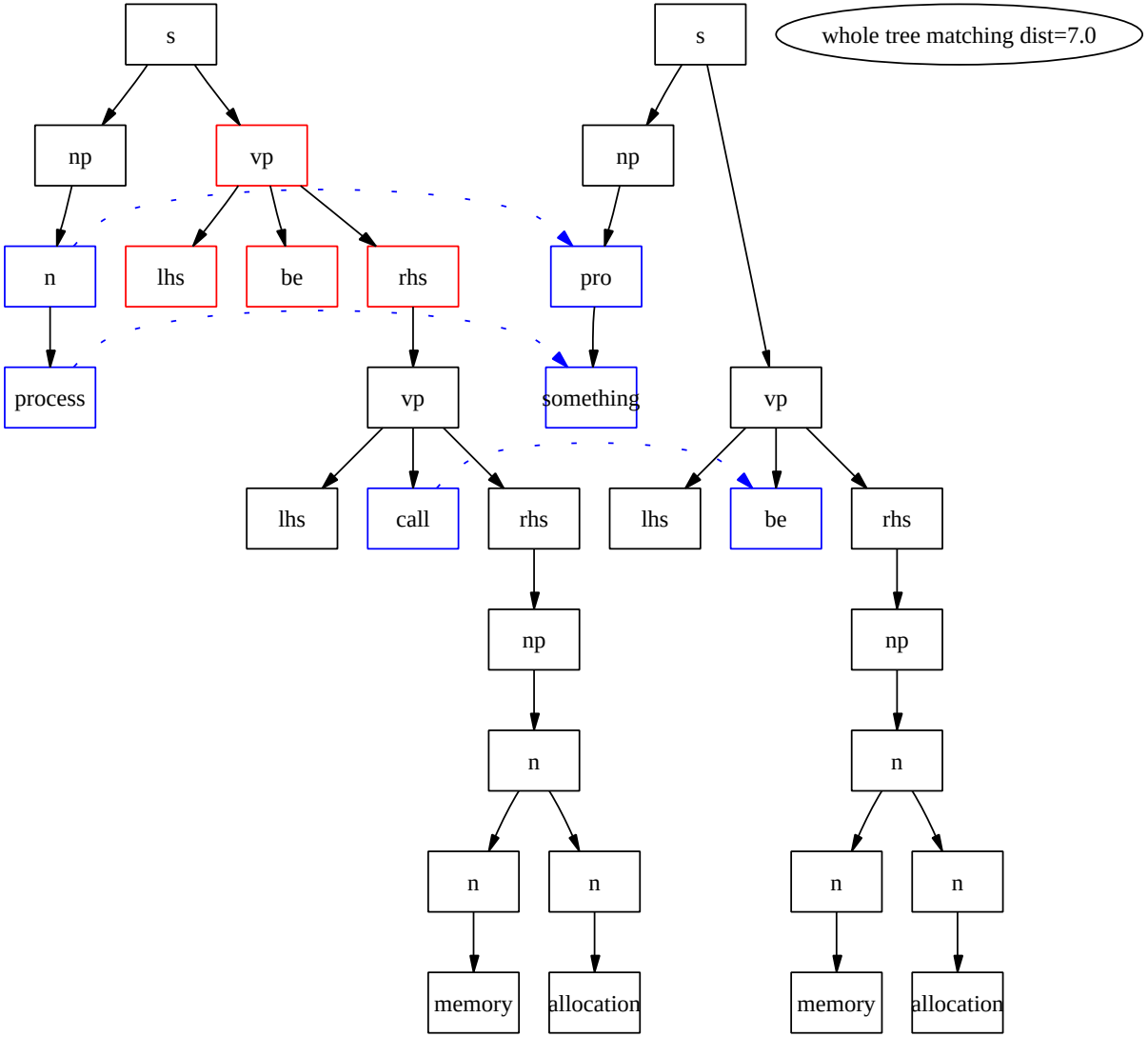
Tree Distance (basic)

- defined by 'best' partial map, $\sigma : S \mapsto T$ (S and T are trees)
- σ preserves left to right order
- σ preserves ancestry
- deletion: node in S not in domain of σ **cost 1**
- insertion: node in T not in range of σ **cost 1**
- substitution: non-identical nodes in S and T mapped **cost1**

