A Neural Graph-based Approach to Verbal MWE Identification

Jakub Waszczuk & Rafael Ehren & Regina Stodden & Laura Kallmeyer
Heinrich Heine University
Düsseldorf, Germany
(waszczuk|ehren|stodden|kallmeyer)@phil.hhu.de

Abstract

We propose to tackle the problem of verbal multiword expression (VMWE) identification using a neural graph parsing-based approach. Our solution involves encoding VMWE annotations as labellings of dependency trees and, subsequently, applying a neural network to model the probabilities of different labellings. This strategy can be particularly effective when applied to discontinuous VMWEs and, thanks to dense, pre-trained word vector representations, VMWEs unseen during training. Evaluation of our approach on three PARSEME datasets (German, French, and Polish) shows that it allows to achieve performance on par with the previous state-of-the-art (Al Saied et al., 2018).

1 Introduction

Multiword expressions (MWEs) are defined as combinations of multiple lexemes whose overall properties are not readily predictable by those of their components (Baldwin and Kim, 2010). This idiosyncrasy makes MWEs a well-known challenge for NLP and their ubiquity forces us to find ways to account for them. While all types of MWEs come with their own set of issues, verbal MWEs (VMWEs) stand out as a particularly challenging subclass because of properties like discontinuity, overlap, varying word order, and syntactic or semantic ambiguity. These properties suggest that we have to rely on both syntactic and semantic features to successfully process VMWEs (Savary et al., 2017). E.g., syntactic information can help us catch long-distance dependencies, while semantic information can prove useful in disambiguating between literal and idiomatic readings.

One of the main tasks that constitute MWE processing is the automatic identification of MWEs in running text which can be used as a preprocessing step for parsing or machine translation. MWE identification can be seen as a sequence labeling task similar to named entity recognition (NER): A system receives sequences of tokens as input and outputs the same sequences with annotation labels added to it (Constant et al., 2017). As in NER, most parts of the sequence will belong to the negative class, that is, the majority of words is not part of an MWE. However, certain issues that occur when dealing with NEs and MWEs are much more prevalent in case of the latter. Especially with respect to discontinuity. In the PARSEME 1.0 corpus (Savary et al., 2018), wich comprises datasets of 18 languages, only three of them have a continuity rate of over 80% when it comes to VMWEs. German, the most striking example in this regard, has a continuity rate of 35.7% and 30.54% of its discontinuities are longer than three words (Savary et al., 2017). In (1), the verb-particle construction *teilnehmen* 'take part' spans over 13 words and this is not even a particularly excessive example. Much more could be inserted in between the two VMWE components *nahmen* and *teil*, e.g. a relative clause, without it sounding marked.

(1)

[In]	Paris	selbst	**nahmen**	zur	gleichen	Zeit
in	*Paris*	*itself*	***took***	*at the*	*same*	*time*

rund	tausend	Studenten	an	einer
roughly	*thousand*	*students*	*in*	*a*

Kundgebung	in	dem	Quartier	Latin	**teil**.
rally	*in*	*the*	*Quartier*	*Latin*	***part***.

'In Paris itself roughly a thousand students **took part** in a rally in the Quartier Latin.'[1]

In this paper, we propose a method which identifies VMWE occurrences directly over dependency structures. Relying on existing dependency trees greatly simplifies the task, since VMWEs

[1]From the German set of the PARSEME Shared Task 1.1.

are usually connected in such trees (Bejček et al., 2012), even if they are discontinuous on the surface of word sequences as in (1).

In the same vein as (Waszczuk, 2018), our method is conceptually divided into two layers. The first is concerned with encoding VMWE occurrences as tree labellings, as well as the inverse process of decoding the labellings into VMWE annotations. In the second layer, a probability model which allows to discriminate between different VMWE labellings is used. We propose two probability models, both based on dense feature representations (i.e. pre-trained word embeddings) of input words. Relying on dense features allows to easier generalize beyond training data, on the one hand, and to possibly capture helpful syntactic and semantic cues, on the other hand.[2]

The paper is structured as follows. In Sec. 2, we describe related work on VMWE identification. In Sec. 3, we give a detailed description of our methods, in particular the encoding schemata and the labelling models. In Sec. 4, we summarize the experiments we performed to evaluate our approach. Finally, we conclude and mention possible future work in Sec. 5.

2 Related Work

The MWE identification strategies can be broadly divided into approaches based on deep learning, sequence labelling, and parsing-based methods. Because all of these have different advantages (and are not necessarily mutually exclusive), some systems also pursue a mix of these approaches, e.g., the system proposed here uses both (graph-based) parsing and deep learning methods.

Gharbieh et al. (2017) were one of the first to apply deep learning to MWE identification. They tested different network architectures, e.g., a layered feed-forward network and a recurrent neural network, and all of them outperformed more traditional MWE identification methods. The approaches based on deep learning have the advantage that they can easily leverage pre-trained word vectors as features (Constant et al., 2017; Taslimipoor and Rohanian, 2018; Ehren et al., 2018). The method described in this work also relies on pre-trained word vectors.

Schneider et al. (2014) addressed the task of MWE identification as a sequence labeling problem. They proposed a sequence labeling scheme (IiOoBb) which allows to represent discontinuous MWEs as well as nested MWEs. The encoding methods we propose also allow to handle discontinuous and, to a certain degree, nested MWEs, with the important difference that they apply to trees rather than sequences.

Previous work on applying parsing-based techniques to MWE identification includes transition-based (Constant and Nivre, 2016; Al Saied et al., 2018; Stodden et al., 2018) and graph-based (Waszczuk, 2018; Boroş and Burtica, 2018) approaches. As shown by the two PARSEME shared tasks (Savary et al., 2017; Ramisch et al., 2018a), both strategies can be very effective, even without relying on pre-trained word vectors. The method we propose is graph-based and it resembles the one of Waszczuk (2018) in that it relies on global modelling and restricts the labelling decisions to dependency fragments, and the one of Boroş and Burtica (2018) in that it relies on a neural architecture. In comparison with the former, the encoding schemata we propose allow to deal with two important phenomena the method of (Waszczuk, 2018) could not handle – adjacent and disconnected MWE occurrences.

