Better method.

The input in many of the algorithms needed for NLP have trees and sequences as input kernel methods have been useful in search algorithms example of such kernel is tree kernel which we have already discussed the tree kernel was a convolution kernel the kernel function counts the number of common subtrees in any two trees to find similarity between them hear the proposed algorithm is tree sequence kernel(TSK) which adopts the structure of sequence of subtrees tsk basically combines the tree kernel and sequence kernel features to produce a more robust tree-based kernel sequence control is integrated with tree kernel.

Tree Sequence Structure

Tree sequence structure can be either contiguous or noncontiguous bracket with the gaps between adjacent subtrees

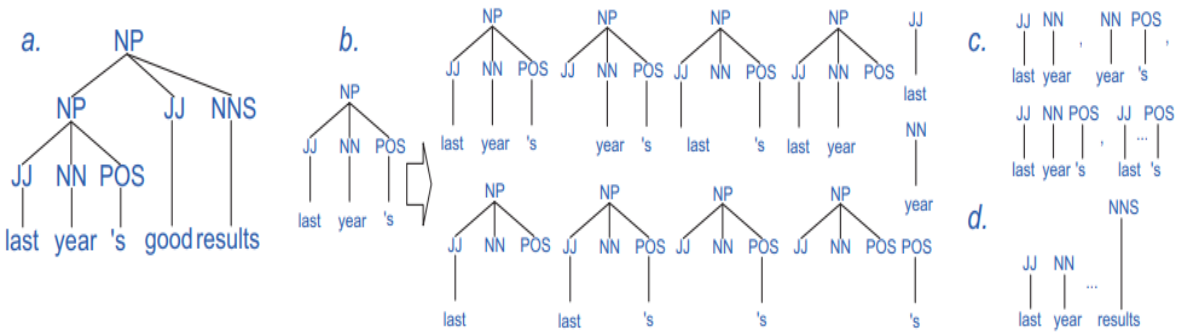


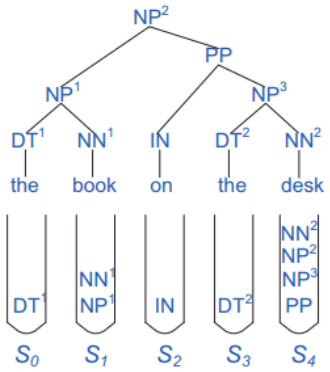
Figure 1: Illustration for Tree Sequence Structure

.

the tree kernel captures only syntactic information while with sequence structure many information can be captured like ok in this case many subtrees related to “last year” can be captured by the tree kernel but the meaningful part like ok “ last year” cannot be individually captured. TSK also help to match last structures which is difficult to do in case of the kernels as TSK uses many disconnected parts and when many parts are made new patterns and learn by tsk for example “ last year” and “results” can be together using non contiguous subtrees which forms a meaningful pattern.

Tree Sequence Kernel

In order to evaluate TSK, we integrate the algorithms of SSK and tree kernel by means of certain modifications.To apply the SSK algorithm, we first transform the parse tree structure into a valid Set Sequence structure. Given a parse tree T with span length L, the node set Si for all the indices $0 \leq i<L$ can be constructed by delivering all the nodes n to set Si, when the end index of the span covered by n equals i.



In the above figure,we construct the Set Sequence for a parse tree with span [0, 4] by sending the internal nodes to the set S0,...,S4. For example, since NN1 ends with index 1, we put NN1 in S1.

The general idea of the TSK evaluation is to match the subtrees in a subtree sequence from left to right and from top to the bottom. Given a subtree sequence to be matched, the key to achieve this goal is the root node rooted at each subtree. When the root node n of a subtree t is matched, we need to vertically move downwards and match the entire subtree structure t. In addition, we also need to horizontally move to the right side of t and match the root node n' of the subtreet t' that is adjacent to t. A gap is allowed between the subtree t and the subtree t' .Consequently, the subtree sequence matching problem is transformed into a problem of matching the root node sequences and the single tree structures.
To implement this idea in dynamic programming, the horizontal evaluation is achieved by incurring SSK on the node sequence set of the given parse tree pair. In other words, any matched node in the node set will be considered as the root of the subtree to be matched. At the same time, the vertical evaluation is achieved by using the tree kernel function Λ(., .) within the root-matched subtrees.

Tree Sequence Kernel (TSK), combines the advantages of both sequence kernel and tree kernel. The key characteristic of the proposed kernel is that the captured structure features are enlarged from the structure of a single tree to the structure of multiple trees (tree sequence). Two kernel functions with different penalty factors are presented. Both TSKs can be efficiently evaluated within O(m|NT |·|NT' |), where m is the maximal number of subtrees allowed in a matched tree sequence and NT is the number of tree nodes. Specifically, the structure of tree sequence more facilitates the task that focuses on disconnected constituents modeling, i.e. Relation Extraction

Ayush Badola
B20MT012