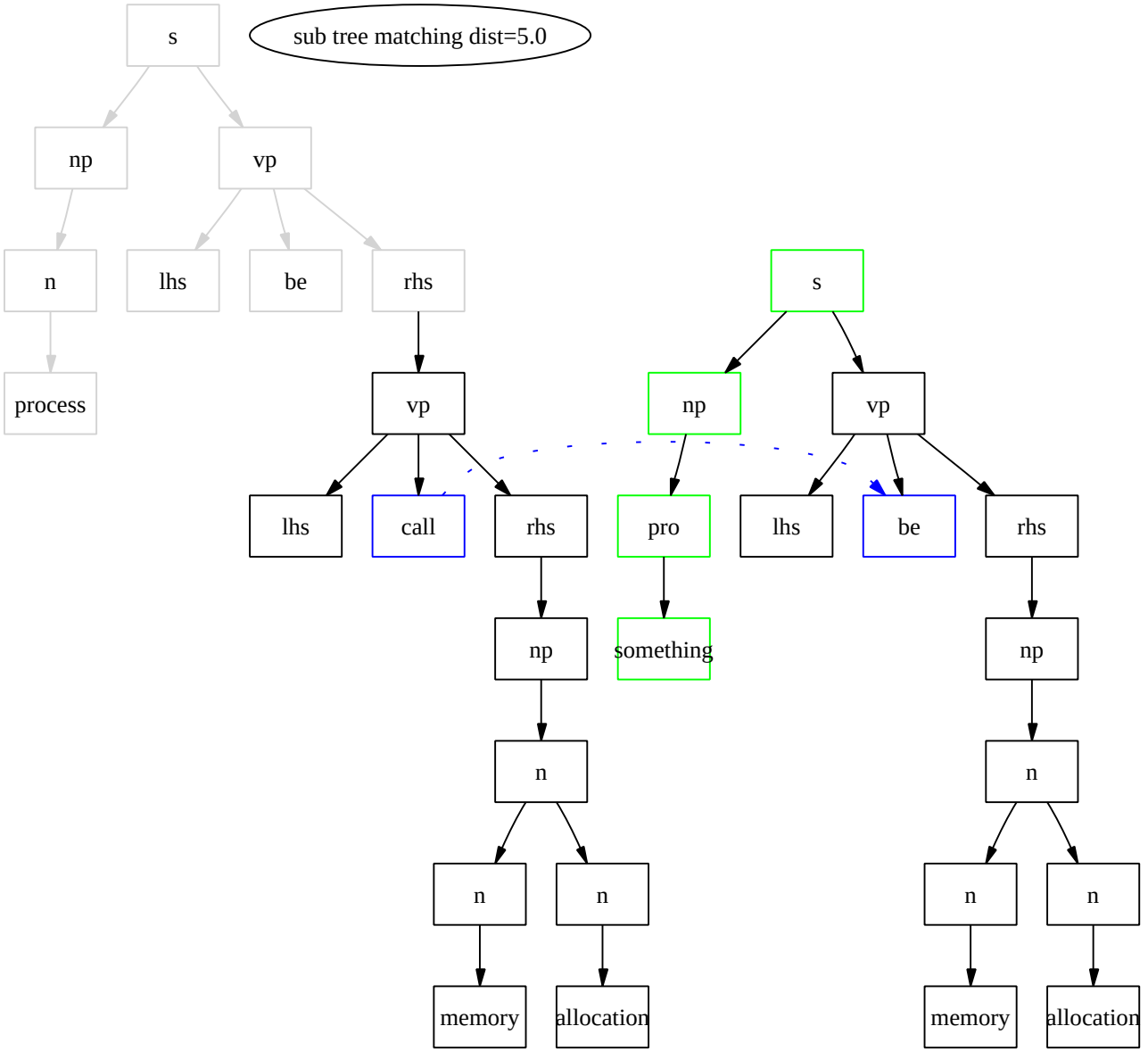

b
whole tree matching dist=3.0
b
c
b
a
b
b
a
a
a
a
a

- based on the Zhang/Shasha algorithm
- adapted code from Fontana et al, for comparing RNA structures
- algorithm works on a *post-order traversal* of a tree (see picture)
- node-types identified by positions in a symbol table (varying this varies what node-types are equated)

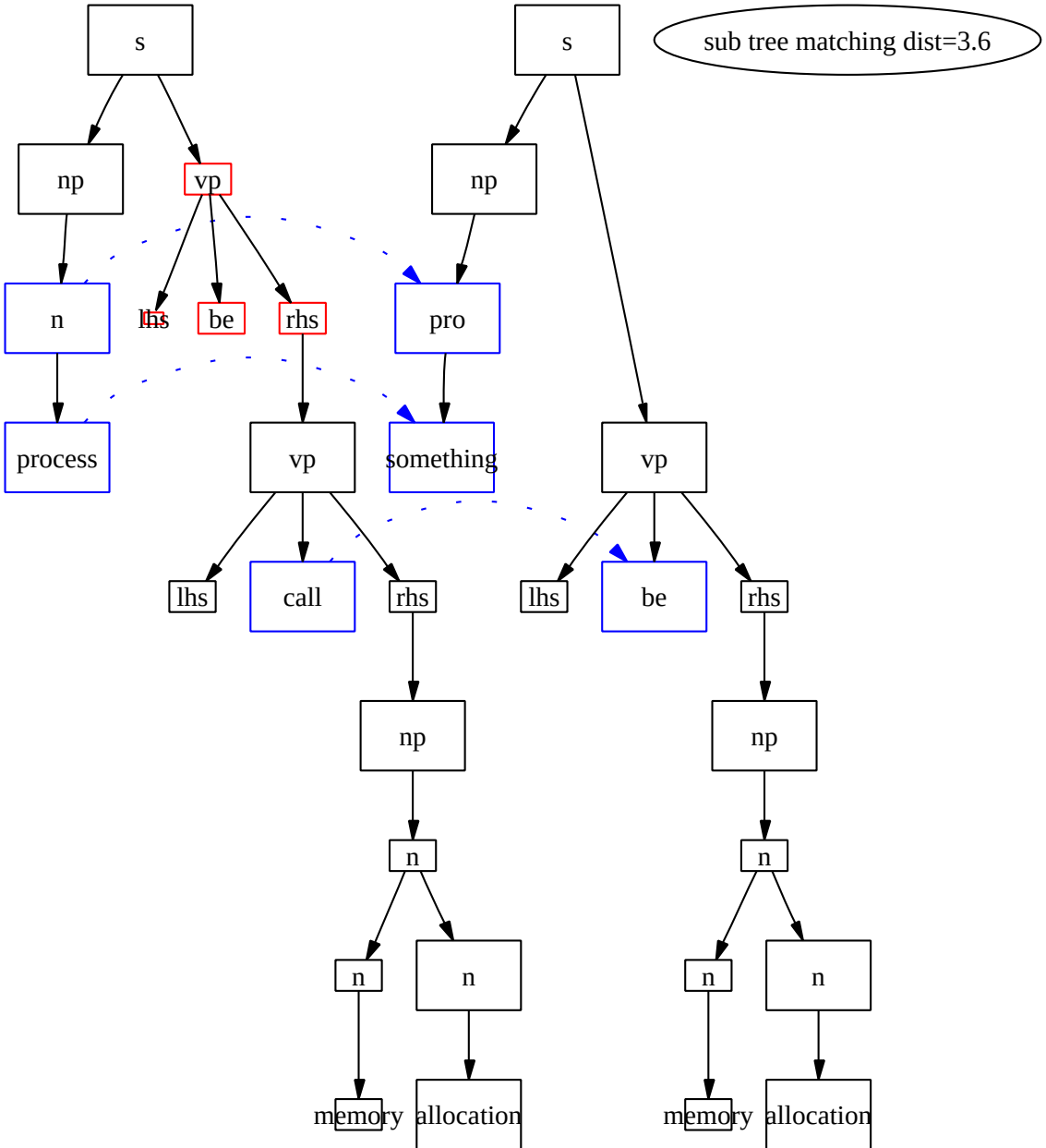
Whole-tree


s
np
vp
n
lhs
be
rhs
process
vp
lhs
call
rhs
np
n
n
n
memory
allocation
s
np
pro
something
vp
lhs
be
rhs
np
n
n
n
memory
allocation
whole tree matching dist=7.0

Sub-tree: in this variant, the *sub*-tree distance is the cost of the least cost mapping from a sub-tree of the source.



