An Automatic Treebank Conversion Algorithm for Corpus Sharing

Jong-Nae Wang
Behavior Design Corporation
No. 28, 2F, R&D Road II
Science-Based Industrial Park
Hsinchu, Taiwan 30077, R.O.C.
wjn@bdc.com.tw

Jing-Shin Chang and Keh-Yih Su
Dept. of Electrical Engineering
National Tsing-Hua University
Hsinchu, Taiwan 30043, R.O.C.
sh±n~hera, ee. nthu. edu. tw
kysu~bdc, com. tw

Abstract
An automatic treebank conversion method is proposed in this paper to convert a treebank into another treebank. A new treebank associated with a different grammar can be generated automatically from the old one such that the information in the original treebank can be transformed to the new one and be shared among different research communities. The simple algorithm achieves conversion accuracy of 96.4% when tested on 8,867 sentences between two major grammar revisions of a large MT system.

Motivation
Corpus-based research is now a major branch for language processing. One major resource for corpus-based research is the treebanks available in many research organizations [Marcus et al. 1993], which carry skeletal syntactic structures or 'brackets' that have been manually verified. Unfortunately, such resources may be based on different tag sets and grammar systems of the respective research organizations. As a result, reusability of such resources across research laboratories is poor, and cross-checking among different grammar systems and algorithms based on the same corpora cannot be conducted effectively. In fact, even for the same research organization, a major revision of the original grammar system may result in a re-construction of the system corpora due to the variations between the revisions. As a side effect, the evolution of a system is often blocked or discouraged by the unavailability of the corresponding corpora that were previously constructed. Under such circumstances, much energy and cost may have to be devoted to the re-tagging or re-construction of those previously available corpora. It is therefore highly desirable to automatically convert an existing treebank, either from a previous revision of the current system or from another research organization, into another that is compatible with the current grammar system.

Several problems may prevent a treebank conversion algorithm from effective conversion of the treebanks. Firstly, the tag sets, including terminal symbols (parts of speech) and nonterminal symbols (syntactic categories) may not be identical in the two systems; the number of such symbols may be drastically different and the mapping may not be one-to-one. Furthermore, the hierarchical structures, i.e., the underlying phrase structure grammars, of two grammar systems may not be easily and uniquely mapped. In fact, the number of mapping units and mapping rules between two systems may become intolerably large if no systematic approach is available to extract the atomic mapping units and the mapping operations [Chang and Su 1993]. In addition, some constructs in one system may not be representable in terms of the grammar of another system; compatibility of two grammar systems thus further complicates the conversion problems.

In many cases, a publicly available corpus may contain only the simplest annotations, like brackets (skeletal structure representations) for some major syntactic categories [Marcus et al. 1993]. In particular, a research organization may not want to contribute its corpora in full detail for free to the public since it may reveal the underlying knowledge, such as the grammar rules, used in the proprietary system. Therefore, the primitive annotations, like brackets, are very likely to be the sole information available to the public in the near future. And corpus exchange is very likely to be limited to such primitive annotations. Such resources may not be directly usable by a system which needs much more information than annotated. In such cases, it is, however, desirable to be able to use the large amount of simply tagged corpus to help construct or bootstrap a large corpus which contains more detailed annotation.

We thus try to address such problems by using a simple and automatic approach for treebank conversion. Since the bracket information from a large treebank is the major external information
required, the proposed algorithm is expected to be very useful and cost-effective for bootstrapping the corpus, in terms of corpus size and annotated information, of a system by using publicly available treebanks or home-made treebanks, which are less costly than fully annotated corpora.

In the following sections, the treebank conversion task is modeled as a transfer problem, commonly encountered in an MT system, between two representations of the same language. A matching metric for selecting the best conversion among all candidates is then proposed, followed by the treebank conversion algorithm. Finally, experiment results are reported, which show a very promising conversion accuracy with the proposed approach.

In the current task, we will assume that the new treebank will be compatible with an underlying target grammar of any appropriate form and a target tag set (including terminal and nonterminal symbols) associated with that grammar; since, otherwise, we could simply use the original treebank directly without doing any conversion. This assumption is reasonable since most natural language research laboratories who deal with syntactic level processing and those who need a treebank is supposed to have an underlying phrase structure grammars or rules for identifying appropriate constituents in the input text.

