A Transition-Based System for Joint Lexical and Syntactic Analysis

Mathieu Constant, Joakim Nivre

A Transition-Based System for Joint Lexical and Syntactic Analysis

Matthieu Constant
Université Paris-Est, LIGM (UMR 8049)
Alpage, INRIA, Université Paris Diderot
Paris, France
Matthieu.Constant@u-pem.fr

Joakim Nivre
Uppsala University
Dept. of Linguistics and Philology
Uppsala, Sweden
joakim.nivre@lingfil.uu.se

Abstract

We present a transition-based system that jointly predicts the syntactic structure and lexical units of a sentence by building two structures over the input words: a syntactic dependency tree and a forest of lexical units including multiword expressions (MWEs). This combined representation allows us to capture both the syntactic and semantic structure of MWEs, which in turn enables deeper downstream semantic analysis, especially for semi-compositional MWEs. The proposed system extends the arc-standard transition system for dependency parsing with transitions for building complex lexical units. Experiments on two different data sets show that the approach significantly improves MWE identification accuracy (and sometimes syntactic accuracy) compared to existing joint approaches.

1 Introduction

Multiword expressions (MWEs) are sequences of words that form non-compositional semantic units. Their identification is crucial for semantic analysis, which is traditionally based on the principle of compositionality. For instance, the meaning of *cut the mustard* cannot be compositionally derived from the meaning of its elements and the expression therefore has to be treated as a single unit. Since Sag et al. (2002), MWEs have attracted growing attention in the NLP community.

Identifying MWEs in running text is challenging for several reasons (Baldwin and Kim, 2010; Seretan, 2011; Ramisch, 2015). First, MWEs encompass very diverse linguistic phenomena, such as complex grammatical words (*in spite of*, *because of*), nominal compounds (*light house*), non-canonical prepositional phrases (*above board*), verbal idiomatic expressions (*burn the midnight oil*), light verb constructions (*have a bath*), multiword names (*New York*), and so on. They can also be discontiguous in the sense that the sequence can include intervening elements (*John **pulled** Mary's **leg***). They may also vary in their morphological forms (*hot dog, hot dogs*), in their lexical elements (*lose one's mind/head*), and in their syntactic structure (*he took a step*, *the step he took*).

The semantic processing of MWEs is further complicated by the fact that there exists a continuum between entirely non-compositional expressions (*piece of cake*) and almost free expressions (*traffic light*). Many MWEs are indeed semi-compositional. For example, the compound *white wine* denotes a type of wine, but the color of the wine is not white, so the expression is only partially transparent. In the light verb construction *take a nap*, *nap* keeps its usual meaning but the meaning of the verb *take* is bleached. In addition, the noun can be compositionally modified as in *take a long nap*. Such cases show that MWEs may be decomposable and partially analyzable, which implies the need for predicting their internal structure in order to compute their meaning.

From a syntactic point of view, MWEs often have a regular structure and do not need special syntactic annotation. Some MWEs have an irregular structure, such as *by and large* which on the surface is a coordination of a preposition and an adjective. They are syntactically as well as semantically non-compositional and cannot be represented with standard syntactic structures, as stated in Candito and Constant (2014). Many of these irregular MWEs are complex grammatical words like *because of*, *in spite of* and *in order to* – fixed (grammatical) MWEs in the sense of Sag et al. (2002). In some treebanks, these are annotated using special structures and labels because they can-

not be modified or decomposed. We hereafter use the term *fixed MWE* to refer to either fixed or irregular MWEs.

In this paper, we present a novel representation that allows both regular and irregular MWEs to be adequately represented without compromising the syntactic representation. We then show how this representation can be processed using a transition-based system that is a mild extension of a standard dependency parser. This system takes as input a sentence consisting of a sequence of tokens and predicts its syntactic dependency structure as well as its lexical units (including MWEs). The resulting structure combines two factorized substructures: (i) a standard tree representing the syntactic dependencies between the lexical elements of the sentence and (ii) a forest of lexical trees including MWEs identified in the sentence. Each MWE is represented by a constituency-like tree, which permits complex lexical units like MWE embeddings (for example, [[*Los Angeles*] *Lakers*], I will [*take a* [*rain check*]]). The syntactic and lexical structures are factorized in the sense that they share lexical elements: both tokens and fixed MWEs.

The proposed parsing model is an extension of a classical arc-standard parser, integrating specific transitions for MWE detection. In order to deal with the two linguistic dimensions separately, it uses two stacks (instead of one). It is synchronized by using a single buffer, in order to handle the factorization of the two structures. It also includes different hard constraints on the system in order to reduce ambiguities artificially created by the addition of new transitions. To the best of our knowledge, this system is the first transition-based parser that includes a specific mechanism for handling MWEs in two dimensions. Previous related research has usually proposed either pipeline approaches with MWE identification performed either before or after dependency parsing (Kong et al., 2014; Vincze et al., 2013a) or workaround joint solutions using off-the-shelf parsers trained on dependency treebanks where MWEs are annotated by specific subtrees (Nivre and Nilsson, 2004; Eryiğit et al., 2011; Vincze et al., 2013b; Candito and Constant, 2014; Nasr et al., 2015).

2 Syntactic and Lexical Representations

A standard dependency tree represents syntactic structure by establishing binary syntactic relations between words. This is an adequate representation of both syntactic and lexical structure on the assumption that words and lexical units are in a one-to-one correspondence. However, as argued in the introduction, this assumption is broken by the existence of MWEs, and we therefore need to distinguish lexical units as distinct from words.