Another way to classify MWE identification approaches is based on whether the process of MWE prediction takes place before, during, or after (syntactic and/or semantic) parsing. The joint solution is typically considered as most promising in that it can potentially improve both MWE identification and parsing results (Constant and Nivre, 2016; Le Roux et al., 2014; Nasr et al., 2015; Simkó et al., 2018). On this scale, our method clearly fits into the family of post-processing approaches, since it requires the dependency trees on input. Nevertheless, it should be straightforward to extend it to a fully joint solution, notably due to its similarities with the graph-based, arc-factored, neural dependency parsing architecture of (Dozat and Manning, 2017).

3 Methods

In this section we describe the methods and models used in the proposed MWE identification approach. In Sec. 3.1, we introduce some basic definitions. In Sec. 3.2, we detail the methods of encoding MWE occurrences as tree labellings. This allows to reduce the problem of MWE identification to the problem of determining the best la-

[2]Implementation of the methods presented in this work can be found at `https://github.com/kawu/vine`.

belling of the given (dependency) tree. We propose two solutions to the latter problem, both described in Sec. 3.3.

3.1 Basic Definitions

Input Sentence. We define an *input sentence* of length n as a sequence $\boldsymbol{w} = (\boldsymbol{w}_i \in \mathbb{R}^d)_{i=1}^n$ of vector representations (with dimension d) corresponding to the subsequent input words. The individual vectors $\boldsymbol{w}_i$ can be simply defined as input word embeddings, but they can also be the result of preliminary processing (e.g., concatenating the input word embeddings with hidden POS representations).

Dependency Tree. We define a *dependency tree* as a directed rooted tree $G = (V, E)$, where V is a set of nodes and $E \subset V \times V$ is a set of arcs. Given $(v, w) \in E$, we say that w is v's head and that v is w's dependent. For simplicity, we blur the distinction between dependency nodes and word identifiers and assume that $V = \{0, 1, 2, \ldots, n\}$, with 0 representing a dummy root node. We additionally define $\mathrm{inc}(i) = \{h \in V \mid (h, i) \in E\}$ and $\mathrm{out}(i) = \{j \in V \mid (i, j) \in E\}$.

3.2 Encoding

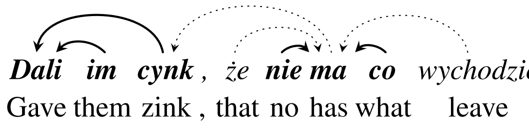
Our methodology relies fundamentally on the idea of *encoding* MWE occurrences as tree labellings. More precisely, given a sentence and the corresponding dependency tree, the set of MWE occurrences present therein is encoded as a labelling of dependency arcs and nodes. A machine learning method is used to model the probabilities of different labellings, which it learns based on a training dataset of encoded MWE annotations. MWE identification requires a reverse procedure of *decoding* a given labelling to the set of MWE occurrences.

MWE Occurrence. Each MWE occurrence is represented as a set of tokens in a particular sentence. We are thus not concerned with determining the category of a MWE occurrence (in our experiments we train one model per MWE category).

3.2.1 Basic Encoding

In the basic encoding scheme, we assume that elements of a single MWE occurrence are connected by dependency arcs. The set of MWE occurrences in a sentence with a given dependency tree is encoded as a single labelling function $\ell_E \colon E \to \mathbb{B}$ defined over the set of arcs $E \subset V \times V$ of the tree.

Encoding. $\ell_E(v, w) := 1$ for a given arc $(v, w) \in E$ iff both v and w belong to a single



Dali im cynk , że nie ma co wychodzić
Gave them zink , that no has what leave

Figure 1: Example of extended encoding applied to a tree fragment with two Polish idioms, *dać komuś cynk* ‘give someone a tip‘ and *nie ma co [wychodzić]* ‘it is not worth [leaving]‘, adjacent in the dependency tree. The nodes and arcs labelled with 1 are marked in bold. The example (simplified) comes from the Polish dataset of the PARSEME Shared Task 1.1.

MWE occurrence.

Decoding is a two-stage process. First, a copy of the dependency tree is created in which only the arcs $(v, w) \in E$ such that $\ell_E(v, w) = 1$ are preserved. Next, each connected component[3] in the copy of the tree is considered to represent a distinct MWE occurrence.

Limitations. The basic encoding scheme does not handle single-token, disconnected, or overlapping MWE occurrences. In the process of encoding, both single-token and disconnected MWE occurrences get either discarded or trimmed. Overlapping MWE occurrences, on the other hand, get merged.

3.2.2 Extended Encoding

In the extended encoding scheme, the set of MWE occurrences is encoded as a pair of labelling functions $\ell_V \colon V \to \mathbb{B}$ and $\ell_E \colon E \to \mathbb{B}$.

Encoding. $\ell_V(v) := 1$ for a given node $v \in V$ iff v is a part of a MWE. $\ell_E(v, w) := 1$ iff both v and w belong to the shortest, undirected path between (any) two component nodes of a single MWE occurrence.

Thanks to node labelling, the extended encoding scheme allows to represent single-token MWE occurrences. Arc labelling, on the other hand, facilitates demarcating adjacent MWE occurrences (see Fig. 1). Finally, using a hybrid (node and arc) labelling allows to represent disconnected MWE occurrences (see Fig. 2).

Decoding. As in the basic encoding scheme, a copy of the dependency tree consisting of arcs labelled with 1 is created first. To accommodate for single-node MWE occurrences, the set of arcs in this copy is further enriched with $\{(v, v) \colon v \in$

[3]Formally, a *connected component* is a set of nodes $C \subset V$ such that every two nodes in C are connected by an undirected path.