Structural weights: nodes have a weight between 0 and 1, assigned according to the syntactic structure.

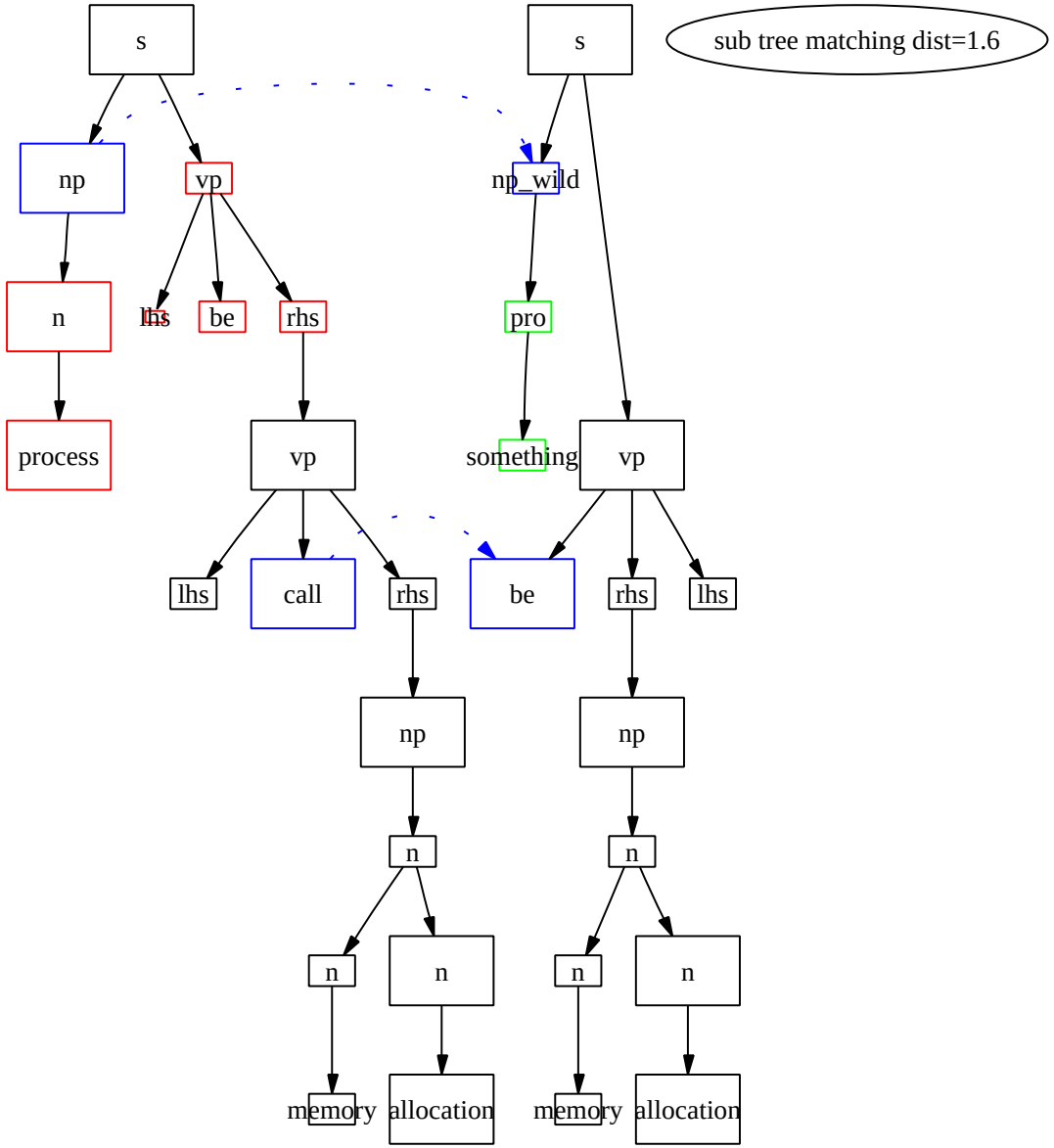


- nodes classified as as heads vs. complements vs. adjuncts vs. the rest
- adjuncts given 1/5th the weights of heads and complements,
- other daughters 1/2
- `assign_weights(rank,node)`:

