TraPacc and TraPaccS at PARSEME Shared Task 2018: Neural Transition Tagging of Verbal Multiword Expressions

Regina Stodden
Heinrich Heine University
Düsseldorf, Germany
regina.stodden@hhu.de

Behrang QasemiZadeh
Heinrich Heine University
Düsseldorf, Germany
zadeh@phil.hhu.de

Laura Kallmeyer
Heinrich Heine University
Düsseldorf, Germany
kallmeyer@phil.hhu.de

Abstract

We describe the TraPacc system and its variant TraPaccS that participated in the closed track of the PARSEME Shared Task 2018 on labeling verbal multiword expressions (VMWEs). TraPacc is a modified arc-standard transition system based on Constant and Nivre's (2016) model of joint syntactic and lexical analysis in which the oracle is approximated using a classifier. For TraPacc, the classifier consists of a data-independent dimension reduction and a convolutional neural network (CNN) for learning and labelling transitions. TraPaccS extends TraPacc by replacing the softmax layer of the CNN with a support vector machine (SVM). We report the results obtained for 19 languages, for 8 of which our system yields the best results compared to other participating systems in the closed-track of the shared task.

1 Introduction

The PARSEME shared task on VMWE identification (Savary et al., 2017) is an initiative aiming at improving automatic methods for VMWE identification in a highly multilingual context. To foster this initiative, the 2018 shared task provides test and training corpora, annotated with VMWEs in 19 different languages[1], and a framework to evaluate supervised methods for identifying VMWEs (Ramisch et al., 2018). In this paper, we present TraPacc and its variant TraPaccS that address the VMWE identification task using a neural transition system.

The general idea behind TraPacc has been motivated by observations from the first edition of the shared task: Saied et al. (2017) showed that the *modified arc-standard transition* method by Constant and Nivre (2016) (hereinafter MAST) can be employed to outperform other systems in which VMWE identification is modeled as a sequence labeling task disregarding of their employed learning model, e.g., conditional random fields (CRFs) (Maldonado et al., 2017), and bidirectional recurrent neural networks (Klyueva et al., 2017). Accordingly, we adapt MAST for the purpose of the PARSEME shared task; but, in contrast to Saied et al. (2017), instead of using a linear SVM for learning and predicting transitions, we use a CNN preceded by a dimension reduction proposed in (QasemiZadeh and Kallmeyer, 2016; QasemiZadeh et al., 2017). Furthermore, TraPaccS extends the TraPacc system by replacing the *output softmax layer* of the CNN with a kernel SVM. The latter is motivated by research, such as Razavian et al. (2014) and Poria et al. (2015), that suggests using CNN combined with a more elaborate classifier (i.e., to use CNN only as a feature selection method) is likely to improve prediction performance.

The remainder of this paper is structured as follows: We describe our method in Section 2, we report results from the experiment and discuss them in Section 3. Section 4 concludes this paper.

2 Method and System Description

The backbone of TraPacc (and subsequently TraPaccS) is the idea behind the MAST proposed by Constant and Nivre (2016). MAST jointly predicts syntactic dependencies and MWEs in a sentence by

[1]Besides Arabic.

introducing a new *lexical stack* (and, accordingly a number of transitions specific to this lexical stack) to the arc-standard incremental dependency parsing (Nivre, 2004; Kubler et al., 2009). Since we are only concerned with the identification of MWEs, the *quintuple* model of MAST is reduced to a triple consisting of a) a buffer B of input tokens to be parsed, b) a stack S of lexical units that are partially processed, and c) a set of processed lexical units P. In effect, the identification of VMWEs becomes very simple with only four general types of transitions (apart from initialize and terminate):

- *Shift*: pushes the head of the buffer B to the stack S;
- *Merge*: reduces the top two elements of the stack S to one element in the stack;
- *Reduce*: is a pop operation, i.e., it removes the element on top of the stack S and adds it to the set of processed elements P;
- *Complete-MWE**: is, similar to the reduce transition, a pop operation, except that it additionally marks the removed element from the stack S as an MWE of the type *. In practice, Complete-MWE* is decomposed to several transitions: each transition is labelled by a specific type of VMWE category derived from the training corpora, e.g., Complete-MWE-*VPC.full*, Complete-MWE-*LVC.cause*, and so on.