Task Definition for Treebank Conversion

Formally, the task for a treebank conversion algorithm is to map a source tree (generated from a source grammar or bracketed by hand) into its corresponding target tree that would be generated from a second grammar (hereinafter, the target grammar) without changing, vaguely speaking, its structures or semantics. The conversion must therefore satisfies several criteria so that the target tree could be reused in the target system. First of all, the target tree must be compatible with the second grammar. This means that the target tree must also be generatable from the second grammar. Secondary, the source tree and target tree must be 'similar' in a sense that their corresponding terminal symbols (parts of speech), nonterminal symbols (syntactic categories) and structures (production rules) preserve essentially similar categorial or structural information.

A simple model for such a conversion problem is shown in Figure 1, where $S$ is a sentence in the treebank, $G_1$ and $G_2$ are the grammars for the original treebank and the target system, respectively, $T_0^s$ is the manually proved tree for $S$ in the treebank, $T_i^s$ are all the possible ambiguous syntax trees for $S$ as generated by the target grammar $G_2$, and $T_0^t$ is the best target tree selected from $T_i^t$ based on a mapping score $Score(T_i^t|T_0^t)$ defined on the treebank tree and the ambiguous constructions. The "conversion" from $T_0^s$ to $T_0^t$ is actually done by a matching algorithm.

To ensure compatibility of the target trees with the target grammar, the sentences from which the source treebank was constructed are parsed by a parser (Parser II) using the target grammar. (It is also possible to enumerate all possible constructs via other apparatus. The parser here is just a characterization of such an apparatus.) All the possible target constructs for a sentence are then matched against the source tree, and the one that best matches the source tree is selected as the preferred conversion. In the above model, it is, of course, possible to incorporate any kind of preference mechanism in the parsing mechanism of Parser II to prevent the converter from enumerating all possible syntactic structures allowed by the target grammar. In fact, the original design of the conversion model is to hook a matching module to the end of any existing parsing mechanism, so that the ambiguous structures are matched against manually verified structure information in the source treebank and pick up the correct parse without human inspection.

To use the proposed model, a mapping metric is required for measuring the mapping preference between the source tree and the candidate target trees. Several frameworks for finding translation equivalents or translation units in machine translation, such as [Chang and Su 1993, Isabelle et al.1993] and other example-based MT approaches, might be used to select the preferred mapping. A general corpus-based statistics-oriented model for statistical transfer in machine translation in [Chang and Su 1993] is especially suitable for such a task. One can, in fact, model the treebank conversion problem as a (statistical) transfer problem in machine translation because both problems deal with the mapping between two structure representations of the same sentence. The difference is: the transfer problem deals with
sentences that are in two different languages while
the treebank conversion problem deals with only
one language. The mechanism used to find the
transfer units and transfer rules together with the
transfer score used in the above frameworks can
thus be used for treebank conversion with little
modification.

Matching Metric for Treebank
Conversion

The matching metric or matching score for tree-
bank conversion is much simpler than the trans-
fer score for the transfer task between two syntax
trees for two languages. The intuition is to assume
that: it is very likely that the tree representation
for a sentence in a particular language will have
essentially the same bracket representation, which
may possibly be associated with different (termi-
nal or nonterminal) symbols, when expressed in
another grammar. We thus use the number of
matching constituents in the source and target
trees as the matching score for converting from
one source tree to a target tree.

Figure 2: An Example for the Tree Matching
Metric

Take Figure 2 as an example. Node ‘9’ in the
source (left) tree contains Nodes ‘3’, ‘4’, ‘5’ as its
children; Node ‘h’ in the target (right) tree also has
Nodes ‘3’, ‘4’, ‘5’ as its children. We therefore add
a constant score of 1 to the matching score for this
tree pair. The same is true for Node ‘10’ and Node
‘i’. Since Node ‘7’ in the source tree and Node ‘f’ in
the target tree do not have any corresponding node
as their counterparts, they contribute nothing to
the matching preference. When there are
single productions, like the construct for Node ‘8’
and its sole child Node ‘6’, such constituents will be
regarded as the same entity. Therefore, the match
between Node ‘8’ (or Node ‘6’) and Node ‘g’ will be
assigned only one constant score of 1. This step
respects the reduction of such single production
rules into only one bracket. (For instance, \(X \rightarrow
Y \rightarrow a b c\) will have the bracket representation
of \([a b c]\), instead of \([[a b c]]\).) As a result, the
matching score for the example tree pair is 3.