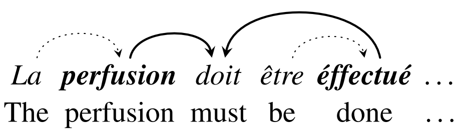

La perfusion doit être éffectué ...
The perfusion must be done ...

Figure 2: Example of extended encoding applied to a tree fragment with a disconnected French light-verb construction. The nodes and arcs labelled with 1 are marked in bold. The example (simplified) comes from the French dataset of the PARSEME Shared Task 1.1.

$V, \ell_V(v)\}$. Finally, each connected component in the resulting structure is considered as a distinct MWE. However, given a particular MWE component C, only the nodes $v \in C$ such that $\ell_V(v) = 1$ are marked as the MWE's elements.

Node and arc labellings can be, in general, inconsistent. An example is a labelling with $\ell_E(v, w) = 1$ for some $(v, w) \in E$ and $\ell_V(v) = 0$ for every $v \in V$. Determining an optimal, consistent labelling for a given sentence is therefore a problem of structured prediction.

Limitations. While the extended encoding scheme is more powerful than the basic one, it still cannot deal with certain phenomena. A notable limitation is the inability to represent overlapping MWE occurrences. Another, more practical drawback is that encoding two MWE occurrence components placed far from each other in the dependency tree entails labelling the entire list of arcs in between (not necessarily related to the MWE) with 1's. Such a situation can in particular occur when the dependency structure is obtained automatically in pre-processing. Joint modelling of dependency structures and MWEs (to which, we believe, our work can be extended) would in principle alleviate this issue.

3.3 Labelling

We consider two labelling models in this work. The first, local model (see Sec. 3.3.2) relies on the basic encoding scheme (see Sec. 3.2.1) and assumes independence between the labelling decisions for the individual dependency arcs. The second, global model (described in Sec. 3.3.3) adopts the extended encoding scheme (see Sec. 3.2.2) and relaxes the independence assumptions of the first model.

3.3.1 Notation

Given two vectors $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathbb{R}^m$, we use $[\boldsymbol{x}; \boldsymbol{y}] \in \mathbb{R}^{n+m}$ to denote the vector concatenation of $\boldsymbol{x}$ and $\boldsymbol{y}$.

The definitions provided below are set in the context of a specific input sentence $\boldsymbol{w}$ and the corresponding dependency tree $G = (V, E)$. These should be understood as implicit arguments of the individual functions defined below.

3.3.2 Local Model

Score. We define the *score* vector $\Phi(i, j) \in \mathbb{R}^2$ of the dependency arc $(i, j) \in E$ as:

$$\Phi(i, j) = \text{MLP}([\boldsymbol{w}_i; \boldsymbol{w}_j]), \quad (1)$$

where MLP is a feed-forward network with a single hidden layer followed by a leaky rectifier and an output layer with two units. The two output scores represent the arc's affinity of not being and being labelled as a MWE component, respectively.

Probability. We define the probability distribution $P(\ell_E(i, j) \mid \boldsymbol{w}, G)$ based on the scores of the arc (i, j) using SoftMax:

$$P(\ell_E(i, j) \mid \boldsymbol{w}, G) = \text{SoftMax}(\Phi(i, j)) \quad (2)$$

Prediction. To determine a most probable labelling for a given sentence, we rely on the adopted independence assumption and set the output label $\ell_E(i, j)$ to 1 iff $P(\ell_E(i, j) = 1 \mid \boldsymbol{w}, G) > \alpha$ for each $(i, j) \in E$ separately, where α is a threshold over which an arc is considered as a MWE. In the rest of this paper, we simply assume $\alpha = 0.5$.

3.3.3 Global Model

The second labelling model we consider in this work is a global model in which the score is assigned to the labelling of the entire dependency tree. This model is based on the extended encoding scheme (see Sec. 3.2.2) and it is concerned with both node and arc labellings.

Compound Labels. We introduce a compound labelling function $\ell \colon E \to \mathbb{B}^3$ as:

$$\ell(i, j) = (\ell_V(i), \ell_E(i, j), \ell_V(j)) \quad (3)$$

This function combines the label of the given arc with the labels of the arc's source and target nodes. Modelling ℓ enables making predictions about both node and arc labellings and, consequently, handling the extended encoding scheme. Moreover, node labels are shared between different compound labels, which allows to capture inter-label interactions and enables global scoring.

To facilitate the interpretation of compound labels, for a given $x = \ell(i,j)$ we denote $x_{\text{dep}} = \ell_V(i)$, $x_{\text{arc}} = \ell_E(i,j)$, and $x_{\text{hed}} = \ell_V(j)$.

Node Score. We define the *node score* $\phi_V(i) \in \mathbb{R}$ of the dependency node $i \in V$ as:

$$\phi_V(i) = \text{MLP}'(\boldsymbol{w}_i)_1, \tag{4}$$

where MLP′ is a feed-forward network with a single hidden layer followed by a leaky rectifier and a single-unit output layer (with the resulting value accessed via $_1$). The score represents the node's affinity of being labeled as a MWE component.

Arc Score. We define the *arc score* $\phi_E(i,j) \in \mathbb{R}^8$ of the arc $(i,j) \in E$ as an 8-element vector whose individual values correspond to the scores of the different $\ell(i,j)$ labelling combinations. Put differently, there are 8 different ways of labelling i, j, and (i,j), and $\phi_E(i,j)$ provides the score for each of these 8 possibilities.

