In a Machine Translation System (MTS), the number of possible analyses for a given sentence is largely due to the ambiguous characteristics of the source language. In this paper, a mechanism, called "Score Function", is proposed for measuring the "quality" of the ambiguous syntax trees such that the one that best fits interpretation by human is selected. It is featured by incorporating the objectiveness of the probability theory and the subjective expertise of linguists. The underlying uncertainty that is fundamental to linguistic knowledge is also allowed to be incorporated into this system. This feature provides an easy resolution to select the best syntax tree and provides some strategic advantages for scored parsing. The linguists can also be relieved of the necessity to describe the language in strictly "correct" linguistic rules, which, if not impossible, is a very hard task.

Motivation

In a Machine Translation System (MTS), where the underlying grammar is large, there are many sources which may cause the system to become highly ambiguous. The system must choose a better syntax tree among all the possible ones to reduce the load of the post-editor. Some systems attack this problem by blocking the rule ordering in a descending order of their relative frequency, following the parsing paths in a depth-first manner, and selecting the first syntax tree successfully parsed as the desired one. However, such an approach is not a locally preferred static scoring mechanism of preference for all possible states allowed by the underlying grammar. This approach has been widely accepted and is useful in eliminating the unnecessary trials. However, there are times when legal paths are blocked inadvertently by condition checks. Therefore, the system must be tuned to prevent the parser from trying all possible rules in a systematic manner. A better solution is to adopt the "Truncation Strategy" (proposed by [Su 87a, 87b] for MT system) to restrict the number of parsing paths to be tried according to the relative preference of all the possible paths. The scoring mechanism of preference for the truncation strategy is called the "Score Function". It bears similar similarity to the select-by-preference at lexicon, syntax and semantics levels, and score can be computed during parsing or after parsing. In this paper, we propose an approach to the semantic and syntactic aspects of the score function.

Criteria for Score Function

In order to define a reasonable score function, it is essential to set up some criteria first. Eight basic criteria are listed here.

1. The score function should reflect the absolute degree of preference of two ambiguous (sub)trees as well as their relative preferences.
2. A good score function should be applicable either locally to a subtree or globally to a complete tree.
3. The score function should be compositional. This means the score of a tree should be directly evaluated from the scores of its constituent subtrees.
4. Relative rule application frequency should be included in the score function. The rule that is used most frequently should receive a higher preference.
5. The score function should also include the semantic information embedded in the sentence, so that the semantic preference can be involved in the score function. (Since our present translation unit is a single sentence, no discourse information need to be included)
6. The implementation of the score function should not be too complicated. In our case, it should be practical for a large-scale MT system.
7. The database for score computation should be easy to build and easy to maintain.
8. The preferenr order of ambiguous trees assigned by the score function should match those assigned by the human. In addition, the way the scores are given had better match the way that people give their preference to the ambiguous trees, i.e., how people recognize the true meaning of a given sentence from several different interpretations)

Keeping these criteria in mind, we define a score function as follows. The score function for a subtree $X_o$ with derivation sequence $D$ of $X_0(i,j) = \rightarrow_{P} X_1(i_1,j_1), X_2(j_1+1,j_2), ..., X_n(j_n-1+j,n),$ is:

$$\text{SCORE}(X_o) = \text{SCsyn}([XI,..,X_n]) \ast \text{SCsem}([XI,KI(XI),KC(XI))] \ast \text{SCsyn}(X_n,KT(X_n),KC(X_n)))$$

In the above, $X_0(i,j)$ is a subtree made up of terminals $X_1$ to $X_n$; $i$ to $j$ are the word index in the sentence; and $\text{SCORE}$ is the score of the subtree $X_0$. $\text{SCsyn}$ is the unweighted syntax score. $\text{SCsem}$ is the semantic weighting. $\text{KI}$ is defined as the knowledge about the inherent properties of the nodes. And $\text{KC}$ is the well-formedness condition, either syntactic or semantic, of the $X_1$ under the given syntactic construction. To decrease the computational complexity, we can convert this multiplication equation into an addition equation with logarithmic entries.
In order to obtain the score without excessive computation and complicated algorithm, the probability model is probably one of the most common and promising approach. Under this approach, the preference measurement in a scoring mechanism can be seen as a probability assignment. The best syntax tree should be the one with highest preference probability assigned to it. This probability model can be divided into two parts. One is the syntactic score model, which is SCsyn, and the other is the semantic score model, which is SCsem. The syntactic score model uses the syntax probability as the base to generate an unweighted syntactic score for each syntax tree. The semantic score model then supplements the unweighted score with weights derived from the semantic knowledge. Incorporation of semantic information is essential for a good score function because pure syntax probability can only provide partial information for sentence preference.

**Syntactic Score Model**

For a syntax tree given below, we define a phrase level as a sequence of terminals and nonterminals that are being reduced at a single step of "derivation, or reduction sequence". The following example shows the reduction sequence of a bottom-up parsing. The sequence is indicated by the time series t1 ... t7.