Additional details and examples about the transition system can be found in Saied et al. (2017). Simply put, the VMWE identification problem is boiled down to finding relationships between words that form VMWEs. When using the modified arc-standard transition approach, the identification of VMWEs in a sentence of length n tokens requires solving $2 \times n$ classification subproblems (without removing the chance of finding most VMWEs in the sentence). However, in other methods, e.g., in a graph-based approach, the number of classification sub-tasks can soar to $n \times (n - 1)$. We assume that the overall VMWE identification problem can be done faster and with a higher accuracy because of the reduced number of classification subproblems in the MAST method.

During the training time, each of the above listed transitions will be converted to a training record, i.e., a labeled feature vector, with its label being the type of the transition and the features being the vector that shows the current state of the system. Each lexical unit in the buffer B and the stack S is accompanied by a set of annotations, e.g., its word form, lemma, part-of-speech-tag, and so on (i.e., whatever is asserted and provided as annotation in the training corpora). These annotations are transformed to (mostly) a set of binary features and (together with a number of other attributes such as the history of the transition system) constitute a feature vectors for each state of the transition system. In our implementation, we use all the features proposed in Saied et al. (2017) and add the following features to them:

- Besides the number of elements in the stack, we use the size of the buffer (i.e. the number of elements that it contains at any time) as an additional feature to take the length of input sentences into account.
- We add the length of tokens as an additional feature, which we believe can be helpful for identifying long single-token VMWEs—e.g., single-token VMWEs in German such as “freigesprochen” (lit: free spoken; to acquit so. of sth., VPC:full).
- In a few languages (e.g., EL, DE, IT), more than 5% of the VMWEs consist of more than three tokens; hence, we use four-grams generated from the elements in the stack and the buffer as additional features to bigram and trigrams used by Saied et al. (2017).
- Given the availability of additional annotations in this year’s shared task, compared to Saied et al. (2017), we could use more linguistic features such as universal part of speech tags and morphological information for a larger number of languages.

From this point, the task is to approximate a so-called static oracle using a supervised multi-way classifier to determine the ‘best’ next transition for the given states of the system. Our method for modelling the oracle differs from Saied et al. (2017) with respect to the classification model that we employ to model the oracle, i.e., instead of classifying high-dimensional and extremely sparse feature vectors using a SVM, we classify feature vectors of reduced dimension using CNN.

The resulting feature vectors from the above-mentioned feature extraction process are of high dimensionality, e.g., in the scale of millions depending on the size of the training corpus and the lexical and morphosynatic diversity of it. These vectors are also highly sparse (i.e., most components of the vector are zero and only a few are non-zero). For classification under these conditions, a linear SVM is usually a fast and reliable choice (Fan et al., 2008) but still overfitting and poor generalization is possible. Given the availability of GPU accelerated computing and in light of the recent advances in deep learning, we replace a classic classifier such as linear SVM—as used by Saied et al. (2017)—or a Perceptron classifier—as used by Constant and Nivre (2016)—with a simple multi-layer CNN.

Usually, when using a deep neural network, the above-stated feature extraction process is altered by using embeddings and particularly pre-trained word embeddings instead of binary-encoded lexical features—e.g., see Chen and Manning (2014). Since the use of pre-trained word embeddings is not an option for systems that participate in the closed track of the shared task[2], we use a data-independent dimensionality reduction method based on random projections. The usage of this dimension reduction method removes engineering issues associated with the use of high-dimensional sparse vectors as input for deep learning methods, e.g., by reducing the memory required to fit the batch of vectors used during training and by decreasing the training time (specially for large corpora).