The score vector $\phi_E(i,j)$ is calculated using a network consisting of a single hidden layer and two output layers. The output layers contain 8 and 3 units, respectively. The 8 elements of the first output vector $\boldsymbol{\gamma} \in \mathbb{R}^8$ correspond to the scores of the different labelling combinations. The 3 elements of the second output vector $\boldsymbol{\delta} = (\delta_1, \delta_2, \delta_3) \in \mathbb{R}^3$ correspond to the scores of (i) labelling i with 1, (ii) labelling (i,j) with 1, and (iii) labelling j with 1, respectively. These scores are combined to produce the final score vector using the binary masks $\boldsymbol{m}_1 = (0,0,0,0,1,1,1,1)$, $\boldsymbol{m}_2 = (0,0,1,1,0,0,1,1)$, and $\boldsymbol{m}_3 = (0,1,0,1,0,1,0,1)$:

$$\phi_E(i,j) = \boldsymbol{\gamma} + \delta_1\boldsymbol{m}_1 + \delta_2\boldsymbol{m}_2 + \delta_3\boldsymbol{m}_3 \tag{5}$$

For instance, $\delta_1\boldsymbol{m}_1$ is the scalar multiplication of the mask $\boldsymbol{m}_1$ by the 1-st element of the vector $\boldsymbol{\delta}$, i.e., $\delta_1\boldsymbol{m}_1 = (0,0,0,0,\delta_1,\delta_1,\delta_1,\delta_1)$. Hence, δ_1 impacts (equally) all the 4 elements of $\phi_E(i,j)$ which correspond to labelling the node i with 1.

The motivation behind using both $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$ is that it can be useful to look at the nodes i, j and the arc (i,j) both jointly and separately, depending on the situation. For instance, if the arc (i,j) belongs to a single MWE, than the score of labelling both i and j with 1 should be high, while the score of labelling only one of them with 1 should be low. This relation can be easily expressed with $\boldsymbol{\gamma}$. Conversely, $\boldsymbol{\delta}$ may be better in expressing the pattern where one of the nodes i, j can be easily pinpointed as a MWE element, while the other is hard to judge without looking at the other neighboring arcs.

Formally, given the input vector representation $[\boldsymbol{w}_i; \boldsymbol{w}_j]$ of the arc $(i,j) \in E$, the hidden vector and the two output vectors are calculated as follows:

$$\boldsymbol{h} = \sigma(\boldsymbol{A}_1[\boldsymbol{w}_i; \boldsymbol{w}_j] + \boldsymbol{b}_1) \tag{6a}$$
$$\boldsymbol{\gamma} = \boldsymbol{A}_2\boldsymbol{h} + \boldsymbol{b}_2 \tag{6b}$$
$$\boldsymbol{\delta} = \boldsymbol{A}_3\boldsymbol{h} + \boldsymbol{b}_3, \tag{6c}$$

where $\boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{A}_3$ are matrices, $\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3$ are the corresponding bias vectors, and σ is the element-wise activation function (leaky rectifier).

Arc Scoring Restrictions. We additionally fix the arc score of each compound label $(x, 0, y)$ for any $x, y \in \mathbb{B}$ to 0. In practice, this means that the non-MWE nodes surrounding a MWE candidate do not influence the choice of marking this candidate as a MWE. This allows to avoid overfitting which could result from relying too much on the surrounding, non-MWE words.

Casting. The set of compound labels $\mathbb{B}^3$ and the set of score vector indices $\{1, 2, \ldots, 8\}$ are isomorphic. In a slight abuse of notation, we therefore treat elements of $\mathbb{B}^3$ and $\{1, 2, \ldots, 8\}$ interchangeably, assuming implicit cast between the objects of these two types.

Local Score. We define the *local score* vector $\phi(i,j) \in \mathbb{R}^8$ for a given arc $(i,j) \in E$ as the sum of the (i,j)'s arc score and the i's node score:

$$\phi(i,j)_x = \phi_V(i)x_{\text{dep}} + \phi_E(i,j)_x, \tag{7}$$

where $x \in \{1, 2, \ldots, 8\}$ represents a particular compound label, $\phi_V(i)x_{\text{dep}}$ is the node score of the dependent, activated only if it is labelled with 1 ($x_{\text{dep}} = 1$), and $\phi_E(i,j)_x$ is the arc score of the compound label x with respect to the arc (i,j).

Global Score. The score $\Phi(\ell) \in R$ of a given labelling ℓ of the entire tree is defined as the sum of the local scores implied by the global labelling:

$$\Phi(\ell) = \sum\nolimits_{(i,j)\in E} \phi(i,j)_{\ell(i,j)} \tag{8}$$

Since we assume the dependency structure to be a tree, summing the local scores of all the arcs in the graph is equivalent to summing the node scores of all the nodes and the arc scores of all the arcs in the tree, modulo the dummy root node.[4]

[4]For simplicity, we assume that the node score of the dummy root is 0. In practice, either this score needs to be explicitely handled, or labelling the root with 1 should be prohibited (the dummy root cannot be a part of a MWE).

Probability. We define the probability of a particular compound labelling ℓ as:

$$P(\ell \mid \boldsymbol{w}, G) = \frac{\exp(\Phi(\ell))}{\sum_{\ell'} \exp(\Phi(\ell'))}, \quad (9)$$

where the calculation of $Z = \sum_{\ell'} \exp(\Phi(\ell'))$, the so called *partition* function (Goldberg, 2017, p. 224), involves summing over all the possible compound labellings of the given tree.[5] The global model is therefore an instance of a log-linear model which combines the scores determined by the non-linear MLP component. A similar solution can be found in (Durrett and Klein, 2015).

Due to the arc-factored nature of the model, it is possible to calculate $P(\ell \mid \boldsymbol{w}, G)$ – in particular, the partition Z – efficiently, without summing over the exponentially many labellings. This can be done using a variant of the standard inside algorithm, which can be specified using the following recursive function inside: $V \times \mathbb{B} \to \mathbb{R}$:

$$\text{inside}(j, x) = \prod_{i \in \text{inc}(j)} \sum_{y \in \mathbb{B}^3 :\, y_{\text{hed}} = x} \exp(\phi(i, j)_y) \times \text{inside}(i, y_{\text{dep}}) \quad (10)$$

The partition factor Z is then equal to $\text{inside}(r, 0) + \text{inside}(r, 1)$, where r is the root of the dependency tree.

