Direct parsing to sentiment graphs

Anonymous ACL submission

Abstract

This paper demonstrates how a graph-based semantic parser can be applied to the task of structured sentiment analysis, directly predicting sentiment graphs from text. We advance the state of the art on 4 out of 5 standard benchmark sets. We release the source code, models and predictions with the camera-ready version.

1 Introduction

The task of structured sentiment analysis (SSA) is aimed at locating all *opinion tuples* within a sentence, where a single opinion contains a) a polar expression, b) an optional holder, c) an optional sentiment target, and d) a positive, negative or neutral polarity. An example is provided in Figure 1. While there have been sentiment corpora annotated with this type of information for decades (Wiebe et al., 2005; Toprak et al., 2010), there have so far been few attempts at modeling the full representation, rather focusing on various subcomponents, such as the polar expressions and targets without explicitly expressing the relations (Peng et al., 2019; Xu et al., 2020) or the polarity (Yang and Cardie, 2013; Katiyar and Cardie, 2016).

Dependency parsing approaches have recently shown promising results for SSA (Barnes et al., 2021; Peng et al., 2021). Here we present a novel sentiment parser which, unlike previous attempts, predicts sentiment graphs directly from text without reliance on heuristic lossy conversions to intermediate dependency representations. The model takes inspiration from successful work in meaning representation parsing, and in particular the permutation-invariant graph-based parser of Samuel and Straka (2020) called PERIN.

Experimenting with several different graph encodings, we evaluate our approach on five datasets from four different languages, and find that it compares favorably to dependency-based baselines across all datasets; most significantly on the more structurally complex ones – **NoReC** and **MPQA**.

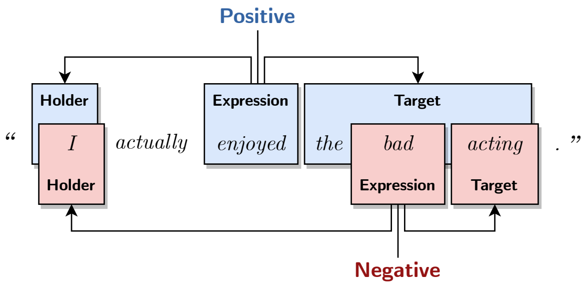

Figure 1: A sentiment graph for the phrase *“I actually enjoyed the bad acting”*, which contains an example of nesting of two opposing opinions.

2 Related work

Proposing a dependency parsing approach to the full task of SSA, Barnes et al. (2021) show that it leads to strong improvements over state-of-the-art baselines. Peng et al. (2021) propose a sparse fuzzy attention mechanism to deal with the sparseness of dependency arcs in the models from Barnes et al. (2021) and show further improvements. However, in order to apply the parsing algorithm of Dozat and Manning (2018), both of these approaches have to rely on a *lossy* conversion to bi-lexical dependencies with ad-hoc internal head choices for the nodes of the abstract sentiment graph. This lossy behaviour is caused by nested text spans in the sentiment graphs, as illustrated by Figure 1, which are ambiguous in their bi-lexical dependency encoding (see Section A in the Appendix).

More generally, decoding structured graph information from text has sparked a lot of interest in recent years, especially for parsing meaning representation graphs (Oepen et al., 2020). There has been tremendous progress in developing complex transition-based and graph-based parsers (Hershcovich et al., 2017; McDonald and Pereira, 2006; Dozat and Manning, 2018). In this paper, we adopt PERIN (Samuel and Straka, 2020), a state-of-the-art graph-based parser capable of modeling a superset of graph features needed for our task.

3 PERIN model

PERIN is a general permutation-invariant text-to-graph parser. We briefly describe our modified SSA version, please consult the original work for more details (Samuel and Straka, 2020).

3.1 Architecture

PERIN processes the input text in four steps, illustrated in Figure 2: 1) To encode the input, PERIN uses contextualized embeddings from XLM-R (base size; Conneau et al., 2020) and combines them with learned character-level embeddings; 2) each token is mapped onto latent *queries* by a linear transformation; 3) a stack of Transformer layers (Vaswani et al., 2017) optionally models the inter-query dependencies; and 4) classification heads select and label queries onto nodes, establish anchoring from nodes to tokens, and predict the node-to-node edges.