In the new representation, each lexical unit – whether a single word or an MWE – is associated with a *lexical node*, which has linguistic attributes such as surface form, lemma, part-of-speech tag and morphological features. With an obvious reuse of terminology from context-free grammar, lexical nodes corresponding to MWEs are said to be *non-terminal*, because they have other lexical nodes as children, while lexical nodes corresponding to single words are *terminal* (and do not have any children).

Some lexical nodes are also *syntactic nodes*, that is, nodes of the syntactic dependency tree. These nodes are either non-terminal nodes corresponding to (complete) fixed MWEs or terminal nodes corresponding to words that do not belong to a fixed MWE. Syntactic nodes are connected into a tree structure by binary, asymmetric dependency relations pointing from a *head* node to a *dependent* node.

Figure 1 shows the representation of the sentence *the prime minister made a few good decisions*. It contains three non-terminal lexical nodes: one fixed MWE (*a few*), one contiguous non-fixed MWE (*prime minister*) and one discontiguous non-fixed MWE (*made decisions*). Of these, only the first is also a syntactic node. Note that, for reasons of clarity, we have suppressed the lexical children of the fixed MWE in Figure 1. (The non-terminal node corresponding to *a few* has the lexical children *a* and *few*.) For the same reason, we are not showing the linguistic attributes of lexical nodes. For example, the node *made-decisions* has the following set of features: surface-form='made decisions', lemma='make decision', POS='V'. Non-fixed MWEs have regular syntax and their components might have some autonomy. For example, in the light verb construction *made-decisions*, the noun *decisions* is modified by the adjective *good* that is not an element of the MWE.

The proposed representation of fixed MWEs is an alternative to using special dependency labels as has often been the case in the past (Nivre and Nilsson, 2004; Eryiğit et al., 2011). In addition

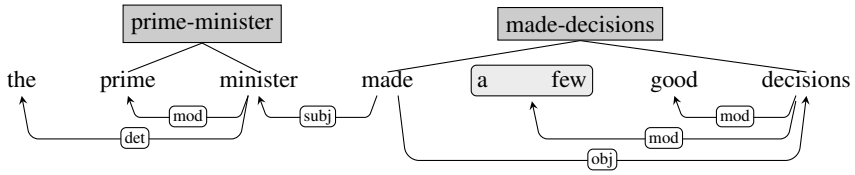

Figure 1: Representation of syntactic and lexical structure.

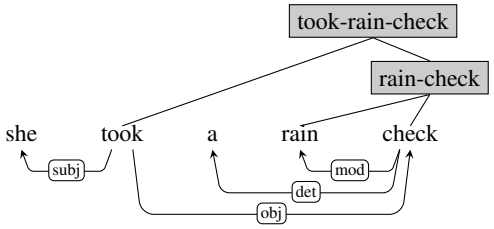

Figure 2: Lexical structure of embedded MWEs.

to special labels, MWEs are then represented as a flat subtree of the syntactic tree. The root of the subtree is the left-most or right-most element of the MWE, and all the other elements are attached to this root with dependencies having special labels. Despite the special labels, these subtrees look like ordinary dependency structures and may confuse a syntactic parser. In our representation, fixed MWEs are instead represented by nodes that are atomic with respect to syntactic structure (but complex with respect to lexical structure), which makes it easier to store linguistic attributes that belong to the fixed MWE and cannot be derived from its components. The new representation also allows us to represent the hierarchical structure of embedded MWEs. Figure 2 provides an analysis of *she took a rain check* that includes such an embedding. The lexical node *took-rain-check* corresponds to a light verb construction where the object is a compound noun that keeps its semantic interpretation whereas the verb has a neutral value. One of its children is the lexical node *rain-check* corresponding to a compound noun.

Let us now define the representation formally. Given a sentence $x = x_1, \ldots, x_n$ consisting of n tokens, the syntactic and lexical representation is a quadruple (V, F, N, A), where

1. V is the set of *terminal nodes*, corresponding one-to-one to the tokens $x_1, \ldots, x_n$,
2. F is a set of n-ary trees on V, with each tree corresponding to a fixed MWE and the root labeled with the part-of-speech tag for the MWE,
3. N is a set of n-ary trees on F, with each tree corresponding to a non-fixed MWE and the root labeled with the part-of-speech tag for the MWE,
4. A is a set of labeled dependency arcs defining a tree over F.

This is a generalization of the standard definition of a dependency tree (see, for example, Kübler et al. (2009)), where the dependency structure is defined over an intermediate layer of lexical nodes (F) instead of directly on the terminal nodes (V), with an additional layer of non-fixed MWEs added on top. To exemplify the definition, here are the formal structures corresponding to the representation visualized in Figure 1.

$$\begin{array}{lcl} V & = & \{1,2,3,4,5,6,7,8\} \\ F & = & \{1,2,3,4,\mathrm{A}(5,6),7,8\} \\ N & = & \{1,\mathrm{N}(2,3),\mathrm{V}(4,8),\mathrm{A}(5,6),7\} \\ A & = & \{(3,\mathrm{det},1),(3,\mathrm{mod},2), \\ & & (4,\mathrm{subj},3),(4,\mathrm{obj},8), \\ & & (8,\mathrm{mod},\mathrm{A}(5,6)),(8,\mathrm{mod},7)\} \end{array}$$

Terminal nodes are represented by integers corresponding to token positions, while trees are represented by n-ary terms $t(c_1, \ldots, c_n)$, where t is a part-of-speech tag and $c_1, \ldots, c_n$ are the subtrees immediately dominated by the root of the tree. The total set of lexical nodes is $L = V \cup F \cup N$, where V contains the terminal and $(F \cup N) - V$ the non-terminal lexical nodes. The set of syntactic nodes is simply F.

