Mumpitz at PARSEME Shared Task 2018: A Bidirectional LSTM for the Identification of Verbal Multiword Expressions

Rafael Ehren
Dept. of Comp. Linguistics
Heinrich Heine University
Duesseldorf, Germany
ehren@phil.hhu.de

Timm Lichte
Dept. of Comp. Linguistics
Heinrich Heine University
Duesseldorf, Germany
lichte@phil.hhu.de

Younes Samih
Dept. of Comp. Linguistics
Heinrich Heine University
Duesseldorf, Germany
samih@phil.hhu.de

Abstract

In this paper, we describe Mumpitz, the system we submitted to the PARSEME Shared Task on automatic identification of verbal multiword expressions (VMWEs). Mumpitz consists of a Bidirectional Recurrent Neural Network (BRNN) with Long Short-Term Memory (LSTM) units and a heuristic that leverages the dependency information provided in the PARSEME corpus data to differentiate VMWEs in a sentence. We submitted results for seven languages in the closed track of the task and for one language in the open track. For the open track we used the same system, but with pretrained instead of randomly initialized word embeddings to improve the system performance.

1 Introduction

Multiword Expressions (MWEs) are linguistic expressions that consist of multiple lexemes and exhibit some form of idiomaticity. The term idiomaticity denotes the fact that the properties of the whole cannot (fully) be predicted by the properties of the components. Idiomaticity can surface on the lexical, syntactic, semantic, pragmatic and/or statistical level (Baldwin and Kim, 2010) and it is a reason why MWEs manifest a well-known challenge for natural language processing (NLP) tasks as parsing and machine translation.

The PARSEME shared task (ST) focuses only on a subset of MWEs, namely verbal multiword expressions (VMWEs), since their attributes like discontinuity (i.e. non-adjacency of the VMWEs' components) and semantic or syntactic ambiguity, make them especially challenging for automatic processing (Ramisch et al., 2018). The goal of the ST is the identification of VMWEs in running text. Besides extraction, which is the process of finding new MWE types, identification of MWE tokens in running text is the most important task with regard to MWE processing. In the context of the ST VMWEs are categorized into types like verb-particle constructions (VPC), light verb constructions (LVC), idioms (VID), inherently reflexive verbs (IRV) or multi-verb constructions (MVC). The ST is multilingual, i.e. the PARSEME organizers provide annotated corpora in 20 different languages which are split in train, dev and test sets. Competing teams can choose to only incorporate the features provided in these data sets or use external features such as MWE lexicons, word embeddings, etc. Systems solely based on preexisting features compete in the closed track, systems that also use external data in the open track of the ST.

Our system, Mumpitz, is a bidirectional recurrent neural network (BRNN) with long short-term memory (LSTM) units coupled with a heuristic that distinguishes different VMWEs in a sentence. We submitted results for 7 of the 20 languages to the closed track and for one language to the open track. In the next section we will provide an overview over last year's entries to the ST, before we will describe Mumpitz in detail. Subsequently, we will present the results and discuss the performance of the system as well as its possibilities for improvement.

2 Related Work

In this section, we will focus on the system description papers that were submitted to the PARSEME shared task 1.0.

The only (official) entry to the open track of last year's edition of the shared task came from Nerima et al. (2017) who employed a multilingual constituency parser that consists of a generic parsing module that can be adapted to the needs of different languages. The key component for the identification task is its manually built lexicon that contains information about collocations, i.e., a relational database.[1] During the parsing process the input is matched against entries in the database to identify VMWEs. Another parsing approach that in contrast does not rely on external resources and thus was submitted to the closed track stems from Simkó et al. (2017). They trained a dependency parser by merging the training MWE annotation into its dependency annotation. A further parsing-based system came from Al Saied et al. (2017) who used a transition-based dependency parser.

Boroş et al. (2017) entered the shared task with a Conditional Random Fields (CRF) classifier. This classifier decides for every word in an input sequence (sentence) if it is part of a VMWE or not with the help of lemma and part-of-speech features for a certain window of words around it. More precisely, they did this in two steps. First the classifier labeled the head of the VMWE, the verb, and then, in a second step, the words that link to the head were identified. A strategy that resulted in an increase in precision according to the authors. Maldonado et al. (2017) also employed a CRF model but instead of only one set of features they used several – one for every broad language family. In addition to the official results of the CRF classifier, they implemented a postprocessing step that would have been submitted to the open track if the results had been achieved in time. In this step, a re-ranking of the 10 most likely output sequences of the CRF classifier was conducted by calculating context vectors using a third-party corpus. Then those vectors were used to calculate new features that were in turn fed to a supervised regression algorithm.