Prediction involves determining a highest-probability (see Eq. 9) or, equivalently, a highest-score (see Eq. 8) labelling. This can be achieved using a variant of the inside algorithm:

$$g(j, x) = \sum_{i \in \text{inc}(j)} \max_{y :\, y_{\text{hed}} = x} \phi(i, j)_y + g(i, y_{\text{dep}})$$

Constraints. Any labelling consistent with the extended encoding scheme (see Sec. 3.2.2) must satisfy the property that, if a node i is labeled with 0, then either zero or more than one arc among $\{(h, i) \mid h \in \text{inc}(i)\} \cup \{(i, j) \mid j \in \text{out}(i)\}$ is labelled with 1. Put differently, all the nodes on the border of a given MWE occurrence must be marked as its elements.

While this constraint is not directly reflected in the definition of the probability (see Eq. 9), we use it in our implementation of the global model for both prediction and (optionally) training.

4 Experiments

We now describe the experiments we performed in order to evaluate the methods described in Sec. 3.

[5]The labellings must be internally consistent, i.e., compound labels must agree on the labels of the nodes they share.

4.1 Dataset

All the experiments were run on the German (DE), French (FR), and Polish (PL) datasets of the edition 1.1 of the PARSEME corpus (Ramisch et al., 2018b). This highly multilingual corpus was created in the context of a shared task on the automatic identification of VMWEs and consists of annotated datasets of 20 different languages. The individual datasets are collections of sentences which were, among other things, tokenized, part-of-speech (POS) tagged, lemmatized, and enriched with dependency information. While FR contains solely manual dependency annotations, the dependencies in DE were annotated partly manually and partly automatically. Besides automatic annotations, PL includes dependencies that were converted from a manually annotated constituency treebank.

The VMWE annotation comprises the identification of the words that belong to a VMWE instance, as well as the categorization of the identified instances. The categories used in the PARSEME annotation framework are light-verb constructions (LVCs), verbal idioms (VIDs), inherently reflexive verbs (IRV), verb-particle constructions (VPC), multi-verb constructions (MVC), and inherently adpositional verbs (IAV).

Our implementation of the global model is currently a prototype and it takes a relatively long time to train a model.[6] We therefore focused on a few languages which come from different families and cover a large spectrum of VMWE-related phenomena. This way, we hope we can test our system on a variety of problems despite the small number of languages. For instance, DE contains a large amount of VPCs, a verb class very common in Germanic, but almost non-existent in Romance or Slavic languages. These VPCs also account for most of the single-token VMWEs in DE which do not occur in FR or PL. The Polish dataset covers a reasonable amount of IAVs,[7] which are rather challenging for our models because of their lack of connectivity in the dependency structures.

4.2 Pre-processing

The first pre-processing step we used in our experiments involved removing (multiword) tokens,

[6]Around 16 hours were required to train a single global model on the 220465 tokens of the Polish dataset using four cores of a 2.40GHz Xeon E5-2630 machine.

[7]IAVs were an experimental PARSEME category and a lot of languages in the PARSEME corpus do not cover them.

such as the contraction *du* of *de le* ‘of the‘ in French, from consideration.[8] In the PARSEME datasets, only the expanded forms (i.e., *de le* rather than *du*) are annotated at the level of dependency structures and VMWEs.

The second pre-processing operation consisted in adding a dummy root node (with special POS and dependency relation values) to each dependency structure, to enforce that it is actually a tree (as required by the global model, see Sec. 3.3.3). This was particularly important for the German dataset, in which some of the dependency structures did not satisfy this property.

The two previous steps are carried out automatically by our VMWE identification systems. However, we also performed one full-fledged pre-processing operation of adding the missing lemmas[9] in the French test set. Even though having no impact on the results of the proposed systems, this step was necessary to obtain reliable comparison with the benchmark system (see Sec. 4.3), configured to use lemma information in case of French.

4.3 Benchmark System

As a benchmark, we use the system of Al Saied et al. (2018), henceforth called ATILF, a transition-based tagger relying on support vector machines and hand-crafted features for classification. The hand-crafted features are separately specified for each language. ATILF addresses several VMWE challenges at the same time – it is able to handle single-token, discontinuous, nested, and (some forms of) overlapping VMWEs. Without relying on word embeddings or any other external resources, the benchmark system yields state-of-the-art results on the PARSEME corpus 1.1 (Taslimipoor and Rohanian, 2018).[10]

4.4 System Implementation

In this subsection we detail the implementation of the proposed systems.

4.4.1 Input

To each word in the input sentence a vector representation is assigned. This representation consists of the concatenation of the corresponding (i) FastText word embedding (Mikolov et al., 2018), (ii) POS embedding, and (iii) dependency label embedding. The latter correspond to the dependency label of the arc connecting the word with its dependency head. The size of the FastText word embeddings is 300. We chose the size of 25 for both POS and dependency embeddings, which should be sufficient given the small number of values they can take. The POS and dependency label embedding vectors are both learned during training, while the FastText embeddings are kept intact.

4.4.2 Network Dimensions

In both labelling models, the size of the network's input layer is determined by the size of the input vector representations. All the scoring networks contain one hidden layer with 200 units, followed by element-wise leaky ReLU. The size of the hidden layer was chosen during preliminary experiments on the French dataset.

4.4.3 Training

Objective. For both labelling models considered in this work we define the training objective for a given tree (V, E) as the sum of the cross-entropies between the target and the estimated distributions for the individual arcs $(i, j) \in E$. In case of the local model the arc labelling distributions $P(\ell_E(i, j) \mid \boldsymbol{w}, G)$ (see Eq. 2) are used, and in case of the global model – the marginal compound labelling distributions $P(\ell(i, j) \mid \boldsymbol{w}, G)$. The marginal distributions can be defined in terms of the global probability (see Eq. 9) and calculated efficiently using the inside-outside algorithm.