It is worth noting that the representation imposes some limitations on what MWEs can be represented. In particular, we can only represent overlapping MWEs if they are cases of embedding, that is, cases where one MWE is properly contained in the other. For example, in an example like *she took a walk then a bath*, it might be argued that *took* should be part of two lexical units: *took-walk* and *took-bath*. This cannot currently be represented. By contrast, we can accommodate cases where two lexical units are interleaved, as in the French example *il prend un cachet et demi*, with the two units *prend-cachet* and *un-et-demi*, which occur in the crossed pattern A1 B1 A2 B2. However, while these cases can be represented in

principle, the parsing model we propose will not be capable of processing them.

Finally, it is worth noting that, although our representation in general allows lexical nodes with arbitrary branching factor for flat MWEs, it is often convenient for parsing to assume that all trees are binary (Crabbé, 2014). For the rest of the paper, we therefore assume that non-binary trees are always transformed into equivalent binary trees using either right or left binarization. Such transformations add intermediate temporary nodes that are only used for internal processing.

3 Transition-Based Model

A transition-based parser is based on three components: a transition system for mapping sentences to their representation, a model for scoring different transition sequences (derivations), and a search algorithm for finding the highest scoring transition sequence for a given input sentence. Following Nivre (2008), we define a transition system as a quadruple $S = (C, T, c_s, C_t)$ where:

1. C is a set of configurations,
2. T is a set of transitions, each of which is a partial function $t : C \rightarrow C$,
3. c_s is an initialization function that maps each input sentence x to an initial configuration $c_s(x) \in C$,
4. $C_t \subseteq C$ is a set of terminal configurations.

A *transition sequence* for a sentence x is a sequence of configurations $C_{0,m} = c_0, \ldots, c_m$ such that $c_0 = c_s(x)$, $c_m \in C_t$, and for every c_i $(0 \leq i < m)$ there is some transition $t \in T$ such that $t(c_i) = c_{i+1}$. Every transition sequence defines a representation for the input sentence.

Training a transition-based parser means training the model for scoring transition sequences. This requires an *oracle* that determines what is an optimal transition sequence given an input sentence and the correct output representation (as given by treebank). Static oracles define a single unique transition sequence for each input-output pair. Dynamic oracles allow more than one optimal transition sequence and can also score non-optimal sequences (Goldberg and Nivre, 2013). Once a scoring model has been trained, parsing is usually performed as best-first search under this model, using greedy search or beam search.

3.1 Arc-Standard Dependency Parsing

Our starting point is the arc-standard transition system for dependency parsing first defined in Nivre (2004) and represented schematically in Figure 3. A configuration in this system consists of a triple $c = (\sigma, \beta, A)$, where σ is a stack containing partially processed nodes, β is a buffer containing remaining input nodes, and A is a set of dependency arcs. The initialization function maps $x = x_1, \ldots, x_n$ to $c_s(x) = ([\,], [1, \ldots, n], \{\,\})$, and the set C_t of terminal configurations contains any configuration of the form $c = ([i], [\,], A)$. The dependency tree defined by such a terminal configuration is $(\{1, \ldots, n\}, A)$. There are three possible transitions:

- **Shift** takes the first node in the buffer and pushes it onto the stack.
- **Right-Arc(**k**)** adds a dependency arc (i, k, j) to A, where j is the first and i the second element of the stack, and removes j from the stack.
- **Left-Arc(**k**)** adds a dependency arc (j, k, i) to A, where j is the first and i the second element of the stack, and removes i from the stack.

A transition sequence in the arc-standard system builds a projective dependency tree over the set of terminal nodes in V. The tree is built bottom-up by attaching dependents to their head and removing them from the stack until only the root of the tree remains on the stack.

3.2 Joint Syntactic and Lexical Analysis

To perform joint syntactic and lexical analysis we need to be able to build structure in two parallel dimensions: the syntactic dimension, represented by a dependency tree, and the lexical dimension, represented by a forest of (binary) trees. The two dimensions share the token-level representation, as well as the level of fixed MWEs, but the syntactic tree and the non-fixed MWEs are independent.

We extend the parser configuration to use two stacks, one for each dimension, but only one buffer. In addition, we need not only a set of dependency arcs, but also a set of lexical units. A configuration in the new system therefore consists of a quintuple $c = (\sigma_1, \sigma_2, \beta, A, L)$, where σ_1 and σ_2 are stacks containing partially processed nodes (which may now be complex MWEs), β is a buffer containing remaining input nodes (which

Initial:	$([\,], [0, \ldots, n], \{\,\})$		
Terminal:	$([i], [\,], A)$		
Shift:	$(\sigma, i\|\beta, A)$	$\Rightarrow$	$(\sigma\|i, \beta, A)$
Right-Arc(k):	$(\sigma\|i\|j, \beta, A)$	$\Rightarrow$	$(\sigma\|i, \beta, A \cup \{(i, k, j)\})$
Left-Arc(k):	$(\sigma\|i\|j, \beta, A)$	$\Rightarrow$	$(\sigma\|j, \beta, A \cup \{(j, k, i)\})$

Figure 3: Arc-standard transition system.

