The ATILF-LLF System for Parseme Shared Task: a Transition-based Verbal Multiword Expression Tagger

Hazem Al Saied
Université de Lorraine, ATILF, CNRS
Nancy, France
halsaied@atilf.fr

Marie Candito
Université Paris Diderot, LLF
Paris, France
marie.candito@linguist.univ-paris-diderot.fr

Matthieu Constant
Université de Lorraine, ATILF, CNRS
Nancy, France
Mathieu.Constant@univ-lorraine.fr

Abstract

We describe the ATILF-LLF system built for the MWE 2017 Shared Task on automatic identification of verbal multiword expressions. We participated in the closed track only, for all the 18 available languages. Our system is a robust greedy transition-based system, in which MWE are identified through a MERGE transition. The system was meant to accommodate the variety of linguistic resources provided for each language, in terms of accompanying morphological and syntactic information. Using per-MWE Fscore, the system was ranked first[1] for all but two languages (Hungarian and Romanian).

1 Introduction

Verbal multi-word expressions (hereafter VMMEs) tend to exhibit more morphological and syntactic variation than other MWEs, if only because in general the verb is inflected, and it can receive adverbial modifiers. Furthermore some VMWEs, in particular light verb constructions (one of the VMWE categories provided in the shared task), allow for the full range of syntactic variation (extraction, coordination etc...). This renders the VMWE identification task even more challenging than general MWE identification, in which fully frozen and contiguous expressions help increasing the overall performance.

The data sets are quite heterogeneous, both in terms of the number of annotated VMWEs and of accompanying resources (for the closed track).[2] So our first priority when setting up the architecture was to build a generic system applicable to all the 18 languages, with limited language-specific tuning. We thus chose to participate in the closed track only, relying exclusively on training data, accompanying CoNLL-U file when available, and basic feature engineering. We developed a one-pass greedy transition-based system, which we believe can handle discontinuities elegantly. We integrated more or less informed feature templates, depending on their availability in the data.

We describe our system in section 2, the experimental setup in section 3, the results in section 4 and the related works in section 5. We conclude in section 6 and give perspectives for future work.

2 System description

The identification system we used is a simplified and partial implementation of the system proposed in Constant and Nivre (2016), which is in itself a mild extension of an arc-standard dependency parser (Nivre, 2004). Constant and Nivre (2016) proposed a parsing algorithm that jointly predicts a syntactic dependency tree and a forest of lexical units including MWEs. In particular, in line with Nivre (2014), this system integrates special parsing mechanisms to deal with lexical analysis. Given that the shared task focuses on the lexical task only and that datasets do not always provide syntactic annotations, we have modified the structure of the original system by removing syntax prediction, in order to use the same system for all 18 languages.

A transition-based system consists in applying a sequence of actions (namely *transitions*) to incrementally build the expected output structure in a bottom-up manner. Each transition is

[1] 2 systems participated for one language only (French), and 5 systems participated for more than one language.

[2] Some of the data sets contain the tokenized sentences plus VMWEs only (BG, ES, HE, LT), some are accompanied with morphological information such as lemmas and POS (CS, MT, RO, SL), and for the third group (the 10 remaining languages) full dependency parses are provided. See (Savary et al., 2017) for more information on the data sets.

Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017), pages 127–132,
Valencia, Spain, April 4.

Transition		Configuration
		$([\,], [1,2,3,4,5,6], [\,])$
Shift	$\Rightarrow$	$([1], [2,3,4,5,6], [\,])$
Complete	$\Rightarrow$	$([\,], [2,3,4,5,6], [1])$
Shift	$\Rightarrow$	$([2], [3,4,5,6], [1])$
Shift	$\Rightarrow$	$([2,3], [4,5,6], [1])$
Complete	$\Rightarrow$	$([2], [4,5,6], [1,3])$
Shift	$\Rightarrow$	$([2,4], [5,6], [1,3])$
Complete	$\Rightarrow$	$([2], [5,6], [1,3,4])$
Shift	$\Rightarrow$	$([2,5], [6], [1,3,4])$
Shift	$\Rightarrow$	$([2,5,6], [\,], [1,3,4])$
Merge	$\Rightarrow$	$([2,(5,6)], [\,], [1,3,4])$
Merge	$\Rightarrow$	$([(2,(5,6))], [\,], [1,3,4])$
Complete	$\Rightarrow$	$([\,], [\,], [1,3,4,(2,(5,6))])$

Figure 1: Transition sequence for tagging *He*$_1$ ***took***$_2$ *this*$_3$ *argument*$_4$ ***into***$_5$ ***account***$_6$.

