Generalized Transition-based Dependency Parsing via Control Parameters

Introduction:
The motivation of this work is to build a unified transition based parser framework in order to compare different parsers and obtain new parsers by changing control parameters.

Transition-based parsers are important parsing algorithms due to their high accuracy and linear runtime. In addition, some of them are also able to handle non-projectivity. There are some fundamental parts that are common between different transition-based parsers; parsers have some states/configuration and some way to transition from one state to another. Additionally, they have so-called buffer(unprocessed tokens) and stack(tokens being operated). In order to achieve a generalized framework for different parsers this work introduces a set of active tokens, which allows us to create and remove arcs. Moreover, they also have some control parameters such as arc distance that helps us to get many common systems e.g arc-standard.

Methodology:
A unified framework is obtained by adding the specific parts, which are configurable, to the parts that are common between different parsers. One of the aspects most parsers differ from one another is making transitions from one state to another. So, the transitions are broken down into some basic operations and these basic operations are then used to make up some specific transition. This allows us to model different transitions.

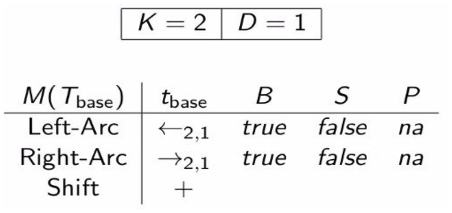
The buffer U has its top most token on left while the stack has its top most token on the right. Figure 2 illustrates four basic operations named LEFT - ARC ($\leftarrow_{i,j}$), RIGHT - ARC ($\rightarrow_{i,j}$), REDUCE ($-_i$), and SHIFT (+) which can be used model transitions for different parsers.

There are two types of control parameters that instantiates different parsers. These parameters are mentioned in the table below. Note that all transitions do not require all of these control parameters e.g Reduce do not need B and S.

Global parameters	Transition Control Parameters
Active token capacity K: determines the size of stack in the start.	**Bottom-up:** B ∈ {true, false} If we say left-arc is bottom-up it indicates that $\leftarrow_{i,j}$ is immediately followed by a reduce $-_i$.
Max arc distance D: Arc distance between tokens in stack. Can draw arc iff $\|i − j\| \leq D$.	**Arc-Shift:** S ∈ true, false. If we say left-arc is arc-shift it indicates that $\leftarrow_{i,j}$ is immediately followed by shift +.
	Periphery: P ∈ {left, right, na}. Left means that the transition must include the left token from the active tokens.

So, in order to build a generalized framework we need states, basic transitions, global parameters, a multi-set of valid transitions and transition-control parameters for valid transitions.

Transition System Instantiations:

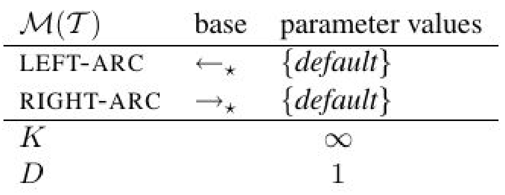


$K = 2$	$D = 1$

$M(T_{\text{base}})$	t_{base}	B	S	P
Left-Arc	$\leftarrow_{2,1}$	*true*	*false*	*na*
Right-Arc	$\rightarrow_{2,1}$	*true*	*false*	*na*
Shift	$+$			

Arc-standard[1]: Active token capacity is set to 2 and max arc distance is 1. As the B is set to true, we have to remove the dependent(2nd element) from the stack when we do the left arc.

Easy First: The number of active tokens is equal to the length of the sentence. There is no need of shift as we already have all the tokens in the stack. The * denotes the set of indexes {(i, j)|i ≤|O|, j ≥ 1, i = j + 1} which means that when doing a left-arc we can do it using various adjacent tokens in the stack.



$\mathcal{M}(\mathcal{T})$	base	parameter values
LEFT-ARC	$\leftarrow_{\star}$	$\{default\}$
RIGHT-ARC	$\rightarrow_{\star}$	$\{default\}$
K		∞
D		1

Novel Transition Systems:

Bounded Capacity Easy-first: Arc-standard is faster and easier to train whereas easy-first is more accurate. The main difference between them is the active token capacity K. This parser is obtained by using a K between 2 and sentence length.

Non-projective Easy-first: A non-projective parser can be obtained by using Max arc distance greater than 1. Changing the D and K(can be fixed) simultaneously will give us this system.

Experimental and results:
Structured perceptron with early-update is used to assign scores for transitions . UAS and LAS are reported excluding the punctuation marks.

Table 1 illustrates that there is not much difference in performance of arc-standard and arc-eager and easy-first has better performance than both of them .However, all three systems were not able to beat the neural-network-based systems.

In the case of bounded capacity easy-first, there is a tradeoff between accuracy and speed when choosing between arc-standard and easy-first. Accuracy increases when active token capacity changes from 2 to 4. However, due to the addition of additional shift transitions in this framework, the accuracy drops as we increase the K further. Table 2 shows the performance of the non-projective easy-first as the max arc distance increases. It can be seen that increasing D helps to model non-projectivity as suggested by the NPS.

The main difference between this system and GR&N13 is that non-projectivity is not included in the GR&N13 and they compose the elementary transition to make up complex transitions as opposed to constraints parameters that are used in this framework.

Conclusion: A framework is proposed to model different traisition-based systems whose results are more comparable. Performance of arc-standard can be increased by using K>2 and non-projectivity can also be modeled by using D>1. Additionally, it can help to design novel systems. However, selecting the right parameters can be computationally expensive.

[1] Figure from the slides.