Since they also used a Bidirectional Recurrent Neural Network (BRNN), the most similar approach to ours came from Klyueva et al. (2017). The main differences (except for the hyperparameters) are the use of gated recurrent units (GRU) instead of Long Short-Term Memory (LSTM) units and randomly initialized word embeddings instead of pretrained word embeddings in their network.

3 System Description

3.1 Why a BRNN with LSTM units?

Mumpitz is a Bidirectional Recurrent Neural Network (BRNN) with Long Short-Term Memory (LSTM) units coupled with a heuristic that leverages the dependency information provided in the PARSEME corpus data to differentiate VMWEs in a sentence. Since, in recent years, RNNs have proved to be very efficient on sequence processing tasks like POS tagging (Labeau et al., 2015) or named entity recognition (NER) (Chiu and Nichols, 2016) and the identification of MWEs can be seen as a sequence labeling task, this choice was quite natural. The great advantage of RNNs over vanilla neural networks is that they are able to draw on an entire history of inputs and not just the most recent input. The following shows a forward propagation step of a simple RNN where g is an activation function like tanh or ReLU, h_t is the hidden state at time step t, x_t is the input at time step t, b_h is the bias vector and W_{hh} and W_{hx} are weight matrices:

$$h_t = g(W_{hh}h_{t-1} + W_{hx}x_t + b_h) \tag{1}$$

The important thing to notice is that the hidden state of the previous time step, h_{t-1}, influences the hidden state of the current time step. Consider the following example from the English train set of the ST:

(1) The Knight of the Sad Countenance does not give up his obsessions.

[1]While collocations are (and were) not considered MWEs according to the PARSEME annotation guidelines, they ironically refer to VMWE categories of the shared task as "verbal collocations". Just another instance of how differently the term is used.

At time step $t = 10$, when the network reads the word *up*, it will have remembered that it read *give* shortly before and thus it could infer that the preposition is part of the VPC *give up*. This example also illustrates a shortcoming of simple RNNs: Just as the co-occurrence of *give* is an indication for *up* being part of a VPC, the co-occurrence of *up* is also an indication for *give* being a part of a VPC. But because the RNN only uses the information of the inputs from the previous and not from future time steps, it cannot use the information that *up* follows *give*. That's why bidirectional RNNs are used for such tasks. As the name suggests, BRNNs read the input from left to right and from right to left and use the hidden states from both passes to make a prediction:

$$\overrightarrow{h}_t = g(\overrightarrow{W}_{hh}\overrightarrow{h}_{t-1} + \overrightarrow{W}_{hx}x_t + \overrightarrow{b}_h) \tag{2}$$

$$\overleftarrow{h}_t = g(\overleftarrow{W}_{hh}\overleftarrow{h}_{t+1} + \overleftarrow{W}_{hx}x_t + \overleftarrow{b}_h) \tag{3}$$

But also BRNNs have the well known problem of not being awfully good at learning long distance dependencies, because of vanishing/exploding gradients. This is especially problematic for some VMWEs as the following example from the German training sentences illustrates:

(2)

Es	scheiden	die Vertreter	von	Ruanda,	Argentinien,	Oman,	Nigeria	und	Tschechiens	aus.
it	drop	the representatives	of	Rwanda,	Argentine,	Oman,	Nigeria	and	Czechia	out.

'The representatives of Rwanda, Argentine, Oman, Nigeria and Czechia drop out.'

The German VPC *ausscheiden* ('drop out') can occur in split word order, where the particle *aus* can be very far apart from the base word *scheiden* – in the sentence above there are nine words in between them. These long distance dependencies are not unusual for VMWEs and are one of the main challenges in their processing.

LSTM units (Hochreiter and Schmidhuber, 1997) can be employed to tackle this problem. The structure of LSTMs allows them to "forget" certain features, while "remembering" others. For the example above this means that in theory the LSTM could remember that it has seen *scheiden* when it comes to *aus* and thus label it a VMWE.