usually predicted by a classifier given the current state of the parser (namely *configuration*). A configuration in our system consists of a triplet $c = (\sigma, \beta, L)$, where σ is a stack containing units under processing, β is a buffer containing the remaining input tokens, and L is a set of processed lexical units. The processed units correspond either to tokens or to VMWEs. When corresponding to a single token, a lexical unit is composed of one node only, whereas a unit representing a (multi-token) VMWE is represented as a binary lexical tree over the input tokens. Every unit is associated with a set of linguistic attributes (when available in the working dataset): its actual form, lemma, part-of-speech (POS) tag, syntactic head and label. The initial configuration for a sentence $x = x_1, ..., x_n$, *i.e.* a sequence of n tokens, is represented by c_s as: $c_s(x) = ([], [x_1, \ldots, x_n], \emptyset)$ and the set of terminal configurations C_t contains any configuration of the form $c = ([], [], L)$. At the end of the analysis, the identified VMWEs are simply extracted from L.

The transitions of this system are limited to the following: (a) the **Shift** transition takes the first element in the buffer and pushes it onto the stack; (b) the **Merge** transition removes the two top elements of the stack, combines them as a single element, and adds it to the stack;[3] (c) the **Complete** transition moves the upper element of the stack to L, whether the element is a single token or an identified VMWE and finally (d) the **Complete-MWT** transition, only valid for multiword tokens (MWT), acts as Complete, but also marks the element moved to L as VMWE.[4]

Training such a system means enabling it to classify a configuration into the next transition to apply. This requires an oracle that determines what is an optimal transition sequence given an input sentence and the gold VMWEs. We created a static oracle using a greedy algorithm that performs Complete as soon as possible (i.e. when a non VMWE token or a gold VMWE is on top of the stack) and Merge as late as possible (i.e. when the right-most component of the VMWE is on top of the stack) (see Figure 1). Note that an oracle sequence is exactly composed of $2n$ transitions: every single token requires one Shift and one Complete, and each multi-token VMWE of length m requires m Shifts, $m-1$ Merges and a single Complete.

The proposed system has some limitations with respect to the shared task annotation scheme. First, for now, our system does not handle embedded VMWEs (only the longest VMWE is considered in the oracle, and the transition system cannot predict embeddings). This feature could be straightforwardly activated as VMWEs are represented with lexical trees. Note also that the system cannot handle overlapping MWEs like ***take***$_{1,2}$ *a* ***bath***$_1$ *then a* ***shower***$_2$, since it requires a graph representation (not a tree).

3 Experimental setup

For replication purposes, we now describe how the system has been implemented (Subsection 3.1), which feature templates have been used (Subsection 3.2) and how they have been tuned (Subsection 3.3). Simple descriptions of the system settings are provided in Table 1. We thereafter use symbol B_i to indicate the *i*th element in the buffer. S_0 and S_1 stand for the top and the second top elements of the stack. For every unit X in the stack or the buffer, we denote Xw its word form, Xl its lemma and Xp its POS tag. The concatenation of two elements X and Y is noted XY.

3.1 Implementation

For a given language, and a given train/dev split, we train three SVM classifiers (one vs all, one vs

[3]The newly created element is assigned linguistic attributes using basic concatenation rules that would deserve to be improved in future experiments: e.g., the lemma is the concatenation of the lemmas of the two initial elements.

[4]We had to add this transition to cope with MWTs, which are present in some data sets (esp. German). Currently this transition is not predicted by a classifier like the other ones. It is activated under certain hard conditions (cf. Subsection 3.1)

one and error-correcting output codes) and we select the majority vote one.[5]

Note that some configurations only allow for a unique transition type, and thus do not require transition prediction. A configuration with a one token stack and empty buffer requires the application of a Complete, as last transition of the transition sequence. Similarly, a configuration with empty stack and non-empty buffer must lead to a Shift transition.

During the feature tuning phase, for a few languages we added a number of hard-coded procedures aiming at enforcing specific transitions in given contexts. These procedures all use a VMWE dictionary extracted from the training set (hereafter the VMWE dictionary). For German and Hungarian, we noticed a high percentage of VMWEs with one token only.[6] We added the Complete-MWT transition for these languages, which we systematically apply when the head of the stack S_0 is a token appearing as MWT in the VMWE dictionary (cf. setting Q in Table 1). For other languages with long and discontinuous expressions, we used other hard-coded procedures that experimentally proved to be beneficial (setting P in Table 1). We systematically apply a Complete transition when $S_1 l B_0 l$ or $S_1 l B_1 l$ forms a VMWE existing in the VMWE dictionary. Moreover, an obligatory Shift is applied when the concatenation of successive elements in the stack and the buffer belongs to the VMWE dictionary. In particular, we test $S_1 l S_0 l B_0 l$, $S_0 l B_0 l$, $S_0 l B_0 l B_1 l$ and $S_0 l B_0 l B_1 l B_2 l$.