Initial:	$([\,], [\,], [0, \ldots, n], \{\,\}, \{\,\})$		
Terminal:	$([x], [\,], [\,], A, L)$		
Shift:	$(\sigma_1, \sigma_2, i\|\beta, A, L)$	$\Rightarrow$	$(\sigma_1\|i, \sigma_2\|i, \beta, A, L)$
Right-Arc(k):	$(\sigma_1\|x\|y, \sigma_2, \beta, A, L)$	$\Rightarrow$	$(\sigma_1\|x, \sigma_2, \beta, A \cup \{(x, k, y)\}, L)$
Left-Arc(k):	$(\sigma_1\|x\|y, \sigma_2, \beta, A, L)$	$\Rightarrow$	$(\sigma_1\|y, \sigma_2, \beta, A \cup \{(y, k, x)\}, L)$
Merge$_F$(t):	$(\sigma_1\|x\|y, \sigma_2\|x\|y, \beta, A, L)$	$\Rightarrow$	$(\sigma_1\|t(x, y), \sigma_2\|t(x, y), \beta, A, L)$
Merge$_N$(t):	$(\sigma_1, \sigma_2\|x\|y, \beta, A, L)$	$\Rightarrow$	$(\sigma_1, \sigma_2\|t(x, y), \beta, A, L)$
Complete:	$(\sigma_1, \sigma_2\|x, \beta, A, L)$	$\Rightarrow$	$(\sigma_1, \sigma_2, \beta, A, L \cup \{x\})$

Figure 4: Transition system for joint syntactic and lexical analysis.

are always tokens), A is a set of dependency arcs, and L is a set of lexical units (tokens or MWEs). The initialization function maps $x = x_1, \ldots, x_n$ to $c_s(x) = ([\,], [\,], [1, \ldots, n], \{\,\}, \{\,\})$, and the set C_t of terminal configurations contains any configuration of the form $c = ([x], [\,], [\,], A, L)$. The dependency tree defined by such a terminal configuration is (F, A), and the set of lexical units is $V \cup L$. Note that the set F of syntactic nodes is not explicitly represented in the configuration but is implicitly defined by A. Similarly, the set L only contains $F \cup N$.

The new transition system is shown in Figure 4. There are now six possible transitions:

- **Shift** takes the first node in the buffer and pushes it onto both stacks. This guarantees that the two dimensions are synchronized at the token level.
- **Right-Arc(k)** adds a dependency arc (x, k, y) to A, where y is the first and x the second element of the syntactic stack (σ_1), and removes y from this stack. It does not affect the lexical stack (σ_2).[1]
- **Left-Arc(k)** adds a dependency arc (y, k, x) to A, where y is the first and x the second element of the syntactic stack (σ_1), and removes x from this stack. Like **Right-Arc(k)**, it does not affect the lexical stack (σ_2).
- **Merge$_F$(t)** applies in a configuration where the two top elements x and y are identical on both stacks and combines these elements into a tree $t(x, y)$ representing a fixed MWE with part-of-speech tag t. Since it operates on both stacks, the new element will be a syntactic node as well as a lexical node.
- **Merge$_N$(t)** combines the two top elements x and y on the lexical stack (σ_2) into a tree $t(x, y)$ representing a non-fixed MWE with part-of-speech tag t. Since it only operates on the lexical stack, the new element will not be a syntactic node.
- **Complete** moves the top element x on the lexical stack (σ_2) to L, making it a final lexical unit in the output representation. Note that x can be a simple token, a fixed MWE (created on both stacks), or a non-fixed MWE (created only on the lexical stack).

A transition sequence in the new system derives the set of lexical nodes and simultaneously builds a projective dependency tree over the set of syntactic nodes. By way of example, Figure 5 shows the transition sequence for the example in Figure 1.

3.3 Implicit Completion

The system presented above has one potential drawback: it needs a separate Complete transition for every lexical unit, even in the default case

[1] We use the variables x and y, instead of i and j, because the stack elements can now be complex lexical units as well as simple tokens.

