Natural Language Inference via Dependency Tree Mapping: An Application to Question Answering

Vasin Punyakanok
Dan Roth
Wen-tau Yih
Department of Computer Science
University of Illinois at Urbana-Champaign

We describe an approach for answer selection in a free form question answering task. In order to go beyond a key-word based matching in selecting answers to questions, one would like to develop a principled way for the answer selection process that incorporates both syntactic and semantic information. We achieve this goal by (1) representing both questions and candidate passages using dependency trees, augmented with semantic information such as named entities, and (2) computing a generalized edit distance between a candidate passage representation and the question representation, a distance which aims to capture some level of meaning similarity.

The sentence that best answers a question is determined to be the one that minimizes the generalized edit distance we define, computed via a dynamic programming based approximate tree matching algorithm. We evaluate the approach on question-answer pairs taken from previous TREC Q/A competitions. Preliminary experiments show its potential by significantly outperforming common bag-of-word scoring methods.

1. Introduction

Open-domain natural language question answering (Q/A) is a challenging task in natural language processing which has received significant attention in the last few years (Voorhees, 2000; Voorhees, 2001; Voorhees, 2002). In the Text REtrieval Conference (TREC) question answering competition, for example, given a free form query like "What was the largest crowd to ever come see Michael Jordan?" (Voorhees, 2002), the system can access a large collection of newspaper articles in order to find the exact answer, e.g. "62,046", along with a short sentence that supports its being the answer.

The overall task is very difficult even for fairly simple questions of the type exemplified above. A complete Q/A requires the ability to (1) analyze questions (question analysis) in order to determine what is the question about (Li and Roth, 2002), (2) retrieve potential candidate answers from the given collection of articles, and (3) determine the final candidate that answers the question. This work is concerned with the last stage only. That is, we assume that a set of candidate answers is already given, and we aim at choosing the correct candidate.

We view the problem as that of evaluating the *distance* between a question and each of their answer candidates. The candidate that has the lowest distance to the question is selected as the final answer. The simple bag-of-word technique does not perform well in this case as shown in the following example taken from Harabagiu and Moldovan (2001).

```
What is the fastest car in the world?
```

The candidate answers are:

1. `The Jaguar XJ220 is the dearest (415000 pounds), fastest (217mph) and most sought after car in the world.`
2. `...will stretch Volkswagen's lead in the world's fastest growing vehicle market.`

Without deep analysis of the sentences, one would not know that the "fastest" in the second candidate does not modify `car` as it does in the first, thus the bag-of-word approach would fail. Therefore, rather than performing inference over the raw representation of the sentence, we first represent the question and the candidate answer using a dependency tree, possibly augmented with more semantic information. Then we define a distance measure between these representations, taking into account their structure and some semantic properties we infer. Figure 1 shows the dependency trees of the question and the candidate answers in the previous example. This information allows us to better match the question and its correct answer.

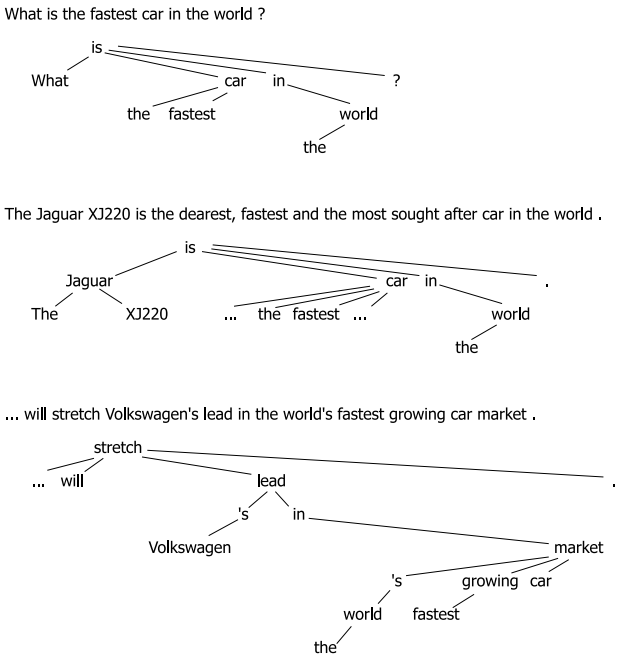

Figure 1
An example of dependency trees for a question and candidate answers. For comprehensibility reasons we omit parts of the tree that are irrelevant.