3.2 Feature Templates

A key point in a classical transition-based system is feature engineering, where feature template design and tuning could play a very important role in increasing the accuracy of system results.

Basic Linguistic Features

First of all, depending on their availability in the working dataset and on the activation of related settings (cf. G and J in Table 1), we extracted linguistic attributes in order to generate features such as $S_0 l$, $S_0 p$ and $S_0 w$ where p, l and w stand for the lemma, the part of speech, and the word form respectively. The same features are extracted for unigrams S_1, B_0 and B_1 (when used) (cf. E in Table 1).

When enabled, the bigrams features for the pair XY of elements are XpYp, XlYl, XwYw, XpYl and XlYp. The trigram-based features are extracted in the same way.

Basically, the involved bigrams are $S_1 S_0$, $S_0 B_0$, $S_1 B_0$ and $S_0 B_1$ (cf. setting F in Table 1), but we also added the $S_0 B_2$ bigram for a few languages (cf. N in Table 1). For trigrams, we only used the features of the $S_1 S_0 B_0$ triple (cf. I in Table 1).

Finally, because the datasets for some languages do not provide the basic linguistic attributes such as lemmas and POS tags, we tried to bridge the gap by extracting unigram "morphological" attributes when POS tag and lemma extraction settings were disabled (cf. G and J in Table 1). The features of S_0 for such languages would be $S_0 w$, $S_0 r$, $S_0 s$ where r and s stand for the last two and three letters of $S_0 w$ respectively.

Syntax-based Features

After integrating classical linguistic attributes, we investigated using more linguistically sophisticated features. First of all, syntactic structure is known to help MWE identification (Fazly et al., 2009; Seretan, 2011; Nagy T. and Vincze, 2014). We therefore inform the system with the

Code	F	Setting description
B	+	use of transition history (length 1)
C	+	use of transition history (length 2)
D	+	use of transition history (length 3)
E	+	use of B_1
F	+	use of bigrams (S_1S_0, S_0B_0, S_1B_0,S_0B_1)
G	+	use of lemma
H	+	use of syntax dependencies
I	+	use of trigrams $S_1S_0B_0$
J	+	use of POS tag
K	+	use of distance between S_0 and S_1
L	+	use of training corpus VMWE lexicon
M	+	use of distance between S_0 and B_0
N	+	use of (S_0B_2) bigram
O	+	use of stack length
P	-	enabling dictionary-based forced transitions
Q	-	enabling Complete-MWT transition

Table 1: System setting code descriptions. The 'F' column indicates whether the setting is a feature-related setting ('+') used by the classifiers or whether ('-') it is a hard-coded implementation enhancement.

[5] The whole system was developed using Python 2.7, with 2,200 lines of code, using the open-source Scikit-learn 0.19 libraries for the SVMs. The code is available on Github: `https://goo.gl/EDFyiM`

[6] These correspond mainly to cases of verb-particle (tagged VPC in the data sets) in which the particle is not separated from the verb.

provided syntactic dependencies when available: for each token B_n that both appears in the buffer and is a syntactic dependent of S_0 with label l, we capture the existence of the dependency using the features $RightDep(S_0, B_n) = True$ and $RightDepLabel(S_0, B_n) = l$. We also use the opposite features $IsGovernedBy(S_0, B_n) = True$ and $IsGovernedByLabel(S_0, G) = l$ when S_0's syntactic governor G appears in the buffer. Other syntax-based features aim at modeling the direction and label of a syntactic relation between the two top elements of the stack (feature $syntacticRelation(S_0, S_1) = \pm l$ is used for S_0 governing/governed by S_1).[7] All these syntactic features (cf. H in Table1) try to capture syntactic regularities between the tokens composing a VMWE.

History-based Features

We found that other traditional transition-based system features were sometimes useful like (local) transition history of the system. We thus added features to represent the sequence of previous transitions (of length one, two or three, cf. settings B, C and D in Table 1).

Distance-based Features

Distance between sentence components is also known to help transition-based dependency parsing (Zhang and Nivre, 2011). We thus added the distance between S_0 and B_0 and the distance between S_0 and S_1 (cf. settings K and M in Table 1).

Dictionary-based Features

We also added features based on the VMWE dictionary automatically extracted from the training set. Such features inform the system when one of the focused elements (S_i, B_j) is a component of a VMWE present in the dictionary (cf. L in Table 1).

