

(19) World Intellectual Property
Organization
International Bureau



(43) International Publication Date
29 September 2005 (29.09.2005)

PCT

(10) International Publication Number
WO 2005/089340 A2

(51) International Patent Classification: **Not classified**

(74) Agent: **HARRIS, Scott, C.**; Fish & Richardson P.C.,
12390 El Camino Real, San Diego, CA 92130 (US).

(21) International Application Number:
PCT/US2005/008648

(22) International Filing Date: 15 March 2005 (15.03.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/553,587 15 March 2004 (15.03.2004) US

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US (patent), UZ, VC, VN, YU, ZA, ZM, ZW.

(63) Related by continuation (CON) or continuation-in-part (CIP) to earlier applications:
US 60/553,587 (CON)
Filed on 15 March 2004 (15.03.2004)
US Not furnished (CIP)
Filed on Not furnished

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

(71) Applicant (for all designated States except US): **UNIVERSITY OF SOUTHERN CALIFORNIA** [US/US]; 3716 S. Hope Street, Suite 313, Los Angeles, CA 90007-4344 (US).

Published:

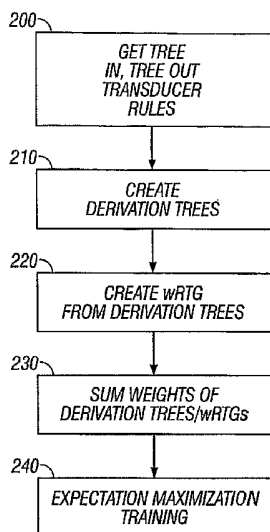
— without international search report and to be republished upon receipt of that report

(72) Inventors; and

(75) Inventors/Applicants (for US only): **KNIGHT, Kevin** [US/US]; 43 6th Street, Hermosa Beach, CA 90245 (US).
GRAEHL, Jonathan [US/US]; 8213 W. Manchester, #2, Playa del Rey, CA 90293 (US).

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: TRAINING TREE TRANSDUCERS



(57) Abstract: Training using tree transducers is described. Given sample input/output pairs as training, and given a set of tree transducer rules, the information is combined to yield locally optimal weights for those rules. This combination is carried out by building a weighted derivation forest for each input/output pair and applying counting methods to those forests.

WO 2005/089340 A2

Training Tree Transducers

[0001] This application claims the benefit of the priority of U.S. Provisional Application Serial No. 60/553,587, filed March 15, 2004 and entitled "TRAINING TREE TRANSDUCERS", the disclosure of which is hereby incorporated by reference.

Background

[0002] Many different applications are known for tree transducers. These have been used in calculus, other forms of higher mathematics. Tree transducers are used for decidability results in logic, for modeling mathematically the theories of syntax direction translations and program schemata, syntactic pattern recognition, logic programming, term rewriting and linguistics.

[0003] Within linguistics, automated language monitoring programs often use probabilistic finite state transducers that operate on strings of words. For example, speech recognition may transduce acoustic sequences to word sequences using left to right substitution. Tree based models based on probabilistic techniques have been used for machine translation, machine summarization, machine paraphrasing, natural language generation, parsing, language modeling, and others.

[0004] A special kind of tree transducer, often called an **R** transducer, operates with its roots at the bottom, with **R** standing for "root to frontier". At each point within the operation, the transducer chooses a production to apply. That choice is based only on the current state and the current root symbol. The

travel through the transducer continues until there are no more state annotated nodes.

[0005] The **R** transducer represents two pairs, T1 and T2, and the conditions under which some sequence of productions applied to T1 results in T2. This is similar to what is done by a finite state transducer.

[0006] For example, if a finite state transition from state q to state r eats symbol A and outputs symbol B, then this can be written as an R production of $q(Ax_0) \rightarrow B(rx_0)$.

[0007] The **R** transducer may also copy whole trees, transform subtrees, delete subtrees, and other operations.

Summary

[0008] The present application teaches a technique of training tree transducers from sample input/output pairs. A first embodiment trains the tree pairs, while a second embodiment trains the tree transducers based on tree/string pairs. Techniques are described that facilitate the computation, and simplify the information as part of the training process.

[0009] An embodiment is described which uses these techniques to train transducers for statistical based language processing: e.g. language recognition and/or language generation. However, it should be understood that this embodiment is merely exemplary, and the other applications for the training of the tree transducers are contemplated.

Brief description of the drawings

[0010] These and other aspects will now be described in detail with reference to the accompanying drawings, wherein:

[0011] Figure 1 shows derivation trees and their simplifications;

[0012] Figure 2A shows a flowchart;

[0013] figure 2B shows a speech engine that can execute the flowchart of figure 2A;

[0014] Figure 2C shows a flowchart of a second embodiment; and

[0015] Figures 3 and 4 show a model and parameter table.

Detailed description

[0016] The present application describes training of tree transducers. The embodiment describes training of tree transducers, e.g., probabilistic **R** transducers. These transducers may be used for any probabilistic purpose. In an embodiment, the trained transducers are used for linguistic operations, such as machine translation, paraphrasing, text compression and the like. Training data may be obtained in the form of tree pairs. Linguistic knowledge is automatically distilled from those tree pairs and transducer information.