Our dimension reduction method is based on the positive-only random projections proposed by QasemiZadeh and Kallmeyer (2016). Put simply, an n-dimensional feature space consisting of p training records, represented by a matrix $\mathbf{M}_{p\times n}$, is mapped to an m-dimensional space $\mathbf{M}'_{p\times m}$ ($m \ll n$) by multiplying $\mathbf{M}_{p\times n}$ to a randomly devised projection matrix (of certain characteristics) $\mathbf{P}_{n\times m}$; i.e., $\mathbf{M}'_{p\times m} = \mathbf{M}_{p\times n} \times \mathbf{P}_{n\times m}$. In the method proposed by QasemiZadeh and Kallmeyer (2016), $\mathbf{P}_{n\times m}$ is a randomly-generated matrix in which only a few elements of each row and column of $\mathbf{P}$ have the value 1 and the remaining are 0. As shown in (QasemiZadeh et al., 2017; QasemiZadeh and Kallmeyer, 2017), the matrix multiplication $\mathbf{M}_{p\times n} \times \mathbf{P}_{n\times m}$ can be efficiently computed using a hash function and a number of addition operations: Given an n-dimensional feature vector $\vec{v}$ that represents a state of the transition system, the corresponding m-dimensional vector $\vec{v'}$ is given by computing

$$\forall i \in \{1, \ldots, n\}, \quad \vec{v'}[|i\%m|] += \vec{v}[i],$$

in which $\vec{v}[i]$ and analogously $\vec{v'}[i]$ denotes the ith component of $\vec{v}$ and $\vec{v'}$, $|x|$ gives the absolute value of its input number x, and $\%$ is the modulo operator that gives the remainder after the division of i by m.

Once these vectors of the reduced dimension m are built, they are fed to our CNN model trained with respect to *categorical cross-entropy loss* using an *ADAM optimizer*. The CNN architecture is straightforward as shown in Figure 1. An input feature vector of the reduced dimension m (we use $m = 500$) is fed to a convolutional layer and it is transformed to a kernel of size 500×128, which is downsampled to a 250×128 kernel using max-pooling. The generated kernel is flattened and mapped to a densely-connected layer of size 512 where a *rectified linear unit* (ReLU) is used as activation function; this is followed by a small drop-out of rate 0.025. For TRAPACC, the last dense layer gives the predicted classes using a *softmax* activation function.

TRAPACC$_\text{S}$ slightly differs from TRAPACC in the way that it predicts class labels. As shown in Figure 1, after training the CNN model, TRAPACC$_\text{S}$ takes the intermediate dense layer of the CNN and uses it as an input feature vector to a Kernel SVM (we use a *radial basis function* (RBF) kernel). Hence, for TRAPACC$_\text{S}$, the CNN model is effectively reduced to a supervised feature extraction system. For implementation, we alter the Python code of Saied et al. (2017) and implement our CNN using *Keras* (Chollet and others, 2015) with TensorFlow as its backend.[3]

[2]The use of any NLP resource or process other than the provided training corpora is not allowed.

[3]Available for download from `https://github.com/rstodden/verbal-multiword-expression-st11`.

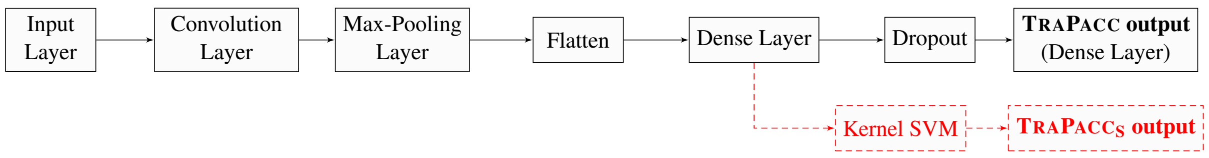

Figure 1: The classification model employed in TRAPACC and TRAPACC$_S$. In both systems the CNN model shown above is employed. CNN is used directly to generate the output of TRAPACC; but in TRAPACC$_S$ (drawn with dashed red lines), the intermediate dense layer of the CNN model is used as input feature vector for a kernel SVM classifier.

3 Result and Discussion

As implied in the previous section, we use a network of the same architecture for developing prediction models for all the languages in the shared task. During the development, the fixed parameters of our CNN model (e.g., the dropout rate) are decided by trial and error over the development set of DE (German) and FA (Farsi). For training, we use both the train and the development sets so that we decrease the chance of unseen VMWEs when testing the system. As mentioned earlier, we participate in the closed track of the shared task so features used in training and test are limited to those that are made available by organizers in the `.cupt` files (hence, some features, e.g., dependency parses, morphological information, and so on are missing for some languages).