Tree matching has recently received attention in the natural langauge processing community in the context of machine translation (Eisner, 2003; Gildea, 2003; Ding et al., 2003) but, so far, not in the context of the Q/A task. Developing an approach to answer selection via the notion of extended tree matching is the first contribution of this work. The second contribution is an algorithmic approach that is different from those used in machine translation. Our approach builds on an edit distance and an approximate tree matching algorithm (Zhang and Shasha, 1989) to measure the distance between trees.

We test our approach on the questions given in the TREC-2002 Q/A track. The comparison between the performance of our approach and a simple bag-of-word approach clearly illustrates the advantage of using dependency trees in this task.

The next section describes our approach for using tree matching over the dependency trees. Then, we explain the edit distance measure and the tree matching method we use. After that we present our experimental results. Our conclusions and future directions are discussed in the final section.

2. Dependency Tree Mapping in Question Answering

We are concerned with finding the (best) sentence that contains the answer to any given question, from a collection of candidate sentences. In doing so, we need a mechanism that can measure how close a candidate answer is to the question, with respect to this criterion. This will allow us the choose the final answer to be the one that matches the question best.

To achieve this, we look at the problem in two levels. First, we need a representation of the sentences that captures useful information in order to accommodate the matching process. Second, we need an efficient matching process that can utilize the chosen representation. At the first level, the representation should capture both the syntactic and semantic information in the sentence. To capture the syntactic information, we represent questions and answers with their dependency trees (Mel'čuk, 1987), which allows us to see clearly the syntactic relations between words in the sentences. Using trees also allows us to flexibly incorporate other information including semantic knowledge. By allowing each node in the tree to contain more than just the surface form of its corresponding word, we can add semantic information to a node, e.g. what type of named entities a word belongs to, synonyms of the words, etc. Moreover, each node may be generalized to represent a larger unit than a word such as a phrase or a named entity.

With an appropriate representation, the only work left is to find the matching between the representations of the question and the answer. In doing so, we use the approximate tree matching approach which we explain in the next section. Formally speaking, we assume, for each question q_i, a given collection of candidate answers, $A_i = \{a_1, a_2, \ldots, a_{n_i}\}$. We output as the final answer for the q_i,

$$a_i = \arg\min_{a \in A_i} DR(q_i, a),$$

where DR returns the minimum approximate tree matching.

3. Edit Distance and Approximate Tree Matching

We first introduce the tree edit distance (Tai, 1979) which is the distance measure used as the criterion for matching between the tree representations. We then explain how this measure is used in the approximate tree matching problem, following an algorithm developed in (Zhang and Shasha, 1989) to determine how similar a given pair of trees are.

Following Tai (1979) and Zhang and Shasha (1989), we consider ordered labeled trees in which each node is labeled by some information and the order from left to right of its children is important. There are three operations that can transform an ordered labeled tree to another. The operations include *deleting* a node, *inserting* a node, and *changing* a node. Figure 2 illustrates the effect of these operations on a tree. Specifically, when a node n is deleted, its children will be attached to the parent of n. Insertion is the inverse of deletion, and changing a node alters its label.

Each operation is associated with a cost. The cost of a sequence of operations is defined to be the sum of the costs of the operations in the sequence. We are interested in finding the minimum cost sequence of operations from among those that can be used

to map one tree into another.

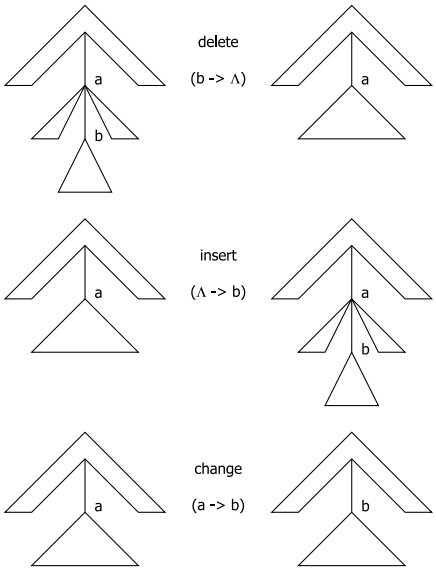

Figure 2
The effect of the operations *delete*, *insert*, and *change*

Formally, we represent an operation as a pair (a, b) where a represents the node to be edited and b is its result. We use (a, Λ) and (Λ, b) to represent the *delete* and *insert* operation respectively.

