Veyn at PARSEME Shared Task 2018: Recurrent neural networks for VMWE identification

Nicolas Zampieri and **Manon Scholivet** and **Carlos Ramisch** and **Benoit Favre**
Aix-Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France
nicolas.zampieri@etu.univ-amu.fr and first.last@lis-lab.fr

Abstract

This paper describes the Veyn system, submitted to the closed track of the PARSEME Shared Task 2018 on automatic identification of verbal multiword expressions (VMWEs). Veyn is based on a sequence tagger using recurrent neural networks. We represent VMWEs using a variant of the begin-inside-outside encoding scheme combined with the VMWE category tag. In addition to the system description, we present development experiments to determine the best tagging scheme. Veyn is freely available, covers 19 languages, and was ranked ninth (MWE-based) and eight (Token-based) among 13 submissions, considering macro-averaged F1 across languages.

1 Introduction

Multiword expressions (MWEs) pose problem for NLP systems such as machine translation. For instance, in English *there are plenty more fish in the sea* would be translated into French as *une de perdu, dix de retrouvées* (lit. *one lost, ten found*) and not word-by-word as *il y a plus de poissons dans la mer*. To be able to translate MWEs correctly, however, we have to first identify them. Automatic identification of MWEs, and in particular of verbal MWEs (VMWEs), is the topic of this paper.

The PARSEME shared task 2018 is an evaluation campaign of systems for the identification of VMWEs (Ramisch et al., 2018).[1] This task presents many challenges, such as the presence of variants, discontinuous and ambiguous MWEs (Constant et al., 2017). Our system "Veyn" is based on a sequence tagger using recurrent neural networks (RNNs). We represent VMWEs using a variant of the standard begin-inside-outside (BIO) encoding scheme. Moreover, we achieve VMWE categorization by combining the VMWE category with BIO tags. The goal of the RNN is to predict the correct BIO+category tag for each token. We use no external corpus or word embeddings to train our system, hence we participated in the closed track. Veyn is freely available,[2] and covers 19 of the 20 languages of the shared task (all except Arabic, which required a special license).

Sequence taggers were successfully employed by many systems for MWE identification in the past. Most existing models and systems, however, represent features as discrete values taken from finite sets instead of continuous vectors. Examples of such systems employ conditional random fields (Constant and Sigogne, 2011; Riedl and Biemann, 2016; Maldonado et al., 2017; Scholivet and Ramisch, 2017) and structured perceptron (Schneider et al., 2014). Most recent NLP systems for sequence tagging, however, are based on RNNs. Our system follows this trend by adapting an RNN model successful in other tagging tasks to VMWE identification.

Our system is similar to MUMULS, submitted to the previous PARSEME shared task, edition 1.0 (Klyueva et al., 2017). MUMULS was evaluated on fifteen languages with variable results. Our system differs from MUMULS in the hyper-parameter configuration, the tag encoding scheme (IO for MUMULS, BIO for Veyn), the use of pre-initialized embeddings (not used by MUMULS) and the number of recurrent layers (1 in MUMULS, 2 in Veyn). In the remainder of this paper, we describe the system

[1] http://multiword.sourceforge.net/sharedtask2018

[2] https://github.com/zamp13/Veyn. The version described in this paper corresponds to commit 845f33a

	La	musique	n'	adoucit	pas	toujours	les	mœurs	.
BIO	B	I	g	I	g	g	I	I	O
IO+cat	IVID	IVID	g	IVID	g	g	IVID	IVID	O
BIO+cat	BVID	IVID	g	IVID	g	g	IVID	IVID	O

Figure 1: Three tagging schemes for an example sentence in French: BIO, IO+cat and BIO+cat.

architecture (Sec. 2), the tuning experiments on the development data (Sec. 3), and the results at the shared task (Sec. 4), before concluding (Sec. 5).

