A LEARNING-BASED DEPENDENCY TO CONSTITUENCY CONVERSION ALGORITHM FOR THE TURKISH LANGUAGE

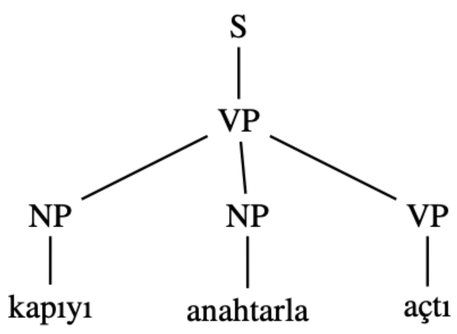
Büşra Marşan,[1] Oğuz Kerem Yıldız,[2] Aslı Kuzgun,[2] Neslihan Cesur,[2] Arife Betül Yenice,[2] Ezgi Sanıyar,[2] Oğuzhan Kuyrukçu,[2] Bilge Nas Arıcan,[2] and Olcay Taner Yıldız.[3]

[1]Starlang Yazılım Danışmanlık, [2]Ahmet Keleşoğlu High School, and [3]Ozyegin University, İstanbul, Turkey.

ABSTRACT

This study aims to create the very first dependency-to-constituency conversion algorithm optimised for Turkish language. For this purpose, a state-of-the-art morphologic analyser and a feature-based machine learning model was used. To enhance the performance of the conversion algorithm, bootstrap aggregating meta-algorithm was integrated. While creating the conversation algorithm, typological properties of Turkish were carefully considered. A comprehensive and manually annotated UD-style dependency treebank was the input, and constituency trees were the output of the conversion algorithm. A team of linguists manually annotated a set of constituency trees. These manually annotated trees were used as the gold standard to assess the performance of the algorithm. The conversion process yielded more than 8000 constituency trees whose UD-style dependency trees are also available on GitHub. In addition to its contribution to Turkish treebank resources, this study also offers a viable and easy-to-implement conversion algorithm that can be used to generate new constituency treebanks and training data for NLP resources like constituency parsers.

FIGURE 1



"[S/he] opened the door with a key"

In this study, we opted for an algorithm that produces flatter phrase structures compared to the X-Bar theory representations and Penn TreeBank trees by allowing ternary branching and bypassing bar-levels. To make trees flat, we followed the strategy also employed by Hajic et al., 1998:

- Each constituent X has only one parent, XP.
- There is no X' level or its equivalent (such as XX).

CONVERSION PROCESS

The conversion algorithm follows a bottom-up approach. Dependency tags and headedness are referred to correctly convert each dependency relation. For each edge in the dependency tree, a sub-tree is generated. Then these sub-branches are conjoined using the dependency relations, headedness, and POS tags indicated in the dependency trees. POS tags determine the phrasal tags.

There are two different oracles created for the purposes of this study, one of which refers to a set of predefined heuristics to conjoin subtrees. These predefined set of heuristics are based on the dependency relations, POS tags and headedness. These rules guide sub-tree merger process. After every terminal node is assigned to a sub-tree in accordance with it headed or dependent, the basic oracle refers to the rules and merges them by one of the three operations: Left, Right and Merge. The operations applied at this stage and their order are critical to create the correct tree output.

BASIC ORACLE

Basic oracle is exclusively rule-based. The main challenge for the basic oracle is grouping the subtrees in the correct order. Thus, a hierarchy of subtree appendage is created to avoid doing the merger operations in the wrong order.

COMPOUND >AUX >DET >AMOD >

NUMMOD >CASE >CCOMP >NEG

The hierarchy of subtree appendage

CLASSIFIER ORACLE

The defining characteristic of the classifier oracle is that it merges the subtrees in accordance with their class information.

This class information is provided by the predetermined classes based on the number of dependents each head has. There are four classes for heads: 1-dependent, 2-dependent, 3-dependent and 4-dependent. The constraints and features regarding these classes are determined making use of dependency tags, POS tags and syntactic particularities of Turkish.

The algorithm follows a bottom-up processing style: It starts from the terminal nodes and builds the constituency structure up until it reaches the root, which is S. Similar to the basic oracle, classifier oracle refers to the morphological analyser to pull basic tags and create phrasal tags like ADJP, NP and VP. Then, dependency relationships between the leaves of the subtree are referred to

EVALUATION PROCESS

After conversion process was concluded, a set of linguists manually annotated constituency trees of the dependency structures used as the input of the conversion algorithm. These manually annotated constituency trees served as the gold standard in parser evaluation process. In order to assess the performance of the conversion algorithm, the evaluation algorithm starts comparing output trees and gold standard trees from the top node. It compares the phrase below each node of the conversion output and gold standard until it reaches the individual leaves. Anything other than exact matches are penalized.

TABLE 1

	Precision	Recall	F-Score
Basic Oracle	88.51	89.90	89.20
Classifier Oracle	89.34	90.06	89.70

RESULTS

The overall performance of the rule based algorithm is satisfactory with an F-score of 89.20 but it is not as successful as the machine learning algorithm which has an F-score of 89.70. The hierarchy of subtree appendage and no bar level representations are the main contributors of these F-score, precision and recall values. Illustrating particularities like nominal predicates, the constituency trees created by the algorithm reflect the syntactic features of Turkish.

REFERENCES

Hajic, Jan et al. (1998). *Core natural language processing technology applicable to multiple languages–Workshop'98*. Tech. rep. Technical report, Johns Hopkins Univ.