3.2 Permutation-invariant query-to-node matching

Traditional graph-based parsers are trained as autoregressive sequence-to-sequence models. PERIN does not assume any prior ordering of the graph nodes. Instead, it processes all queries in parallel and then dynamically maps them to gold nodes.

Based on the predicted probabilities of labels and anchors, we create a weighted bipartite graph between all queries and nodes. Our goal is to find the most probable matching, which can be done efficiently in polynomial time by using the Hungarian algorithm. Finally, every node is assigned to a query and we can backpropagate through standard cross-entropy losses to update the model weights.

3.3 Graph encodings

PERIN defines an overall framework for general graph parsing, it can cater to specific graph encodings by changing the subset of its classification heads. In parsing the abstract sentiment structures, there are several possible lossless graph encodings depending on the positioning of the polarity information and the sentiment node type (see Figure 3):

1. **Node-centric encoding**, with labeled nodes and directed unlabeled arcs. Each node corresponds to a target, holder or sentiment expression; edges form their relationships. The parser uses a multi-class node head, an anchor head and a binary edge classification head.

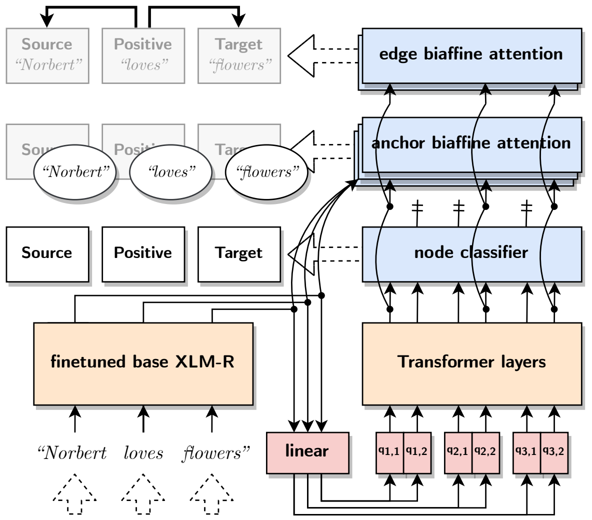

Figure 2: Diagram of the PERIN architecture; 1) each token gets a contextualized embedding and 2) generates queries, 3) queries are further processed and 4) they are put through node, anchor and edge classification heads.

2. **Labeled-edge encoding**, with deduplicated unlabeled nodes and labeled arcs. Each node corresponds to a unique text span from some sentiment graph, while edge labels denote their relationships and functions. The model has a binary node classifier, an anchor classifier and a binary and multi-class edge head.
3. **Opinion-tuple encoding**, which represents the structured sentiment information as a sequence of opinion four-tuples. This encoding is the most restrictive, having the lowest degrees of freedom. The parser utilizes a multi-class node head and three anchor classifiers, it does not need an edge classifier.

4 Experiments

Following Barnes et al. (2021) we perform experiments on five structured sentiment datasets in four languages, the statistics of which are shown in Table 1. The largest dataset is the **NoReC**$_{fine}$ dataset (Øvrelid et al., 2020), a multi-domain dataset of professional reviews in Norwegian. **EU** and **CA** (Barnes et al., 2018) contain hotel reviews in Basque and Catalan, respectively. **MPQA** (Wiebe et al., 2005) annotates news wire text in English. Finally, **DSU** (Toprak et al., 2010) annotates English reviews of online universities. We use the SemEval 2022 releases of **MPQA** and **DSU**.[1]

[1]Available from https://competitions.codalab.org/competitions/33556.