2 System Description

We use CoNLL-U's LEMMA and UPOS fields as input features (falling back to FORM and XPOS, respectively, if the former are absent).[3] Each token's LEMMA and UPOS are converted into one-hot vectors, which are then transformed into embeddings and concatenated. Input LEMMA and UPOS embeddings are pre-initialized on the shared task training corpora, but fine-tuned during the training phase. These embeddings are then forwarded to a double bidirectional recurrent layer using gated recurrent units (GRU). Finally, each BIO label prediction is based on a softmax layer that takes as input the concatenation of the GRU cell outputs in both directions for each token. VMWEs annotations are reconstructed on the output based on heuristic rules that group together 'B' and 'I' tags with the same category.

BIO encoding For all our experiments, we used a variant of the BIO scheme (Ramshaw and Marcus, 1995). In the original BIO scheme, every token is tagged 'B', 'I' or 'O'. If the tag is 'B', it means that the token is the at the beginning of a VMWE. The tag 'I' means the token is inside a VMWE. Finally, if the token's tag is 'O', it means the token does not belong to an VMWE. Since BIO was originally designed for continuous sequences, a special label (lowercase 'g') is used for tokens not belonging to an expression, but occurring in between words that belong to an expression (Schneider et al., 2014). BIO allows abstracting away the order of VMWEs in the sentence. However, it does not allow representing overlaps, that is, tokens belonging to more than one VMWE at the same time. These are quite rare, corresponding to 3,293 out of 982,155 tokens that are part of a VMWE (0.34%) in the training, development and test corpora. We deal with overlaps simply by repeating the sentence and adding one VMWE annotation to each copy of the sentence.

Figure 1 shows different tagging schemes used to tune our system on a French discontinuous VMWE of the verbal idiom (VID) category.[4] The first one is BIO: ***La*** is tagged B to indicate where the VID begins, ***musique***, ***adoucit***, ***les***, and ***moeurs*** are inside the expression, *n'*, *pas*, and *toujours* do not belong to the expression, but occur in between words that belong to the expression, and the period is outside of the expression. Notice that the system must identify the precise words that are part of the VMWE, and not its span, thus the need for a special tag 'g' to indicate intermediate tokens.

During system development, one of our goals was to evaluate different tagging schemes and choose the best one based on the development corpus performances. Therefore, in addition to the extended BIO scheme, we also tested an adaptation that includes category labels (BIO+cat). 'B' and 'I' tags are thus concatenated with the provided VMWE's category labels (IRV, LVC.full, VID, etc). The idea is that categories present quite heterogeneous characteristics, so it may be a good idea to model/learn them separately in the neural network. This is illustrated in the last row of Figure 1. Finally, We have also evaluated our system using an inside-outside scheme similar to the one used in MUMULs (Klyueva et al., 2017). This scheme does not distinguish the token that begins an expression from the others (IO+cat).[5]

[3] `http://universaldependencies.org/format.html`

[4] Categories are described at `http://parsemefr.lif.univ-mrs.fr/parseme-st-guidelines/1.1`

[5] Though both are not fully comparable because MUMULS does not use a special label (lowercase 'g') for intervening

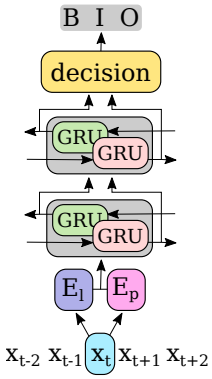

Figure 2: Veyn's architecture, with two layers of bidirectional GRU based on lemma (E_l) and POS (E_p) embeddings, Source: (Scholivet et al., 2018).

Given the performance of these tagging schemes (Sec. 3), we decided not experiment with IO tags, as they were not considered as a promising representation choice.

MWE tagger Veyn is based on a recurrent neural network. Figure 2 shows the architecture of our system. It uses as inputs LEMMA and UPOS (falling back to FORM and XPOS, respectively, if the former are absent). Because of limitations of the library used to implement the system, we must set a fixed length for all sentences in the corpus. Therefore, we considered that each sentence contains 128 tokens. If the sentence is shorter, we pad it with zeros until we reach 128 tokens. If it is longer, we crop it and ignore all remaining tokens. Padding should not pose problems, as the network should easily learn that zeroes are always tagged using a special padding label. Cropping, on the other hand, affects 400 out of 317,816 sentences (0.13%) in the training, development and test corpora, especially in Romanian (148 sentences), Turkish (125 sentences) and Italian (85 sentences). Raising the sentence length above 128 could avoid cropping sentences at the expense of significantly slowing down the RNN training.