Transition		Configuration
		$([\,],[\,],[1,2,3,4,5,6,7,8],A_0=\{\,\},L_0=\{\,\})$
Shift	$\Rightarrow$	$([1],[1],[2,3,4,5,6,7,8],A_0,L_0)$
Complete	$\Rightarrow$	$([1],[\,],[2,3,4,5,6,7,8],A_0,L_1=L_0\cup\{1\})$
Shift	$\Rightarrow$	$([1,2],[2],[3,4,5,6,7,8],A_0,L_1)$
Shift	$\Rightarrow$	$([1,2,3],[2,3],[4,5,6,7,8],A_0,L_1)$
Merge$_N$(N)	$\Rightarrow$	$([1,2,3],[\mathrm{N}(2,3)],[4,5,6,7,8],A_1,L_1)$
Complete	$\Rightarrow$	$([1,2,3],[\,],[4,5,6,7,8],A_0,L_2=L_1\cup\{\mathrm{N}(2,3)\})$
Left-Arc(mod)	$\Rightarrow$	$([1,3],[\,],[4,5,6,7,8],A_1=A_0\cup\{(3,\mathrm{mod},2)\},L_2)$
Left-Arc(det)	$\Rightarrow$	$([3],[\,],[4,5,6,7,8],A_2=A_1\cup\{(3,\mathrm{det},1)\},L_2)$
Shift	$\Rightarrow$	$([3,4],[4],[5,6,7,8],A_2,L_2)$
Left-Arc(subj)	$\Rightarrow$	$([4],[4],[5,6,7,8],A_3=A_2\cup\{(4,\mathrm{subj},3)\},L_2)$
Shift	$\Rightarrow$	$([4,5],[4,5],[6,7,8],A_3,L_2)$
Shift	$\Rightarrow$	$([4,5,6],[4,5,6],[7,8],A_3,L_2)$
Merge$_F$(A)	$\Rightarrow$	$([4,\mathrm{A}(5,6)],[4,\mathrm{A}(5,6)],[7,8],A_3,L_2)$
Complete	$\Rightarrow$	$([4,\mathrm{A}(5,6)],[4],[7,8],A_3,L_3=L_2\cup\{\mathrm{A}(5,6)\})$
Shift	$\Rightarrow$	$([4,\mathrm{A}(5,6),7],[4,7],[8],A_3,L_3)$
Complete	$\Rightarrow$	$([4,\mathrm{A}(5,6),7],[4],[8],A_3,L_4=L_3\cup\{7\})$
Shift	$\Rightarrow$	$([4,\mathrm{A}(5,6),7,8],[4,8],[\,],A_3,L_4)$
Left-Arc(mod)	$\Rightarrow$	$([4,\mathrm{A}(5,6),8],[4,8],[\,],A_4=A_3\cup\{(8,\mathrm{mod},7)\},L_4)$
Left-Arc(mod)	$\Rightarrow$	$([4,8],[4,8],[\,],A_5=A_4\cup\{(8,\mathrm{mod},\mathrm{A}(5,6))\},L_4)$
Merge$_N$(V)	$\Rightarrow$	$([4,8],[\mathrm{V}(4,8)],[\,],A_5,L_4)$
Complete	$\Rightarrow$	$([4,8],[\,],[\,],A_5,L_5=L_4\cup\{\mathrm{V}(4,8)\})$
Right-Arc(obj)	$\Rightarrow$	$([4],[\,],[\,],A_6=A_5\cup\{(4,\mathrm{obj},8)\},L_5)$

Figure 5: Transition sequence for joint syntactic and lexical analysis.

when a lexical unit is just a token. This makes sequences much longer and increases the inherent ambiguity. One way to deal with this problem is to make the Complete transition implicit and deterministic, so that it is not scored by the model (or predicted by a classifier in the case of deterministic parsing) but is performed as a side effect of the Right-Arc and Left-Arc transitions. Every time we apply one of these transitions, we check whether the dependent x of the new arc is part of a unit y on the lexical stack satisfying one of the following conditions: (i) $x = y$; (ii) x is a lexical child of y and every lexical node z in y either has a syntactic head in A or is the root of the dependency tree. If (i) or (ii) is satisfied, we move y from the lexical stack to the set L of lexical units as a side effect of the arc transition.

4 Experiments

This section provides experimental results obtained with a simple implementation of our system using a greedy search parsing algorithm and a linear model trained with an averaged perceptron with shuffled examples and a static oracle. More precisely, the static oracle is defined using the following transition priorities: Merge$_F$ > Merge$_N$ > Complete > LeftArc > RightArc > Shift. At each state of the training phase, the static oracle selects the valid transition that has the higher priority.

We evaluated the two variants of the system, namely *Explicit* and *Implicit*, with explicit and implicit completion, respectively. They were compared against the joint approach proposed in Candito and Constant (2014) that we applied to an arc-standard parser, instead of a graph-based parser. The parser is trained on a treebank where MWE status and grammatical function are concatenated in arc labels. We consider it as the *Baseline*. We used classical transition-based parsing features consisting of patterns combining linguistic attributes of nodes on the stacks and the buffer, as well as processed subtrees and transition history. We can note that the joint systems do not contain features sharing elements of both stacks. Preliminary tuning experiments did not show gains when using such features.

We also compared these systems against weaker ones, obtained by disabling some transitions and using one stack only. Two systems, namely *Syntactic-baseline* and *Syntactic* only predict the syntactic nodes and the dependency structure by using respectively a baseline parser and our system where neither the lexical stack nor the Merge$_N$ and Complete transitions are used. The latter one is an implementation of the proposal in Nivre (2014). Two systems are devoted only to the lexical layer: *Lexical* only recognizes the lexical units (only the lexical stack and the Merge$_N$ and Complete transitions are activated); *Fixed* only identifies the fixed expressions.

Corpus	EWT		FTB		
	Train	Test	Train	Dev	Test
# sent.	3,312	500	14,759	1,235	2,541
# tokens	48,408	7,171	443,113	38,820	75,216
# MWEs	2,996	401	23,556	2,119	4,043
# fixed	-	-	10,987	925	1,992

Table 1: Dataset statistics.

We also implemented pipeline systems where: (i) fixed MWEs are identified by applying only the *Fixed* system; (ii) elements of predicted MWEs are merged into single tokens; (iii) the retokenized text is parsed using the *Baseline* or *Implicit* systems trained on a dataset where fixed MWEs consist of single tokens.