To facilitate such matching operations and
matching score evaluation, the word indices of the
sentence for the source/target tree pair is perce-
lated upward (and recursively) to the tree nodes
by associating each nonterminal node with the
list of word indices, called an index list, acquired
by concatenating the word indices of its children.
(The index lists are shown near the nodes in Figure
2.) Two nonterminal nodes which have the
same index list form an aligned node pair; the
subtrees rooted at such aligned nonterminal nodes
and terminated with aligned nodes then consti-
tute the mapping units between the two trees.
The number of such matches thus represents a
simple matching score for the tree pair. The
index lists can be easily established by a depth-first
traversal of the tree. Furthermore, the existence of
one constituent which consists of terminal nodes
\((l, l + 1, \ldots, m)\) can be saved in a chart (a lower
triangular matrix), where chart\((l, m)\) records the
number of nodes whose terminal children are num-
bered from \(l\) to \(m\). By using a chart for a tree, all
nodes in a chain of single productions will cor-
respond to the same count for a particular chart
entry. A match in a source/target node pair will
correspond to a pair of nonzero cells in the charts;
the matching score then reduces to the number
of such pairs. We therefore have the following
treebank conversion algorithm based on the simple
matching metric described here.

The Baseline Treebank Conversion
Algorithm

With the highly simplified mapping model, we can
convert a tree in a treebank into another which
is compatible with the target grammar with the
following steps:

- 1. Parse the sentence of the source tree with a
parser of the target system based on the target
grammar.
- 2. For each ambiguous target tree produced
in step 1 and the source tree in the original
treebank, associate each terminal word with its
word index and associate each nonterminal node
with the concatenation of the word indices of its
children nodes. This can be done with a depth-
first traversal of the tree nodes.
- 3. For the trees of step 2, associate each tree
with a Chart (a lower triangular matrix), which
is initially set to zero in each matrix cell. Make
a traversal of all the tree nodes, say in the
depth-first order, and increment the number in
Chart\((l, m)\) by one each time a node with the
indices \((l, \ldots, m)\) is encountered.
• 4. For each chart of the candidate target trees, compare it with the chart of the source tree and associate a mapping score to the target tree by scanning the two charts. For each index range \((i, m)\), increment the score for the target tree by one if both the Chart\((i, m)\) entries for the source tree and the target tree are non-zero.

• 5. Select the target tree with the highest score as the converted target tree for the source tree. When there are ties, the first one encountered is selected.

In spite of its simplicity, the proposed algorithm achieves a very promising conversion accuracy as will be shown in the next section. Note that the parser and the grammar system of the target system is not restricted in any way; therefore, the annotated information to the target treebank can be anything inherent from the target system; the bracket information of the original treebank thus provides useful information for bootstrapping the corpus size and information contents of the target treebank.

Note also that we do not use any information other than the index lists (or equivalently the bracket information) in evaluating the matching metric. The algorithm is therefore surprisingly simple. Further generalization of the proposed conversion model, which uses more information such as the mapping preference for a source/target tag pair or mapping unit pair, can be formulated by following the general corpus-based statistics-oriented transfer model for machine translation in [Chang and Su 1993]. In [Chang and Su 1993], the transfer preference between two trees is measured in terms of a transfer score: 

\[
P(T'_{1,2} | T'_{0,2}) = \prod_{j=1}^{n} P(t'_{i,j} | t'_o) \]

where \(T'_{0,2}\) and \(T'_{1,2}\) are the source tree and the \(i^{th}\) possible target tree, which can be decomposed into pairs of transfer (i.e., mapping) units \((t'_{i,j}, t'_o)\) (local subtrees). The transfer pairs can be found by aligning the terminal and nonterminal nodes with the assistance of the index lists as described previously [Chang and Su 1993].

In fact, the current algorithm can be regarded as a highly simplified model of the above cited framework, in which the terminal words for the source tree and the target tree are identical and are implicitly aligned exactly 1-to-1; the mapping units are modeled by the pairs of aligned nodes; and the probabilistic mapping information is replaced with binary constant scores. Such assignment of constant scores eliminate the requirement for estimating the probabilities and the requirement of treebank corpora for training the mapping scores.

The following examples show a correctly matched instance and an erroneously matched one.