An operation $(a, b) \neq (\Lambda, \Lambda)$ is associated with a nonnegative cost $\gamma(a \rightarrow b)$. The cost of a sequence of operations $S = \langle s_1, s_2, \ldots, s_k \rangle$ is $\gamma(S) = \sum_{i=1}^{k} \gamma(s_i)$.

Given a tree T, we denote by $s(T)$ the tree resulting from applying operation s on T, and $S(T) = s_k(s_{k-1}(\ldots(s_1(T))\ldots))$. Given two trees T_1 and T_2, we would like to find the *edit distance* which is the cost of the minimal cost edit operations,

$$\delta(T_1, T_2) = \min_S \{\gamma(S) | S(T_1) = T_2\}.$$

A *mapping* corresponds to a restricted sequence of operations, which we define as follows. A mapping M from T_1 to T_2 is a set of integer pairs satisfying the following properties. Let $T[i]$ denote ith node of the tree T in a given order, and N_1 and N_2 the numbers of nodes in T_1 and T_2 respectively.

1. For any pair $(i, j) \in M$, $1 \leqslant i \leqslant N_1$ and $1 \leqslant j \leqslant N_2$.
2. For any pairs (i_1, j_1) and $(i_2, j_2) \in M$,
   (a) $i_1 = i_2$ if and only if $j_1 = j_2$,
   (b) $T_1[i_1]$ is to the left of $T_1[i_2]$ if and only if $T_2[j_1]$ is to the left of $T_2[j_2]$,
   (c) $T_1[i_1]$ is an ancestor of $T_1[i_2]$ if and only if $T_2[j_1]$ is an ancestor of $T_2[j_2]$.

The cost of a mapping M is

$$\gamma(M) = \sum_{(i,j) \in M} \gamma(T_1[i] \rightarrow T_2[j]) + \sum_{i \in I} \gamma(T_1[i] \rightarrow \Lambda) + \sum_{j \in J} \gamma(\Lambda \rightarrow T_2[j]),$$

where I is the set of indices of nodes in T_1 that are not mapped by M, and J is the corresponding set in T_2.

A mapping may be thought as a restricted edit operation sequence where only at most one operation is allowed to each node. Interestingly, when the cost function satisfies the triangular inequality, that is, $\forall a, b, c : \gamma(a \to c) \leqslant \gamma(a \to b) + \gamma(b \to c)$, then the minimum cost $\delta(T_1, T_2)$ is a minimum cost of a mapping (Tai, 1979).

In general, we can use the notion of edit distance defined above to determine how similar two given trees are. However, in the question answering domain, when matching a question and a candidate answer, an exact answer to a question may be only a clause or a phrase in a sentence, rather than the whole sentence. Therefore, matching the question with the whole candidate sentence may result in a poor match even though the sentence actually contains the correct answer. Our goal is therefore to match a question only with parts of the sentence. Specifically, there should be no additional cost if some subtrees of the answer are deleted. We achieve this by employing an approximate tree matching approach (Zhang and Shasha, 1989).

A forest S of a tree T is a set of disjoint subtrees in T, and $T \backslash S$ is the new tree resulting from cutting all subtrees in S from T. Let $\mathcal{S}(T)$ represent the set of all possible forests of T. Let T_1 and T_2 be two trees we would like to match. The approximate tree matching problem between T_1 and T_2 is to find:

$$DR(T_1, T_2) = \min_{S \in \mathcal{S}(T_2)} \delta(T_1, T_2 \backslash S)$$

We use the *SUBTREE REMOVAL* algorithm developed in Zhang and Shasha (1989)—an efficient dynamic programming based algorithm—with a slight modification to compute the approximate tree matching. We note that in our experiments we allow the cost functions to violate the triangularity property. Although, the algorithm as presented in Zhang and Shasha (1989) does not support this directly, the problem can be easily got around by modifying Lemma 4 in Zhang and Shasha (1989) to reconsider this exception, and deriving the new algorithm accordingly. The complexity of the algorithm is $O(|T_1| \times |T_2| \times \min(depth(T_1), leaves(T_1)) \times \min(depth(T_2), leaves(T_2)))$ where $depth$ returns the maximum depth of the tree and $leaves$ returns the number of leaves in the tree[1]. For the details of the modified Lemma 4, see Appendix.