We carried out our experiments on two different datasets annotating both the syntactic structure and the MWEs: the French Treebank [FTB] (Abeillé et al., 2003) and the STREUSLE corpus (Schneider et al., 2014b) combined with the English Web Treebank [EWT] (Bies et al., 2012). They are commonly used for evaluating the most recent MWE-aware dependency parsers and supervised MWE identification systems. Concerning the FTB, we used the dependency version developed in Candito and Constant (2014) derived from the SPMRL shared task version (Seddah et al., 2013). Fixed and non-fixed MWEs are distinguished, but are limited to contiguous ones only. The STREUSLE corpus (Schneider et al., 2014b) corresponds to a subpart of the English Web Treebank (EWT). It consists of reviews and is comprehensively annotated in contiguous and discontiguous MWEs. Fixed and non-fixed expressions are not distinguished though the distinction between non-compositional and collocational MWEs is made. This implies that the Merge$_F$ transition is not used on this dataset. Practically, we used the LTH converter (Johansson and Nugues, 2007) to obtain the dependency version of the EWT constituent version. We also used the predicted linguistic attributes used in Constant and Le Roux (2015) and in Constant et al. (2016). Both datasets include predicted POS tags, lemmas and morphology, as well as features computed from compound dictionary lookup. None of them is entirely satisfying with respect to our model, but they allow us to evaluate the feasibility of the approach. Statistics on the two datasets are provided in Table 1.

Results are provided in Table 2 for French and in Table 3 for English. In order to evaluate the syntactic layer, we used classical UAS and LAS metrics. Before evaluation, merged units were automatically decomposed in the form of flat subtrees using specific arcs as in Seddah et al. (2013), so all systems can be evaluated and compared at the token level. MWE identification is evaluated with the F-score of the MWE segmentation, namely *MWE* for all MWEs and *FMWE* for fixed MWEs only. An MWE segment corresponds to the set of its component positions in the input token sequence.

First, results show that our joint system consistently and significantly outperforms the baseline in terms of MWE identification on both datasets. The merge transitions play a key role. In terms of syntax, the *Explicit* system does not have any positive impact (on par or degraded scores), whereas the *Implicit* system allows us to obtain slightly better results on French and a significant improvement on English. The very good performances on English might be explained by the fact that it contains a non-negligeable set of discontiguous MWEs which complicates the prediction of explicit Complete transitions.

When compared with weaker systems, we can see that the addition of the lexical layer helps improve the prediction of the syntactic layer, which confirms results on symbolic parsing (Wehrli, 2014). The syntactic layer does not seem to impact the lexical layer prediction: we observe comparable results. This might be due to the fact that syntax is helpful for long-distance discontiguity only, which does not appear in our datasets (the English dataset contains MWEs with small gaps). Another explanation could also be that syntactic parsing accuracy is rather low due to the use of a simple greedy algorithm. Developing more advanced transition-based parsing methods like beam-search may help improve both syntactic parsing accuracy and MWE identification. When comparing joint systems with pipeline ones, we can see that preidentifying fixed MWEs seems to help MWE identification whereas syntactic parsing accuracy tends to be slightly lower. One hypothesis could be that Merge$_F$ transitions may confuse the prediction of Merge$_N$ transitions.

When compared with existing state-of-the-art systems, we can see that the proposed systems achieve MWE identification scores that are comparable with the pipeline and joint approaches used in Candito and Constant (2014) with a graph-

	DEV				TEST			
System	UAS	LAS	MWE	FMWE	UAS	LAS	MWE	FMWE
Baseline	86.28	83.67	77.2	83.2	84.85	82.67	75.5	81.9
Explicit	86.36	83.77	79.7	86.0	84.98	82.79	**79.3**	**84.8**
Implicit	**86.61**	**84.10**	**80.0**	**86.2**	**85.04**	**82.93**	78.4	84.3
Syntactic only -Baseline	86.31	83.69	-	83.5	84.89	82.70	-	82.0
Syntactic only	86.39	83.77	-	85.0	85.02	82.84	-	83.8
Lexical only	-	-	80.0	-	-	-	79.5	-
Fixed only	-	-	-	85.7	-	-	-	85.7
Pipeline (Fixed only → Baseline)	85.33	83.29	80.6	85.7	84.86	82.86	80.4	85.7
Pipeline (Fixed only → Implicit)	85.49	83.50	**81.8**	85.7	84.84	82.89	**81.1**	**85.7**
graph-based (Candito and Constant, 2014)	89.7	87.5	77.6	85.4	89.21	86.92	77.0	85.1
CRF+graph-based (Candito and Constant, 2014)	89.8	87.4	79.0	85.0	86.97	89.24	78.6	86.3
CRF (SPMRL) (Le Roux et al., 2014)	-	-	82.4	-	-	-	80.5	-

Table 2: Results on the FTB. To reduce bias due to training with shuffled examples, scores are averages of 3 different training/parsing runs.