[0017] $T\Sigma$ represents the set of trees over the alphabet Σ . An alphabet is a finite set of symbols. Trees may also be written as strings over the set Σ .

[0018] A regular tree grammar or RTG allows compactly representing a potentially infinite set of trees. A weighted regular tree grammar is a set of values, where trees in the set have weights associated with them. The trees

can be described as a quadruple $G(\Sigma, N, S, P)$, where Σ is the alphabet, and N is the set of non-terminals, S is the starting (initial) terminal, and P is the set of weighted productions. The productions are written left to right. A weighted RTG can accept information from an infinite number of trees. More generally, the weighted RTG can be any list which includes information about the trees in a tree grammar, in a way that allows the weight to change rather than a new entry each time the same information is reobtained.

[0019] The RTG can take the following form:

$$\begin{aligned} \Sigma &= \{S, NP, VP, PP, PREP, DET, N, V, \text{run, the, of, sons, daughters}\} \\ N &= \{qnp, qpp, qdet, qn, qprep\} \\ S &= q \\ P &= \{q \rightarrow^{1.0} S(qnp, VP(VB(\text{run}))), \\ &\quad qnp \rightarrow^{0.6} NP(qdet, qn), \\ &\quad qnp \rightarrow^{0.4} NP(qnp, qpp), \\ &\quad qpp \rightarrow^{1.0} PP(qprep, np), \\ &\quad qdet \rightarrow^{1.0} DET(\text{the}), \\ &\quad qprep \rightarrow^{1.0} PREP(\text{of}), \\ &\quad qn \rightarrow^{0.5} N(\text{sons}), \\ &\quad qn \rightarrow^{0.5} N(\text{daughters})\} \end{aligned}$$

Table I

[0020] The tree is parsed from left to right, so that the leftmost non-terminal is the next one to be expanded as the next item in the RTG. The left most derivations of G build a tree pre-order from left to right according to

$$LD(G) \equiv \left\{ (t, ((p_1, r_1), \dots, (p_n, r_n))) \in D_G \mid \forall 1 \leq i < n : p_{i+1} \not\prec_{lex} p_i \right\}$$

The *total weight of t in G* is given by $W_G : T_\Sigma \rightarrow \mathbb{R}$, the sum of leftmost derivations producing t :

$$W_G(t) \equiv \sum_{(t,h) \in LD(G)} \prod_{i=1}^n w_i \text{ where } h = (h_1, \dots, h_n) \text{ and } h_i = (p_i, (l_i, r_i, w_i))$$

[0021] Therefore, for every weighted context free grammar, there is an equivalent weighted RTG that produces weighted derivation trees. Each weighted RTG is generated from exactly the recognizable tree language.

[0022] An extended transducer are is also used herein. According to this extended transducer $\mathbf{xR}$, an input subtree matching pattern in state q is converted into its right hand side ("rhs"), and it's Q paths are replaced by their recursive transformations. The right hand side of these rules may have no states for further expansions (terminal rules) or may have states for further expansion.

In notation form

$$\Rightarrow_X \equiv \left\{ ((a, h), (b, h \cdot (i, (q, pattern, rhs, w)))) \mid (q, pattern, rhs, w) \in R \wedge \right. \\ \left. i \in paths_n \wedge q = label_a(i) \wedge pattern(a \downarrow (i \cdot (1))) = 1 \wedge \right. \\ \left. b = a \left[i \leftarrow rhs \left[\forall p \in paths_{rhs} : label_{rhs}(p) = (q', i') \right] \right] \right\}$$

where, b is derived from a by application of a

rule (queue, pattern) $\rightarrow$ rhs

to an unprocessed input subtree a_i which is in state q .

Its output is replaced by the output given by rhs. Its non-terminals are replaced by the instruction to transform descendent input subtrees.

The *sources* of a rule $r = (q, l, rhs, w) \in R$ are the input-paths in the *rhs*:

$$sources(rhs) \equiv \{i' \mid \exists p \in paths_{rhs}(Q \times paths), q' \in Q : label_{rhs}(p) = (q', i')\}$$

The reflexive, transitive closure of $\Rightarrow_X$ is written $\Rightarrow_X^*$, and the *derivations of X*, written $D(X)$, are the ways of transforming input tree I (with its root in the initial state) to an output tree O :

$$D(X) \equiv \{(I, O, h) \in T_\Sigma \times T_\Delta \times (paths \times P)^* \mid (Q_i(I), ()) \Rightarrow_X^* (O, h)\}$$

The *leftmost derivations of X* transform the tree preorder from left to right (always applying a transformation rule to the state-labeled subtree furthest left in its string representation):

$$LD(X) \equiv \{(I, O, ((p_1, r_1), \dots, (p_n, r_n))) \in D(X) \mid \forall 1 \leq i < n : p_{i+1} \not\prec_{lex} p_i\}$$

The *total weight of (I, O) in X* is given by $W_X : T_\Sigma \times T_\Delta \rightarrow \mathbb{R}$, the sum of leftmost derivations transforming I to O :

$$W_X(I, O) \equiv \sum_{(I, O, h) \in LD(X)} \prod_{i=1}^n w_i \text{ where } h = (h_1, \dots, h_n) \text{ and } h_i = (p_i, (l_i, r_i, w_i))$$

[0023] The tree transducers operate by starting at an initial state root and recursively applying output generating rules until no states remain, so that there is a complete derivation. In this way, the information (trees and transducer information) can be converted to a derivation forest, stored as a weighted RTG.