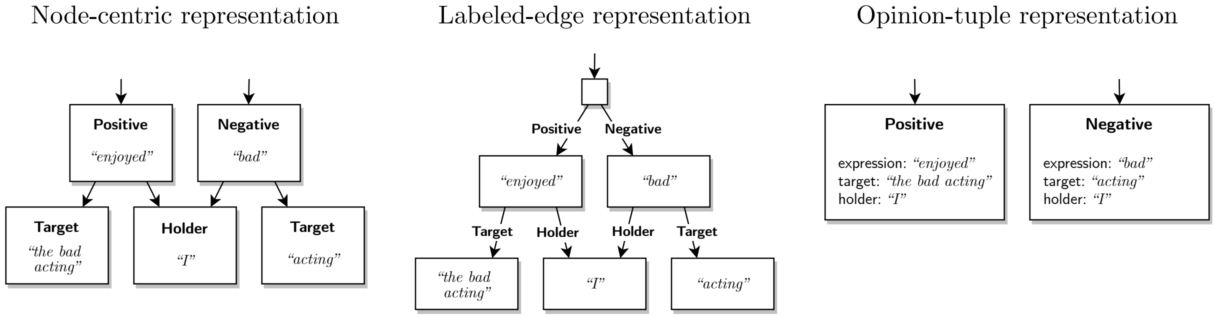

Figure 3: Three representations of the structured sentiment graph for sentence *"I actually enjoyed the bad acting."*

		sentences	holders	targets	exps.	+	neu	−
NoReC	train	8634	898	6778	8448	5684	—	2756
	dev	1531	120	1152	1432	988	—	443
	test	1272	110	993	1235	875	—	358
CA	train	1174	169	1695	1981	1272	—	708
	dev	168	15	211	258	151	—	107
	test	336	52	430	518	313	—	204
EU	train	1064	205	1285	1684	1406	—	278
	dev	152	33	153	204	168	—	36
	test	305	58	337	440	375	—	65
MPQA	train	5873	1431	1487	1715	671	337	698
	dev	2063	414	503	581	223	126	216
	test	2112	434	462	518	159	82	223
DSU	train	2253	65	836	836	349	104	383
	dev	232	9	104	104	31	16	57
	test	318	12	142	142	59	12	71

Table 1: Statistics of the datasets, including number of sentences per split, as well as number of holder, target, and polar expression annotations. Additionally, we include the distribution of polarity – restricted to positive, neutral, and negative – in each dataset.

4.1 Evaluation

Following Barnes et al. (2021), we evaluate our models using both token-level F_1 for extraction of Holders, Targets, and polar Expressions, as well as the graph-level metrics Non-polar Sentiment Graph F_1 (**NSF$_1$**) and Sentiment Graph F_1 (**SF$_1$**), weighing the overlap in predicted and gold spans for each entity, averaged across all three spans. SF$_1$, which also includes polarity, is considered the primary metric for the full SSA task.

4.2 Models

We compare our models to the head-final dependency graph parsers from Barnes et al. (2021) as well as the second-order Sparse Fuzzy Attention parser of Peng et al. (2021). For all models, we perform 5 runs with 5 different random seeds and report the mean and standard deviation. Results on development splits are provided in Appendix D, training details are in Appendix E.

4.3 Results

Table 2 shows the main results. Our models outperform both dependency graph models on SF$_1$, although the results are mixed for span extraction. The opinion-tuple encoding gives the best performance on SF$_1$ (an average of 6.2 percentage points (pp.) better than Peng et al. (2021)), followed by the labeled edge encoding (3.0) and finally the node-centric encoding (2.1).

For extracting spans, the opinion tuple encoding also achieves the the best results on **NoReC**, either labeled-edge or node centric on **CA** and **MPQA**, while Peng et al. (2021) is best on **EU** and **DSU**. This suggests that the main benefit of PERIN is at the structural level, rather than local extraction.

5 Analysis

There are a number of architectural differences between the dependency parsing approaches compared above. In this section, we aim to isolate the effect of predicting intermediate dependency graphs vs. directly predicting sentiment graphs by creating more comparable dependency[2] and PERIN models. We adapt the dependency model from Barnes et al. (2021) by removing the token, lemma, and POS embeddings and replacing mBERT (Devlin et al., 2019) with XLM-R (Conneau et al., 2020). The 'XLM-R dependency' model thus has character LSTM embeddings and token-level XLM-R features. Since these are not updated during training, for the opinion-tuple 'Frozen PERIN' model, we fix the XLM-R weights to make it comparable.

As shown in Table 3, predicting the sentiment graph directly leads to an average gain of 3.7 pp. on the Sentiment Graph F_1 metric. For extracting the spans of holder, target, and polar expressions, the

[2] We do not use the model from Peng et al. (2021) as the code is not available.