Stack-length Features

Using the length of the stack as an additional feature (cf. O in Table 1) has also proven beneficial during our feature tuning.

[7] For the shared task, we used gold syntactic features for the languages accompanied with gold dependency companion files, as authorized in the closed track. Performance when using predicted syntax will be evaluated in future work.

Finally, it is worthwhile to note that system settings (cf. Table 1) interact when used to generate the precise set of features. For instance if lemma extraction is disabled (code G) while bigram extraction is enabled (code F), the produced features for e.g. the S_1S_0 bigram would not include the following features: $S_1 l S_0 l$, $S_1 p S_0 l$ and $S_1 l S_0 p$.

3.3 Feature Tuning

We first divided the data sets into 3 groups, based on the availability of CoNLL-U files: (a) for **BG, HE and LT** only the VMWEs on tokenized sentences are available; (b) **CS, ES, FA, MT and RO** are accompanied by CoNLL-U files but without syntactic dependency annotations, and (c) **the other languages** are accompanied by a fully annotated CoNLL-U file. In the first tuning period, we tested the various configurations using three pilot languages (BG, CS, FR) representing one group each. In the latest days of the experiments, the set of languages tested was enlarged to all of them and systematic tuning was performed for every language.

4 Results

Table 2 summarizes the results of the system performance over all the languages proposed by the shared task. Each row of the table displays its per-MWE and per-token F-scores for a given language (identified by its ISO 639-1 code) for test dataset, on top of a 5-fold cross-validation (CV) per-MWE F-score on training dataset. The system settings are represented as a sequence of codes described in Table 1.

We can observe that results are very heterogeneous. For instance, five languages (CS, FA, FR, PL, RO) are above 0.70 per-MWE F-score in the case of cross-validation, while seven languages (DE, HE, HU, IT, LT, MT, SV) are below 0.30. In general, we can see an approximative linear correlation between the number of training VMWEs and the performance. This suggests that the size of training datasets is not large enough as systems' performance does not converge. We note though that some languages like CS and TR reach relatively low scores given the size of training data, which shows the high complexity of this task for these languages.

When comparing to the other shared task systems, we can observe that our system is the only one that handled all 18 languages, showing the

	Corpus		Shared Task						CV	
	#VMWE		MWE-based			token-based			MWE	
	Train	Test	F	Rank	delta	F	Rank	delta	F	System setting
BG	1933	473	0,613	1/2	26,59	0,662	1/2	6,99	0,57	B C F I L M
CS	12852	1684	0,717	1/4	7,49	0,736	1/4	0,79	0,71	B C D E F G J K L M O
DE	2447	500	0,411	1/5	0,57	0,411	2/5	-4,36	0,28	F H I K L N P Q
EL	1518	500	0,401	1/5	8,19	0,469	1/5	3,74	0,56	E F G H J K L N
ES	748	500	0,574	1/5	13,06	0,584	1/5	9,22	0,63	B C D E F G H I J K L M
FA	2707	500	0,867	1/2	6,56	0,902	1/2	4,84	0,88	E F G J K M
FR	4462	500	0,577	1/6	6,86	0,603	2/6	-1,24	0,71	E F G H J K L M N
HE	1282	500	0,334	1/2		0,313	1/2		0,17	B C D E F L N P
HU	2999	500	0,699	2/5	-4,14	0,675	3/5	-3,34	0,24	B C D E F G H I J L Q
IT	1954	500	0,399	1/4	16,81	0,436	1/4	8,67	0,27	C F G H J K P
LT	402	100	0,284	1/2		0,253	1/2		0,086	B C D E F I K L M N O P
MT	772	500	0,144	1/4	8,03	0,163	1/4	7,42	0,081	B C D F G J K L O P
PL	3149	500	0,691	1/4	1,14	0,706	2/4	-2,18	0,7	D F G H J L
PT	3447	500	0,673	1/4	9,19	0,71	1/4	0,76	0,65	B C D E F G H I J K L M N O Q
RO	4040	500	0,753	3/4	-2,44	0,791	3/4	-4,46	0,86	B C D E F G I J K M N
SL	1787	500	0,432	1/4	6,14	0,466	1/4	0,93	0,48	D F G J L P
SV	56	236	0,304	1/4	0,04	0,307	2/4	-0,79	0,25	B C D E F G H I J K L M N O
TR	6169	501	0,554	1/4	3,64	0,553	1/4	2,43	0,58	B C D E F G H J M
AVG			*0,524*			*0.541*			*0,484*	