	TRAIN Cross-validation			TEST		
System	UAS	LAS	MWE	UAS	LAS	MWE
Baseline	86.16	81.76	49.6	86.31	82.02	46.8
Explicit	86.25	82.09	52.9	86.05	81.68	**53.4**
Implicit	**86.81**	**82.68**	**55.0**	**87.05**	**83.14**	51.6
Syntactic only	86.35	82.23	-	86.41	82.20	-
Lexical only	-	-	54.5	-	-	53.6
(Schneider et al., 2014a)	-	-	-	-	-	53.85

Table 3: Results on the reviews part of the English Web Treebank, via cross-validation on the training set with 8 splits, and simple validation on the test set.

based parser for French, and the base sequence tagger using a perceptron model with rich MWE-dedicated features of Schneider et al. (2014a) for English. It reaches lower scores than the best simple CRF-based MWE tagging system of Le Roux et al. (2014). These scores are obtained on the SPMRL shared task version, though they are not entirely comparable with our system as they do not distinguish fixed from non-fixed MWEs.

5 Related work

The present paper proposes a new representation for lexical and syntactic analysis in the framework of syntactic dependency parsing. Most existing MWE-aware dependency treebanks represent an MWE as a flat subtree of the syntactic tree with special labels, like in the UD treebanks (Nivre et al., 2016) or in the SPMRL shared task (Seddah et al., 2013), or in other individual treebanks (Nivre and Nilsson, 2004; Eryiğit et al., 2011). Such representation enables MWE discontinuity, but the internal syntactic structure is not annotated. Candito and Constant (2014) proposed a representation where the irregular and regular MWEs are distinguished: irregular MWEs are integrated in the syntactic tree as above; regular MWEs are annotated in their component attributes while their internal structure is annotated in the syntactic tree. The Prague Dependency Treebank (Bejček et al., 2013) has several interconnected annotation layers: morphological (*m*-layer), syntactic (*a*-layer) and semantic (*t*-layer). All these layers are trees that are interconnected. MWEs are annotated on the *t*-layer and are linked to an MWE lexicon (Bejček and Straňák, 2010). Constant and Le Roux (2015) proposed a dependency representation of lexical segmentation allowing annotations of deeper phenomena like MWE nesting. More details on MWE-aware treebanks (including constituent ones) can be found in Rosén et al. (2015).

Statistical MWE-aware dependency parsing has received a growing interest since Nivre and Nilsson (2004). The main challenge resides in finding the best orchestration strategy. Past research has explored either pipeline or joint approaches. Pipeline strategies consist in positioning the MWE recognition either before or after the parser itself, as in Nivre and Nilsson (2004), Eryiğit et al. (2011), Constant et al. (2013), and Kong et al. (2014) for pre-identification and as in Vincze et al. (2013a) for post-identification. Joint strategies have mainly consisted in using off-the-shelf

parsers and integrating MWE annotation in the syntactic structure, so that MWE identification is blind for the parser (Nivre and Nilsson, 2004; Eryiğit et al., 2011; Seddah et al., 2013; Vincze et al., 2013b; Candito and Constant, 2014; Nasr et al., 2015).

Our system includes a special treatment of MWEs using specific transitions in a classical transition-based system, in line with the proposal of Nivre (2014). Constant et al. (2016) also proposed a two-dimensional representation in the form of dependency trees anchored by the same words. The annotation of fixed MWEs is redundant on both dimensions, while they are shared in our representation. They propose, along with this representation, an adaptation of an easy-first parser able to predict both dimensions. Contrary to our system, there are no special mechanisms for treating MWEs.

The use of multiple stacks to capture partly independent dimensions is inspired by the multiplanar dependency parser of Gómez-Rodríguez and Nivre (2013). Our parsing strategy for (hierarchical) MWEs is very similar to the deterministic constituency parsing method of Crabbé (2014).

6 Conclusion

This paper proposes a transition-based system that extends a classical arc-standard parser to handle both lexical and syntactic analysis. It is based on a new representation having two linguistic layers sharing lexical nodes. Experimental results show that MWE identification is greatly improved with respect to the mainstream joint approach. This can be a useful starting point for several lines of research: implementing more advanced transition-based techniques (beam search, dynamic oracles, deep learning); extending other classical transition systems like arc-eager and hybrid as well as handling non-projectivity.

Acknowledgments

The authors would like to thank Marie Candito for her fruitful inputs. This work has been partly funded by the French Agence Nationale pour la Recherche, through the PARSEME-FR project (ANR-14-CERA-0001). This work has also been supported in part by the PARSEME European COST Action (IC1207).

References

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a treebank for French. In Anne Abeillé, editor, *Treebanks*. Kluwer, Dordrecht.

Timothy Baldwin and Su Nam Kim. 2010. Multiword Expressions. In Nitin Indurkhya and Fred J. Damerau, editors, *Handbook of Natural Language Processing*, pages 267–292. CRC Press, Taylor and Francis Group, Boca Raton, FL, USA, 2 edition.

Eduard Bejček and Pavel Straňák. 2010. Annotation of Multiword Expressions in the Prague Dependency Treebank. *Language Resources and Evaluation*, 44(1-2):7–21.

Eduard Bejček, Eva Hajičová, Jan Hajič, Pavlína Jínová, Václava Kettnerová, Veronika Kolářová, Marie Mikulová, Jiří Mírovský, Anna Nedoluzhko, Jarmila Panevová, Lucie Poláková, Magda Ševčíková, Jan Štěpánek, and Šárka Zikánová. 2013. Prague Dependency Treebank 3.0.

Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012. English web treebank.