Dataset	Model	Span F_1			Sent. graph	
		Holder	Target	Exp.	$NSF_1 \uparrow$	$SF_1 \uparrow$
NoReC	Barnes et al. (2021)	60.4	54.8	55.5	39.2	31.2
	Peng et al. (2021)	63.6	55.3	56.1	40.4	31.9
	PERIN – node-centric	$60.3^{\pm 1.8}$	$51.8^{\pm 2.5}$	$54.2^{\pm 0.9}$	$42.7^{\pm 0.6}$	$39.3^{\pm 0.7}$
	PERIN – labeled edge	$64.0^{\pm 1.5}$	$52.3^{\pm 4.2}$	$56.1^{\pm 2.7}$	$43.7^{\pm 2.2}$	$40.4^{\pm 2.1}$
	PERIN – opinion-tuple	$65.1^{\pm 2.5}$	$*58.3^{\pm 1.5}$	$*60.7^{\pm 1.1}$	$47.8^{\pm 1.2}$	$\mathbf{41.6}^{\pm 0.7}$
EU	Barnes et al. (2021)	60.5	64.0	72.1	58.0	54.7
	Peng et al. (2021)	65.8	71.0	76.7	66.1	**62.7**
	PERIN – node-centric	$58.9^{\pm 1.1}$	$63.5^{\pm 1.5}$	$73.9^{\pm 0.6}$	$59.8^{\pm 0.7}$	$58.6^{\pm 0.7}$
	PERIN – labeled edge	$57.6^{\pm 2.5}$	$64.9^{\pm 0.8}$	$72.5^{\pm 1.9}$	$60.0^{\pm 1.4}$	$58.8^{\pm 1.3}$
	PERIN – opinion-tuple	$64.2^{\pm 2.5}$	$67.4^{\pm 0.8}$	$73.2^{\pm 1.2}$	$62.5^{\pm 1.2}$	$61.3^{\pm 1.0}$
CA	Barnes et al. (2021)	37.1	71.2	67.1	59.7	53.7
	Peng et al. (2021)	46.2	74.2	71.0	64.5	59.3
	PERIN – node-centric	$56.1^{\pm 3.0}$	$69.8^{\pm 0.4}$	$70.5^{\pm 0.5}$	$63.5^{\pm 0.6}$	$61.7^{\pm 0.6}$
	PERIN – labeled edge	$60.8^{\pm 5.1}$	$70.8^{\pm 1.9}$	$72.5^{\pm 0.8}$	$64.5^{\pm 1.4}$	$62.1^{\pm 1.3}$
	PERIN – opinion-tuple	$48.0^{\pm 3.9}$	$72.5^{\pm 0.7}$	$68.9^{\pm 0.2}$	$65.7^{\pm 0.7}$	$\mathbf{63.3}^{\pm 0.6}$
MPQA	Barnes et al. (2021)	46.3	49.5	46.0	26.1	18.8
	Peng et al. (2021)	47.9	50.7	47.8	38.6	19.1
	PERIN – node-centric	$58.4^{\pm 2.3}$	$60.3^{\pm 2.0}$	$55.8^{\pm 1.5}$	$38.7^{\pm 1.6}$	$28.3^{\pm 0.9}$
	PERIN – labeled edge	$53.6^{\pm 1.2}$	$53.4^{\pm 1.9}$	$53.4^{\pm 1.1}$	$33.8^{\pm 1.5}$	$27.0^{\pm 0.9}$
	PERIN – opinion-tuple	$55.7^{\pm 1.7}$	$*64.0^{\pm 0.6}$	$53.5^{\pm 1.2}$	$*45.1^{\pm 1.1}$	$*\mathbf{34.1}^{\pm 1.1}$
DSU	Barnes et al. (2021)	37.4	42.1	45.5	34.3	26.5
	Peng et al. (2021)	50.0	44.8	43.7	35.0	27.4
	PERIN – node-centric	$31.4^{\pm 5.6}$	$35.0^{\pm 1.6}$	$35.1^{\pm 2.2}$	$24.8^{\pm 0.7}$	$22.9^{\pm 1.5}$
	PERIN – labeled edge	$32.5^{\pm 6.8}$	$38.0^{\pm 3.7}$	$36.2^{\pm 2.5}$	$28.8^{\pm 2.0}$	$27.3^{\pm 1.5}$
	PERIN – opinion-tuple	$42.2^{\pm 4.6}$	$40.6^{\pm 2.7}$	$39.3^{\pm 2.5}$	$33.2^{\pm 2.4}$	$\mathbf{31.2}^{\pm 2.4}$