Table 1 reports the official shared task results and ranks per language for both systems. Overall, based on our official submissions, with respect to the MWE-based F1 metric, TRAPACC and TRAPACC$_S$ rank third and second, respectively. In particular, compared to other participating systems, our TRAPACC and TRAPACC$_S$ show the best performance for 8 languages including RO (Romanian), HU (Hungarian), and EN (English); and on the contrary, they perform poorly for SL (Slovenian), PT (Portuguese), FR (French), HR (Croatian) and IT (Italian). For TR (Turkish), neither TRAPACC nor TRAPACC$_S$ converge and the obtained results are unusually bad (near to zero).

Concerning the comparison of TRAPACC and TRAPACC$_S$, the performance obtained from both systems remains very similar. At first sight, on the basis of per language comparison, TRAPACC seems to perform better than TRAPACC$_S$ for most languages; and, it is only for DE (German), EL (Greek), LT (Lithuanian), and SL that TRAPACC$_S$ yields better results than TRAPACC. However, using a rank correlation metric such as Spearman's ρ for comparing TRAPACC and TRAPACC$_S$ (where results are sorted by language), we observe a high correlation between the results obtained by the two systems—i.e. $\rho = 98.07$ (for MWE-based F1) and $\rho = 97.89$ (for Token-based F1)—compare it, for instance, to $\rho = 81.40$ when comparing MWE-based F1 scores between TRAPACC and *TRAVERSAL*, another participating system in the shared task. This observation suggests that our attempt (i.e., using RBF-SVM instead of softmax layer for classification) to enhance results has not been as effective as we expected it. This can be due to several reasons: E.g., we trained RBF-SVM only using its preset default parameter values whereas we know tuning the parameters of SVM can enhance results to a certain extent; similarly, we would achieve better results using a different kernel. These remain as unanswered research questions for our future investigations.

Concerning TR, the performance gap between TRAPACC (as well as TRAPACC$_S$) and other participating systems is apparent. Simply put, we failed to deliver any meaningful result for TR. Initially, we associate this failure to the presence of very long sentences, particularly in TR test and train corpora[4] (this is also the case for SL). However, analyzing sentence length in other corpora such as ES (which also contains long sentences but our systems perform relatively well for them) as well additional experiments (i.e., by removing very long sentences from train and test corpora and repeating experiments) ruled out the aforementioned hypothesis (i.e., the long sentence length aggravates poor performance). In our second hypothesis, we related this problem to the inconsistent lemmatization and tagging of the TR corpus, e.g., in a sentence only a few tokens are tagged with their part-of-speech (only 27.6% of the tokens in

[4]Some appear to be due to some mistakes.

	Corpus		**TraPacc Results**				**TraPacc$_S$ Results**					
	#VMWEs		***MWE-based***		***Tok-based***		***MWE-based***		***Tok-based***		**Best Using TraPacc**	
Lang	**Train+**	**Test**	**F1**	**Rank**	**F1**	**Rank**	**F1**	**Rank**	**F1**	**Rank**	**F1 MWE**	**F1 Token**
BG	6,034	670	60.83	2/9	62.35	2/10	52.57	8/9	53.47	8/10	61.05	64.18
DE	3,323	500	44.05	2/11	48.37	4/11	45.27	1/11	49.97	3/11	45.14	48.09
EL	1,904	501	46.43	3/10	49.14	5/11	49.76	1/10	53.1	3/11	50.6	52.73
EN	331	501	32.88	1/10	34.37	1/10	30.28	3/10	30.23	2/10	32.88	34.37
ES	2,239	500	31.64	3/10	38.04	4/11	33.98	1/10	39.75	2/11	32.33	38.47
EU	3,323	500	73.23	2/9	74.4	3/10	75.8	1/9	76.83	1/10	73.23	74.4
FA	2,951	501	75.48	3/9	78.12	4/10	74.23	4/9	77.03	5/10	75.48	78.12
FR	5,179	498	46.97	3/12	52.93	4/13	45.96	6/12	50.12	7/13	48.28	53.86
HE	1,737	502	20.37	3/8	24.26	4/10	16.95	6/8	18.97	6/10	21.97	26.08
HI	534	500	69.38	3/7	71.22	3/8	68.39	4/8	68.31	5/8	69.38	71.22
HR	1,950	501	43.39	3/9	49.97	2/10	44.27	2/9	47.95	4/10	47.94	52.39
HU	6,984	776	90.31	1/10	88	1/10	90.12	2/10	87.25	2/10	90.31	88
IT	3,754	503	38.52	2/11	40.64	4/12	33.33	4/11	34.02	8/12	39.7	41.45
LT	312	500	30.82	2/7	34.43	1/10	32.17	1/7	34.1	2/10	33.63	36
PL	4,637	515	60.54	2/10	63.68	3/11	59.86	4/10	61.51	6/11	62.50	65.05
PT	4,983	553	52.73	4/12	59.96	5/13	52.29	5/12	54.51	7/13	56.96	59.35
RO	5,302	589	85.28	1/9	85.69	1/10	83.36	2/9	83.45	5/10	85.28	85.69
SL	2,878	500	23.04	9/9	34.72	9/10	31.36	7/9	38.33	7/10	36.63	44.74
TR	6,635	506	1.61	8/9	4.68	9/10	0.78	9/9	3.01	10/10	39.34	44.09
			49.57	**3/13**	**53.09**	**3/13**	**49.74**	**2/13**	**52.13**	**4/13**	**53.28**	**56.42**