Backpropagation. In order to use $P(\ell(i, j) \mid \boldsymbol{w}, G)$ as a part of the training objective, the inside-outside algorithm needs to be specified in a backpropagation-enabled way. We achieve this by using a library[11] which automatically determines the way to backpropagate the gradient from the output to the input of the inside-outside algorithm. Conveniently, this requires no changes in the structure of the algorithm itself. This is similar to how the inside algorithm can be extended with its outside counterpart automatically, using automatic differentiation (Eisner, 2016).

Optimization. We used stochastic gradient descent (SGD) to train the models for the individual datasets and VMWE categories. We used mini-

[8]The discarded tokens are restored after identification in order to allow for comparison with gold data.

[9]Provided by the shared task's organizers via `http://groups.google.com/group/verbalmwe`.

[10]ATILF was originally developed for the edition 1.0 of the PARSEME shared task. We therefore converted the relevant files of the PARSEME corpus 1.1 to the format supported by the tool. We used the published, default feature configurations for the individual languages.

[11]`https://backprop.jle.im/`

		DE			FR			PL			AVG		
		P	R	F	P	R	F	P	R	F	P	R	F
ATILF	MWE-based	71.56	46.71	**56.52**	82.69	71.38	76.62	85.23	68.35	**75.86**	79.82	62.15	**69.67**
	Tok-based	76.43	45.72	57.21	85.73	72.96	78.83	88.69	67.9	**76.92**	83.61	62.19	**70.99**
Local	MWE-based	49.64	27.15	35.10	71.04	62.08	66.67	75.54	53.98	62.97	65.41	47.98	55.36
	Tok-based	68.22	39.78	50.25	80.03	68.12	73.60	79.45	54.37	64.56	75.90	54.09	63.17
Global	MWE-based	68.48	47.70	56.24	84.92	70.75	**77.19**	80.83	64.66	71.84	78.08	61.04	68.52
	Tok-based	72.74	47.83	**57.72**	86.84	73.24	**79.47**	83.13	66.19	73.69	80.90	62.42	70.47

Table 1: General results per language and system on the development data.

		DE			FR			PL			AVG		
		P	R	F	P	R	F	P	R	F	P	R	F
ATILF	MWE-based	70.82	39.96	51.09	74.57	61.24	**67.25**	80.94	60.19	69.04	75.44	53.80	62.81
	Tok-based	76.03	39.69	52.16	79.83	65.93	**72.22**	83.21	59.48	69.37	79.69	55.03	65.10
Local	MWE-based	54.36	26.31	35.45	60.26	55.42	57.74	74.46	60.00	66.45	63.03	47.24	54.00
	Tok-based	70.3	36.82	48.38	73.96	62.08	67.50	78.95	59.57	67.90	74.48	52.82	61.81
Global	MWE-based	69.72	44.38	**54.23**	74.57	60.64	66.89	82.01	66.41	**73.39**	75.43	57.14	**65.02**
	Tok-based	74.52	44.10	**55.41**	78.56	63.54	70.25	83.85	66.06	**73.90**	78.98	57.90	**66.82**

Table 2: General results per language and system on the test data.

batches of size 30 and the training length of 60 epochs. We did not apply drop-out.

We used the Adam variant (Kingma and Ba, 2015) of the SGD algorithm with the default parameters: initial stepsize $\alpha_0 = 0.001$, the exponential decay rates $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. We additionally used a gradually decreasing stepsize $\alpha = \frac{\alpha_0 \times \tau}{\tau + t}$, where $t \in [0, 60]$ is the epoch number (fractional) and $\tau = 15$. The suitable hyperparameter values were determined during preliminary experiments on the French dataset.

4.5 Evaluation Results

We evaluated the two implemented systems and the benchmark (ATILF) system on the development and the test parts of the German (DE), French (FR), and Polish (PL) PARSEME datasets.[12] We trained one local and three global models per language and per VMWE category. Training separate models for different categories allows to partially handle the issue of overlapping VMWE instances, as well as to simplify the architecture (no need to encode the categories in terms of tree labellings). The benchmark system predicts all the categories in one pass. We used the official evaluation script[13] provided by the shared task organizers to calculate all the scores.

As mentioned above, we trained three global models per language and per VMWE category. One model was obtained using constrained training and two models using unconstrained training (see Sec. 3.3.3). We observed that the results between different training runs can differ significantly. For instance, the LVC.full identification F-scores differed by almost 3% on the FR development set between the two unconstrained models. We therefore used all three models (for each language and VMWE category) to calculate ensemble node scores (see Eq. 4) and ensemble arc scores (see Eq. 5) by simply summing up the corresponding scores coming from the three models. Such ensemble averaging should have a smoothing effect and alleviate the issue of diverging results.

The general results of the three systems on the development and the test sets are presented in Tab. 1 and Tab. 2, respectively. The benchmark system and the global model achieve comparable results: ATILF has better overall performance on the development sets, the global model – on the test sets. The results of the local model are consistently lower than those of the global model, which shows the usefulness of extended encoding combined with global scoring. Nevertheless, the local model achieves very competitive results, comparable to those obtained with the best systems participating in the PARSEME shared task 1.1.

More detailed evaluation results are presented in Tab. 3 and Tab. 4. The former shows the performance of the systems across different VMWE-related challenges, while the latter presents their

[12]In contrast to several systems participating in the PARSEME shared task 1.1, we didn't use the development parts for training. This is important in that the development set can contain VMWEs unseen in the training part.

[13]Available at https://gitlab.com/parseme/sharedtask-data/tree/master/1.1/bin/.

	Contin-uous	**Discon-tinuous**	**Multi-token**	**Single-token**	**Seen-in-train**	**Unseen-in-train**	**Variant-of-train**	**Identical-to-train**
ATILF	72.19	44.79	60.26	69.08	**82.15**	18.9	71.87	**92.72**
Local	56.68	47.96	56.37	0.0	72.29	29.59	68.06	75.88
Global	**72.58**	**53.30**	**62.67**	**69.89**	81.65	**32.28**	**74.07**	89.23

Table 3: Results (MWE-based F-scores) per VMWE challenge averaged over the three language test sets. The single-token score is only calculated on German because single-token VMWEs do not occur in the other languages.

