Tree Distance in Answer Retrieval and Parser Evaluation

Martin Emms

Department of Computer Science, Trinity College, Dublin, Ireland
Martin.Emms@tcd.ieSpringer.de
http://www.cs.tcd.ie/Martin.Emms

Abstract. The use of syntactic tree-distance as a surrogate for semantic distance in an answer retrieval task is investigated. The feasibility of this is confirmed by showing that retrieval performance increases with parse quality, and an application of this to parser evaluation is discussed. Variant definitions of tree-distance involving parameters such as whole vs sub-tree, node weighting, wild-card trees and lexical emphasis are compared with each other and with sub-string distance.

1 Introduction

The concern of this paper is the use of *tree-distance* in an answer retrieval (AR) task. Given a question (such as Q below) and a collection of sentences (eg. a manual), the task is to retrieve the sentences that best answer the question, (such as A below):

Q: *what does malloc return ?*
A: *the malloc function returns a null pointer*

The paper presents results on this AR task in which questions and answers are compared on their syntactic structures, using *tree-distance* to quantify the amount of editing needed to derive the question structure from the answer structure. The less editing an answer needs, the higher it is rated as an answer to the question.

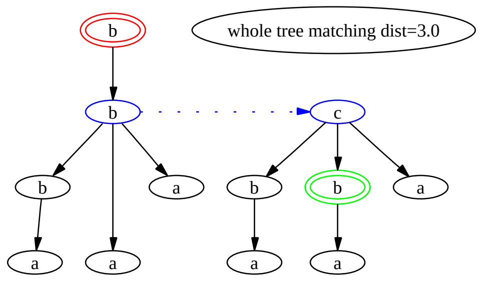
Two results are presented. The work uses tree-distance as a stand-in for semantic distance. As evidence that this can be done, the first result is that improving the syntax structures does improve performance on the AR task. This also indicates that using tree-distance in AR has a potential for parser evaluation. The second result compares some variants of tree-distance with each other, and also with standard *string-distance*. We show that a particular variant of tree-distance out-performs string-distance.

2 Tree Distance

The *tree distance* between two trees can be defined by considering all the possible 1-to-1 partial maps, σ, between source and targets trees S and T, which preserve left to right order and ancestry[1]. Nodes of S which are not in the domain (resp. range)

[1] if S_{i_1} and S_{i_2} are mapped to T_{j_1} and T_{j_2}, then (i) S_{i_1} precedes S_{i_2} iff T_{j_1} precedes T_{j_2} and (ii) S_{i_1} is descended from S_{i_2} iff T_{j_1} is descended from T_{j_2}

of σ are considered *deleted* (resp. *inserted*) with an associated cost. Otherwise, where $T_j = \sigma(S_i) \neq S_i$, there is a *substitution* cost. The least cost mapping between the trees defines the tree distance. In the example below, a distance of 3 assuming unit costs is obtained. Deleted (resp. inserted) nodes are shown with a double outline in the lefthand (resp. righthand) tree, substituted nodes are linked with an arrow, and unaltered nodes are displayed at the same height.



In the work reported in this paper the definition of tree distance is varied along a number of dimensions.

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: in this version, nodes have a weight between 0 and 1, and weights are assigned according to the syntactic structure. The procedure for assigning structural depends on classifying the daughters as head vs. complement vs. adjunct vs. other, with essentially adjuncts given 1/5th the weights of heads and complements, and other daughters 1/2.

Wild cards: If a node is labeled as a *wild card*, it (and the sub tree it dominates) can have zero cost matching with sub-trees in the source. In the tree structures assigned to wh-questions, such wild card trees can be put in the position of the gap.

Lexical emphasis: the leaf-nodes of the tree are words. In a lexically emphasized version of tree distance, the leaf nodes have weights which are scaled up relative to tree-internal nodes.

An alignment between two trees is shown in Figure 1, using the weighted, wild-card variant. The nodes associated with the auxiliary verb get a low weight and are cheaply deleted. The subject position of the target is taken by a `np_wild` tree, and matches for 0 cost with the subject np in the source tree.

The basis of the algorithm used is the *ZhangShasha algorithm* [3] to compute the cost of the least cost mapping. The implementation is an adaptation of [2], allowing for wild-cards, sub-tree matching and human-readable display of the chosen alignments (such as seen in figure 1).