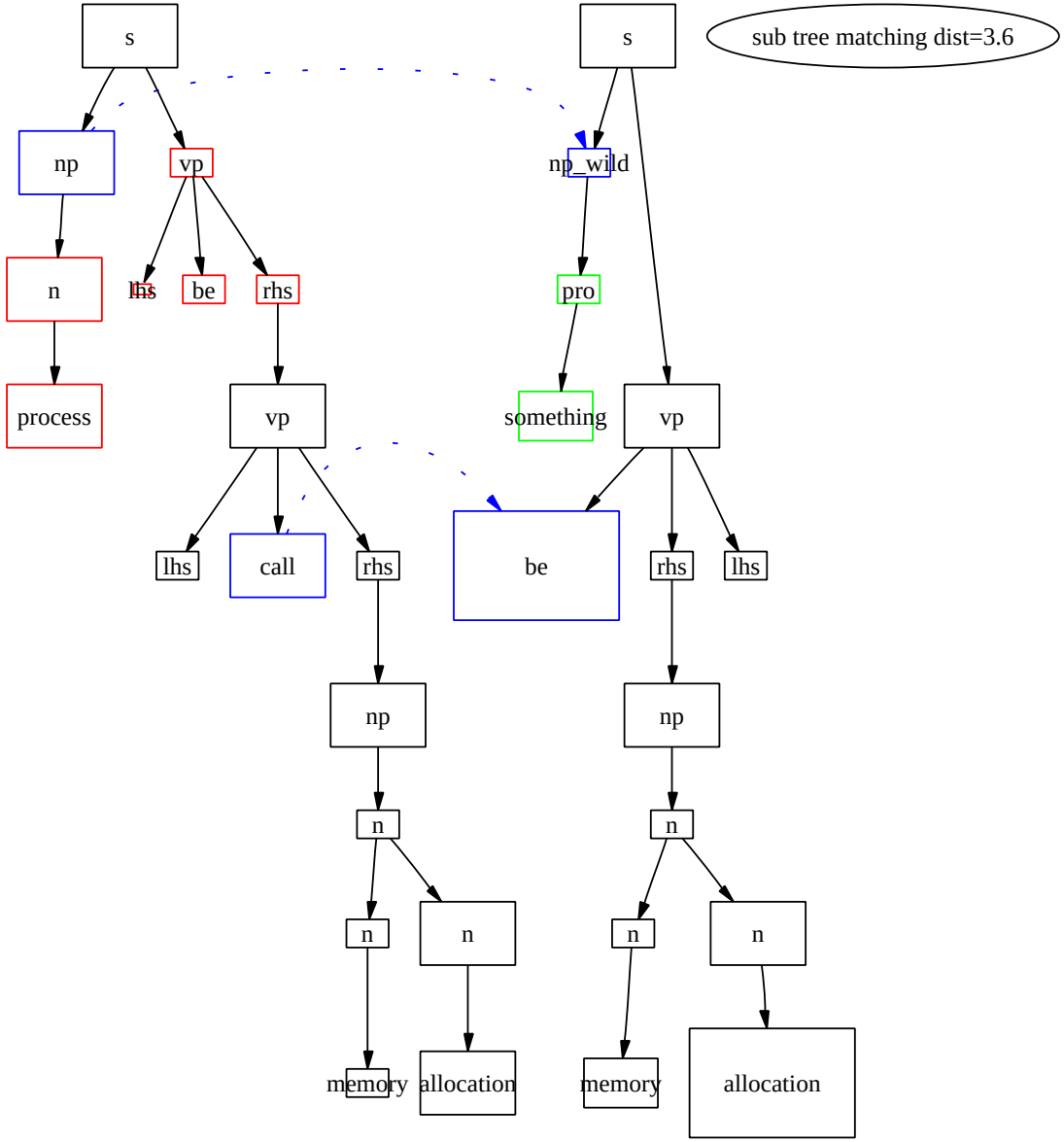
```
assign weight 1/rank to node

for each daughter d
  if (d is head or complement) {
     assign weight = 1/rank
     assign_weights(rank,d)
  }
  else if (d is adjunct) {
     assign weight = 1/(5 * rank)
     assign_weights(5 * rank,d)
  }
  else {
     assign weight = 1/(2 * rank)
     assign_weights(2 * rank)
```

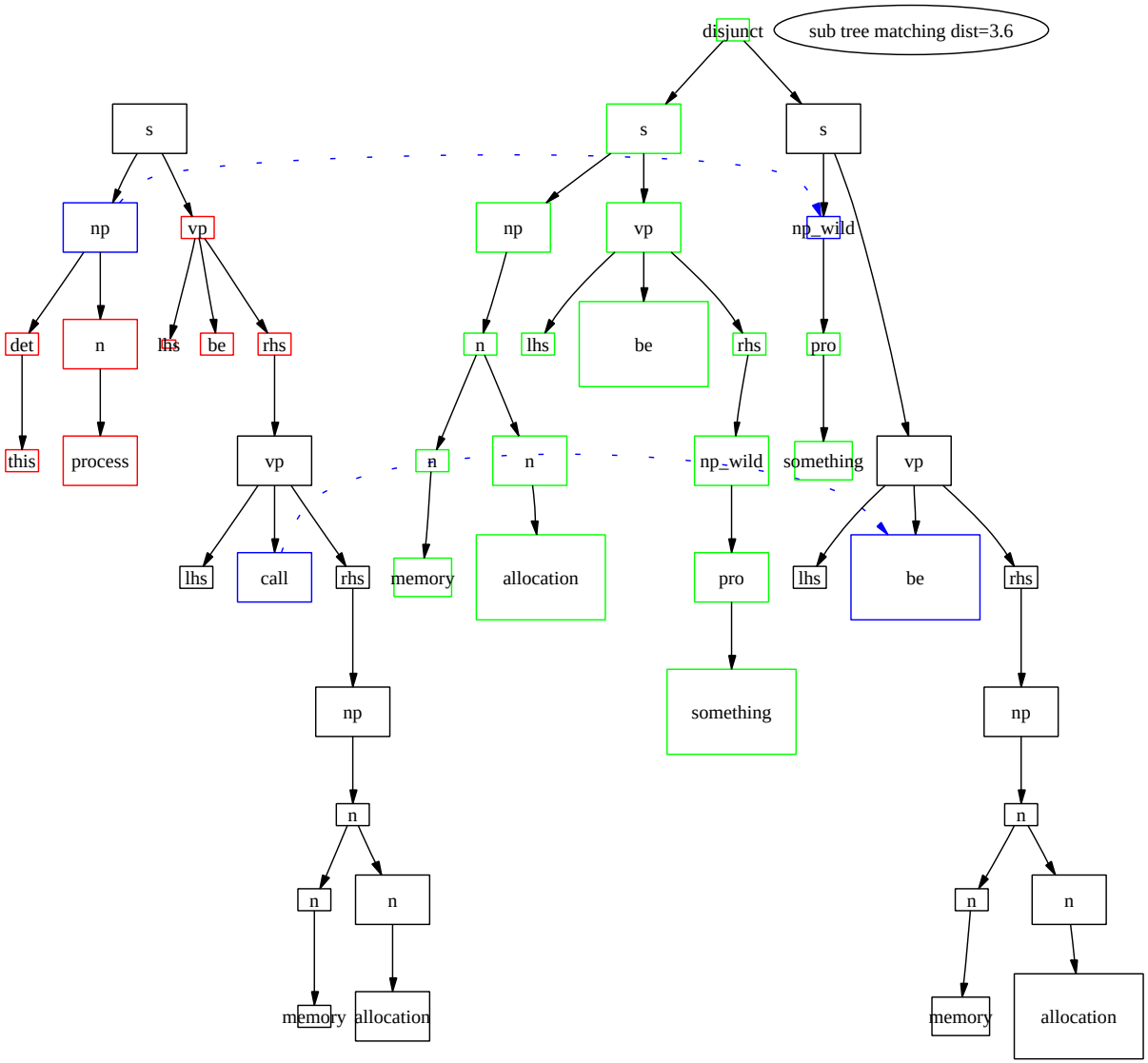
Target wild cards: marked target sub-trees can have zero cost matching with sub-trees in the source. eg. gap in wh-questions



Lexical Emphasis: the leaf nodes have weights which are scaled up in comparision to tree-internal nodes.



