Appendices

A Lexical categories in STREUSLE

Lexcat	SS	Definition
`N`	n.*	noun, common or proper
`PRON.POSS`	p.*	possessive pronoun
`POSS`	p.*	possessive clitic (*'s*)
`P`	p.*	adposition
`PP`	p.*	(idiomatic) adpositional phrase MWE
`INF.P`	p.*	semantically annotatable infinitive marker
`V`	v.*	**single-word** full verb or copula
`V.VID`	v.*	**MWE:** verbal idiom
`V.VPC.full`	v.*	**MWE:** full verb-particle construction
`V.VPC.semi`	v.*	**MWE:** semi verb-particle construction
`V.LVC.full`	v.*	**MWE:** full light verb construction
`V.LVC.cause`	v.*	**MWE:** causative light verb construction
`V.IAV`	v.*	**MWE:** idiomatic adpositional verb

Lexcat	Definition
`NUM`	number
`PRON`	non-possessive pronoun
`ADJ`	adjective
`ADV`	adverb
`DET`	determiner
`INF`	nonsemantic infinitive marker
`AUX`	auxiliary, not copula
`DISC`	discourse/pragmatic expression
`CCONJ`	coordinating conjunction
`SCONJ`	subordinating conjunction
`INTJ`	interjection
`SYM`	symbol
`PUNCT`	punctuation
`X`	foreign or nonlinguistic

Table 1: Lexcats (lexical categories) that are annotated for strong lexical units, i.e., single-word expressions or strong MWEs. Weak MWEs are treated as compositional and thus do not receive a holistic lextag or supersense. ***Left:*** Lexcats that require supersenses of the class designated in the second column: nominal (n.*), verbal (v.*), or adpositional/possessive (p.*). Verbal MWEs are syntactically subtyped in the lexcat, and the simple `V` lexcat applies to non-MWEs only. ***Right:*** Lexcats that are incompatible with supersenses. Most of these are defined in line with Universal POS tag definitions, but may also apply to MWEs. Definitions come from `https://github.com/nert-nlp/streusle/blob/master/CONLLULEX.md`.

B Baseline Implementation Details

Table 2 lists the hyperparameter values we found by tuning on the STREUSLE development set, with BERT pre-trained contextualized embeddings (large-cased; Devlin et al., 2019), predicted POS tags and lemmas. BERT parameters are not fine-tuned.

BiLSTM #layers	2
BiLSTM total dim. per layer	512
Learning rate	0.001
Batch size	64

Table 2: Hyperparameter values.

Our tagger uses the BERT (large, cased) pretrained model to produce input word representations; these input word representations are a learned scalar mixture of the BERT representations, following observations that the topmost layer of BERT is highly attuned to the pretraining task and generalizes poorly (Liu et al., 2019). The representation for a token is taken to be BERT output corresponding to its first wordpiece representation. We freeze the BERT representations during training.

The word representations from the frozen BERT contextualizer are then fed into a 2-layer bidirectional LSTM with 256 hidden units in each direction. The LSTM outputs then are projected into the label space with a learned linear function, and a linear chain conditional random field produces the final output.

For training, we minimize the negative log-likelihood of the tag sequence with the Adam optimizer, using a batch size of 64 sequences and a learning rate of 0.001.

We train our model for 75 epochs, and gradient norms are rescaled to a maximum of 5.0. We apply early stopping with a patience of 25 epochs. Our model is implemented in the AllenNLP framework (Gardner et al., 2018).

In our ablated models that use GloVe vectors and character-level CNNs instead of BERT, we use 200 output filters with a window size of 5 in the CNN. The input to the CNN are 64-dimensional character embeddings.

C Per-Lexcat STREUSLE Results

Table 3 shows STREUSLE test set results for the BERT tagger with only MWE constraints (no POS/lemma constraints), broken down by lexical category. The numbers reported here differ from the evaluation in table 1—these metrics are calculated by extracting the predicted and gold spans, and then computing an exact-match F1 measure between the predicted and gold sets.

Frequency counts are for STREUSLE-test; OOV token rates are relative to STREUSLE-train. Examples are lemmatized lexical units ("lexlemmas"). Lexlemmas are used to calculate OOV rates.