		VID	**LVC.full**	**VPC.full**	**IRV**	**IAV**
DE	**ATILF**	**39.29**	19.23	64.55	28.57	-
	Local	33.67	21.87	40.29	30.77	-
	Global	35.56	**22.95**	**72.40**	**32.84**	-
	#	37%	8%	42%	8%	0%
FR	**ATILF**	64.47	60.9	-	73.53	-
	Local	51.08	53.25	-	75.93	-
	Global	**66.12**	**61.29**	-	**78.47**	-
	#	43%	32%	0%	22%	0%
PL	**ATILF**	**46.73**	50.81	-	86.08	60.0
	Local	13.01	64.86	-	85.71	0.0
	Global	35.51	**65.62**	-	**87.32**	**69.57**
	#	14%	29%	0%	48%	6%

Table 4: Results (MWE-based F-scores) for the selected VMWE categories on the test sets. The last row per language reports the percentage of occurrences per category in the test data.

results for the VMWE categories occurring most frequently in the three test sets. These results clearly show that our approach performs particularly well for both discontinuous and unseen VMWEs. Despite its relative simplicity, the local model also yields better results than the benchmark system in these two categories. The global model under-performs in the identification of the identical-to-train VMWEs. This also applies to the local model, which does not perform very well in the category of seen-in-train VMWEs in general.

Concerning the VMWE categories, VIDs proved challenging for both our models, especially in `DE` and `PL`. This may be due to the arc-factored nature of our approach, which may be inadequate for handling VIDs, often composed from more than two words. On the other hand, our approach proved very effective in case of LVCs (especially in `PL`) and, somewhat surprisingly, in case of IAVs, consistently disconnected in the PARSEME dependency structures.

5 Conclusions and Future Work

In this paper we propose a neural, graph-based VMWE identification method relying on the idea that VMWE annotations can be represented as labellings of dependency trees. Couching the task of VMWE identification as syntax-driven labelling allows to transparently handle the issue of discontinuity, a challenging property of VMWEs which makes sequential models poorly adapted for this task. Relying on neural scoring and pre-trained word embeddings, on the other hand, facilitates identifying unseen VMWEs. While the idea of applying parsing-based (in particular, graph-based) methods to VMWE identification is not novel, we show that combining it with a neural scoring component and supplying the system with pre-trained word embeddings allows to achieve overall results on par with state-of-the-art on all three PARSEME datasets we experimented with (German, French, and Polish), and to surpass it as far as handling discontinuous and unseen VMWEs is concerned.

Some VMWE-related challenges, such as overlapping instances, are not very well supported by the encoding schemata we propose. Their enhancement is thus one of the major points for future work. Furthermore, we believe that the system could better support certain classes of VMWEs, in particular verbal idioms, often continuous and consisting of several lexemes. It is difficult to identify such VMWEs by focusing on pairs of input tokens at a time. Possible solutions to this issue include adding a BiLSTM layer in order to contextualize the input word embeddings, as well as increasing the scope of the factors underlying the global model. We plan to explore these possibilities in future work as well. Finally, we would like to extend our model to a joint dependency parsing and VMWE identification solution, and to experimentally check how effective it can be in the setting where the dependency structures are not available on input.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. The work presented in this paper was funded by the German Research Foundation (DFG) within the CRC 991 and the Beyond CFG project, as well as by the Land North Rhine-Westphalia within the NRW-Forschungskolleg Online-Partizipation.

References

Hazem Al Saied, Marie Candito, and Matthieu Constant. 2018. A transition-based verbal multiword expression analyzer. In Stella Markantonatou, Carlos Ramisch, Agata Savary, and Veronika Vincze, editors, *Multiword expressions at length and in depth: Extended papers from the MWE 2017 workshop*, pages 209–226. Language Science Press., Berlin.

Timothy Baldwin and Su Nam Kim. 2010. Multiword expressions. *Handbook of natural language processing*, 2:267–292.

Eduard Bejček, Jarmila Panevová, Jan Popelka, Pavel Straňák, Magda Ševčíková, Jan Štěpánek, and Zdeněk Žabokrtský. 2012. Prague Dependency Treebank 2.5 – a revisited version of PDT 2.0. In *Proceedings of COLING 2012*, pages 231–246, Mumbai, India. The COLING 2012 Organizing Committee.

Tiberiu Boroş and Ruxandra Burtica. 2018. GBD-NER at PARSEME shared task 2018: Multi-word expression detection using bidirectional long-short-term memory networks and graph-based decoding. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pages 254–260, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Mathieu Constant, Gülşen Eryiğit, Johanna Monti, Lonneke van der Plas, Carlos Ramisch, Michael Rosner, and Amalia Todirascu. 2017. Multiword expression processing: A survey. *Comput. Linguist.*, 43(4):837–892.

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. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Greg Durrett and Dan Klein. 2015. Neural CRF parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 302–312, Beijing, China. Association for Computational Linguistics.

Rafael Ehren, Timm Lichte, and Younes Samih. 2018. Mumpitz at PARSEME shared task 2018: A bidirectional LSTM for the identification of verbal multiword expressions. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pages 261–267, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Jason Eisner. 2016. Inside-outside and forward-backward algorithms are just backprop (tutorial paper). In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 1–17, Austin, TX. Association for Computational Linguistics.

Waseem Gharbieh, Virendrakumar Bhavsar, and Paul Cook. 2017. Deep learning models for multiword expression identification. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 54–64, Vancouver, Canada. Association for Computational Linguistics.

Yoav Goldberg. 2017. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the Third International Conference on Learning Representations (ICLR)*, San Diego, California, USA.