4. An Experiment

We describe an experiment with 500 questions given in the TREC-2002 Q/A competition (Voorhees, 2002). 454 of the questions had answers in the text collection. The correct answers for each question, if any, were given along with the answers, returned by all participants after the completion of the competition. We built the pool of candidate sentences for each question by including the sentence containing correct answers as well as all answers returned by the TREC participants to the question. Clearly, this made the problem harder for our answer selector. Typically, an answer selection process is evaluated using a candidate collection built from the correct answers and the output from an information retrieval engine. However, with a candidate collection that contains incorrect answers chosen by other systems, the answer selection needs to be more accurate.

Since the structure of a sentence might be quite different from that of a question, we reformulated each question to a statement form using simple heuristics. Specifically, the question word (e.g. *what*, *when*, or *where*) was replaced with a special token `*ANS*` (which is supposed to stand for the answer phrase that will be extracted). For example:

[1] For a comprehensive reading in tree matching and its algorithms, see Shasha and Zhang (1997)

```
Where is Devil's Tower?
Devil's Tower is in *ANS*
```

Each sentence was preprocessed first by a SNoW-based part-of-speech tagger (Roth and Zelenko, 1998; Even-Zohar and Roth, 2001). Then, Collins' parser (Collins, 1997) was run to produce the parse trees. Since this parser also outputs the head word of each constituent, we could directly convert the parse trees to their corresponding dependency tree by simply taking the head word as the parent. Moreover, we extracted named-entity information with the named-entity recognizer used in Roth et al. (2001). In addition, for each question, we also ran a question classifier (Li and Roth, 2002) which predicted the type of the answer expected by the question.

After an answer was selected, the document id that contained the answer was returned. We counted the selected answer as correct if the returned document id matched that of the correct answer.

We defined three types of cost functions, for the delete, insert and change operations, as shown in Figure 3. In these definition, the stop word list contained some common words that would not be very meaningful, e.g. articles such as "a", "an", "the". The word lemma forms were extracted using WordNet (Miller et al., 1990).

1. delete:

   if a is a stop word, $\gamma(a \rightarrow \Lambda) = 5$,

   else $\gamma(a \rightarrow \Lambda) = 200$.

2. insert:

   if a is a stop word, $\gamma(\Lambda \rightarrow a) = 200$,

   else $\gamma(\Lambda \rightarrow a) = 5$.

3. change:

   if a is *ANS*,

   if b matches the expected answer type, $\gamma(a \rightarrow b) = 5$,

   else $\gamma(a \rightarrow b) = 200$,

   else

   if word a is identical to word b, $\gamma(a \rightarrow b) = 0$,

   else if a and b have the same lemma form, $\gamma(a \rightarrow b) = 1$,

   else $\gamma(a \rightarrow b) = 200$.

Figure 3
The definition of the cost functions

We compared our approach with a simple bag-of-word strategy. In that approach, the similarity between a question and a candidate answer is measured as the number of words in common between the question and a candidate answer (either in their surface forms or lemma forms), divided by the length of the answer. The final answer was chosen to be the one that produced the highest similarity.

Note that the evaluation method we used here is different from that in the TREC-2002 Q/A competition. In TREC, an answer produced by a system consists of the answer key and the document that supports the answer. The answer is considered correct only when both the answer key and the supporting document are correct. Since our system does not provide the answer key, we relax the evaluation of our system by finding only the correct supporting document. However, this does not greatly simplify the

task as the harder part of answer selection is to find the correct supporting document. The answer key can be extracted later from the chosen sentence using some heuristics. Also, in practice, a user who uses a Q/A system is very unlikely to believe the system without a correct supporting document. Even though the system does not provide a correct answer key, the user can easily find it given a correct supporting document at hand.

The results of the experiments are shown in Table 1. It shows the large improvement of using this method of mapping dependency trees over the simple bag-of-word strategy.

Table 1
The comparison of the performance of the approximate tree matching approach and the simple bag-of-word. The last column shows the percentage improvement counting the 454 questions that have an answer.