Each token's LEMMA and UPOS, represented as one-hot vectors, are transformed into dense vectors of dimension 250 by the first trainable layer of the model, represented by E_l and E_p in Figure 2. In the submitted system, we have pre-initialized these embeddings using word2vec on the training corpus itself.[6] Since pre-initialized embeddings are learned on lemmas and POS tags rather than surface forms, the reduced size of the training corpus does not prevent us from generating useful representations in the form of 250-dimensional vectors.

The first recurrent layer takes as input the concatenation of the embeddings layers E_l and E_p. The second recurrent layer takes as input the concatenation of the forward and backward vectors of the previous one. Both are composed of identical gated recurrent units (GRU), a variant of long short-term memory units (LSTMs). Each recurrent layer is bidirectional, and contains 512 neurons per input position. We use two recurrent layers because preliminary experiments indicated that models using one or three recurrent layers presented lower performance. We have used no regularization or drop-out, but we believe that this could further improve the performance of our system in the future.

For the decision, the activation layer takes the output of the forward and backward units of the second recurrent layer, and use a softmax function to transform results into probabilities for each possible tag. We then choose the tag with the highest probability. Finally, the predicted tags use the BIO scheme provided in the training data. We used heuristics to recognize which tokens are part of the same expression, that is, to transform all tags from the BIO format into the required `cupt` format.

These heuristics will first convert all 'O' and 'g' tags into an asterisk, indicating that the token is not part of a VMWE.[7] Then, for every token tagged with a 'B', we assign it a new VMWE identifier. Furthermore, when BIO+cat or IO+cat are used, we categorize the token according to the predicted

tokens.

[6]Word2vec parameters for pre-initialized embeddings: Skip-gram, 5-word window, 10 negative samples, downsampling rate of 1e-3, minimum count of 1, no hierarchical softmax, 15 iterations.

[7]`http://multiword.sourceforge.net/cupt-format`

	MWE-based																
	BG	DE	EL	ES	EU	FA	FR	HE	HR	HU	IT	PL	PT	RO	SL	TR	**avg**
BIO	48.60	17.60	16.89	18.55	50.70	65.04	58.68	10.06	18.00	84.60	24.13	56.94	53.17	72.60	39.20	45.09	**38.96**
BIO+cat	47.60	25.91	16.15	7.80	57.05	67.46	49.95	11.53	19.46	85.37	18.76	62.79	51.79	75.58	40.88	48.18	**41.56**
IO+cat	48.51	22.74	17.46	7.80	38.70	58.72	45.44	9.34	13.98	74.37	16.98	53.32	18.87	69.96	30.16	31.17	**35.44**
BIO+cat+PI	46.29	29.27	23.65	20.27	60.18	65.30	53.07	15.95	22.44	86.75	33.87	62.31	55.41	80.10	43.29	42.73	**46.31**
	Token-based																
BIO	53.76	42.63	44.53	37.41	62.76	78.25	65.83	22.91	40.69	87.16	36.58	65.68	60.32	78.78	53.37	51.21	**53.90**
BIO+cat	57.83	42.90	32.39	19.48	66.15	75.99	58.89	14.68	31.01	87.36	30.65	69.05	58.49	80.65	52.64	54.03	**50.88**
IO+cat	58.52	44.23	44.12	19.48	59.65	76.61	60.62	23.10	40.02	79.9	38.22	67.29	39.87	78.14	49.38	71.72	**52.13**
BIO+cat+PI	53.88	44.51	42.41	33.74	66.26	73.72	65.32	24.56	41.43	88.38	42.44	68.31	57.36	82.94	52.68	47.43	**55.34**

Table 1: Performance of tested tagging schemes on the development corpus, according to MWE-based and Token-based F1. PI stands for pre-initialized embeddings layer.

category. All subsequent 'I' tokens are associated to the VMWE identifier of the closest preceding 'B', as long as both have the same category. Given that the RNN does not have hard constraints on allowed label sequences, a token can be labelled 'I' even if there is no preceding token labelled 'B'. In this case, we treat the 'I' token as a 'B' starting a new VMWE. For example, in the Spanish sentence *Gracias a Dios* ***vamos conociendo*** *un poquito mas de lo que* ***podria ser*** *un gobierno [...]*, the tokens ***vamos*** and ***conociendo*** are respectively tagged 'B+MVC' and 'I+MVC', and so are the tokens ***podria*** and ***ser***. Our heuristics will thus identify a first VMWE ***vamos conociendo*** and a second one ***podria ser***.