Joseph Le Roux, Antoine Rozenknop, and Matthieu Constant. 2014. Syntactic parsing and compound recognition via dual decomposition: Application to french. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1875–1885, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

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 (Volume 1: Long Papers)*, pages 1116–1126, Beijing, China. Association for Computational Linguistics.

Carlos Ramisch, Silvio Ricardo Cordeiro, Agata Savary, Veronika Vincze, Verginica Barbu Mititelu, Archna Bhatia, Maja Buljan, Marie Candito, Polona Gantar, Voula Giouli, Tunga Güngör, Abdelati Hawwari, Uxoa Iñurrieta, Jolanta Kovalevskaitė, Simon Krek, Timm Lichte, Chaya Liebeskind, Johanna Monti, Carla Parra Escartín, Behrang QasemiZadeh, Renata Ramisch, Nathan Schneider, Ivelina Stoyanova, Ashwini Vaidya, and Abigail Walsh. 2018a. Edition 1.1 of the PARSEME shared task on automatic identification of verbal multiword expressions. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG 2018)*, Santa Fe, New

Mexico, USA. Association for Computational Linguistics.

Carlos Ramisch, Silvio Ricardo Cordeiro, Agata Savary, Veronika Vincze, Verginica Barbu Mititelu, Archna Bhatia, Maja Buljan, Marie Candito, Polona Gantar, Voula Giouli, Tunga Güngör, Abdelati Hawwari, Uxoa Iñurrieta, Jolanta Kovalevskaitė, Simon Krek, Timm Lichte, Chaya Liebeskind, Johanna Monti, Carla Parra Escartín, Behrang QasemiZadeh, Renata Ramisch, Nathan Schneider, Ivelina Stoyanova, Ashwini Vaidya, Abigail Walsh, Cristina Aceta, Itziar Aduriz, Jean-Yves Antoine, Špela Arhar Holdt, Gözde Berk, Agnė Bielinskienė, Goranka Blagus, Loic Boizou, Claire Bonial, Valeria Caruso, Jaka Čibej, Matthieu Constant, Paul Cook, Mona Diab, Tsvetana Dimitrova, Rafael Ehren, Mohamed Elbadrashiny, Hevi Elyovich, Berna Erden, Ainara Estarrona, Aggeliki Fotopoulou, Vassiliki Foufi, Kristina Geeraert, Maarten van Gompel, Itziar Gonzalez, Antton Gurrutxaga, Yaakov Ha-Cohen Kerner, Rehab Ibrahim, Mihaela Ionescu, Kanishka Jain, Ivo-Pavao Jazbec, Teja Kavčič, Natalia Klyueva, Kristina Kocijan, Viktória Kovács, Taja Kuzman, Svetlozara Leseva, Nikola Ljubešić, Ruth Malka, Stella Markantonatou, Héctor Martínez Alonso, Ivana Matas, John McCrae, Helena de Medeiros Caseli, Mihaela Onofrei, Emilia Palka-Binkiewicz, Stella Papadelli, Yannick Parmentier, Antonio Pascucci, Caroline Pasquer, Maria Pia di Buono, Vandana Puri, Annalisa Raffone, Shraddha Ratori, Anna Riccio, Federico Sangati, Vishakha Shukla, Katalin Simkó, Jan Šnajder, Clarissa Somers, Shubham Srivastava, Valentina Stefanova, Shiva Taslimipoor, Natasa Theoxari, Maria Todorova, Ruben Urizar, Aline Villavicencio, and Leonardo Zilio. 2018b. Annotated corpora and tools of the PARSEME shared task on automatic identification of verbal multiword expressions (edition 1.1). LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Agata Savary, Marie Candito, Verginica Barbu Mititelu, Eduard Bejek, Fabienne Cap, Slavomr pl, Silvio Ricardo Cordeiro, Glen Eryiit, Voula Giouli, Maarten van Gompel, Yaakov HaCohen-Kerner, Jolanta Kovalevskait, Simon Krek, Chaya Liebeskind, Johanna Monti, Carla Parra Escartn, Lonneke van der Plas, Behrang QasemiZadeh, Carlos Ramisch, Federico Sangati, Ivelina Stoyanova, and Veronika Vincze. 2018. PARSEME multilingual corpus of verbal multiword expressions. In Stella Markantonatou, Carlos Ramisch, Agata Savary, and Veronika Vincze, editors, *Multiword expressions at length and in depth: Extended papers from the MWE 2017 workshop*, pages 87–149. Language Science Press., Berlin.

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 multiword expressions. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 31–47, Valencia, Spain. Association for Computational Linguistics.

Nathan Schneider, Emily Danchik, Chris Dyer, and Noah A. Smith. 2014. Discriminative lexical semantic segmentation with gaps: Running the MWE gamut. *Transactions of the Association for Computational Linguistics*, 2:193–206.

Katalin Ilona Simkó, Viktria Kovcs, and Veronika Vincze. 2018. Identifying verbal multiword expressions with POS tagging and parsing techniques. In Stella Markantonatou, Carlos Ramisch, Agata Savary, and Veronika Vincze, editors, *Multiword expressions at length and in depth: Extended papers from the MWE 2017 workshop*, pages 227–243. Language Science Press., Berlin.

Regina Stodden, Behrang QasemiZadeh, and Laura Kallmeyer. 2018. TRAPACC and TRAPACCS at PARSEME shared task 2018: Neural transition tagging of verbal multiword expressions. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pages 268–274, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Shiva Taslimipoor and Omid Rohanian. 2018. SHOMA at PARSEME shared task on automatic identification of VMWEs: Neural multiword expression tagging with high generalisation. *arXiv preprint arXiv:1809.03056*.

Jakub Waszczuk. 2018. TRAVERSAL at PARSEME shared task 2018: Identification of verbal multiword expressions using a discriminative tree-structured model. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pages 275–282, Santa Fe, New Mexico, USA. Association for Computational Linguistics.