CRF-Seq and CRF-DepTree at PARSEME Shared Task 2018: Detecting Verbal MWEs Using Sequential and Dependency-Based Approaches

Erwan Moreau
Adapt Centre
Trinity College Dublin
erwan.moreau@adaptcentre.ie

Ashjan Alsulaimani
School of Computer Science and Statistics
Trinity College Dublin
alsulaia@tcd.ie

Alfredo Maldonado
Adapt Centre
Trinity College Dublin
alfredo.maldonado@adaptcentre.ie

Carl Vogel
Trinity Centre for Computing
and Language Studies
Trinity College Dublin
vogel@scss.tcd.ie

Abstract

This paper describes two systems for detecting Verbal Multiword Expressions (VMWEs) which both competed in the closed track at the PARSEME VMWE Shared Task 2018. *CRF-DepTree-categs* implements an approach based on the dependency tree, intended to exploit the syntactic and semantic relations between tokens; *CRF-Seq-nocategs* implements a robust sequential method which requires only lemmas and morphosyntactic tags. Both systems ranked in the top half of the ranking, the latter ranking second for token-based evaluation. The code for both systems is published under the GNU General Public License version 3.0[1] and is available at http://github.com/erwanm/adapt-vmwe18.

1 Introduction

This paper describes two systems for identifying verbal multiword expressions (VMWEs). This work builds on the authors' previous attempt at the same task (Maldonado et al., 2017; Moreau et al., 2018). The two systems were designed and developed in the context of the 2018 edition of the PARSEME VMWE Shared Task, in which training data annotated with VMWEs is provided for 20 different languages (Ramisch et al., 2018). This setting naturally leads to considering the task as a supervised learning problem.[2]

In this edition of the PARSEME shared task, we attempted an approach which focuses on leveraging the syntactic and semantic dependency relations between tokens; the aim is to take into account the phrase structure of the expressions in order to improve their identification. We expect this to be useful especially in the case of discontinuous VMWEs, which can be hard to identify using sequential approaches. In §2.1 we motivate this approach with a detailed analysis of the expressions found in the data; the rest of §2 describes the design and implementation of our first and main system, called *CRF-DepTree-categs*.

Comparatively, the second system *CRF-Seq-nocategs* (described in §3) is very simple and was originally intended as a baseline and/or a fallback option. It relies on a method inspired from the one presented in (Moreau and Vogel, 2018) to optimize a sequential CRF model. To our surprise, the system performed better than *CRF-DepTree-categs* with most datasets. A brief comparative analysis of the results of the two systems is provided in §5, which validates our initial assumption that, despite its weaknesses, *CRF-DepTree-categs* is better at identifying discontinuous VMWEs.

[1] https://www.gnu.org/licenses/gpl-3.0.en.html. Last verified: May 2018.

[2] It is worth noticing that this view also entails a complete dependency on the annotations with respect to what is defined as a VMWE. In particular, there can be differences between datasets in the number, diversity and syntactic or semantic complexity of the annotated expressions (Maldonado et al., 2017).

Proceedings of the Joint Workshop on
Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018), pages 241–247
Santa Fe, New Mexico, USA, August 25-26, 2018.