[0024] The overall operation is illustrated in the flow chart of figure 2A; and figure 2B illustrates an exemplary hardware device which may execute that flowchart. For the application of language translation, a processing module 250 receives data from various sources 255. The sources may be the input and output trees and transducer rules described herein. Specifically, this may be the translation memories, dictionaries, glossaries, Internet, and human-created translations. The processor 250 processes this information as described herein to produce translation parameters which are output as 260. The translation parameters are used by language engine 265 in making translations based on input language 270. In the disclosed embodiment, the speech engine is a language translator which translates from a first language to a second language. However, alternatively, the speech engine can be any engine that operates on strings of words such as a language recognition device in speech recognition device, a machine paraphraser, natural language generator, modeler, or the like.

[0025] The processor 250 and speech engine 265 may be any general purpose computer, and can be effected by a microprocessor, a digital signal processor, or any other processing device that is capable of executing the steps described herein.

[0026] The flowchart described herein can be instructions which are embodied on a machine-readable medium such as a disc or the like. Alternatively, the flowchart can be executed by dedicated hardware, or by any known or later discovered processing device.

[0027] The system obtains a plurality of input and output trees or strings, and transducer rules with parameters. The parameters may then be used for statistical machine translation. More generally, however, the parameters can be used for any tree transformation task.

[0028] At 210, the input tree, output tree and transducer rules are converted to a large set of individual derivation trees, "a derivation forest".

[0029] The derivation forest effectively flattens the rules into trees of depth one. The root is labeled by the original rule. All the non-expanding Δ labeled nodes of the rule are deterministically listed in order. The weights of the derivation trees are the products of the weights of the rules in those derivation trees.

[0030] Figure 1 illustrates an input tree 100 being converted to an output tree 110 and generating derivation trees 130. Figure 1 also shows the transducer rules 120. All of these are inputs to the system, specifically the input and output tree are the data that is obtained from various language translation resources 255, for example. The transducer rules are known. The object of the parsing carried out in figure 1 is to derive the derivation trees 130 automatically.

[0031] The input/output tree pairs are used to produce a probability estimate for each production in P, that maximizes the probability of the output trees given

the input trees. The result is to find a local maximum. The present system uses simplifications to find this maximum.

[0032] The technique describes the use of memoization by creating the weighted RTG's. Memoization means that the possible derivations for a given produced combination are constant. This may prevent certain combinations from being computed more than once. In this way, the table, here the wRTG can store the answers for all past queries and return those instead of recomputing.

[0033] Note the way in which the derivation trees are converted to weighted RTG's. At the start, rule one will always be applied, so the first RTG represents a 1.0 probability of rule one being applied. The arguments of rule one are 1.12 and 2.11. If 1.12 is applied, rule 2 is always used, while 2.11 can be either rule 3 or rule 4, with the different weightings for the different rules being also shown.

[0034] At 230, the weighted RTG is further processed to sum the weights of the derivation trees. This can use the "inside-outside" technique, (Lari, et al, "The estimation of stochastic context free grammars using the inside-outside algorithm, Computer Speech and Language, 4, pp 35-36). The inside-outside technique observes counts and determines each time a rule gets used. When a rule gets used, the probability of that rule is increased. More specifically, given a weighted RTG with parameters, the inside outside technique enables computing the sums of weights of the trees derived using each production. Inside weights are the sum of all weights that can be derived for a non-terminal or production. This is a recursive definition. The inside weights for a production are the sum of all the weights of the trees that can be derived from that production.

$$\beta_G(n \in N) \equiv \sum_{(n,r,w) \in P} w \cdot \beta_G(r)$$

$$\beta_G(r \in T_N(N) \mid (n, r, w) \in P) \equiv \prod_{p \in \text{paths}_r(N)} \beta_G(\text{label}_r(p))$$

[0035] The outside weights for a non-terminal are the sum of weights of trees generated by the weighted RTG that have derivations containing it but exclude its inside weights, according to

$$\alpha_G(n \in N) \equiv \begin{cases} 1 & \text{if } n = S \\ \underbrace{\sum_{p, (w', r, w) \in P: \text{label}_r(p) = n} w \cdot \alpha_G(n')}_{\text{uses of } n \text{ in productions}} \cdot \underbrace{\prod_{p' \in \text{paths}_r(N) - \{p\}} \beta_G(\text{label}_r(p'))}_{\text{sibling nonterminals}} & \text{otherwise.} \end{cases}$$

[0036] Estimation maximization training is then carried out at 240. This maximizes the expectation of decisions taken for all possible ways of generating the training corpus, according to expectation, and then maximization, as:

$$\forall p \in \text{parameters} : \text{counts}_p \equiv E_{d \in \text{derivations}} \left[\frac{\sum_{d \in \text{derivations}_i} (\# \text{ of times } p \text{ used in } d) \cdot \text{weight}_{\text{parameters}}(d)}{\sum_{d \in \text{derivations}_i} \text{weight}_{\text{parameters}}(d)} \right]$$

2. Maximizing by assigning the counts to the parameters and renormalizing:

$$\forall p \in \text{parameters} : p \leftarrow \frac{\text{counts}_p}{Z(p)}$$

[0037] Each iteration increases the likelihood until a local maximum is reached.