3 The Answer Retrieval Task

Queries were formulated whose answers are single sentences in the manual of the GNU C Library [2]. For example

[2] `www.gnu.org`

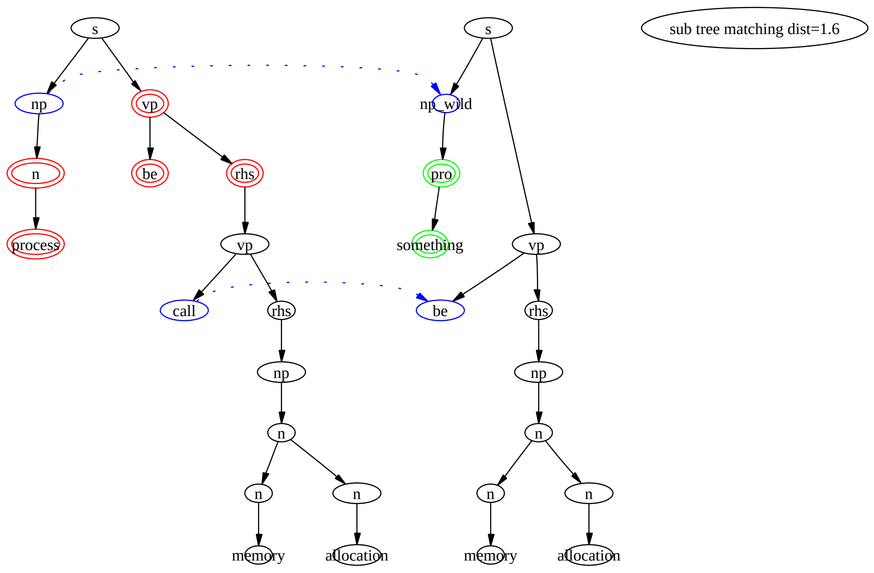


Fig. 1. An alignment using weights and wild cards

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 *what does the free function do*
A - *Function : void free (void * ptr) The free function deallocates the block of memory pointed at by ptr*

In the retrieval task, each sentence in the manual is assigned a distance rating, measuring the distance from it, to the question. For a given query, q, its *correct cutoff* – $cc(q)$ – is the proportion of the answers whose distance is less than or equal to the distance for the correct answer[3]. The lower this number is for a given query, the better the system performs on that query.

There are 88 queries. The text from which the answers come was turned into a part-of-speech tagged version which contains 360326 tokens, split into 31625 units.

4 Parse Quality vs Retrieval Performance

The above-defined approach to AR could be described as using syntactic distance to serve as a stand-in for semantic distance. Can this work ? Grouping items which are semantically related is one of the aims of syntactic structure, but there are rival aims, the fulfilling of which may pull syntactic structures away from semantics. On a pessimistic

[3] equivalently this is the rank of the correct answer divided by the total number of answers

view, these non-semantic aims have the upper hand to such an extent that semantic structure is mostly very difficult to discern from syntactic structure; syntactic structure is then more of an encryption of semantics than a useful gateway into it. This section addresses this question, and provides some evidence that one need not be overly pessimistic.

What we look at is the relationship between parse-quality and AR performance. First we establish the performance of a particular parsing system, making uses of particular linguistic knowledge bases (see *full* in the tables below). For this discussion the details of this parsing system are not important, save to say that it is a chartparser whose final step is to select a sequence of edges spanning the input. Though the parsing system is imperfect, it could be worse. Randomly removing 50% of the linguistic knowledge base should make for worse structures (see *thin50*), as should manually stripping out parts of it (see *manual*), while worst of all should be the entirely flat parses which result if the knowledge bases are empty. In each case, the AR performance was determined. To try to get a picture of AR performance if the parse quality improved, we hand-corrected the parses of the queries and their correct answers – see *gold* in the tables below.

The tables below give the results using the sub-tree distance and *weighted* sub-tree distance. The numbers describe the distribution of the *correct cutoff* over the queries, so lower numbers are better. Figure 2 shows the empirical cumulative density function (ecdf) for the correct cutoff obtained with weighted sub-tree with wild cards measure: so for each cut-off d, it plots the percentage of queries q with $cc(q) \leq d$. Clearly as we move through the different parse settings, the performance improves.

	sub tree			
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

	weighted sub tree			
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

The results indicate that the more successful the parser is in recovering correct syntax structure, the more successfully tree-distance can be used in the AR task. By reversing this implication, this is also suggestive of a way in which the AR task could be used as an evaluation metric for a parser. Such an application has some attractions, the main one being that the raw materials for the AR task are questions and answers, as plain text, not syntactic structures. This avoids the difficulties that arise when trying to evaluate by comparing a parser's native output with non-native gold-standard treebank parses. Just so long as the parser can assign tree-structures to queries and answers, the tree-distance measure can compare them, regardless of that parser's native notational or theoretical commitments.