**INPUT:** Depending on the type of control used, it may or may not respond quickly enough to protect against spikes and faults. (Correct answer and selected output are #3.)

1. [[[Depending-on [the type] [of [control used]]] [if] [may-or-may-not respond [quickly enough] to [protect [against [spikes and faults]]]]].

2. [[[Depending-on [the type] [of [control used]]] [if] [may-or-may-not respond [quickly enough] to [protect [against [spikes and faults]]]]].

3. [[[Depending-on [the type] [of [control used]]] [if] [may-or-may-not respond [quickly enough] to [protect [against [spikes and faults]]]]].

4. [[[Depending-on [the type] [of [control used]]] [if] [may-or-may-not respond [quickly enough] to [protect [against [spikes and faults]]]]].

**INPUT:** The PC's power-supply is capable of absorbing most noise, spikes, and faults.

1. [[[The PC's] power-supply] [is] [capable of [absorbing [most noise] , [spikes] , and faults]]].

2. [[[The PC's] power-supply] [is] [capable of [absorbing [most noise] , [spikes] , and faults]]].

3. [[[The PC's] power-supply] [is] [capable of [absorbing [most noise] , [spikes] , and faults]]].

4. [[[The PC's] power-supply] [is] [capable of [absorbing [most noise] , [spikes] , and faults]]].

5. [[[The PC's] power-supply] [is] [capable of [absorbing [most noise] , [spikes] , and faults]]].

6. [[[The PC's] power-supply] [is] [capable of [absorbing [most noise] , [spikes] , and faults]]].

**Experiment Results**

The performance of the proposed approach is evaluated on a treebank consisting of 8,867 English sentences (about 140,634 words in total) from the statistical database of the BehaviorTran (formerly the ArchTran [Su and Chang 1990, Chen et al.1991]) MT system. The English sentences are acquired from technical manuals for computers and electronic instruments. Two versions of the grammar used in this MT system are used in the experiment. The basic parameters for these two grammars are shown in Table 1, where \(G_1\) and \(G_2\) are the source and target grammars, \#P is the number of production rules (i.e., context-free phrase structure rules), \(\#\Sigma\) is the number of terminal symbols, \(\#\mathcal{N}\) is the number of nonterminal symbols and \(\#A\) is the number of semantic constraints or actions associated with the phrase structure rules.
The target grammar shown here is an improved version of the source grammar. It has a wider coverage, a little more ambiguous structures, and shorter processing time than the old one. The major changes are the representations of some constructs in addition to the changes in the parts of speech and nonterminal syntactic categories. For instance, the hierarchy is revised in the new revision to better handle the 'gaps' in relative clauses, and the tag set is modified to better characterize the classification of the various words. Such modifications are likely to occur between any two grammar systems, which adopt different tag sets, syntactic structures and semantic constraints. Therefore, it, in some sense, characterizes the typical operations which may be applied across two different systems.

Each sentence produces about 16.9 ambiguous trees on the average under the new grammar $G_2$. The source trees contain brackets corresponding to the fully parsed structures of the input sentences; however, multiple brackets which correspond to "single productions" are eliminated to only one bracket. For instance, a structure like $X \rightarrow Y \rightarrow Z \rightarrow ab$ will reduce to the equivalent bracket structure of $\{a b\}$. This reduction process is implied in the proposed algorithm since we increment the matching score by one whenever the two charts have the same word index range which contains non-zero counts; we do not care how large the counts are. This also implies that the target tree brackets are also reduced by the same process. The reduced brackets, on which the matching is based, in the source and target trees are thus less detailed than their fully parsed trees structures.

After feeding the 8,867 sentences into the parser and selecting the closest match among the target trees against the source trees in the treebank, it is found that a total of 115 sentences do not produce any legal syntactic structures under the new grammar, 158 sentences produce no correct structure in terms of the new grammar (including 12 sentences which produce unique yet erroneous parses), and 1,546 sentences produce unambiguously, one correct analysis. The former two cases, which is mostly attributed to the coverage of the target grammar, indicate the degree of incompatibility between the two grammars. The latter case will not indicate any difference between any tree conversion algorithms. Therefore, they are not considered in evaluating the performance of the conversion procedure.

For the remaining 7,048 sentences, 6,799 source trees are correctly mapped to their counterparts in the new grammar; only 249 trees are incorrectly mapped; therefore, excluding unambiguously parsed sentences, a conversion accuracy of 96.46% (6,799/7,048) is obtained. The results appear to be very promising with this simple algorithm. It also shows that the bracket information and the mapping metric do provide very useful information for treebank conversion.