[0038] The step 230 can be written in pseudocode as:

For each $(i, o, w_{\text{example}}) \in T$: *// Estimate*

- i. *Let* $D \equiv d_{i,o}$
- ii. compute α_D, β_D using latest W *// inside-outside weights*
- iii. *For each* $\text{prod} = (n, \text{rhs}, w) \in P : \text{label}_{\text{rhs}}() \in R$ in derivation $\text{wRTG } D = (R, N, S, P)$:
 - A. $\gamma_D(\text{prod}) \leftarrow \alpha_G(n) \cdot w \cdot \beta_G(\text{rhs})$
 - B. *Let* $\text{rule} \equiv \text{label}_{\text{rhs}}()$
 - C. $\text{count}_{\text{rule}} \leftarrow \text{count}_{\text{rule}} + w_{\text{example}} \cdot \frac{\gamma_D(\text{prod})}{\beta_D(S)}$
- iv. $L \leftarrow L + \log \beta_D(S) \cdot w_{\text{example}}$

For each $r = (q, \text{pattern}, \text{rhs}, w) \in R$: *// Maximize*

- i. $w_r \leftarrow \frac{\text{count}_r}{Z(\text{counts}, r)}$

$\delta \leftarrow \frac{L - \text{lastL}}{|L|}$
 $\text{lastL} \leftarrow L, \text{itno} \leftarrow \text{itno} + 1$

By using the weighted RTG's, each estimation maximum iteration takes an amount of time that is linear to the size of the transducer. For example, this may compute the sum of all the counts for rules having the same state, to provide model weights for a joint probability distribution of the input output tree pairs. This joint normalization may avoid many different problems.

[0039] The above has described tree-to-tree transducers. An alternative embodiment describes tree to string transducers is shown in the flowchart of figure 2C. This transducer will be used when a tree is only available at the input side of the training corpus. Note that figure 2C is substantially identical to figure 2A other than the form of the input data.

[0040] The tree to string transduction is then parsed using an extended **R** transducer as in the first embodiment. This is used to form a weighted derivation tree grammar. The derivation trees are formed by converting the input tree and the string into a flattened string of information which may include trees and strings. 285 of Figure 5c simply refers to this as derivation information. The parsing of the tree to string transduction may be slightly different then the tree to tree transduction. Instead of derivation trees, there may be output string spans. A less constrained alignment may result.

[0041] This is followed in Figure 2C by operations that are analogous to those in Figure 2A: specifically, creation of the weighted RTG, the same as the weight summing of 230 and the expectation maximization of 240.

[0042]

Example

[0043] An example is now described here in of how to cast a probabilistic language model as an R transducer.

Table 2 shows a bilingual English tree Japanese string training corpus.

```

-----
ENGLISH:  (VB (NN hypocrisy)
           (VB is)
           (JJ (JJ abhorrent)
              (TO (TO to) (PRP them))))

JAPANESE:  kare ha gizen ga daikirai da
-----
ENGLISH:  (VB (PRP he)
           (VB has)
           (NN (JJ unusual) (NN ability))
           (IN (IN in) (NN english)))

JAPANESE:  kare ha eigo ni zabanuke-ta sainou wo mot-te iru
-----
ENGLISH:  (VB (PRP he)
           (VB was)
           (JJ (JJ ablaze)
              (IN (IN with) (NN anger))))

JAPANESE:  kare ha mak-ka ni nat-te okot-te i-ta
-----
ENGLISH:  (VB (PRP i)
           (VB abominate)
           (NN snakes))

JAPANESE:  hebi ga daikirai da
-----
etc.

```

Table 2

[0044] Figures 3 and 4 respectively show the generative model and its parameters. The parameter values that are shown are learned via expectation maximization techniques as described in Yamada and Knight 2001.

[0045] According to the model, an English tree becomes a Japanese string in four operations. Figure 3 shows how the channel input is first reordered, that is its children are permuted probabilistically. If there are three children, then there are six possible permutations whose probabilities add up to one. The reordering is done depending only on the child label sequence.

[0046] In 320, a decision is made at every node about inserting a Japanese function word. This is a three-way decision at each node, requiring determination of whether the word should be inserted to the left, to the right, or not inserted at all. This insertion technique at 320 depends on the labels of the node and the parent. At 330, the English leaf words are translated probabilistically into Japanese, independent of context. At 340, the internal nodes are removed, leaving only the Japanese string.

[0047] This model can effectively provide a formula for $P(\text{Japanese string} | \text{English tree})$ in terms of individual parameters. The expectation maximization training described herein seeks to maximize the product of these conditional probabilities based on the entire tree-string corpus.

[0048] First, an xRs tree to string transducer is built that embodies the probabilities noted above. This is a four state transducer. For the main-start state, the function q , meaning translate this tree, has three productions:

$$\bullet q\ x \rightarrow i\ x, r\ x$$

$$\bullet q\ x \rightarrow r\ x, i\ x$$

$$\bullet q\ x \rightarrow r\ x$$

State i means "produce a Japanese word out of thin air." There is an i production for each Japanese word in the vocabulary.