Table 2: Experiments comparing the PERIN model with previous results. We show the average values and their standard deviations from 5 runs. **Bold** numbers indicate the best result for the main SF_1 metric in each dataset. * marks significant difference between our two best approaches, determined by bootstrap testing (see Appendix C).

Dataset	Model	Span F_1			Sent. graph	
		H.	T.	E.	NSF_1	$SF_1 \uparrow$
NoReC	XLM-R dependency	58.5	49.9	58.5	37.4	31.9
	Frozen PERIN	48.3	51.9	57.9	*41.8	$*\mathbf{35.7}^{\pm 0.6}$
EU	XLM-R dependency	50.0	60.3	70.0	55.1	51.0
	Frozen PERIN	55.5	58.5	68.8	53.1	$\mathbf{51.3}^{\pm 1.2}$
CA	XLM-R dependency	24.9	67.7	67.3	54.8	50.5
	Frozen PERIN	*39.8	69.2	66.3	*60.2	$*\mathbf{57.6}^{\pm 1.2}$
MPQA	XLM-R dependency	49.3	*56.9	47.6	30.5	18.9
	Frozen PERIN	44.0	49.0	46.6	30.7	$\mathbf{23.1}^{\pm 1.0}$
DSU	XLM-R dependency	26.8	33.6	36.4	22.9	18.0
	Frozen PERIN	13.8	37.3	33.2	24.5	$\mathbf{21.3}^{\pm 2.9}$

Table 3: Results from comparable experiments, where the dependency graph model (XLM-R dependency) and frozen PERIN models use the same input and similar number of trainable parameters. * marks significant difference, determined by bootstrap (see Appendix C).

benefit is less clear. Here, the PERIN model only outperforms the XLM-R dependency model 5 of 15 times, which seems to confirm that its benefit is at the graph level. This is further supported by the fact that the highest gains are found on the datasets with the most nested sentiment expressions and dependency arcs lost due to overlap, which are difficult to encode in bi-lexical graphs (see Appendix A).

6 Conclusion

Previous work cast the task of structured sentiment analysis (SSA) as dependency parsing, converting the sentiment graphs into lossy dependency graphs. We present a novel sentiment parser which, unlike previous attempts, predicts sentiment graphs directly from text without reliance on lossy dependency representations. We adapted a state-of-the-art meaning representation parser to SSA and experimentally evaluated three candidate graph encodings of the sentiment structures. The results suggest that our approach to SSA has clear performance benefits, advancing the state of the art on four out of five commonly used benchmarks. Specifically, the most direct opinion-tuple encoding provides the highest performance gains. More detailed analysis of the results shows that the benefits stem from better extraction of global structures, rather than local span prediction. We will release the source code, models and predictions in the camera-ready version of this paper at `https://github.com/censored/for-review`.

References

Jeremy Barnes, Toni Badia, and Patrik Lambert. 2018. MultiBooked: A corpus of Basque and Catalan hotel reviews annotated for aspect-level sentiment classification. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Jeremy Barnes, Robin Kurtz, Stephan Oepen, Lilja Øvrelid, and Erik Velldal. 2021. Structured sentiment analysis as dependency graph parsing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3387–3402, Online. Association for Computational Linguistics.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An Empirical Investigation of Statistical Significance in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005, Jeju Island, Korea. Association for Computational Linguistics.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, pages 794–803.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia. Association for Computational Linguistics.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1127–1138, Vancouver, Canada. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Arzoo Katiyar and Claire Cardie. 2016. Investigating LSTMs for joint extraction of opinion entities and relations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 919–929, Berlin, Germany. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy. Association for Computational Linguistics.

