The total number of binary trees for a sentence of length n is given by the Catalan number C_{n-1}, where $C_n = (2n)!/((n+1)!\,n!)$. Thus while a sentence of 4 words has 5 binary trees, a sentence of 8 words has already 429 binary trees, and a sentence of 10 words has 4862 binary trees. Of course, we can represent the set of binary trees of a string in polynomial time and space by means of a chart, resulting in a chart-like parse forest if we also include pointers. But if we want to extract rules or subtrees from these binary trees -- as in DOP -- we need to unpack the parse forest. And since the total number of binary trees that can be assigned to the WSJ10 is almost 12 million, it is doubtful whether we can apply the unrestricted U-DOP model to such a corpus.

However, for longer sentences the binary trees are highly redundant. In these larger trees, there are many rules like $X \rightarrow XX$ which bear little information. To make parsing with U-DOP possible we therefore applied a simple heuristic which takes random samples from the binary trees for sentences ≥ 7 words before they are fed to the DOP parser. These samples were taken from the distribution of all binary trees by randomly choosing nodes and their expansions from the chart-like parse forests of the sentences (which effectively favors trees with more frequent subtrees). For sentences of 7 words we randomly sample 60% of the trees, and for sentences of 8, 9 and 10 words we sample respectively 30%, 15% and 7.5% of the trees. In this way, the set of remaining binary trees contains 8.23 * 10^5 trees, which we will refer to as the *binary tree-set*. Although it can happen that the correct tree is deleted for some sentence in the binary tree-set, there is enough redundancy in the tree-set such that either the correct binary tree can be generated by other subtrees or that a remaining tree only minimally differs from the correct tree. Of course, we may expect better results if *all* binary trees are kept, but this involves enormous computational resources which will be postponed to future research.

Step 2: Convert the trees into a PCFG-reduction of DOP

The underlying idea of U-DOP is to take all subtrees from the binary tree-set to compute the most probable tree for each sentence. Subtrees from the trees in figure 1 include for example the subtrees in figure 2 (where we again added words for readability). Note that U-DOP takes into account both contiguous and non-contiguous substrings.

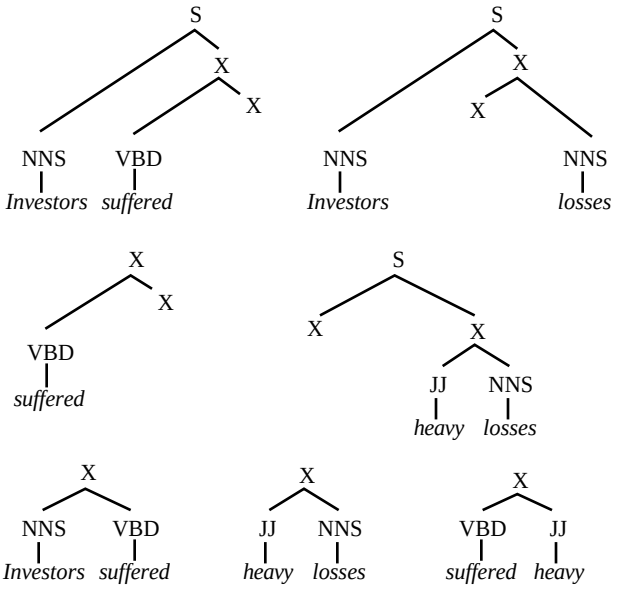

Figure 2. Some subtrees from the binary trees for NNS VBD JJ NNS given in figure 1

As in the supervised DOP approach (Bod 1998), U-DOP parses a sentence by combining corpus-subtrees from the binary tree-set by means of a leftmost node substitution operation, indicated as ∘. The probability of a parse tree is computed by summing up the probabilities of all derivations producing it, while the probability of a derivation is computed by multiplying the (smoothed) relative frequencies of its subtrees. That is, the probability of a subtree t is taken as the number of occurrences of t in the binary tree-set, $|\,t\,|$, divided by the total number of occurrences of all subtrees t' with the same root label as t. Let $r(t)$ return the root label of t:

$$P(t) = \frac{|\,t\,|}{\sum_{t':\, r(t')=r(t)} |\,t'\,|}$$

The subtree probabilities are smoothed by applying simple Good-Turing to the subtree distribution (see Bod 1998: 85-87). The probability of a derivation $t_1 \circ ... \circ t_n$ is computed by the product of the probabilities of its subtrees t_i:

$$P(t_1 \circ ... \circ t_n) = \prod_i P(t_i)$$

Since there may be distinct derivations that generate the same parse tree, the probability of a parse tree T