Target disjunctions: marked target sub-trees compete as alternatives. allows *what is x* to match both *x is y* and *y is x*.



Sub-traversal: the least cost mapping from a sub-traversal of the left-to-right post-order traversal of the source.

String Distance: if you code source and target word sequences as vertical trees, the string distance = the tree-distance, and the sub-string distance = sub-traversal distance.

Source 'self effacers': in this variant, marked source sub-trees (such as optional adjuncts) can be deleted in their entirety for no cost.

Target 'self inserters': in this variant, marked target sub-trees (such as optional adjuncts) can be inserted in their entirety for no cost.

Parse Quality vs Retrieval Performance

- Is syntactic structure

  optimist: more-or-less an approximation of semantic structure

  or

  pessimist: more-or-less an encryption of semantic structure

- Does improving parse performance lead to improved retrieval ?

Question Answering by Tree Distance (QATD) tasks

- a set of queries, $\mathcal{Q}$
- for each query q

  $\mathcal{COR}_q$: a corpus of potential answer sentences

  a_c: the correct answer in $\mathcal{COR}_q$
- for each $a \in \mathcal{COR}_q$, determine $td(a, q)$ the *tree-distance* between a and q, and use this to sort $\mathcal{COR}_q$ into $\mathcal{A}_q$.
- *correct-answer-rank* is the rank of the correct answer a_c in $\mathcal{A}_q$:

$$| \{a \in \mathcal{A}_q : td(a, q) \leq td(a_c, q)\} |$$

- *correct-answer-cutoff* is the proportion of $\mathcal{A}_q$ cut off by the correct answer a_c:

$$| \{a \in \mathcal{A}_q : td(a, q) \leq td(a_c, q)\} | \ / \ | \mathcal{A}_q |$$

- **lower values** for *correct-answer-cutoff* are **better**

GNU QATD Task

- $\mathcal{Q}$: 88 hand-created queries
- $\mathcal{COR}_q$: the sentences of the manual of the GNU C Library (shared by all the queries)

  Q *what is a page fault*

  A *When a program attempts to access a page which is not at that moment backed by real memory , this is known as a page fault*

  Q *must you free blocks at the end of program ?*

  A *There is no point in freeing blocks at the end of a program ...*

- $\mathcal{COR}_q$ was generated from XML sources of the manual, and after part-of-speech tagging contains 360326 tokens, split into 31625 sentences.

- performance on the GNU QATD task was determined for various versions of a particular parsing system
- Parser settings

full	full linguistic knowledge bases
thin50	randomly remove 50% of the linguistic knowledge base
manual	manually strip out parts
flat	attaching unanalysed words to a top-most node
gold	hand-correct each query q and *correct answer* a_c

GNU task, trinity parser

Table 1: Correct Answer Cutoff *in different parse settings, ranking by sub-tree distance (GNU task, trinity parser)*

Parsing	1st Qu.	Median	Mean	3rd Qu.
flat	0.1559	0.2459	0.2612	0.3920
manual	0.0349	0.2738	0.2454	0.3940
thin50	0.01936	0.1821	0.2115	0.4193
full	0.0157	0.1195	0.1882	0.2973
gold	0.00478	0.04	0.1450	0.1944

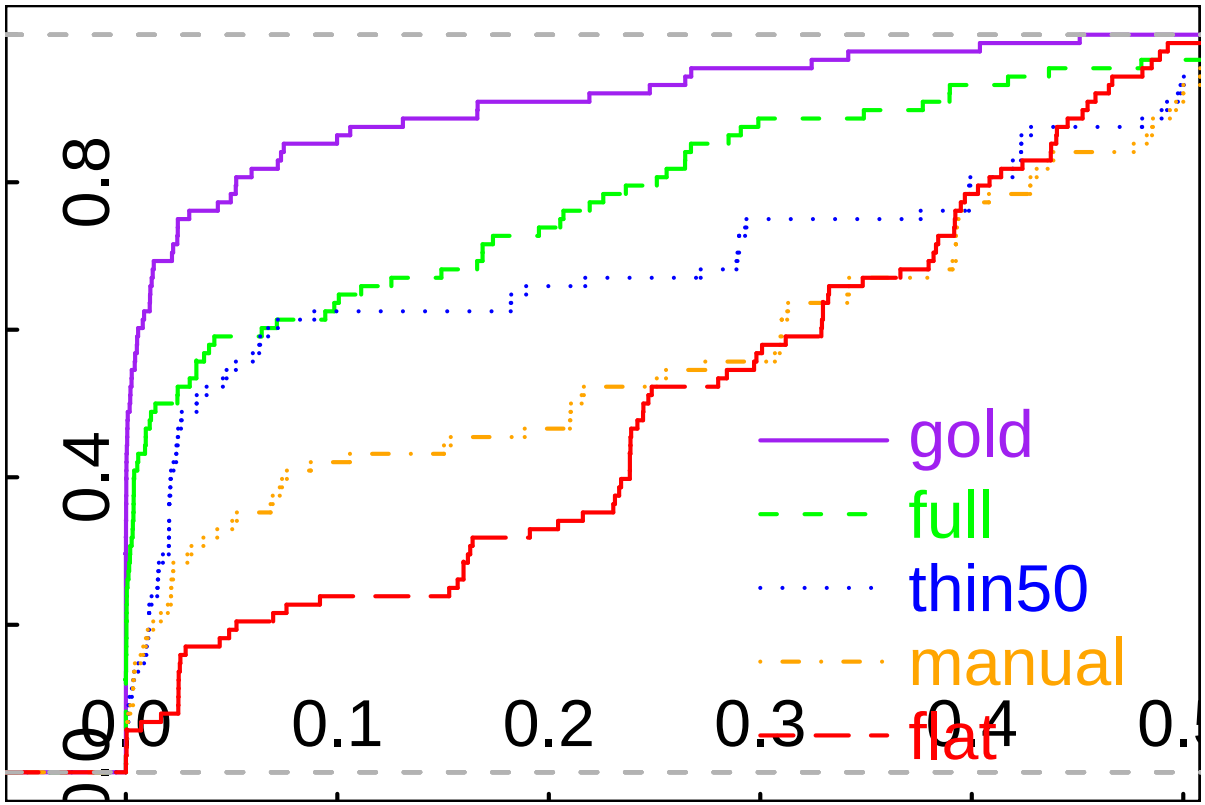
Recall for each query q, *correct-answer-cutoff* is the proportion of $\mathcal{A}_q$ cut off by the correct answer a_c:

$$\mid \{a \in \mathcal{A}_q : td(a,q) \leq td(a_c,q)\} \mid \ / \ \mid \mathcal{A}_q \mid$$

Table 2: Correct Cutoff *in different parse settings, ranking by weighted sub-tree distance (GNU task, trinity parser)*

Parsing	1st Qu.	Median	Mean	3rd Qu.
flat	0.1559	0.2459	0.2612	0.3920
manual	0.0215	0.2103	0.2203	0.3926
thin50	0.01418	0.02627	0.157	0.2930
full	0.00389	0.04216	0.1308	0.2198
gold	0.00067	0.0278	0.1087	0.1669

Empirical Cumulative Density Function of correct-answer-cutoff



x = correct-answer-cutoff

y = proportion of queries whose correct-answer-cutoff ≤ x

td = weighted sub-tree with wild cards

GNU task

trinity parser

Closer look at low correct-rank values

parse	**1**	10	50	100	1000
flat	1	2	5	5	15
manual	0	3	7	8	27
thin50	1	4	8	10	43
full	11	18	25	28	47
gold	**26**	37	43	48	67

n in column c: number of queries with correct-rank $\leq c$

eg. **26** queries had correct-rank = **1** with **gold** parses

GNU task

trinity parser

TREC 11 QATD task

- $\mathcal{Q}$: the 500 questions of the the TREC11 QA track [**?**]
- answers drawn from the AQUAINT corpus of newspaper articles, eg.

  Q *What year did poet Emily Dickinson die?*

  A *In 1886 , poet Emily Dickinson died in Amherst , Mass*

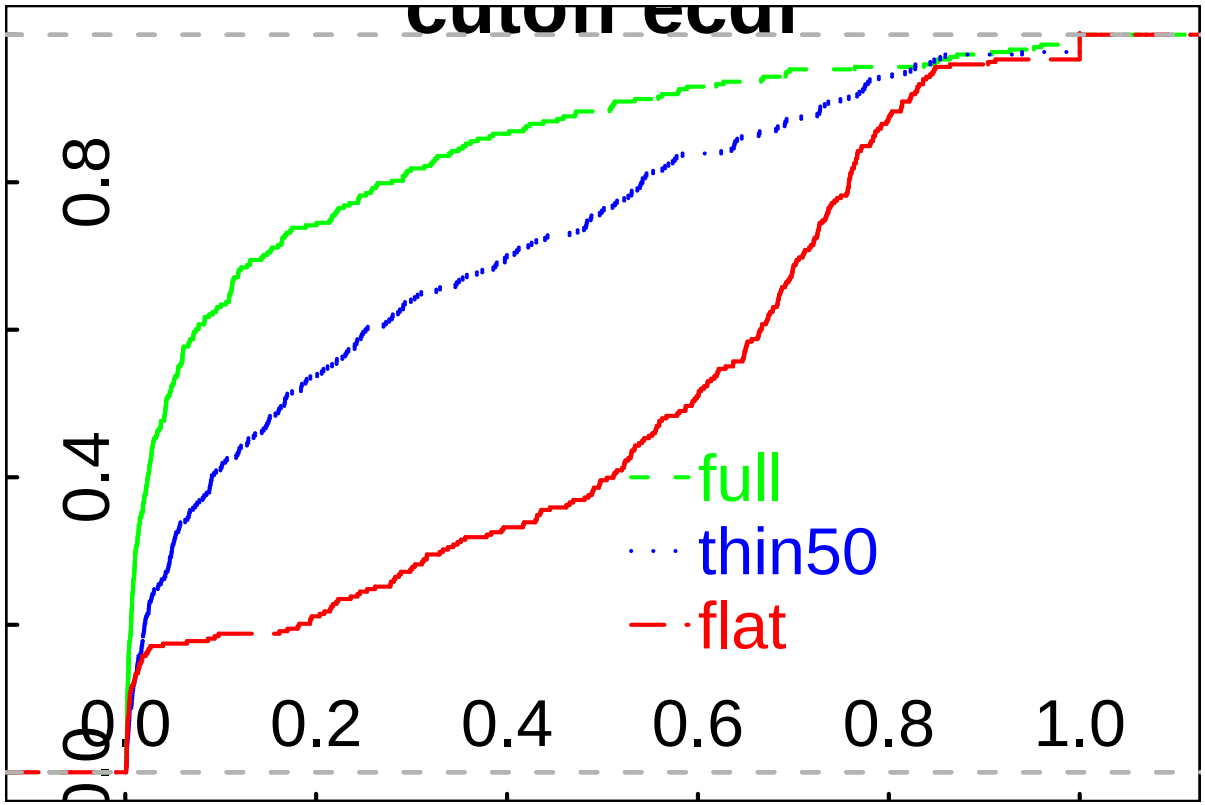
- $\mathcal{COR}_q$: the sentences of the top 50 from the top-1000 ranking of articles provided by TREC11 for each question ($\mid \mathcal{COR}_q \mid \approx 1000$)
- Answer correctness was determined using the TREC11 answer regular expressions

Table 3: Correct-Answer-Cutoff *in different parse settings, ranking by weighted sub-tree distance (TREC11 task, trinity parser)*

Parsing	1st Qu.	Median	Mean	3rd Qu.
flat	0.2657	0.5939	0.5007	0.7326000
thin50	0.03432	0.1669	0.2745	0.4867
full	0.00824	0.0428	0.1552	0.2156

Empirical Cumulative Density Function of

correct-answer-cutoff



x = correct-answer-cutoff

y = proportion of queries with correct-answer-cutoff $\leq x$

td = weighted sub-tree with wild cards

TREC11 task

trinity parser

closer look at low correct-rank values

parse	1	10	50	100	1000
flat	11	43	54	57	289
thin50	8	43	95	130	293
full	23	95	161	199	296

n in column c: number of queries with correct rank $\leq c$

TREC task

trinity parser

so ..