Table 2: Detailed results of all experiments over all the languages. F columns provide F-score results and delta columns display the difference in F-score (times 10^{-2}) between our system and the best other system of the shared task for the current evaluation/language configuration.

robustness of our approach. Moreover, evaluation using per-MWE F-score (i.e. exact VMWE matching) ranks our system first on all languages but two (HU:2nd:, RO:3rd), displaying an average difference of 6.73 points with the best other system in the current evaluation/language pair. Concerning per-token scores (which allow partial matchings), results are relatively lower: our system is ranked first for 12 languages (out of 18), with a positive average difference of 1.84 points as compared with the best other system. Such very enthusiastic results for per-MWE evaluations seem to show that our system succeeds more in considering a MWE as a whole. Further error analysis is needed to explain this trait, and in particular to check the impact of the Merge transition, which transforms sequences of elements into one.

5 Related Work

Previous approaches for VMWE identification include the two-pass method of candidate extraction followed by binary classification (Fazly et al., 2009; Nagy T. and Vincze, 2014).

VMWE identification has also been performed using sequence labeling approaches, with IOB-scheme. For instance, Diab and Bhutada (2009) apply a sequential SVM to identify verb-noun idiomatic combinations in English. Such approaches were used for MWE identification in general (including verbal expressions) ranging from contiguous expressions (Blunsom and Baldwin, 2006) to gappy ones (Schneider et al., 2014).

A joint syntactic analysis and VMWE identification approach using off-the-shelf parsers is another interesting alternative that has shown to help VMWE identification such as light verb constructions (Eryiğit et al., 2011; Vincze et al., 2013).

6 Conclusion and future work

This article presents a simple transition-based system devoted to VMWE identification. In particular, it offers a simple mechanism to handle discontinuity since foreign elements are iteratively discarded from the stack, which is a crucial point for VMWEs. It also has the advantage of being robust, accurate and efficient (linear time complexity). As future work, we would like to apply more sophisticated syntax-based features, as well as more advanced machine-learning techniques like neural networks and word embeddings. We also believe that a dynamic oracle could help increase results to better deal with cases where the system is unsure.

Acknowledgements

This work was partially funded by the French National Research Agency (PARSEME-FR ANR-14-CERA-0001).

References

Phil Blunsom and Timothy Baldwin. 2006. Multilingual deep lexical acquisition for hpsgs via supertagging. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 164–171, Sydney, Australia, July. Association for Computational Linguistics.

Matthieu Constant and Joakim Nivre. 2016. A transition-based system for joint lexical and syntactic analysis. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 161–171, Berlin, Germany, August. Association for Computational Linguistics.

Mona Diab and Pravin Bhutada. 2009. Verb noun construction mwe token classification. In *Proceedings of the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications*, pages 17–22, Singapore, August. Association for Computational Linguistics.

Gülşen Eryiğit, Tugay İlbay, and Ozan Arkan Can. 2011. Multiword expressions in statistical dependency parsing. In *Proc. of IWPT Workshop on Statistical Parsing of Morphologically-Rich Languages (SPMRL 2011)*, pages 45–55, Dublin.

Afsaneh Fazly, Paul Cook, and Suzanne Stevenson. 2009. Unsupervised type and token identification of idiomatic expressions. *Computational Linguistics*, 35(1):61–103.

István Nagy T. and Veronika Vincze. 2014. Vpctagger: Detecting verb-particle constructions with syntax-based methods. In *Proceedings of the 10th Workshop on Multiword Expressions (MWE)*, pages 17–25, Gothenburg, Sweden, April. Association for Computational Linguistics.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In Frank Keller, Stephen Clark, Matthew Crocker, and Mark Steedman, editors, *Proceedings of the ACL Workshop Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57, Barcelona, Spain, July. Association for Computational Linguistics.

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

Agata Savary, Carlos Ramisch, Silvio Cordeiro, Federico Sangati, Veronika Vincze, Behrang QasemiZadeh, Marie Candito, Fabienne Cap, Voula Giouli, Ivelina Stoyanova, and Antoine Doucet. 2017. The PARSEME Shared Task on Automatic Identification of Verbal Multi-word Expressions. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, Valencia, Spain.

Nathan Schneider, Emily Danchik, Chris Dyer, and Noah A. Smith. 2014. Discriminative lexical semantic segmentation with gaps: running the MWE gamut. *TACL*, 2:193–206.

Violeta Seretan. 2011. *Syntax-based collocation extraction*. Text, Speech and Language Technology. Springer.

Veronika Vincze, János Zsibrita, and István Nagy T. 2013. Dependency parsing for identifying hungarian light verb constructions. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 207–215, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA, June. Association for Computational Linguistics.