Table 1: Results and rankings for TraPacc and TraPacc$_S$ from official submission together with the best results we obtained using TraPacc after an additional adjustment of our system's parameters/features. The denominators in the rank column show the number of submitted systems per language. Train+ denotes the number of VMWE instances in both training and development set.

the TR training corpus has a POS tag), and it seems as if lemmas appear in wrong places, e.g., as shown in Table 2. Table 2 shows the annotations for a sentence in the TR training corpus: only a few tokens of the sentence are tagged with their part-of-speech (5th column). Moreover, for many tokens in the TR corpus, the lemmas seem to appear in the wrong places such as the case for the word "göstermek" (in line/position 11) for which its lemma (göster) is asserted wrongly in line/position 10.

Hence, we retrained our systems using only word forms and those that can be extracted from them (e.g., suffix, prefix, bigrams, etc.). As shown in Table 1 for the TraPacc system, the MWE-based F1 score increases from almost zero to 39.34. This leads us to another observation that despite our expectations, our CNN-based method is vulnerable to the presence of noise. Similar to TR, we observe that for PT, only 40.05% of the part-of-speech tags in the training data are available. When we removed features based on part-of-speech tags, the MWE-based F-Score increase by 4.23. For FR, only 4.6% of

ID	**WORD FORM**	**LEMMA**	**UPOS**	**XPOS**	**FEATS**	**HEAD**	**DEPREL**	**DEPS**	**MISC**	**MWE**
1	_	ülke	Noun	_	A3sg|Loc|P1pl	2	DERIV	_	_	*
2	Ülkemizdeki	_	Adj	Rel	_	4	MODIFIER	_	_	*
3	KOBİ	Kobi	Noun	Prop	A3sg|Nom|Pnon	4	POSSESSOR	_	_	*
4	potansiyelini	potansiyel	Noun	_	A3sg|Acc|P3sg	5	OBJECT	_	_	*
5	_	kullan	Verb	_	Able|Pos	6	DERIV	_	_	*
6	kullanabilmek	_	Noun	Inf1	A3sg|Nom|Pnon	7	ARGUMENT	_	_	*
7	için	için	Postp	PCNom	_	10	MODIFIER	_	_	*
8	bu	bu	Det	_	_	9	DETERMINER	_	_	*
9	çabayı	çaba	Noun	_	A3sg|Acc|Pnon	10	OBJECT	_	_	1:LVC.full
10	_	göster	Verb	_	Pos	11	DERIV	_	_	*
11	göstermek	_	Noun	Inf1	A3sg|Nom|Pnon	12	SUBJECT	_	_	1
12	şart	şart	Noun	_	A3sg|Nom|Pnon	0	PREDICATE	_	_	*
13	.	.	Punc	_	_	12	PUNCTUATION	_	_	*

Table 2: An example of untypical lemmatization and missing part-of-speech tags in sentences from the train set for TR.

the part-of-speech tags in the train data are available. Likewise TR and PT, removing features related to them leads to an increase of 1.31 in the MWE-based F1-Score. We add to this the observation that the linear-SVM+high-dimensional-sparse-vectors generalize better than CNN over the noisy input. This can be due to the network architecture that we use, criteria used when training, the employed dimension reduction method, or a combination of them. For instance, a slight change in the training procedure (i.e., the use of different conditions for terminating training) leads to an increase of 13.59 in the MWE-based F1-score for SL (the over-fitting problem).[5] The effect of these factors must be investigated further in our future work.