Stephan Oepen, Omri Abend, Lasha Abzianidze, Johan Bos, Jan Hajic, Daniel Hershcovich, Bin Li, Tim O'Gorman, Nianwen Xue, and Daniel Zeman. 2020. MRP 2020: The second shared task on cross-framework and cross-lingual meaning representation parsing. In *Proceedings of the CoNLL 2020 Shared Task: Cross-Framework Meaning Representation Parsing*, pages 1–22, Online. Association for Computational Linguistics.

Lilja Øvrelid, Petter Mæhlum, Jeremy Barnes, and Erik Velldal. 2020. A fine-grained sentiment dataset for Norwegian. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5025–5033, Marseille, France. European Language Resources Association.

Haiyun Peng, Lu Xu, Lidong Bing, Fei Huang, Wei Lu, and Luo Si. 2019. Knowing what, how and why: A near complete solution for aspect-based sentiment analysis.

Letian Peng, Zuchao Li, and Hai Zhao. 2021. Sparse fuzzy attention for structured sentiment analysis.

David Samuel and Milan Straka. 2020. ÚFAL at MRP 2020: Permutation-invariant semantic parsing in PERIN. In *Proceedings of the CoNLL 2020 Shared Task: Cross-Framework Meaning Representation Parsing*, pages 53–64, Online. Association for Computational Linguistics.

Cigdem Toprak, Niklas Jakob, and Iryna Gurevych. 2010. Sentence and expression level annotation of opinions in user-generated discourse. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 575–584, Uppsala, Sweden. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2-3):165–210.

Lu Xu, Hao Li, Wei Lu, and Lidong Bing. 2020. Position-aware tagging for aspect sentiment triplet extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2339–2349, Online. Association for Computational Linguistics.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1640–1649, Sofia, Bulgaria. Association for Computational Linguistics.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2021. Revisiting few-sample {bert} fine-tuning. In *International Conference on Learning Representations*.

A Problems with dependency encoding

As briefly mentioned in the main text, previous dependency parsing approaches have relied on a *lossy* bi-lexical conversion. We use this appendix to describe this problem in more detail. There is an inherent ambiguity in the encoding of two nested text spans with the same head (defined as either the first or the last token in (Barnes et al., 2021)). To be concrete, we can use the running example *"I actually enjoyed the bad acting"*, which has two opinions with nested targets *"the bad acting"* and *"acting"*. As shown in Figure 4, both expression-target edges correctly lead to the word *"acting"* but it is impossible to disambiguate the prefix of both targets in the bi-lexical encoding. For that, we need a more abstract graph encoding, such as the ones suggested in the main text.

Table 4 shows that the amount of nesting in the SSA datasets is not negligible. This is especially true for **NoReC** and **MPQA**, two datasets experiencing significant performance gains from our proposed graph encoding. Table 5 further shows the amount of dependency edges lost because of overlap. Finally, Table 6 shows the SF_1 score when converting the gold sentiment graphs to bi-lexical dependency graphs and back – an inherent upper bound for any dependency parser.

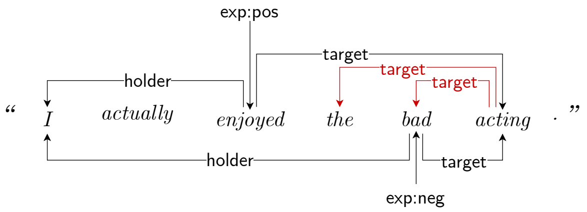

Figure 4: Ambiguous *targets* when encoding the sentence *"I actually enjoyed the bad acting"* as a head-final bi-lexical dependency graph (Barnes et al., 2021).

Dataset	Holders		Targets		Exps.	
	#	%	#	%	#	%
NoReC	95	1.5	1187	14.1	1075	9.3
EU	30	2.2	79	4.5	16	0.7
CA	43	2.9	28	1.2	23	0.9
MPQA	48	2.2	250	9.3	145	5.6
DSU	0	0.0	10	1.1	7	0.5

Table 4: Count and percentage of nesting for each dataset.

NoReC	8.8%
EU	4.5%
CA	6.7%
MPQA	4.2%
DSU	0.5%

Table 5: Percentages of dependency arcs lost due to overlap.