All parameters other than the input vectors are initialized randomly.[8] We used `train.cupt` files to train our models for all languages, `dev.cupt` to find the optimal tag encoding scheme, and `test.blind.cupt` to obtain the predicted results for submission. We used the Python library Keras to implement our system, using Tensorflow as backend. We use `sparse_categorical_crossentropy` as loss function, Nadam as optimizer, training takes 10 epochs, and batches are of size 128.

3 Tuning the system

In section 2 we discussed several tagging schemes that could be used to represent VMWEs. To choose which one we would submit to the shared task, we have searched the best tagging scheme on all languages based on their performance on the development corpus. We have tuned our system only on the languages for which we have a training and a development corpus (that is, EN, LT and HI are not considered).

Table 1 shows the results of our experiments to tune our system with different tagging schemes on the development corpus. Each column represents a language code (e.g. BG for Bulgarian, DE for German, etc.) and the last column contains the macro-average over all languages. Two F1 scores were considered: MWE-based performs strict per-MWE comparison, whereas Token-based F1 is a fuzzy score, taking partial matches into account (Savary et al., 2017; Ramisch et al., 2018).

When BIO tags are compared with BIO+cat tags, we noticed that the results depend on the training corpus size. For example, the Romanian corpus is much larger than the Spanish corpus. In Romanian, using category labels in BIO+cat seems to increase performance, whereas in Spanish it seems to degrade performance. These results indicate that the size of the training corpus is important to determine the tagging scheme. Other factors that may affect the quality of predictions include the quality of the corpus, which can be estimated as the inter-annotator agreement. For instance, inter-annotator agreement is lower for Spanish than for Romanian (Ramisch et al., 2018).

When we compare the BIO+cat and BIO schemes on average, BIO+cat scores are higher on the MWE-based evaluation (41.56 for BIO+cat vs. 38.96 for BIO) and BIO scores are higher on the Token-based evaluation (50.88 for BIO+cat vs. 53.90 for BIO). A similar trend is observed when comparing IO+car and BIO+cat, with IO+cat performing better than BIO+cat on the Token-based evaluation (52.13 for IO+car vs. 50.88 for BIO+cat). Both BIO and IO+cat use reduced tagsets with respect to BIO+cat, and

[8] As a consequence, the results reported in this paper are not fully reproducible and may oscillate because of random initialization. This has been corrected in the latest version of the system, after the submission.

		avg F1 on dev	avg F1 on test
General	MWE-based	46.68	36.94
	TOK-based	56.32	44.9
Continuous	MWE-based	**49.75**	39.66
Discontinuous	MWE-based	34.23	25.94
Multi-token	MWE-based	48.58	28.96
Single-token	MWE-based	9.23	24.25
Seen-in-train	MWE-based	**69.71**	**55.65**
Unseen-in-train	MWE-based	10.55	11.05
Variant-of-train	MWE-based	**60.16**	46.47
Identical-to-train	MWE-based	**65.49**	**64.36**

Table 2: Average (avg) of characteristics and categories on the development corpus and on the test corpus.

this helps in recognizing words that are parts of an expression. However, these tagsets are not accurate enough to represent full expressions, especially after the use of heuristics to reconstruct the VMWE annotation from the predicted tags. Therefore, we chose BIO+cat as our submission tagging scheme for all languages, assuming that MWE-based evaluation has priority over Token-based evaluation.

Furthermore, the performance of the model depends on the initialization of the RNN. Therefore, we tried to pre-initialize (PI) the input layer with embeddings pre-trained on the training corpus for LEMMA and UPOS. On average, this considerably improves the model performance. However, for some languages such as Farsi, Polish, Portuguese and Turkish, BIO+cat scores are higher than BIO+cat+PI. This outlier behaviour should be investigated in the future to improve our system for these languages.