Last but not least, after slight modifications in the original implementation of Saied et al. (2017) and a careful feature selection process, we observe that modeling oracle using high dimensional sparse feature vectors and linear SVM (or multilayer perceptron) can be equally effective as TRAPACC; in other words, the gain from using a deep learning model instead of simpler models such as linear SVM is negligible in the closed track. However, the deep learning modules of TRAPACC and TRAPACC$_S$ permit the use of features such as word embeddings. These features are not easily accessible (and supposedly effective) to simpler learning methods such as linear SVM. As a result, using TRAPACC and TRAPACC$_S$ in the open track and exploiting more complex unsupervised features for their training remain questions for our future work.

4 Conclusion and Future Work

We report the results obtained from our methods TRAPACC and TRAPACC$_S$ for identifying VMWEs in the closed track of the PARSEME shared task. Among 13 participating systems in the closed track of the shared task, TRAPACC ranks and TRAPACC$_S$ ranks third in the official shared task results.

This research is a work in progress and it can be extended in several ways. Firstly, we can improve our results by removing limitations due to partaking in the closed track of the shared task. For instance, we could use word vectors and various association measures such as PMI and χ^2 that are obtained directly from large corpora in order to improve results. Similarly, the availability of large corpora would permit us to look into features that capture 'structural' properties of words and tokens that are inherently a strong cue of the presence of VMWEs, e.g. as seen for DE in which many single-token VMWEs are expected to appear also as multi-token VMWEs.

Our employed learning method is novel in the sense that it combines a data-independent dimension reduction with CNNs. However, it can be changed and improved in several ways, too. For instance, we are currently extending our input feature vectors to matrices, in which a vector replaces the scalar values currently used to represent the value of a feature. Comparing different dimension reduction methods, even removing them from the classification model, is another research avenue. The identification of VMWEs in our current models TRAPACC and TRAPACC$_S$ is carried out per VMWE category; instead, this can be replaced by a two-step process. I.e., we could use a binary classifier for detecting VMWEs disregarding of their category, followed by a category classification process. Similarly, we could utilize a better informed strategy for the training and parameter setting of our models. For example, instead of a language-agnostic method for setting parameters of CNNs (such as layer size and drop-out rate), we could adapt a language-wise methodology.

Acknowledgements

This work was supported by the CRC 991 "The Structure of Representations in Language, Cognition, and Science" funded by the German Research Foundation (DFG). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

[5]With these improvements, TRAPACC outperforms the TRAVERSAL system and it ranks first with respect to the average MWE-based F1 scores.

References

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October. Association for Computational Linguistics.

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

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. Association for Computational Linguistics.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874.

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@EACL 2017, Valencia, Spain, April 4, 2017*, pages 60–65.

Sandra Kubler, Ryan McDonald, Joakim Nivre, and Graeme Hirst. 2009. *Dependency Parsing*. Morgan and Claypool Publishers.

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.

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

Soujanya Poria, Erik Cambria, and Alexander Gelbukh. 2015. Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2539–2544, Lisbon, Portugal, September. Association for Computational Linguistics.

Behrang QasemiZadeh and Laura Kallmeyer. 2016. Random positive-only projections: Ppmi-enabled incremental semantic space construction. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 189–198, Berlin, Germany, August. Association for Computational Linguistics.

Behrang QasemiZadeh and Laura Kallmeyer. 2017. HHU at SemEval-2017 task 2: Fast hash-based embeddings for semantic word similarity assessment. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics.

Behrang QasemiZadeh, Laura Kallmeyer, and Peyman Passban. 2017. Sketching word vectors through hashing. *CoRR*, abs/1705.04253.

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.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. Cnn features off-the-shelf: An astounding baseline for recognition. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, CVPRW '14, pages 512–519, Washington, DC, USA. IEEE Computer Society.

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@EACL 2017, Valencia, Spain, April 4, 2017*, pages 127–132.

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.