Marie Candito and Matthieu Constant. 2014. Strategies for Contiguous Multiword Expression Analysis and Dependency Parsing. In *ACL 14 - The 52nd Annual Meeting of the Association for Computational Linguistics*, Baltimore, United States, June. ACL.

Matthieu Constant and Joseph Le Roux. 2015. Dependency Representations for Lexical Segmentation. In *6th Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2015)*, Bilbao, Spain, July.

Matthieu Constant, Marie Candito, and Djamé Seddah. 2013. The LIGM-Alpage Architecture for the SPMRL 2013 Shared Task: Multiword Expression Analysis and Dependency Parsing. In *Fourth Workshop on Statistical Parsing of Morphologically Rich Languages*, pages 46–52, Seattle, United States, October.

Matthieu Constant, Joseph Le Roux, and Nadi Tomeh. 2016. Deep lexical segmentation and syntactic parsing in the easy-first dependency framework. In *Proceeedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2016)*.

Benoît Crabbé. 2014. An LR-inspired generalized lexicalized phrase structure parser. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 541–552.

Gülşen Eryiğit, Tugay İlbay, and Ozan Arkan Can. 2011. Multiword Expressions in Statistical Dependency Parsing. In *Proceedings of the Second Workshop on Statistical Parsing of Morphologically Rich*

Languages, SPMRL '11, pages 45–55, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yoav Goldberg and Joakim Nivre. 2013. Training Deterministic Parsers with Non-Deterministic Oracles. *TACL*, 1:403–414.

Carlos Gómez-Rodríguez and Joakim Nivre. 2013. Divisible Transition Systems and Multiplanar Dependency Parsing. *Comput. Linguist.*, 39(4):799–845, December.

Richard Johansson and Pierre Nugues. 2007. Extended Constituent-to-dependency Conversion for English. In *Proceedings of NODALIDA 2007*, pages 105–112, Tartu, Estonia, May 25-26.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. 2014. Dependency parsing for tweets. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar, October. Association for Computational Linguistics.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Morgan and Claypool.

Joseph Le Roux, Antoine Rozenknop, and Matthieu Constant. 2014. Syntactic Parsing and Compound Recognition via Dual Decomposition: Application to French. In *COLING*.

Alexis Nasr, Carlos Ramisch, José Deulofeu, and André Valli. 2015. Joint Dependency Parsing and Multiword Expression Tokenization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1116–1126.

Joakim Nivre and Jens Nilsson. 2004. Multiword units in syntactic parsing. *Proceedings of Methodologies and Evaluation of Multiword Units in Real-World Applications (MEMURA)*.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Dan Zeman. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*.

Joakim Nivre. 2004. Incrementality in Deterministic Dependency Parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together (ACL)*, pages 50–57.

Joakim Nivre. 2008. Algorithms for Deterministic Incremental Dependency Parsing. *Comput. Linguist.*, 34(4):513–553, December.

Joakim Nivre. 2014. Transition-Based Parsing with Multiword Expressions. In *2^nd PARSEME General Meeting, Athens, Greece*.

Carlos Ramisch. 2015. *Multiword Expressions Acquisition: A Generic and Open Framework*, volume XIV of *Theory and Applications of Natural Language Processing*. Springer.

Victoria Rosén, Gyri Smørdal Losnegaard, Koenraad De Smedt, Eduard Bejček, Agata Savary, Adam Przepiórkowski, Petya Osenova, and Verginica Mititelu. 2015. A survey of multiword expressions in treebanks. In *Proceedings of the 14th International Workshop on Treebanks and Linguistic Theories (TLT14)*.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword Expressions: A Pain in the Neck for NLP. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 2276 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin Heidelberg.

Nathan Schneider, Emily Danchik, Chris Dyer, and Noah A. Smith. 2014a. Discriminative lexical semantic segmentation with gaps: running the MWE gamut. *Transactions of the Association for Computational Linguistics (TACL)*.

Nathan Schneider, Spencer Onuffer, Nora Kazour, Emily Danchik, Michael T. Mordowanec, Henrietta Conrad, and Noah A. Smith. 2014b. Comprehensive Annotation of Multiword Expressions in a Social Web Corpus. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 455–461, Reykjavík, Iceland, May. ELRA.

Djamé Seddah, Reut Tsarfaty, Sandra K$'$ubler, Marie Candito, Jinho Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiorkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clérgerie. 2013. Overview of the SPMRL 2013 Shared Task: A Cross-Framework Evaluation of Parsing Morphologically Rich Languages. In *Proceedings of the 4th Workshop on Statistical Parsing of Morphologically Rich Languages*, Seattle, WA.

Violeta Seretan. 2011. *Syntax-Based Collocation Extraction*. Text, Speech and Language Technology. Springer.

Veronika Vincze, István Nagy T., and Richárd Farkas. 2013a. Identifying english and hungarian light verb constructions: A contrastive approach. In *Proceedings of the 51st Annual Meeting of the Association*

for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers, pages 255–261.

Veronika Vincze, János Zsibrita, and Istvàn Nagy T. 2013b. Dependency parsing for identifying hungarian light verb constructions. In *Proceedings of International Joint Conference on Natural Language Processing (IJCNLP 2013)*, Nagoya, Japan.

Eric Wehrli. 2014. The Relevance of Collocations for Parsing. In *Proceedings of the 10th Workshop on Multiword Expressions (MWE)*, pages 26–32, Gothenburg, Sweden, April. Association for Computational Linguistics.