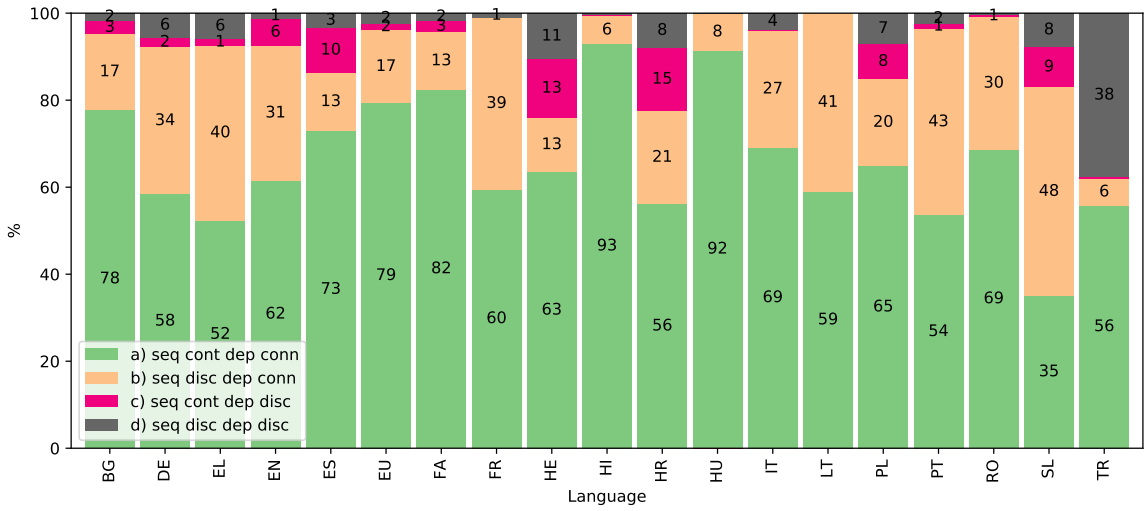

Figure 1: Proportion of a) sequentially-continuous and dependency-connected VMWEs (seq cont dep conn), b) sequentially-discontinous but dependency-connected VMWEs (seq disc dep conn), c) sequentially-continuous but dependency-disconnected VMWEs (seq cont dep disc) and d) sequentially-discontinuous and dependency-disconnected VMWEs (seq disc dep disc), for each language in the 2018 training dataset.[4]

The full implementation of the two systems is published under the GNU GPL License v 3.0 and available at `http://github.com/erwanm/adapt-vmwe18`.

2 Dependency Tree Approach

2.1 Motivations

The previous edition of this shared task (Savary et al., 2017) found that a significant minority of VMWEs in most language datasets were discontinuous, that is, VMWEs whose individual tokens did not necessarily occur as an uninterrupted sequence within a sentence (e.g. *take something into account*). Therefore, a system able to handle discontinuities could potentially make significant gains in performance. For a sequential tagging method to be able to handle these discontinuities, sufficiently large word windows would need to be considered. However, it can be difficult to determine an ideal word window size that can cover a majority of cases and yet be tractable.

Nevertheless, we hypothesise that tokens in a VMWE should be directly connected to each other via their syntactic dependency links, even if they are not sequentially continuous. We expect that most VMWEs in a given language would tend to be fully dependency-connected, regardless of whether they are sequentially continuous or not. If this is the case indeed, then a tree-based tagging method that exploits these direct dependency links should perform well, even in sequentially-discontinuous VMWEs.

To estimate the potential gains to be made through such a dependency tree-based tagging method, we counted the proportion of VMWEs in each language that are sequentially continuous/discontinuous and dependency-connnected/not fully dependency-connected. Figure 1 shows the proportions of each continuity and connectedness combination in each language. For the purposes of this figure, a VMWE is deemed to be sequentially continuous if each of its member tokens appear in a sentence sequentially with no tokens foreign to the VMWE breaking this continuity. Analogously, a VMWE is deemed to be dependency connected if its member tokens are fully connected via their syntactic dependency links. That is, all member tokens of a dependency-connected VMWE, depend on another member with the exception of at most one member, which depends on a non-member token (i.e. the head for the whole VMWE). We also consider a VMWE to be dependency connected if it has more than one member depending on the same non-member token. But if at least two members depend on two different non-member tokens, then the VMWE is deemed to be dependency disconnected for our purposes.

```
<Feature xsi:type="Current-Parent" name="example">
  <Ycur xsi:type="Label" value="label_O"/>
  <Ypar xsi:type="Label" value="label_I"/>
  <TestX value="self::TOKEN[@upos=='P']"/>
  <TestX value="ancestor::TOKEN[@upos=='V']"/>
</Feature>
```

Figure 2: Example of an XCRF feature generated from the French training set. The value of this feature is true if and only if the current node is labeled O, its parent node is labeled I,[8]the attribute *upos* of the current node has value *P* and the current node has an ancestor containing value *V* for its attribute *upos*.