4 Results and analysis

Sixteen languages (listed in Table 1) contain a development file, whereas three languages are too small to have a development file. Table 2 shows the average F1 scores on the development corpora (16 languages) and on the test corpora (19 languages), both in general and according to linguistic characteristics of the VMWEs reported by the evaluation script.

If we focus on the average performance on the development corpus, we can see that Veyn is better at recognising continuous expressions (49.75%) than discontinuous expressions (34.23%). In fact, discontinuous expressions are more difficult to tag even if we have a special tag to identify tokens not belonging but occurring in the middle of an expression. More sophisticated models (e.g. tree-based or graph-based) would be required to represent discontinuous VMWEs more accurately.

Our system has better scores when it has seen a VMWE or its variants in the training corpus, rather than when VMWEs are unseen in the training corpus. It is indeed unlikely that, by training the system on the training corpus only, we could tag an expression which it has not been seen in the training corpus at all. Nonetheless, if it saw one token which is part of an expression, we may succeed to tag this token. This is why we have a non-zero score of 10.55% MWE-based F1 on expressions which are unseen in the training corpus. Most expressions are formed by several tokens. Therefore, Veyn has higher F1 scores for multi-token expressions (48.58%) than for single-token ones (9.23%).

Table 2 shows that the system obtains an F1 score which is about 10% higher on the development corpus than on the test corpus. The global results on the test corpus are $36.94\%(\leq 46.68\%)$ for general MWE-based evaluation and $44.9\%(\leq 56.32\%)$ for Token-based evaluation. Veyn performed better on the test corpus than on the development corpus for Single-token VMWEs. Only expressions identical to training corpus are stable: 65.49% for the development corpus and 64.36% for the test corpus.

These discrepancies could be explained by the low performance for the three languages for which no development set was available, which were also the languages with the smallest training corpora in terms of number of annotated VMWEs. Moreover, our system could not predict any correct VMWE for Bulgarian, whereas the MWE-based F1 score on the development corpus was 46.29. By observing the learning curves of the system, we do not believe that this is due to overfitting. Instead, we think that it can be related to the random initialization of parameters, associated with an insufficient number of training epochs, which led to lower F1 scores on the test corpus than on the development corpus. Our system

was ranked ninth (eighth) on the average MWE-based (Toekn-based) F1 score on the official shared task 2018 ranking.

As for the prediction of different categories, the main factor influencing performance seems to be category frequency. While we do admit that some categories are probably harder to identify than others, we have observed that Veyn could not identify most VMWEs that appeared rarely in the training corpus. For instance, in French, only 2% of the VMWEs are tagged as LVC.cause in the development corpus, and 1% in the training corpus. Therefore, our system did not identify any LVC.cause correctly. On the other hand, IRVs correspond to 27% of the VMWEs in the training corpus in French. As a result, our system identified them with an F1 score of 0.72 in the development corpus.

Upon inspection of a sample of predictions for the French data, we have noticed some interesting facts. First, Veyn was capable of correctly identifying some VMWEs that were never observed in the training corpus. For example, Veyn tagged the VMWE ***donner résultat*** (*give result*) as LVC.full in the sentence *Les obligations de parité prévues par la loi ont* ***donné*** *des* ***résultats*** (dev). In the training corpus, the verb *donner* (*give*) is often annotated as part of an LVC.full, and the noun *résultat* (*result*) was also seen as part of the LVC.full ***produire résultat*** (*produce result*) and ***avoir pour résultat*** (*have as result*). We hypothesize that the RNN was able to combine information from several training instances to be able to tag a new, unseen VMWE.

On the other hand, some seen VMWEs such as ***mener réflexion*** (*carry out reflection*) were missed, as in the sentence *Une* ***réflexion*** *commune est* ***menée*** *aves les enseignants [...]* (dev). This probably happens because this VMWE only occurs in the training corpus in active voice, with its components in reversed order. The RNN was not able to handle long-distance discontinuity and order change.