3.2 Experimental setup for the closed track

We used Keras with TensorFlow as back-end to build our network (Chollet and others, 2015). Keras is a deep learning API written in Python that makes it very convenient to write complex neural network architectures in just a few lines of code. The back-end, TensorFlow, is an open source software library developed by Google Brain (Abadi et al., 2015) that is optimized for large-scale machine learning applications. This is achieved by defining the models as computational graphs which then can be split to distribute them across different servers.

The model we employed for the identification task is a BRNN with 100 LSTM units. The features that were fed as input to the BRNN were mapped to embeddings and concatenated. Of all the features that were provided in the train and test files, combinations of the following were tested as input for the neural network: word form (WF), lemma (L), universal POS (UP), language specific POS (LP), head of the current word (H) and the dependency relation to the head (D). The combination of features that achieved the highest F1 score on the German dev set consisted of L, LP and D and it was subsequently used for the predictions on the test sets for all the languages. Of course, it would have made more sense to choose the best performing set of features for every language or language family (as Maldonado et al. (2017) did), but we chose this course of action because we did not have the time to do so and we were mostly interested in the German data set. One of the reasons for this focus is the fact that two of the authors did some annotation work on the German corpus of this year's edition of the ST and we were eager to see how well a deep learning algorithm would do at this challenging data set.

To form the input for the network the lemmas were mapped to randomly initialized embeddings of dimensionality 50, while the embeddings for LP and D both had 20 dimensions. Thus, the concatenated feature vectors were of size 90 and every sentence was represented by a certain number of those feature vectors. Because Keras requires all input sequences to have the same length, this number had to be fixed.

This was achieved by padding all the feature sequences to the length of the longest sentence (which, for example, was 100 words long in the German corpus) with an arbitrary value of L, LP or D. Hence, every sentence was represented by a matrix of shape *maximum-sentence-length* $\times$ 90 (e.g. 100 $\times$ 90 for German).

To avoid overfitting, we applied a Dropout of 0.1 to the LSTM layer. Dropout is a regularization method that randomly deletes a specific number of nodes (in our case 10% of the nodes) during the training process. This basically has the effect that the network is inhibited from "memorizing" the training data too well which in turn prevents overfitting. We experimented with different dropout rates and found that the performance of the model decreased when using larger rates, so we maintained a rate of 0.1 for all the experiments.

The last part of the model is a softmax layer that returns the probability distribution over the tags. The focus was exclusively on the identification of VMWEs. The classifier did only decide if a certain word was part of a VMWE or not, that is, the classifier did not consider any of the VMWE type labels. One problem that arises by applying this strategy is that the different VMWEs in an sentence are not distinguished, because every word that belongs to a VMWE has the same label, even if they do not belong to the same expression.

Differentiating the words tagged as VMWEs in an sentence was the purpose of the heuristic mentioned above which worked in different steps: first, it identified all the words in a sentence that were labeled as VMWEs and had the universal POS-tag "VERB" and enumerated them. In the next step, every word that was a dependent of an enumerated verb received the same number as this verb. Finally, every verb not labeled as a VMWE by the classifier that had a dependent labeled as such, also got the VMWE label.

To summarize, here is how the classification proceeded: Every sentence was fed into the BRNN one word at a time, i.e. one feature vector at a time. At every time step the hidden states of the LSTM units[2] were passed on to the softmax layer to compute a prediction for the current word that included the information from all the time steps before and all the time steps after. For example, if a sequence of 100 words (feature vectors) was fed to the network, the output was a 100 $\times$ 2 matrix, i.e. the probabilities for every word if it belonged to a VMWE or not. The training was not conducted with a fixed set of epochs, but the loss of a cross validation set (10% of the train set) was monitored and the training was stopped when it did not decrease anymore for two consecutive epochs. The convergence was relatively fast, usually it took between 4 and 7 epochs to train. The loss was computed with binary cross-entropy while the optimization algorithm used was RMSProp. After the network made its predictions the heuristic differentiated the annotated VMWEs as described above.