5 Distance Measures Compared

In section 2 some parameters of variation in the definition of tree-distance were introduced: sub-tree vs whole tree, weights, wild cards, and lexical emphasis. The perfor-

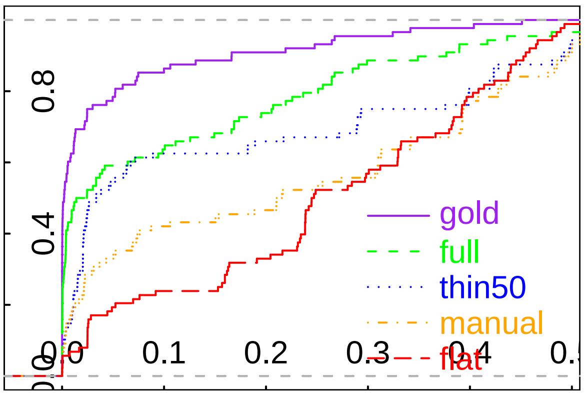


Fig. 2. Success vs Cut-off: x = correct cutoff, y = proportion of queries whose correct cutoff $\leq x$

mance of these distance measures was compared, along with the performance of the *sub-string* distance[4]. The table below summarises the distribution of the correct cut-off over the queries for the different settings, whilst in Figure 3 the ecdf plots are given (we = structural weights, wi = wild cards, lex = lexical emphasis, sub = sub-tree).

sub	weights	wild	lex	1st Qu.	Median	Mean	3rd Qu.
+	+	+	+	0.00009	0.00152	0.04662	0.02929
	sub string			0.00022	0.00361	0.05137	0.04375
+	+	+	-	0.00071	0.01919	0.1119	0.2058
+	+	-	-	0.00389	0.04216	0.1308	0.2198
+	-	-	-	0.01517	0.1195	0.1882	0.2973
-	-	-	-	0.04071	0.1596	0.2846	0.5077

What the data show is that the version of tree distance which uses sub-trees, weights, wild-cards and lexical emphasis, performs better than the sub-string distance, and that each of the parameters make a contribution to improved performance.

Lexical emphasis makes a large contribution to the performance. Without it, what can happen is that a structurally similar but lexically completely dissimilar false answer can be rated about the same as a lexically more similar but structurally less similar correct answer.

6 Conclusion and Future Work

By showing that improved parse quality leads to better AR driven by tree-distance, we have shown that syntax structures can be used as a stand-in for semantic structures – the syntax structures do not encode matters entirely unrelated to semantics. Also a particular variant of tree-distance was shown to perform better than string-distance. We

[4] this is the least number of insertions, deletions and substitutions needed to turn a sub-string of the answer word sequence into the query word sequence [5]

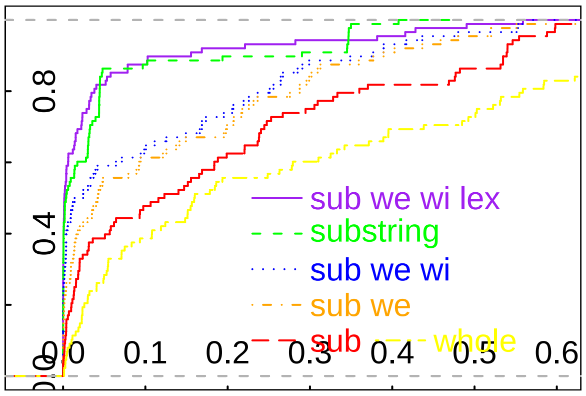


Fig. 3. Success vs Cut-off: x = correct cutoff, y = proportion of queries whose correct cutoff $\leq x$

also suggested that performance on AR driven by tree-distance could be used as an evaluator for parsers.

This paper does not argue that tree-distance be used as the sole ingredient in practical AR. In all probability, practical AR will use syntactic structure as an input to further processing. For example [4] convert parser output to relational triples, and model the question/answer relation as entailment. Perhaps the places where AR by tree-distance works worst can function as hot-spots for deeper processing by such systems.

We have argued that AR using tree-distance allows for a portable assessment of how successful, and how *semantic* a parser is. A direction for future work is to prove portability by using further parsers, such as the statistical parser of [1]. A question deserving attention is whether the methodology can be adapted to parsers which generate dependency structures rather than trees.

References

1. Michael Collins. *Head-driven statistical models for natural language parsing*. PhD thesis, 1999.
2. Walter Fontana, Ivo L Hofacker, and Peter F Stadler. Vienna rna package. `www.tbi.univie.ac.at/ĩvo/RNA`.
3. 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.
4. Diego Molla and Ben Hutchison. Intrinsic vs extrinsic evaluations of parsing systems. In *Proceedings European Association for Computational Linguistics (EACL), workshop on Evaluation Initiatives in Natural Language Processing*, pages 43–50, 2003.
5. V.I.Levenshtein. Binary codes capable of correcting insertions and reversals. *Sov. Phys. Dokl*, 10:707–710, 1966.