Not surprisingly for most languages, the majority of VMWEs are both sequentially continuous and dependency connected (a in Figure 1), but the proportion of sequentially discontinuous but dependency connected VMWEs (b) is quite significant. In fact, the vast majority of the VMWEs in all languages are dependency connected (a + b), confirming that there are significant gains in performance to be made by developing a method that is able to exploit the connectedness of dependency trees. The other two combinations, (c and d), tend to be a minority in most languages. In some cases they are negligible (less than 10%), but in others they are relatively significant (in particular for Turkish). We do not propose strategies to deal with these combinations in this paper and leave their further analysis for future work.

2.2 Tree-Structured Dependencies in Conditional Random Fields

Conditional Random Fields (CRFs) can in theory handle dependencies of any level of complexity between variables (Lafferty et al., 2001; Sutton and McCallum, 2012).[5] In practice, however, most applications where CRF proved sucessful are based on the most simple kind of dependency network, which takes only sequential dependencies into account; this is largely due to the high computational cost of adding dependency relations between variables for training a CRF model. Nonetheless, there are several CRF software tools which are able to handle non-sequential dependencies; in this work we chose XCRF, a Java library for labeling XML trees (Jousse et al., 2006; Jousse, 2007).[6]

Based on the analysis presented in §2.1, our approach relies on the dependency tree structure of a sentence, provided in the Shared Task data for most languages (see §4). XCRF can deal with three kinds of dependency relations between nodes in a tree; a group of inter-dependent nodes is called a *clique*, and the different kinds of cliques, called the *clique levels*, are:

- Level 1: the label of the current node does not depend on any other label;
- Level 2: the label of the current node depends on its parent label;
- Level 3: the label of the current node depends on its parent label as well as on the label of its next sibling.

2.3 Implementation

2.3.1 Feature Generation

The XCRF software requires boolean features which consist of two parts: the *clique* is the set of classes for the current node (level 1), possibly with its parent (level 2) and possibly with its next sibling (level 3); the second part is a set of *tests* expressed as XPath expressions over the XML tree. Figure 2 shows an example of such a feature.

After converting the *cupt* dataset to a set of XML tree, the system iterates through the data in order to collect and count all the existing combinations of *cliques* and *tests*. Atomic tests are based on a triple $\langle Neighbourhood, Attribute, Value \rangle$, where: $Neighbourhood$ describes a set of target nodes,

[4]In the case of Lithuanian (LT) the dependency structure was not provided; in this particular case, VMWEs are arbitrarily represented as fully dependency-connected in figure 1.

[5]Here the word *dependency* refers to the probabilistic concept of conditional independence.

[6]`http://treecrf.gforge.inria.fr/`. Last verified: May 2018.

[8]The labeling scheme is explained in §4.

which can be: the node itself, its parent, its children its preceding siblings or its following siblings; $Attribute$ specifies the attribute to test: the lemma, the Part-of-Speech tag (*upos*) or the dependency relation (*deprel*); $Value$ is the value of the attribute. Additionally, conjunctions of a pair of tests are generated by cartesian product for each node. All the possible features for XCRF are generated by combining each test with the actual cliques that this node belongs to: for example, if the node has a parent, each of the tests is counted at both clique levels one and two. Finally, the collected frequencies (by clique, by clique level, by test and by combination clique+test) are used to calculate the probabilities used for feature selection.

2.3.2 Feature Selection

In order to select the most relevant features but also to avoid memory issues with XCRF, we implemented three different feature selection methods; the maximum number of features is a parameter (we used 25,000 for most languages). The number of features are shared equally between the atomic tests and the multi-tests, as well as between the different clique levels. Thus for every clique level, we reduce the number of features depending on the method, as explained below.