	Correct		
Method	#	%	%(454)
Tree Matching	183	36.60	40.31
Bag-of-Word	151	30.20	33.26

The main reason of the improvement is the ability to exploit the structure of the sentence as illustrated before in Figure 1. Figure 4 provides another example selected from an actual question in our experiment. Although the tree matching approach does not match all keywords in the questions, the dependency tree structure leads it to choose a correct answer while the bag-of-word strategy simply tries to match as many keywords as possible.

5. Conclusion and Discussion

We presented a novel approach that models the answer selection stage in a Q/A process as a problem of approximate tree matching over richer representations of the question and candidate answers. Algorithmically, our approach builds on an algorithm developed in Zhang and Shasha (1989). This approach provides a principled way to incorporate into the decision process both useful syntactic information—in the form of dependency trees in this case, and some semantic information—we used here named entity information. We evaluated our approach on the TREC-2002 questions, and the result clearly illustrates the potential for structure mapping approaches over the common bag-of-word strategy.

We view our approach as a simple instance of a more general structure mapping framework to answer selection, and are planning to extend it in several directions. First, we plan to use more semantic information such as synonyms and related words in our approach. Structurally, at this point each node in a tree represents only a word in a sentence; we believe that appropriately combining nodes into meaningful phrases may allow our approach to perform better. In addition, a limitation of the current implementation is that it makes use of ordered trees, which restricts the possibility of mapping between structures where the order of children is rearranged. An obvious example of this is in the case of active and passive voices. We plan to investigate the use of unordered trees. Although, in general, unordered tree mapping is proved to be a hard problem (Zhang et al., 1992), there is an efficient algorithm in some restricted

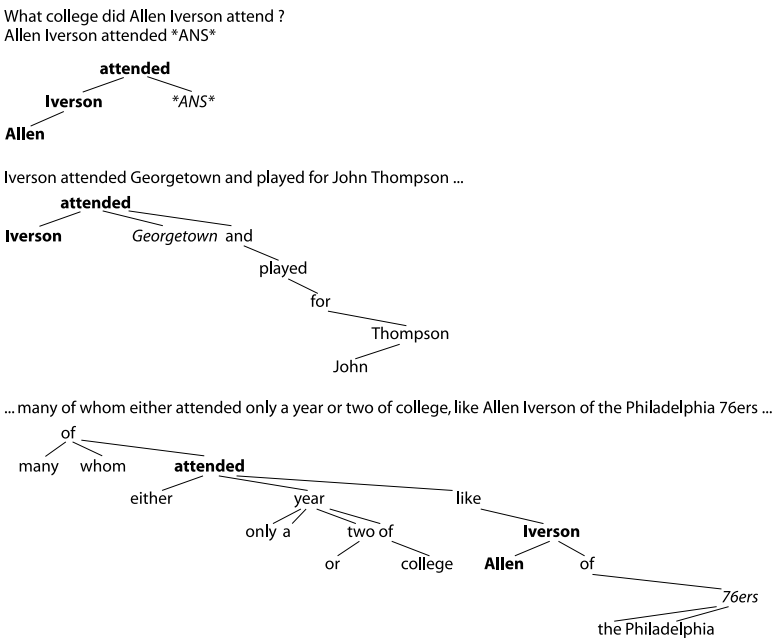

Figure 4
An example of answers found by tree matching and bag-of-word approaches respectively. The bold text represents words that appear in the question and the italic text represents the answer token.

case (Zhang, 1996). Finally, we plan to use learning techniques to learn the cost functions which are now defined manually.

Acknowledgments

This research is supported by NSF grants ITR-IIS-0085836, ITR-IIS-0085980 and IIS-9984168, an ONR MURI Award and by the Advanced Research and Development Activity (ARDA)s Advanced Question Answering for Intelligence (AQUAINT) Program.

References

Collins, M. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association of Computational Linguistics*, pages 16–23, Madrid, Spain.

Ding, Y., D. Gildea, and M. Palmer. 2003. An algorithm for word-level alignment of parallel dependency trees. In *The 9th Machine Translation Summit of International Association of Machine Translation*, New Orleans, LA.

Eisner, J. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (companion volume)*, Supporo, Japan, July.

Even-Zohar, Y. and D. Roth. 2001. A sequential model for multi-class classification. In *Proceedings of 2001 Conference on Empirical methods in Natural Language Processing*, Pittsburgh, PA.