• $i\ x \rightarrow \text{"de"}$

• $i\ x \rightarrow \text{"kuruma"}$

• $i\ x \rightarrow \text{"wa"}$

• ...

[0049] State r means "reorder my children and then recurse". For internal nodes, this includes a production for each parent/child sequence, and every permutation thereof:

• $r\ NN(x0:CD, x1:NN) \rightarrow q\ x0, q\ x1$

• $r\ NN(x0:CD, x1:NN) \rightarrow q\ x1, q\ x0$

• ...

[0051] The RHS then sends the child subtrees back to state q for recursive processing. For English leaf nodes, the process instead transitions to a different state t to prohibit any subsequent Japanese function word insertion:

- $r \text{ NN}(x_0: \text{"car"}) \rightarrow t \ x_0$
- $r \text{ CC}(x_0: \text{"and"}) \rightarrow t \ x_0$
- ...

[0052] State t means "translate this word". There is a production for each pair of cooccurring in English and Japanese words.

- $t \text{ "car"} \rightarrow \text{"kuruma"}$
- $t \text{ "car"} \rightarrow \text{"wa"}$
- $t \text{ "car"} \rightarrow *e*$
- ...

[0053] Each production in the XRS transducer has an associated weight, and corresponds to exactly 1 of the model parameters.

[0054] The transducer is unfaithful in one respect, specifically the insert function word decision is independent of context. It should depend on the node

label and the parent label. This is addressed by fixing the q and r production.

Start productions are used:

$$\bullet q\ x:VB \rightarrow q.TOP.VB\ x$$

$$\bullet q\ x:JJ \rightarrow q.TOP.JJ\ x$$

$$\bullet \dots$$

[0055] States are used, such as q.top.vb which states mean something like "translate this tree, whose route is vb". Every parent-child payer in the corpus gets its own set of insert function word productions:

$$\bullet q.TOP.VB\ x \rightarrow i\ x, r\ x$$

$$\bullet q.TOP.VB\ x \rightarrow r\ x, i\ x$$

$$\bullet q.TOP.VB\ x \rightarrow r\ x$$

$$\bullet q.VB.NN\ x \rightarrow i\ x, r\ x$$

$$\bullet q.VB.NN\ x \rightarrow r\ x, i\ x$$

$$\bullet q.VB.NN\ x \rightarrow r\ x$$

$$\bullet \dots$$

[0056] Finally, the **R** productions need to send parent child information when they recurse to the q.parent.child states.

[0057] The productions stay the same. Productions for appraisal translations and others can also be added.

[0058] Although only a few embodiments have been disclosed in detail above, other modifications are possible, and this disclosure is intended to cover all such modifications, and most particularly, any modification which might be predictable to a person having ordinary skill in the art. For example, an alternative embodiment could use the same techniques for string to string training, based on tree based models or based only on string pair data. Another application is to generate likely input trees from output trees or vide versa. Also, and to reiterate the above, many other applications can be carried out with tree transducers, and the application of tree transducers to linguistic issues is merely exemplary.

[0059] Also, only those claims which use the words "means for" are intended to be interpreted under 35 USC 112, sixth paragraph. Moreover, no limitations from the specification are intended to be read into any claims, unless those limitations are expressly included in the claims

[0060] All such modifications are intended to be encompassed within the following claims

What is claimed is

1. A method, comprising:

obtaining tree transducer information including input/ output pair information and transducer information;

converting said input/output pair information and said transducer information into a set of values in a weighted tree grammar; and

using said weighted tree grammar to solve a problem that requires information from the input/output pair information and transducer information.

2. A method as in claim 1, wherein said using comprises training a linguistic engine which solves a linguistic problem.

3. A method as in claim 2, wherein said linguistic problem includes training of a linguistic engine of a type that converts from one language to another.

4. A method as in claim 1, wherein said tree transducer information includes an input tree and an output tree as said input/output pair.

5. A method as in claim 1, wherein said tree transducer information includes an input tree and an output string as said input/output pair.

6. A method as in claim 1, further comprising converting said tree transducer

information into a derivation forest.

7. A method as in claim 1, wherein said set of values represents information about the tree transducer information and transducer information in a specified grammar, associated with a weight for each of a plurality of entries.

8. A method as in claim 7, wherein said set of values are in a weighted regular tree grammar.

9. A method as in claim 8, wherein said converting comprises storing the set of values as a weighted regular tree grammar, and returning certain stored information instead of recomputing said certain stored information.

10. A method as in claim 1, further comprising further processing the set of values to sum weights based on information learned from said tree transducer information and said transducer information.

11. A method as in claim 10, wherein said further processing comprises using an inside-outside algorithm to observe counts and determine each time a rule gets used and to adjust said rule weights based on said observing.

12. A method as in claim 10, wherein said further processing comprises observing counts and determining each time a rule gets used, and increasing a

probability of that rule each time the rule gets used.

13. A method as in claim 1, wherein said training further comprises maximizing an expectation of decisions.

14. A method as in claim 1, wherein said training further comprises computing a sum of all the counts for rules having the same state, to provide model weights for one of a joint or conditional probability distribution of the tree transducer information.