- Clique-blind: This method assumes that features with a similar proportion of true and false cases are more likely to help predict the clique. To this end, for every feature we take the minimum frequency difference between the case where the test is true and where it is false, and select the features for which this value is the highest. This method is based on the frequency of the test only, without taking into account the relation between the test and the clique within a feature.

- Binary Split: with this method, we select the tests which behave the most differently when occuring with the default clique versus with another clique. The default clique is defined as the one which contains only the default class (i.e. the class for the tokens which are not part of an expression). More precisely, we select the tests which maximize $|p(test = T|dc = T) - p(test = T|dc = F)|$, where dc is the default clique.

- Conditional entropy: $H(Y|X) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log \frac{p(x)}{p(x, y)}$, where Y is the set of cliques and X is the boolean test. The features which have the lowest conditional entropy are selected.

Additionally, a boolean option specifies whether to use the probability of the clique as a weight. That is, by not using a clique prior, it is assumed that all the cliques are equally important, even though this is not representative of the actual clique distribution.

3 Sequential Approach

3.1 Motivations

The sequential CRF approach was originally implemented as a simple and robust alternative to the dependency-tree approach (see §2). It uses only the lemma and the POS tag as features, so that it can be applied to a dataset even if no syntactic information is provided.[9] This can be useful in particular in the case of languages for which no (good) parser exists.

As explained in §2.1, the majority of the VMWEs are continuous in almost every language. Thus sequential CRFs are potentially capable of identifying most VMWEs, even though they are not well equipped to capture discontinuous VMWEs (at least when the VMWE components are distant from each other). Moreover, the efficiency of sequential CRFs implementations makes it possible to optimize the parameters of the model, in particular the length of the window of tokens to consider around the target token. Using this method, we expect the system to achieve a high performance on the continuous cases.

[9] In fact, the original intention was to use the sequential approach only for Lithuanian and Slovenian (see §2.2); the high level of performance achieved by this method motivated its application to all the datasets.

3.2 Implementation

To train a CRF model, we use the Wapiti sequence labelling toolkit (Lavergne et al., 2010).[10]. As in most sequence labelling software tools, the features to be used in the model are specified as a list of patterns in a template file; each pattern describes the observations to use as features for predicting the current token.

The *CRF-Seq-nocategs* system implements a simple brute-force method to optimize the sequential CRF model, similar to the one used in (Moreau and Vogel, 2018): for every dataset, we generate a large set of templates by varying the use of a column or not (lemma, POS or both), the maximum length of the window (i.e. the number of tokens before and after the target token), and the maximum size n of n-grams to use in the representation of the sequence of tokens. During the development stage, a model is trained using every template, and each such model is evaluated against the development set; then we simply select the model which achieves the highest performance.

4 Data and Training Process

For both systems, the data is first converted to a sequential labelling scheme which can be any of the following options:

- IO: a token which belongs to a VMWE is labelled *I*, any other is labelled *O*;
- BIO: IO with an additional label *B* for the first token in a VMWE;
- BILOU: BIO with label *L* for the last token in a VMWE and label *U* for a single token span.

Embedded and overlapping VMWEs cannot be represented adequately in any of these labelling schemes. Additionally, three options control how categories are handled: the default *joint* preserves categories by suffixing a label with the category name, e.g. *I_IAV*; in the *independent* mode, a distinct version of the data is created for every category which contains only the expressions belonging to this category, and an independent model is trained for each category; finally the *ignore* mode does not take categories into account. The *CRF-DepTree-categs* system uses the best of the first two options by language, while *CRF-Seq-nocategs* uses the last option.

The *CRF-DepTree-categs* system also requires converting the sentences to XML dependency trees. The Shared Task data files are provided in *.cupt* format, and include a column *HEAD* which contains the id of the parent for every token.[11] Due to the complexity of the training process and memory issues with the XCRF library, we had to reduce the size of the training set to a maximum 8,000 sentences.[12]

For every of the 19 datasets, the two systems have been trained using the training set provided; various parameters were tuned using the development set if provided, or a 20% subset of the training set if not. For *CRF-DepTree-categs* (resp. *CRF-Seq-nocategs*) we selected the model which achieves the highest performance for the MWE-based evaluation (resp. token-based evaluation).