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Percentage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag Error</td>
<td>19.6</td>
</tr>
<tr>
<td>Conjunction Error</td>
<td>51.4</td>
</tr>
<tr>
<td>Attachment Error</td>
<td>23.6</td>
</tr>
<tr>
<td>Drastic Structural Error</td>
<td>5.4</td>
</tr>
</tbody>
</table>

Table 2: Error Type Analysis

A sampling of 146 trees from the 249 incorrectly mapped trees reveals the error types of mismatch as tabulated in Table 2. The error introduced by inappropriate tags is about 19.6%. Structural error, on the other hand, is about 80.4%, which can be further divided into errors due to: incorrect mapping of conjunct elements and/or appositions (51.4%), incorrect attachment patterns between heads and modifiers (23.6%) and drastic structure variation (5.4%). Note that tagging error is far less than structural error; furthermore, two trees with drastically different structures are rarely matched. A closer look shows that 2.72% (185/6799) of the correctly mapped trees and 31.73% (79/249) of the incorrectly mapped trees have the same scores as the other competing trees; they are selected because they are the first candidate. The current solution to tie, therefore, tends to introduce incorrectly mapped trees. A better way may be required to avoid the chance of tie. For instance, we may increment different scores for different types of matches or different syntactic categories.

The above experiment results confirm our previous assumption that even the simplest skeletal structure information, like brackets, provides significant information for selecting the most likely structure in another grammar system. This fact partially explains why the simple conversion algorithm achieves a satisfactory conversion accuracy.

Note that a mapping metric against the source tree may introduce systematic bias that prefers the
source structures rather than the target grammar. This phenomenon could prevent the improvement of the new grammar from being reflected in the converted corpus if the new grammar is a revision of the old one. Attachment and conjunction scopes, which may vary from system to system, are more likely to suffer from such a bias as shown in the above experiment results. A wise way to incorporate preference form the target grammar may be necessary if such bias introduces a significant fraction of errors. Such preference information may include mapping preference acquired from other extra information or by using other more complicated models.

From the low error rate of the overall performance, however, it seems that we need not be too pessimistic with such a bias since most major constituents, like noun phrases and verb phrases, recognized by different persons are in agreement to a large extent. It is probably also true even for persons across different laboratories.

Since the conversion rate is probably high enough, it is possible simply to regard errors in the converted treebank as noise in probabilistic frameworks, which use the converted treebank for parameter training. In these cases, further manual inspection is not essential and the conversion is basically automatic. This situation is particularly true if the original source treebank had been manually verified, since we can at least make sure that the target trees are legal, even though not preferred. If serious work is necessary to avoid error accumulation in the treebank, say in the grammar revision process, it is suggested only to check a few high-score candidates to save checking time. If, in addition, the major differences of the two grammars are known, the checking time could be further reduced by only applying detailed checking to the trees that have relevant structure changes.

Of course, there are many factors which may affect the performance of the proposed approach among different grammar systems. In particular, we did not use the information between the mapping of the parts of speech (terminal symbols) and the syntactic categories (nonterminal symbols), which may be useful in the cases where the mapping is applied to two trees with the same bracket representation. In our future research, we will try to convert large treebanks, such as the Penn Treebank, available in the community into our grammar system, and make use of more information on the parts of speech and syntactic categories so that a robust conversion algorithm can be developed.

Concluding Remarks

It is important to be able to share treebanks among different research organizations. The significance for developing a treebank conversion technique includes at least: (1) corpus sharing among different grammar systems and research organizations; (2) automatic system corpus updating between two major revisions; (3) corpus bootstrapping with a large and cheaply tagged treebank; (4) avoidance of duplicated investment in the construction and maintenance of proprietary corpora; (5) promoting continuous evolution of an old grammar system for a corpus-based system.

In this paper, we therefore proposed a simple approach for converting one treebank into another across two different grammar systems using a simple conversion metric based on the bracket information of the original treebank. The simple metric, which evaluates the number of bracket matching, turns out to be effective in preserving the structures across two different grammars. The experiment results show that, excluding unambiguous sentences, the conversion accuracy, in terms of the number of correctly converted trees, achieves as high as 96.4%.

References