- for 2 QATD tasks, evidence that improving parse-quality improves retrieval performance
- evidence that syntactic structures can be used a substitute for semantic structures

The Collins parser [**?**] (*Model 3* variant)

- probabilistic parser
- using a model of trees as built top-down with a repertoire of moves
- learnt from the Penn Treebank

'argument' Node	`CAT(A,hd,NoOfDtrs,HdDtrIndex)`
'non argument' Node	`CAT(hd,NoOfDtrs,HdDtrIndex)`

- moves available to the Collins parser is defined by its grammar file *grammar.grm* containing 4 parts

  L: possibilities for modifier+head (eg DT-mod NN-head)
  R: possibilities for head+modifier (eg NP-head PP-mod)
  U: possibilities for unary rules (eg NP $\rightarrow$ NN)
  X: possiblites arguments+head (eg NP-arg VP-head)
  Y: possibilities head+arguments (eg VB-head VB-arg)

- 5916 possibilities altogether

- GNU and TREC QATD tasks were repeated with versions of Collins parser, where *grammar.grm* was reduced to different sized random subsets of itself.

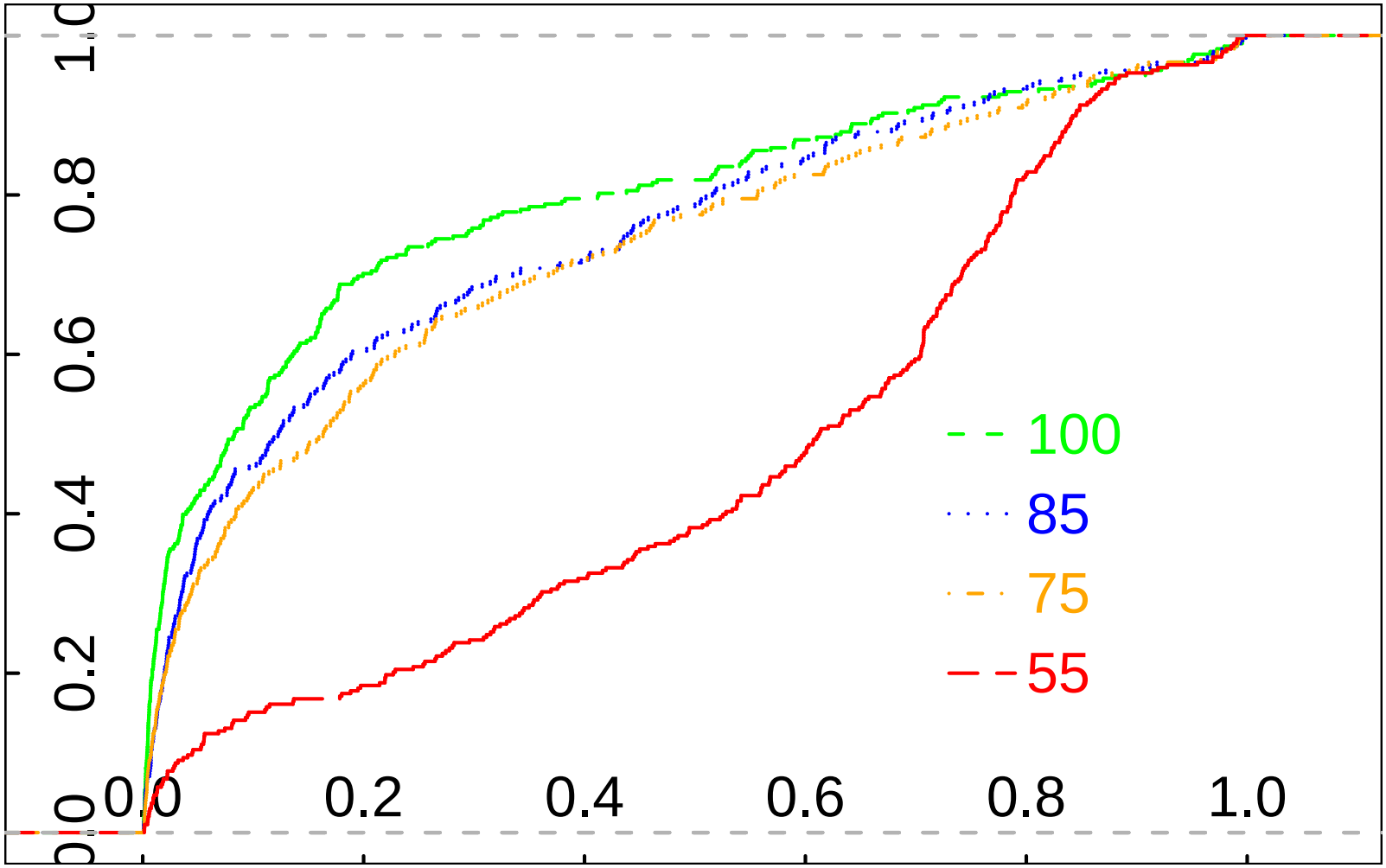
Table 4: *Reducing possible Collins parser moves: n in column c is number of queries whose correct rank is $\leq c$ (GNU task, collins)*

% moves	1	10	50	100	1000
65	6	9	17	31	68
85	11	21	45	51	81
100	11	22	49	58	82

Table 5: Correct-Answer-Cutoff *in different parse settings, ranking by unweighted sub-tree distance (TREC11 task, collins parser)*

Parsing	1st Qu.	Median	Mean	3rd Qu.
55	0.3157	0.6123	0.5345	0.766400
75	0.02946	0.1634	0.2701	0.4495
85	0.0266	0.1227	0.2501	0.4380
100	0.01256	0.08306	0.2097	0.2901

Empirical Cumulative Density Function of correct-answer-cutoff



x = correct-answer-cutoff

y = proportion of queries with correct-answer-cutoff $\leq x$

td = unweighted sub-tree

TREC11 task, collins parser

closer look at low correct-rank values

% moves	1	10	50	100	1000
55	6	41	48	49	49
75	6	27	84	119	276
100	10	57	125	153	292

n in column c: number of queries with correct-answer-rank $\leq c$

TREC11

collins parser

so ..