3.3 Experimental setup for the open track

The model for the open track is essentially the same as for the closed track. The only difference is that we didn't use randomly initialized embeddings, but pretrained word embeddings. We used the word2vec implementation (more precisely the skip-gram model) of the python package gensim to create the word embeddings (Řehůřek and Sojka, 2010). The word vectors were trained on the DECOW16 corpus (Schäfer and Bildhauer, 2012) which is a German web corpus that comes in two variants: one with 20 billion tokens and full documents and one with 11 billion tokens and shuffled sentences. The latter was used for the training. The word embeddings have dimensionality 100 and represent lemmas, not the word surface forms. Besides the embeddings we trained ourselves, we also tested pretrained embeddings that were trained on Wikipedia with fastText (Bojanowski et al., 2016), an extension of word2vec.[3] After evaluation on the dev set we opted for the selftrained embeddings, as they performed considerably better then those pretrained with fastText. And since the training of embeddings on a very large corpus like DECOW16[4] is somewhat time-consuming, we only managed to submit results for the German dataset in time for the open track.

[2]The hidden states were activated with the hyperbolic tangent and the gates with the hard sigmoid function.

[3]Pretrained word vectors for 294 languages can be found here: `https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md`

[4]Which of course has to exist in the first place for a specific language.

Ahead of the training process every lemma in the vocabulary was mapped to its respective word2vec embedding to form an embeddings matrix which was then used instead of the random instantiations. For some words such an embedding did not exist (because it did not appear often enough in DECOW16), so in these cases the words were mapped to zero vectors. Like the random instantiations the pretrained word embeddings were concatenated with the (still randomly initialized) embeddings for LP and D which resulted in feature vectors of 140 dimensions to represent the words in the sentences. Everything else stayed exactly the same as for the setup of the closed track.

4 Results

Because the general ranking of the shared task seems to depend on the average performance over 19 languages and we only submitted results for 7 languages, it may not be a good indicator for Mumpitz' performance. Therefore we will concentrate on the language-specific evaluation. As already evident from the system description, Mumpitz did not make any predictions for the VMWE type, so we will also skip this part of the evaluation.

The language-specific results for the closed and the open track can be seen in Table 1. Since the selection of the features was conducted on the basis of how well it performed on the German dev set, it is not particularly surprising that Mumpitz achieved its highest rankings on the German test set and considerably lower rankings on other languages. For the closed track it ranks 5th (of 11) in the MWE-based and first (of 11) in the token-based evaluation. For the open track it ranks last (of 4) in the MWE-based, and again first (of 4) in the token-based evaluation. Furthermore Mumpitz ranks relatively high for Bulgarian (4th of 10), French (5th of 13) and Portuguese (4th of 13) in the token-based evaluation, but except for German it ranks quite low in the MWE-based evaluation.

System	**Track**	**Language**	**MWE-based**				**Token-based**			
			P	**R**	**F1**	**Rank**	**P**	**R**	**F1**	**Rank**
Mumpitz	closed	BG	75.12	46.42	57.38	6/9	86.99	48.16	62	4/10
Mumpitz	closed	DE	32.15	38.35	34.98	5/11	55.91	48	51.66	1/11
Mumpitz	closed	EL	45	30.54	36.39	8/10	73.21	36.82	49	6/11
Mumpitz	closed	ES	9.66	13	11.08	10/10	31.83	28.87	30.28	6/10
Mumpitz	closed	FR	56.8	33.53	42.17	7/12	81.25	38.86	52.57	5/13
Mumpitz	closed	PL	62.07	38.45	47.48	8/10	80.92	41.34	54.72	8/11
Mumpitz	closed	PT	44.77	47.2	45.95	7/12	63.96	52.37	57.58	4/13
Mumpitz-preinit	open	DE	43.37	36.14	39.43	4/4	70.5	44.62	54.65	1/4

Table 1: Language-specific results

For us, the most interesting part of the evaluation is the comparison of Mumpitz and Mumpitz-preinit, i.e. the comparison of the systems that were completely equal except for the fact that the former was fed randomly initialized word embeddings, while the latter was given pretrained word embeddings. The results seen in Table 1 were achieved with the word embeddings we trained ourselves on the DECOW16 corpus. We used the selftrained embeddings for the test set predictions, rather then the fastText embeddings, because the selftrained embeddings outperformed the fastText word vectors on the dev set. For both evaluation schemes, the precision was increased substantially (11.22 for MWE-based, 14,59 for token-based), while the recall decreased (2.21 for MWE-based, 3,38 for token-based). But since the increase in precision outweighed the decrease, the F1-Score in turn was increased by 4.51 percentage points for the MWE-based, and by 2.99 for the token-based evaluation.