Lexcat	**Example**	**# Gold**	% OOV	**P**	**R**	**F**	**Lexcat**	**Example**	**# Gold**	% OOV	**P**	**R**	**F**
`N`	*food*	946	24.7%	85.5	88.9	87.2	`NUM`	*five*	41	17.1%	92.9	95.1	94.0
`PRON.POSS`	*my*	94	0.0%	98.9	93.6	96.2	`PRON`	*it*	393	0.0%	95.1	98.2	96.6
`POSS`	*'s*	1	0.0%	100.0	100.0	100.0	`ADJ`	*best*	532	8.9%	85.8	94.0	89.7
`P`	*with*	322	0.3%	88.0	93.2	90.5	`ADV`	*extremely*	358	2.1%	91.7	92.2	91.9
`PP`	*by far*	18	0.0%	87.5	77.8	82.4	`DET`	*the*	376	0.0%	92.4	96.8	94.5
`INF.P`	*to* see	20	0.0%	87.0	100.0	93.0	`INF`	*to*	36	0.0%	91.9	94.4	93.2
`V`	*go*	587	3.5%	90.0	95.4	92.6	`AUX`	*have*	160	0.0%	95.7	96.2	96.0
`V.VID`	*take time*	24	0.0%	64.3	37.5	47.4	`DISC`	*thanks*	21	4.5%	63.6	66.7	65.1
`V.VPC.full`	*turn out*	11	0.0%	58.3	63.6	60.9	`CCONJ`	*and*	204	0.0%	95.3	99.5	97.4
`V.VPC.semi`	*add on*	5	0.0%	50.0	60.0	54.5	`SCONJ`	*lest*	21	4.8%	90.5	90.5	90.5
`V.LVC.full`	*have fun*	8	0.0%	60.0	37.5	46.2	`INTJ`	*hey*	17	35.3%	78.6	64.7	71.0
`V.LVC.cause`	*give chance*	1	0.0%	0.0	0.0	0.0	`SYM`	*:)*	12	0.0%	100.0	75.0	85.7
`V.IAV`	*treat to*	17	5.9%	81.8	52.9	64.3	`PUNCT`	.	597	0.3%	99.0	99.7	99.3
							`X`	*etc*	2	50.0%	0.0	0.0	0.0

Table 3: STREUSLE test set results for the BERT-based tagger with only MWE constraints (no POS/lemma constraints), broken down by lexcat. The numbers reported here differ from the evaluation in table 1—these metrics are calculated by extracting the predicted and gold spans, and then computing an exact-match F1 measure between the predicted and gold sets.

D Per-VMWE Category PARSEME Results

Table 4 shows PARSEME (English) test set results for the BERT tagger with only MWE constraints (no POS/lemma constraints), broken down by VMWE category.

Frequency counts are for PARSEME-EN-test and reflect the number of gold MWEs; OOV token rates are relative to STREUSLE-train. Examples are lemmatized lexical units ("lexlemmas"). Lexlemmas are used to calculate OOV rates.

PARSEME 1.1				**MWE-based**			**Token-based**		
VMWEs (EN-test)	**Example**	**# Gold**	% OOV	**P**	**R**	**F**	**P**	**R**	**F**
`V.VID`	*tide turn*	79	80.6%	8.8	17.7	11.8	11.8	20.9	15.1
`V.VPC.full`	*bring in*	146	44.3%	41.8	59.6	49.2	43.3	63.7	51.5
`V.VPC.semi`	*speak up*	26	61.5%	12.7	30.8	18.0	12.5	30.8	17.8
`V.LVC.full`	*make promise*	166	90.5%	30.8	4.8	8.3	38.2	6.1	10.6
`V.LVC.cause`	*yield result*	36	100.0%	0.0	0.0	0.0	0.0	0.0	0.0
`V.IAV`	*turn into*	44	52.2%	30.4	38.6	34.0	29.1	37.8	32.9
`V.MVC`	*cross examine*	4	80.0%	0.0	0.0	0.0	0.0	0.0	0.0

Table 4: PARSEME (English) test set results for the BERT tagger with only MWE constraints (no POS/lemma constraints), broken down by VMWE category.

E VMWE Performance in STREUSLE vs. PARSEME

PARSEME appears to be a much more challenging task, even considering just the VMWE identification performance in STREUSLE: in the main text, compare BERT model F-scores of 64% in STREUSLE versus 40% in PARSEME (where the state-of-the-art result is 42%). Why is this the case? We suspect at least three factors are at play:

- Substantial domain shift: PARSEME covers a wide range of genres, including literary genres, which is likely to contribute to lower precision and recall in general.
- Base rate of MWEs: STREUSLE contains about 10 times as many MWEs per word as PARSEME, in part due to the comprehensive nature of MWE annotation in STREUSLE. Considering just verbal MWEs per word, STREUSLE-train has 763/44,815 = 1.7% and STREUSLE-test has 66/5,381 = 1.2%, whereas PARSEME-test has 501/71,002 = 0.7%. So it is not surprising that the STREUSLE-trained tagger would overpredict VMWEs in PARSEME. Note that precision is lower than recall overall and for most VMWE subtypes.
- OOV rate: MWE identification of lexical items unseen in training is generally more challenging. We see above that the VMWE vocabularies of STREUSLE and PARSEME are largely disjoint, with OOV rates above 50% for most subtypes. This would be expected to mainly impact recall, and in fact, the higher the OOV rate, in general the lower the recall. In particular, recall is much lower than precision for the LVC.full subtype, with an OOV rate of 90.5%, suggesting that it is able to correctly identify some known LVCs but unable to generalize to new ones. The 8 instances correctly identified had, in fact, been seen in training.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL-HLT*, pages 4171–4186, Minneapolis, Minnesota.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proc. of NAACL-HLT*, pages 1073–1094, Minneapolis, Minnesota.