**Example 2**:

<table>
<thead>
<tr>
<th>t7</th>
<th>t6</th>
<th>t5</th>
<th>t4</th>
<th>t3</th>
<th>t2</th>
<th>t1</th>
</tr>
</thead>
<tbody>
<tr>
<td>X6</td>
<td>X5</td>
<td>X4</td>
<td>X3</td>
<td>X4</td>
<td>X4</td>
<td>X3</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>D</td>
<td>B</td>
<td>E</td>
</tr>
<tr>
<td>{w1,w2}</td>
<td>{w3,w4}</td>
<td>{w5,w6}</td>
<td>{w7,w8}</td>
<td>{w3,w4}</td>
<td>{w1,w2}</td>
<td>{w3,w4}</td>
</tr>
</tbody>
</table>

The unweighted score for this tree A is modeled as the following conditional probability.

\[
\text{SCsyn}(A) = \log(S(X_{t_1}, X_{t_2}, \ldots X_{t_n})) = \log(\text{SCsyn}) + \log(\text{SCsem})
\]

In the above, it satisfies all eight criteria we had set initially and it is a good systematic approach for assigning references to a set of ambiguous trees.

**Semantic Score Model**

The weight-assigning process of the semantic score can be seen as an expert task where the linguist is giving the syntax tree a diagnosis. The linguist will assign a preference to a tree according to some linguistic knowledge or heuristic rules. Very often these linguistic rules are not very precise. Therefore, a good semantic score model must allow this type of inexact knowledge. The problem of incorporation into building a rule-based expert system that can calculate semantic scores (weightings) and handle inexact knowledge encountered during calculation. We propose a model similar to the CF model (certainty factor model) in MYCIN [Buch 85] system. It has a knowledge-rule base where each rule has a certainty factor based on the degree of belief and disbelief. The confirmation of a hypothesis then is calculated from the applicable rules and from other pieces of evidence. The CF of a hypothesis is then accumulated gradually with each additional evidence.

Each tree node will have a well-formedness factor (WFF), which is the CF for the derivation of this node, associated with it. As the knowledge, which may come in the word sense, syntactic category, attribute, etc., of leaf nodes percolates up along the syntax structure, every node's WFF will be calculated according to the rules stored in the knowledge rule-base. This WFF then becomes the semantic score of the subtree.

**Simulation Result**

A simulation, based on 1408 source sentences, was conducted to test the syntactic score model. The probability assigned to the entries, e.g. P(E[w2,w3]), in the SCsyn equation is estimated with the relative frequency of these entries. That is, we approximate P(E[w2,w3]) by the ratio of the number of events (E[w2,w3]) in the database and the number of events (w2,w3). Several tests are conducted to check the influence of the context on the probability assignment. These tests include LL, LR, RR, LLR, LRR, LLRR and LLLR. Table 1 is some of the result of the simulation using sentences in the database as the test inputs. The number of entries in the table is the number of different conditional probability, e.g. P(E[w2,w3]), in the database. Each entry is assigned a probability according to its usage frequency as we explained earlier. The preference of a tree is then the sum of these probability weights estimated from these entries. If the size of database is not large enough then these probability weights are reduced at a single step of "derivation, or reduction sequence".

\[
\text{WFF}(X_0) = \text{SCsem}(X_1, K_1(X_1), K_2(X_1), \ldots, X_n, K_1(X_n), K_2(X_n))
\]

where derivation sequence D : X = X0 \Rightarrow X1, \ldots Xn.
In this paper, we will investigate two databases, one having 1468 source sentences and the other having 820 sentences. If the simulation result from different base is close then we may assume that the database size is large enough. Comparing the results from these two databases, it is apparent that the size is adequate for the present simulation. Furthermore, it is also apparent that a context-sensitive scoring function must be adopted for a good preference estimation.

Conclusions can be drawn from this simulation result. First, we should adopt three constituents in calculating the probability. The reason is that although the result of LLRR case is better than that of LRR case, the size of entries required by LLRR is considerably greater. Second, approximately 85% of syntax trees is accurately selected with only syntactic information available. Therefore, if we want to improve this result further we must include the semantic information.

**Conclusion and Perspective**

In a Machine Translation System, to reduce the load of the post-editor we must select the best syntax tree from a set of ambiguous trees and pass it to the post-editor. There are systems that rely on a set of ordered grammar rules or on a set of restrictive condition checks to achieve this. Unfortunately, they all have some drawbacks: one being too uncertain and the other being too restrictive. In this paper we have proposed a score mechanism for the truncation strategy to perform disambiguation during parsing. The score function, with the adoption of three context symbols, gives the power of context-sensitive grammar to an efficient context-free parser. From our simulation, the score function with just syntactic information will achieve an accuracy rate of 85%. In the near future when the semantic information is included, this accuracy rate is expected to increase. Currently, two databases, one for unweighted score computation and the other for linguistic rule base (for weighting assignment), are under the development at the BTC R&D center. After completion they will be incorporated into the truncation parsing algorithms for our third generation parser.

**Acknowledgment**

We would like to express our deepest appreciation to Wen-Hueh Li and Hsue-Hueh Hsu for their work on the simulations, to the whole linguistic group at BTC R&D center for their work on the database, and Mei-Hui Su for her editing. Special thanks are given to Behavior Tech. Computer Co. for their full financial support of this project.

**References**