- for 2 parsers, 2 QATD tasks, seen that parse-quality relates to retrieval performance
- what about inverting that

  use QATD performance for *parser evaluation* ?

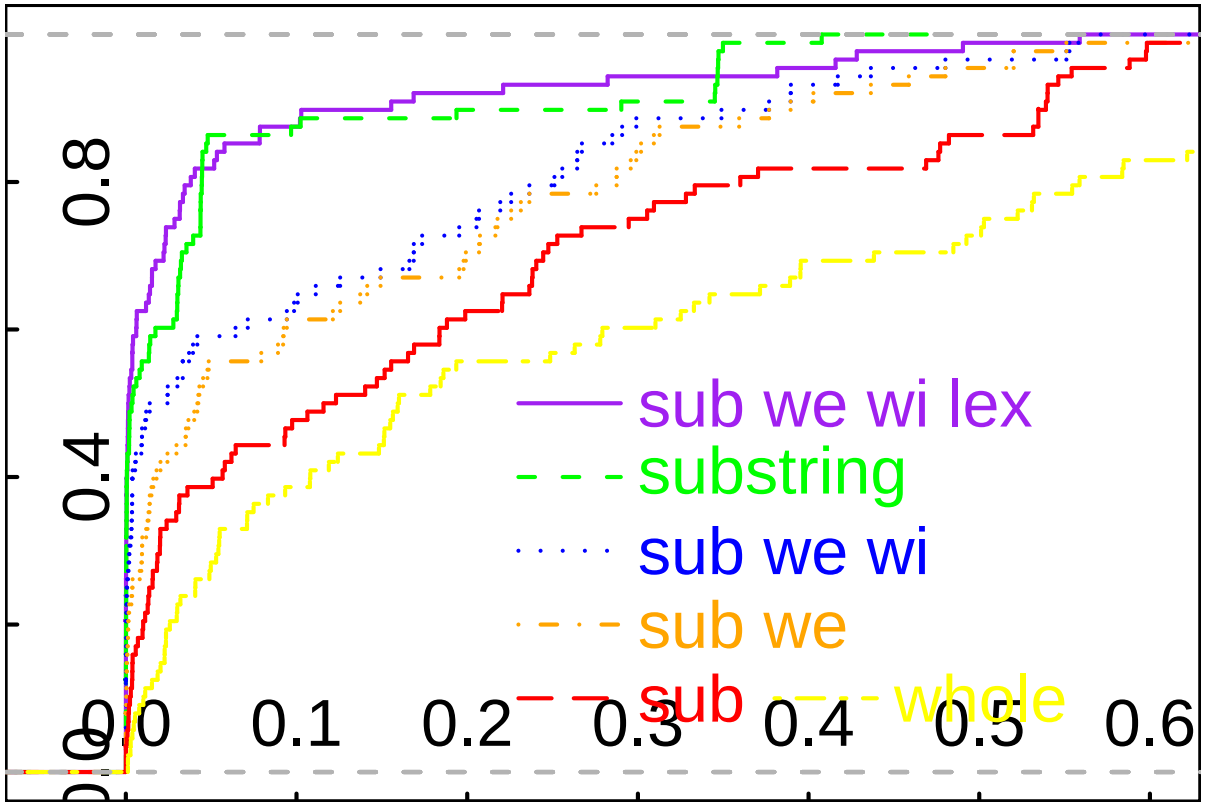
- why that might be nice
  - 'gold standard' queries/answers are plain text, not trees. So easier to make and more plentiful than treebanks
  - portable to any parser
  - no mapping between notations of a given parser $\mathcal{P}$ and those of a given tree-bank $\mathcal{G}$

Distance Measures Compared

Table 6: Correct Answer CutOff *for different distance measures -we = structural weights, -wi = wild cards, -lex = lexical emphasis, sub = sub-tree*

distance type	1st Qu.	Median	Mean
sub-we-wi-lex	9.414e-05	1.522e-03	4.662e-02
substring	2.197e-04	3.609e-03	5.137e-02
sub-we-wi	7.061e-04	1.919e-02	1.119e-01
sub-we	3.891e-03	4.216e-02	1.308e-01
sub	1.517e-02	1.195e-01	1.882e-01
whole	0.040710	0.159600	0.284600

Empirical Cumulative Density Function of correct-answer-cutoff



x = correct answer cutoff

y = proportion of queries with cutoff $\leq x$

GNU task

Trinity parser

closer look at low correct-rank values

distance	1	10	50	100	1000
sub we wild lex	18	29	44	47	66
substring	11	29	38	43	59

n in column c: number of queries with correct-rank $\leq c$

GNU task

Trinity parser

Conclusions

To summarise

- we have given evidence that tree-distance can be used as surrogate for semantic in a QA task
- we have given evidence that (a variant of) tree-distance can perform better than the string-distance measure.
- we have given evidence that improved parse quality leads to better QA performance, and suggested that this shows that performance on QATD tasks could be used as an evaluator for parsers.

In the future

- flatness of trees, grammar compaction
- closer look at dependency structures
- parameters in cost functions:
  - semantically enriched node descriptions
  - leaf-nodes: weights based on frequencies
  - internal nodes: weights based on tree-bank frequencies
- use QATD-performance as a driver in machine-learning of probabilities for a parser
- use in Recognising Text Entailment
- use in Document Summarisation
- using Tree-Distance clustering to train NE recognisers
- using Tree-Distance clustering to look at Levins claims about semantic classes and verb-alternations