5 Results and Discussion

Table 1 gives a brief summary of the performance of our two systems in the PARSEME VMWE Shared Task 2018.[13] The two systems performed rather consistently across the 19 languages compared to other participant systems: *CRF-Seq-nocategs* ranked from position 1 to 6 for all but one language, and *CRF-DepTree-categs* ranked from position 2 to 7 for all but two languages (according to the token-based evaluation). Both performed particularly well for Hindi, quite bad for Hebrew and especially bad for Lithuanian, but overall the two methods work fairly well independently from the language.

[10] `https://wapiti.limsi.fr/`. Last verified: May 2018.

[11] This information is provided in full for 17 languages out of 19, excluding Lithuanian (no information in the *HEAD* column) and Slovenian (incomplete information in the *HEAD* column); for these two languages the predictions made by *CRF-Seq-nocategs* are used. For several languages, the dependency tree can have multiple roots (DE, EU, HU, IT, PL, TR).

[12] Thus the *CRF-DepTree-categs* system does not use the full training set for the following datasets: BG, EU, FR, HE, IT, PL, RO, SL.

[13] The results cannot be detailed and analyzed fully in this paper, because there are many categories of evaluation and languages; for more details, see the overview paper (Ramisch et al., 2018) and the full official results at `http://multiword.sourceforge.net/PHITE.php?sitesig=CONF&page=CONF_04_LAW-MWE-CxG_2018&subpage=CONF_50_Shared_task_results`.

System	MWE-based evaluation				Token-based evaluation			
	Precision	Recall	F1-score	Rank	Precision	Recall	F1-score	Rank
Best system (TRAVERSAL)	67.58	44.97	54.00	1	77.41	48.55	59.67	1
CRF-Seq-nocategs	56.13	39.12	46.11	4	73.44	43.49	54.63	2
CRF-DepTree-categs	52.33	37.83	43.91	6	64.65	41.56	50.60	5
Median system (GBD-NER-standard)	36.56	48.30	41.62	7	41.11	52.21	46.00	7

Table 1: Performance and ranking of the two systems at the PARSEME VMWE Shared Task 2018 (closed track, 13 participant systems, macro-average scores).

Lang.	Percentage of **continuous** expressions found by system:				Percentage of **non-continuous** expressions found by system:			
	none	both	*CRF-Seq*	*CRF-DepTree*	none	both	*CRF-Seq*	*CRF-DepTree*
BG	32.56	42.65	**18.07**	6.72	75.26	12.89	4.12	**7.73**
DE	60.30	23.22	**8.99**	7.49	67.38	8.15	4.29	**19.31**
EL	39.64	37.45	**16.73**	6.18	62.83	7.52	5.75	**23.89**
EN	60.34	18.31	**16.27**	5.08	87.86	0.00	0.00	**12.14**
ES	39.55	23.40	**25.35**	11.70	86.52	0.71	2.84	**9.93**
EU	20.39	62.90	**12.78**	3.93	79.57	5.38	0.00	**15.05**
FA	19.60	63.82	**12.31**	4.27	74.76	8.74	7.77	**8.74**
FR	44.48	36.65	**13.52**	5.34	61.75	12.44	5.53	**20.28**
HE	88.80	0.00	0.00	**11.20**	95.76	0.00	0.00	**4.24**
HI	23.66	59.78	6.88	**9.68**	42.86	31.43	0.00	**25.71**
HR	52.22	17.06	**17.41**	12.63	79.81	2.88	3.85	**12.98**
HU	14.06	71.73	**9.99**	4.22	38.46	24.62	12.31	**24.62**
IT	51.34	20.00	**19.40**	8.06	77.98	1.19	0.00	**19.05**
PL	24.93	36.57	**25.21**	13.30	64.29	7.79	3.90	**24.03**
PT	34.50	33.87	**24.60**	7.03	58.33	4.58	0.42	**36.67**
RO	12.98	57.76	**26.46**	2.80	20.92	56.63	**13.78**	8.67
TR	46.63	35.10	**12.02**	6.25	71.14	15.10	**7.05**	6.71