Gildea, D. 2003. Loosely tree-based alignment for machine translation. In *Proceedings of the 41st Annual Meeting of the Association of Computational Linguistics (ACL-03)*, Supporo, Japan.

Harabagiu, S. and D. Moldovan. 2001. Open-domain textual question answering. In *Tuturial of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*.

Li, X. and D. Roth. 2002. Learning question classifiers. In *COLING 2002, The 19th International Conference on Computational Linguistics*, pages 556–562.

Mel'čuk, I. A. 1987. *Dependency Syntax: Theory and Practice*. State University of New York Press, Albany.

Miller, G., R. Beckwith, C. Fellbaum, D. Gross, and K.J. Miller. 1990. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312.

Roth, D., G. K. Kao, X. Li, R. Nagarajan, V. Punyakanok, N. Rizzolo, W. Yih, C. Ovesdotter, and L. Moran. 2001. Learning components for a question-answering system. In *Proceedings of The Tenth Text REtrieval Conference (TREC 2001)*, Gaithesburg, Maryland.

Roth, D. and D. Zelenko. 1998. Part of speech tagging using a network of linear separators. In *Proceedings of COLING-ACL '98*, pages 1136–1142.

Shasha, D. and K. Zhang. 1997. Approximate tree pattern matching. In A. Apostolico and Z. Galil, editors, *Pattern Matching Algorithms*. Oxford University Press, pages 341–371.

Tai, K. 1979. The tree-to-tree correction problem. *Journal of the Association for Computing Machinery*, 26(3):422–433, July.

Voorhees, E. 2000. Overview of the trec-9 question answering track. In *The Ninth Text Retrieval Conference (TREC-9)*, pages 71–80. NIST SP 500-249.

Voorhees, E. 2001. Overview of the trec 2001 question answering. In *The Tenth Text Retrieval Conference (TREC 2001)*, pages 42–51. NIST SP 500-250.

Voorhees, E. 2002. Overview of the trec 2002 question answering. In *The Eleventh Text Retrieval Conference (TREC 2002)*. NIST SP 500-251.

Zhang, K. 1996. A constrained edit distance between unordered labeled trees. *Algorithmica*, 15:205–222.

Zhang, K. and D. Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, December.

Zhang, Z., R. Statman, and D. Shasha. 1992. On the editing distance between unordered labeled trees. *Information Processing Letters*, 42(3):133–139, May.

Appendix

The following notations comply with those defined in Zhang and Shasha (1989). All nodes in any tree are ordered from left-to-right and by postorder numbering. $T[i]$ indicates the ith node in the tree T. $l(i)$ represents the leftmost leaf of the subtree rooted at $T[i]$. $forestdist(T_1[i' \ldots i], T_2[j' \ldots j])$ returns the edit distance between two forests $T_1[i' \ldots i]$ and $T_2[j' \ldots j]$ where $T[i' \ldots i]$ is the forest extracted from nodes between $T[i']$ and $T[i]$.

Modified Lemma 4 in Zhang and Shasha (1989) *Let $i_1 \in anc(i)$ and $j_1 \in anc(j)$. Then*

$$
\begin{aligned}
&forestdist(l(i_1) \ldots i, l(j_1) \ldots j) = \\
&\min \begin{cases}
forestdist(l(i_1) \ldots i-1, l(j_1) \ldots j) + \gamma(T_1[i] \rightarrow \Lambda), \\
forestdist(l(i_1) \ldots i, l(j_1) \ldots j-1) + \gamma(\Lambda \rightarrow T_2[j]), \\
forestdist(l(i_1) \ldots l(i)-1, l(j_1) \ldots l(j)-1) + \\
\quad + forestdist(l(i) \ldots i-1, l(j) \ldots j-1) \\
\quad + \gamma(T_1[i] \rightarrow T_2[j]), \\
forestdist(l(i_1) \ldots l(i)-1, l(j_1) \ldots l(j)-1) + \\
\quad + forestdist(l(i) \ldots i-1, l(j) \ldots j-1) + \\
\quad + \gamma(T_1[i] \rightarrow \Lambda) + \gamma(\Lambda \rightarrow T_2[j]).
\end{cases}
\end{aligned}
$$

The last case is ignored in Zhang and Shasha (1989) because its value is no less than the third case due to the assumed triangularity property.