When looking further into the evaluation, it shows that Mumpitz-preinit outperforms Mumpitz in almost every aspect of the continuity and number of tokens metrics (see Table 2 and 3).[5]

[5]VMWEs with adjacent components are continuous, VMWEs with non-adjacent components are discontinuous. The number of tokens refers to how many components a VMWE has.

System	Continuous VMWEs			Discontinuous VMWEs		
	MWE-based			MWE-based		
	P	R	F1	P	R	F1
Mumpitz	27.62	48.31	35.15	48.82	26.84	34.64
Mumpitz-preinit	37.38	43.82	40.31	61.76	27.27	37.84

Table 2: Continuity (DE)

System	Multi-token VMWEs			Single-token VMWEs		
	MWE-based			MWE-based		
	P	R	F1	P	R	F1
Mumpitz	46.63	27.79	34.83	24.35	63.09	35.14
Mumpitz-preinit	56.91	30.66	39.85	32.16	48.99	38.83

Table 3: Number of tokens (DE)

5 Conclusion and future work

Mumpitz treats MWE identification as sequence tagging problem using a BRNN with LSTM units. We tried to remain as faithful as possible to this neural approach, leaving orthogonal pre- or post-processing steps aside. The features used are lemma, language-specific POS-tag and the dependency relation to the head; feature selection was conducted based on German, for which Mumpitz also obtained the highest F1 measure as to token-based classification compared to competing systems. The heuristic used to detect the span of MWEs was deliberately kept simple and only consisted in adding MWE-classified tokens to the span of the governing verb. Within the open track, we used pretrained embeddings, which lead to considerable improvements.

There are several possibilities of improving Mumpitz that are worth exploring. One of the most obvious is to choose language-specific (or language family-specific) features instead of optimizing a feature set for one language and then use it for all the others. Furthermore it is unsatisfactory to rely on a heuristic for the differentiation of VMWEs appearing in a single sentence, rather than having the classifier do it. Connected to this issue is the categorization of the VMWE candidates into the different VMWEs types which Mumpitz is unable to do so. For example, if the classifier would assign different labels to different VMWEs in a sentence, that would already automatically distinguish them. Another possibility for improvement would be the usage of character-level embeddings as features. They could improve the performance of the network as well as the word embeddings did.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Hazem Al Saied, Matthieu Constant, and Marie Candito. 2017. The atilf-llf system for parseme shared task: a transition-based verbal multiword expression tagger. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 127–132, Valencia, Spain, April. Association for Computational Linguistics.

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

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Tiberiu Boroş, Sonia Pipa, Verginica Barbu Mititelu, and Dan Tufiş. 2017. A data-driven approach to verbal multiword expression detection. parseme shared task system description paper. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 121–126, Valencia, Spain, April. Association for Computational Linguistics.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

François Chollet et al. 2015. Keras. `https://keras.io`.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

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.

Matthieu Labeau, Kevin Löser, and Alexandre Allauzen. 2015. Non-lexical neural architecture for fine-grained pos tagging. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 232–237, Lisbon, Portugal, September. 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 2017)*, pages 114–120, Valencia, Spain, April. Association for Computational Linguistics.

Luka Nerima, Vasiliki Foufi, and Eric Wehrli. 2017. Parsing and mwe detection: Fips at the parseme shared task. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 54–59, Valencia, Spain, April. 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. 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.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. `http://is.muni.cz/publication/884893/en`.

Roland Schäfer and Felix Bildhauer. 2012. Building large corpora from the web using a new efficient tool chain. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 486–493, Istanbul, Turkey, May. European Language Resources Association (ELRA). ACL Anthology Identifier: L12-1497.

Katalin Ilona Simkó, Viktória Kovács, and Veronika Vincze. 2017. Uszeged: Identifying verbal multiword expressions with pos tagging and parsing techniques. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 48–53, Valencia, Spain, April. Association for Computational Linguistics.