Finally, the transformation heuristics used to "glue" the BIO tags into a VMWE in the `cupt` sometimes led to errors. For instance, a sentence may start with two consecutive VMWEs, such as ***Il faut tenir compte*** *de [...]* (dev). Here, ***il faut*** (*One must*) is the first expression and ***tenir compte*** (*take [into] account*) is the second one, both categorized as VID. Our system tagged both as a single VID composed of 4 tokens. This shows that the heuristics could probably be improved if they took POS and syntactic dependencies into account.

One particular limitation of our system is that it cannot tag a single token with several VMWE tags, but we can have sentences containing tokens that belong to several VMWEs. For example, *Il devrait comparaître dans les prochains jours pour indiquer s'il* ***plaide coupable*** *ou* ***non coupable*** *[...]* (train) contains two expressions which involve the same token: ***plaide coupable*** (*plead guilty*) is the first VID, and ***plaide non coupable*** (*plead non guilty*) is the second one. Our system cannot recognize these two expressions because it can only predict one tag per token.

5 Conclusions and future work

We have presented the Veyn system for the automatic identification of verbal multiword expressions. The system is based on a sequence tagger using an RNN over an extended BIO scheme enriched with category labels. The system has participated in the PARSEME shared task 2018 on the closed track, obtaining reasonably good results. It was ranked ninth and eighth considering the cross-lingual macro-averaged MWE-based and Token-based F1 scores. It performed better on VMWEs seen in the training corpus (identical or variants). It was fifth at the shared task ranking for single-token VMWEs.

Veyn can be improved in several points, especially to deal with expressions never seen in the training corpora. We would like to train our system using embeddings already pre-initialized on larger non-annotated corpora, beyond those provided for the shared task. Moreover, we believe that integrating external lexical resources could improve the system's coverage. A second point for future work concerns the initialization of the RNN, to make it more robust and performing. The initialization of the RNN is currently random, but sometimes the system training procedure fails and predicts no tag at all. These runs were simply manually detected and discarded for the moment. We would like to study why this happens, and find automatic ways to prevent it. A third idea for future work would be to achieve representing overlapping VMWEs. This is quite rare in general, but does occur in many languages, such as English or French. Our system does not handle these cases.

Acknowledgments

This work was supported by the IC1207 PARSEME COST action[9] and by the PARSEME-FR project (ANR-14-CERA-0001).[10]

References

Matthieu Constant and Anthony Sigogne. 2011. MWU-aware part-of-speech tagging with a CRF model and lexical resources. In *Proceedings of the ALC Workshop on Multiword Expressions: From Parsing and Generation to the Real World*, MWE 2011, pages 49–56. 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. *Computational Linguistics*, 43(4):837–892.

Natalia Klyueva, Antoine Doucet, and Milan Straka. 2017. Neural networks for multi-word expression detection. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 60–65, Valencia, Spain, April. Association for Computational Linguistics.

Alfredo Maldonado, Lifeng Han, Erwan Moreau, Ashjan Alsulaimani, Koel Dutta Chowdhury, Carl Vogel, and Qun Liu. 2017. Detection of verbal multi-word expressions via conditional random fields with syntactic dependency features and semantic re-ranking. In *Proceedings of the 13th Workshop on Multiword Expressions* , MWE '17, pages 114–120. Association for Computational Linguistics, April.

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. 2018. 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, August. Association for Computational Linguistics.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *3rd Workshop on Very Large Corpora*, pages 82–94.

Martin Riedl and Chris Biemann. 2016. Impact of MWE resources on multiword recognition. In *Proceedings of the 12th Workshop on Multiword Expressions* , MWE '16, pages 107–111. Association for Computational Linguistics.

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, April. 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(1):193–206.

Manon Scholivet and Carlos Ramisch. 2017. Identification of ambiguous multiword expressions using sequence models and lexical resources. In *Proceedings of the 13th Workshop on Multiword Expressions*, MWE '17, pages 167–175. Association for Computational Linguistics, April.

Manon Scholivet, Carlos Ramisch, and Benoit Favre. 2018. Identification d'expressions polylexicales avec réseaux de neurones récurrents. *Traitement Automatique des Langues*. (submitted).

[9] `http://www.parseme.eu`

[10] `http://parsemefr.lif.univ-mrs.fr/`