15. A method as in claim 1, wherein said using comprises solving a logic problem.

16. A method as in claim 15, wherein said logic problem is a problem that uses a machine to analyze and the aspect of at least one language.

17. A training apparatus comprising:

an input part which includes tree transducer information that includes input samples and output samples, and includes transducer information;

a processor that processes said tree transducer information and said transducer information and produces an output table indicative of said tree transducer information and said transducer information, said output table including a set of values, and weights for said set of values.

18. A training apparatus as in claim 17, further comprising a linguistic engine, receiving said output tables, addressing a linguistic problem based on said output tables.

19. A training apparatus as in claim 17, further comprising using an inside-outside algorithm to observe counts and determine each time a rule gets used and to adjust said rule weights based on the observing.

20. A training apparatus as in claim 17, wherein said input part is a tree transducer.

21. A training apparatus as in claim 20, wherein said output part is a tree transducer.

22. A training apparatus as in claim 20, wherein said output of part is a string transducer.

23. A training apparatus as in claim 17, wherein said output table stores a set of values as a regular tree grammar that is weighted with weights related to a number of times that each rule is used.

24. A training apparatus as in claim 23, wherein said processor is also operative

to adjust said weights.

25. A training apparatus as in claim 17, further comprising a logic engine that analyzes at least one logical problem based on said output table.

26. A computer readable storage medium containing a set of instructions to enable a computer to carry out a function, the set of instructions including instructions to:

obtain tree transducer information including at least an input tree, output information corresponding to said input tree, and transducer information;

convert said tree transducer information into a weighted set of values indicative of said tree transducer information and said transducer information; and

use said set of values to solve a problem.

27. A storage medium as in claim 26, wherein said output information within said tree transducer information includes an output tree.

28. A storage medium as in claim 26, wherein said output information within said tree transducer information includes an output string.

29. A storage medium as in claim 26, wherein said instructions to convert include instructions to maximize an expectation that a rule will be used.

30. A method comprising:

using a computer to obtain information in the form of a first-tree, second information corresponding to said first tree, and transducer information; and using said computer to automatically distill information from said first tree, said from said second information, and said transducer information into a list of information in a specified tree grammar with weights associated with entries in the list and to produce locally optimal weights for said entries.

.

31. A method as in claim 30, further comprising using said list of information in said computer to solve a problem.

32. A method as in claim 30, wherein said automatically distill comprises parsing the information from a first portion to a second portion.

33. A method as in claim 32, wherein said second information comprises a second tree.

34. A method as in claim 32, wherein said second information comprises a string.

35. A method as in claim 30, wherein said automatically distill comprises

observing when a specified rule in said list of information is used, and increasing said weight associated with said specified rule when said specified rule is used.