Table 2: Percentage of continuous and non-continuous expressions found by (1) none of the two systems, (2) both of them, (3) *CRF-DepTree-categs* only and (4) *CRF-Seq-nocategs* only.[16] The percentage is based on the number of expressions according to the gold-standard labels, therefore these figures do not take false positive cases into account (thus are akin to recall statistics: the recall measure for a given system corresponds to the sum of the percentage for this system and the one for both). For every case, the highest of the two values between (3) and (4) is displayed in bold.

Although the simple sequential approach significantly outperforms the more sophisticated dependency tree-based one, there are indications that the latter might be able to deal with more complex cases: in the special evaluation categories for discontinuous VMWEs and unseen-in-train VMWEs, *CRF-DepTree-categs* ranks respectively 3rd and 4th, i.e. comparatively better than in the other evaluation categories.[14] This suggests that despite its moderate success at detecting VMWEs in general, the method might be good at capturing some of the hardest cases; in order to confirm this interpretation, we analyze the predictions made by both systems by contrasting the continuous and discontinuous cases: table 2 unambiguously shows that even if both systems are able to identify both kinds of expressions, each system tends to specialize on a specific kind; for most languages, *CRF-Seq-nocategs* identifies significantly more continuous expressions than *CRF-DepTree-categs*, but the latter identifies significantly more discontinuous expressions than the former.

6 Conclusion and Future Work

In this paper we presented two CRF-based systems for detecting VMWEs: *CRF-DepTree-categs* which exploits the dependency structure of the sentence, and *CRF-Seq-nocategs* which implements a simple sequential approach. While the latter achieved better performance in the PARSEME VMWE Shared Task 2018, our analysis shows that the two methods are complementary: *CRF-Seq-nocategs* identifies contin-

[14] For the sake of comparison, *CRF-Seq-nocategs* ranks 9th for discontinuous VMWEs and 7th for unseen VMWEs.

[16] Language with no dependency information (Lithuanian and Slovenian) are excluded; see §4.

uous VMWEs better, while conversely *CRF-DepTree-categs* identifies discontinous VMWEs better. As a consequence, combining the two approaches into a single system seems a very promising direction for future work.

Acknowledgements

The ADAPT Centre for Digital Content Technology is funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund.

References

Florent Jousse, Rémi Gilleron, Isabelle Tellier, and Marc Tommasi. 2006. Conditional Random Fields for XML Trees. In *Proceedings of the ECML Workshop on Mining and Learning in Graphs*.

Florent Jousse. 2007. *XML Tree Transformations with Probabilistic Models*. Theses, Université Charles de Gaulle - Lille III, October.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale CRFs. In *Proceedings the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 504–513. Association for Computational Linguistics, July.

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 Agata Savary Stella Markantonatou, Carlos Ramisch and Veronika Vincze, editors, *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 114–120, Valencia, Spain, April. Association for Computational Linguistics.

Erwan Moreau and Carl Vogel. 2018. Multilingual Word Segmentation: Training Many Language-Specific Tokenizers Smoothly Thanks to the Universal Dependencies Corpus. In Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 7-12, 2018. European Language Resources Association (ELRA).

Erwan Moreau, Ashjan Alsulaimani, Alfredo Maldonado, Lifeng Han, Carl Vogel, and Koel Dutta Chowdhury. 2018. Semantic Re-Ranking of CRF Label Sequences for Verbal Multiword Expression Identification. In Stella Markantonatou, Carlos Ramisch, Agata Savary, Veronika Vincze, editor, *to appear*. Language Science Press.

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.

Agata Savary, Carlos Ramisch, Silvio Ricardo 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*, pages 31–47, Valencia.

Charles Sutton and Andrew McCallum. 2012. An Introduction to Conditional Random Fields. *Found. Trends Mach. Learn.*, 4(4):267–373, April.