36. A method as in claim 31, wherein said problem is a linguistic problem.

1/5

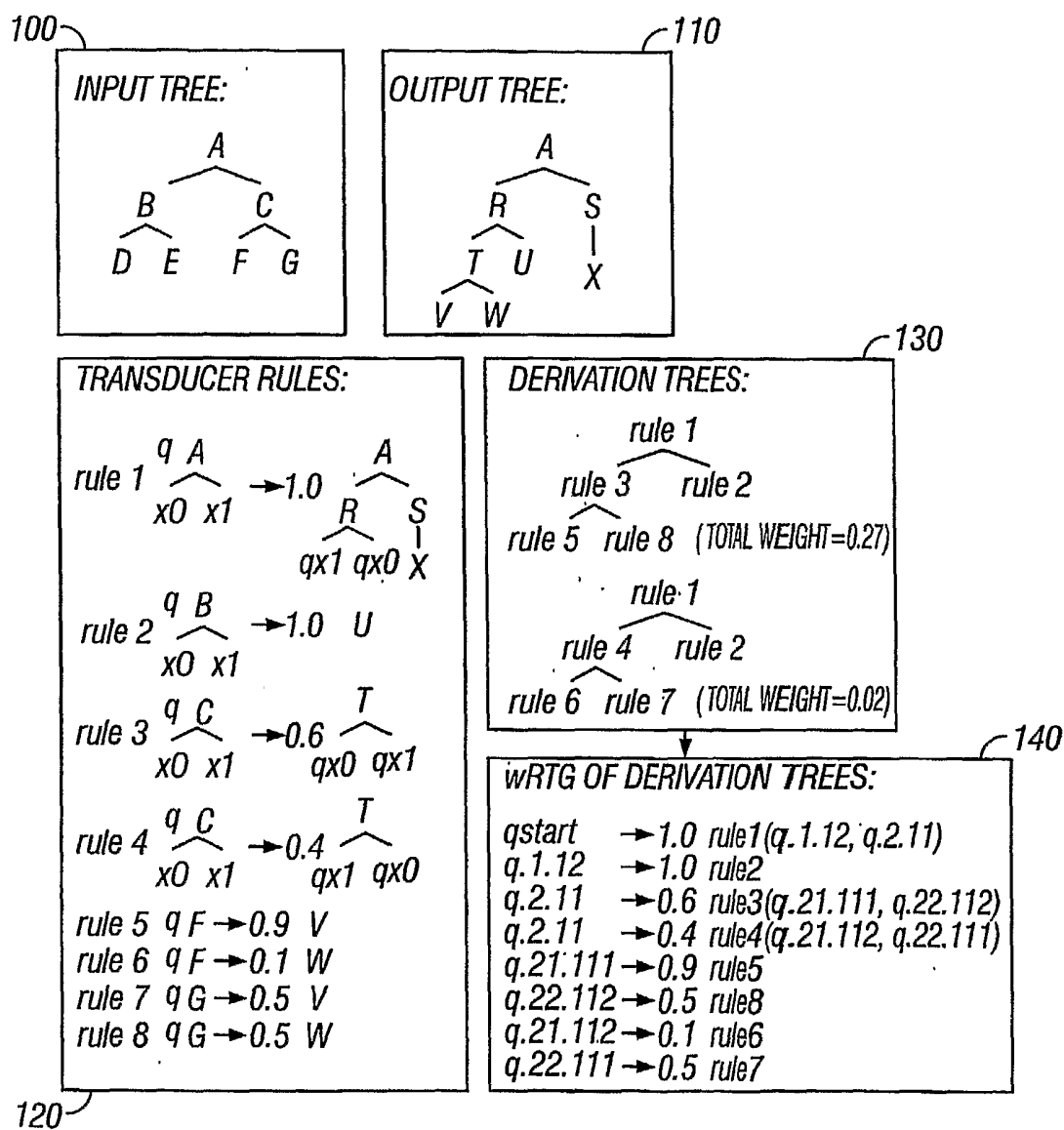


FIG. 1

2/5

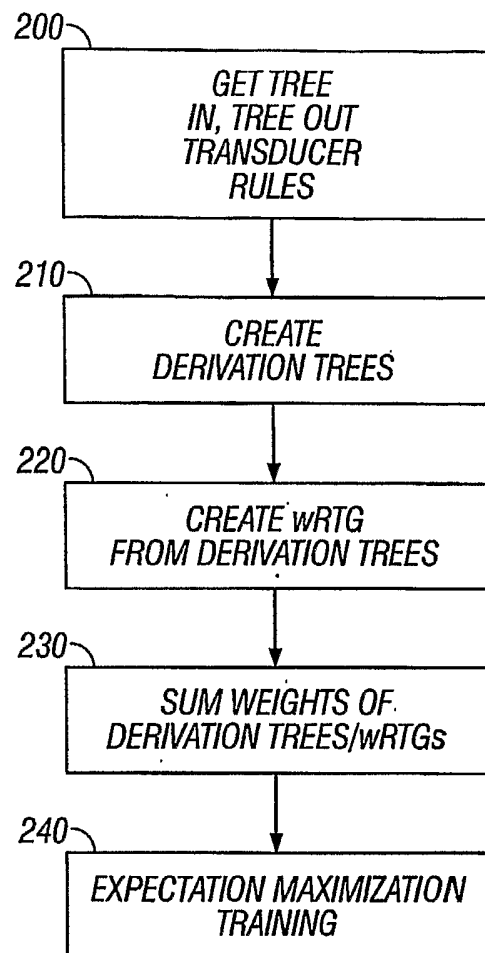


FIG. 2A

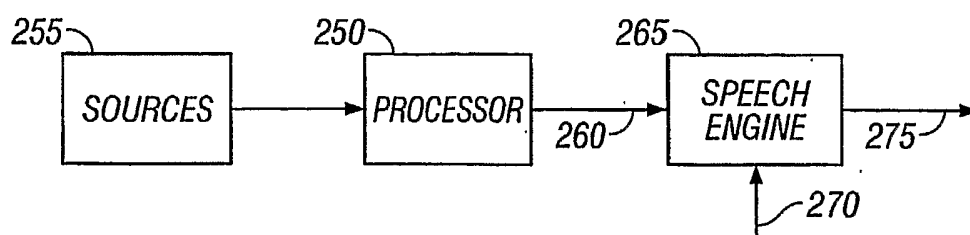


FIG. 2B

SUBSTITUTE SHEET (RULE 26)

3/5

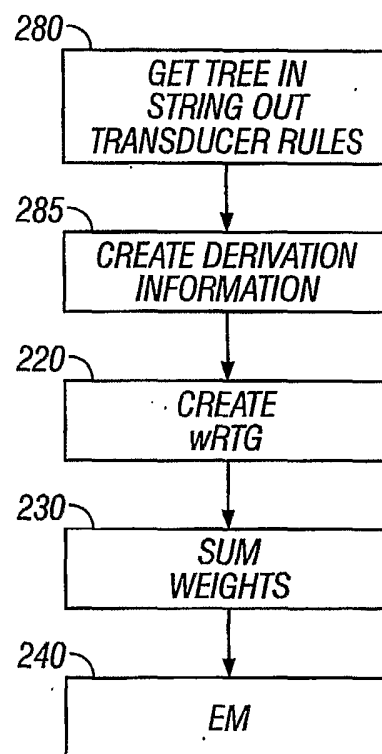


FIG. 2C

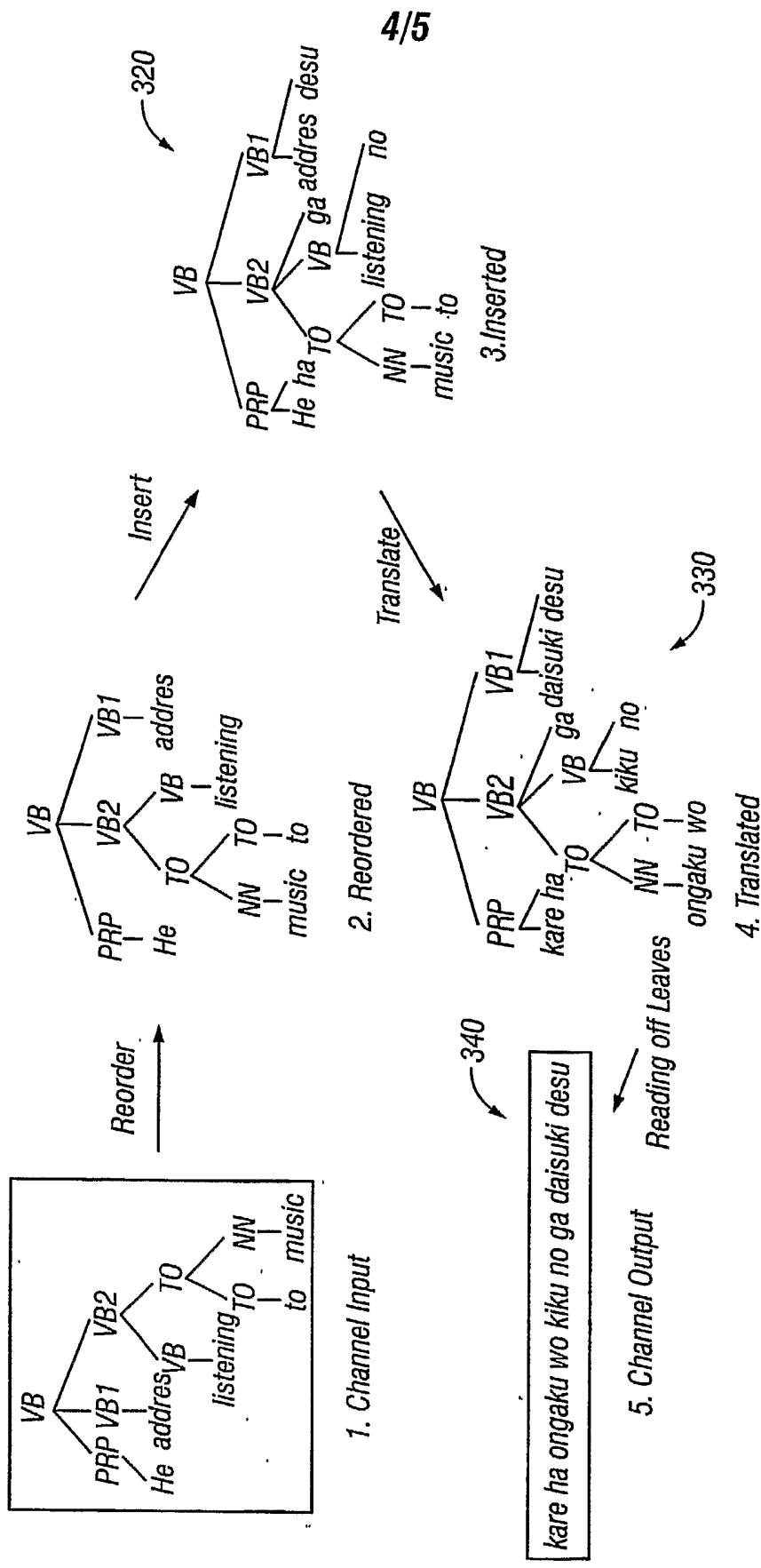


FIG. 3

w	P(ins-w)
ha	0.219
ta	0.131
wo	0.099
no	0.094
ni	0.080
te	0.078
ga	0.062
...	...
desu	0.0007
...	...

parent	TOP	VB	VB	VB	TO	...
node	VB	VB	PRP	TO	NN	...
P(None)	0.735	0.687	0.344	0.709	0.900	0.800
P(Left)	0.004	0.061	0.004	0.030	0.003	0.096
P(Right)	0.260	0.252	0.652	0.261	0.007	0.104

n-table

original order	reordering	P(reorder)
PRP VB1 VB2	PRP VB1 VB2	0.074
	PRP VB2 VB1	0.723
	VB1 PRP VB2	0.061
	VB1 VB2 PRP	0.037
	VB2 PRP VB1	0.083
	VB2 VB1 PRP	0.021
VB TO	VB TO	0.251
TO VB	TO VB	0.749
TO NN	TO NN	0.107
NN TO	NN TO	0.893
...	...	...

r-table

E	adores	he	i	listening	music	to
J	daisuki 1.000	kare 0.952	NULL 0.471	kiku 0.333	ongaku 0.900	ni 0.216
		NULL 0.016	watasi 0.111	kii 0.333	naru 0.100	NULL 0.204
		nani 0.005	kare 0.055	mi 0.333		to 0.133
		da 0.003	shi 0.021			no 0.046
		shi 0.003	nani 0.020			wo 0.038
		...	...			...

t-table

FIG. 4