NoReC	93.6
EU	95.2
CA	97.6
MPQA	96.6
DSU	99.8

Table 6: Sentiment Graph F1 after converting test sets to head-final and then reconverting to json format.

B Changes to datasets

We found out that the official data published at `https://competitions.codalab.org/competitions/33556` was slightly changed from the data used in previous related work. Specifically the **MPQA** and **DSU** datasets had removed a number of errors resulting from the annotation and from the conversion scripts used to create the sentiment graph representations. We re-run the experiments

Dataset	Model	Span F_1			Sent. graph		Runtime	# Params
		Holder	Target	Exp.	NSF_1 ↑	SF_1 ↑		
NoReC	PERIN – node-centric	$54.9^{\pm4.3}$	$52.7^{\pm2.0}$	$57.4^{\pm1.5}$	$44.8^{\pm1.8}$	p: 46.4 r: 36.4 $40.8^{\pm1.5}$	9:52 h	108.9 M
	PERIN – labeled edge	$59.4^{\pm2.8}$	$52.0^{\pm2.3}$	$57.5^{\pm2.7}$	$44.4^{\pm1.7}$	p: 45.7 r: 37.7 $41.1^{\pm1.5}$	9:58 h	109.5 M
	PERIN – opinion-tuple	$59.2^{\pm1.3}$	$59.6^{\pm1.3}$	$61.5^{\pm1.0}$	$49.4^{\pm1.0}$	p: 42.5 r: 45.5 $43.9^{\pm0.9}$	9:25 h	108.1 M
	Frozen PERIN – opinion-tuple	$50.1^{\pm2.5}$	$53.8^{\pm1.6}$	$59.4^{\pm1.0}$	$44.0^{\pm0.6}$	p: 33.6 r: 42.2 $37.4^{\pm0.9}$	0:25 h	23.1 M
EU	PERIN – node-centric	$57.1^{\pm3.1}$	$68.7^{\pm1.5}$	$69.9^{\pm1.0}$	$61.1^{\pm1.1}$	p: 62.8 r: 56.8 $59.7^{\pm1.3}$	1:02 h	87.6 M
	PERIN – labeled edge	$51.2^{\pm4.7}$	$66.1^{\pm2.1}$	$66.0^{\pm1.0}$	$59.4^{\pm1.2}$	p: 60.1 r: 55.1 $57.4^{\pm1.2}$	0:57 h	88.2 M
	PERIN – opinion-tuple	$57.3^{\pm3.0}$	$65.1^{\pm2.3}$	$68.6^{\pm0.3}$	$59.9^{\pm1.0}$	p: 64.5 r: 54.7 $59.2^{\pm0.6}$	1:04 h	86.9 M
	Frozen PERIN – opinion-tuple	$57.0^{\pm10.4}$	$61.1^{\pm3.2}$	$65.1^{\pm3.9}$	$55.5^{\pm2.9}$	p: 56.3 r: 48.8 $52.2^{\pm3.2}$	0:06 h	0.7 M
CA	PERIN – node-centric	$57.1^{\pm2.0}$	$73.8^{\pm2.5}$	$74.2^{\pm1.6}$	$68.4^{\pm2.6}$	p: 69.9 r: 62.9 $66.2^{\pm2.1}$	1:17 h	87.6 M
	PERIN – labeled edge	$48.9^{\pm4.3}$	$72.1^{\pm0.9}$	$72.6^{\pm1.1}$	$67.1^{\pm1.6}$	p: 69.5 r: 61.8 $65.4^{\pm1.6}$	1:13 h	88.2 M
	PERIN – opinion-tuple	$46.1^{\pm3.0}$	$74.4^{\pm1.0}$	$72.9^{\pm0.5}$	$68.4^{\pm1.5}$	p: 73.6 r: 61.6 $67.0^{\pm1.2}$	1:20 h	86.9 M
	Frozen PERIN – opinion-tuple	$48.1^{\pm6.4}$	$65.5^{\pm1.8}$	$69.2^{\pm5.5}$	$62.2^{\pm2.7}$	p: 64.7 r: 56.0 $59.9^{\pm2.5}$	0:07 h	0.7 M
MPQA	PERIN – node-centric	$58.2^{\pm1.3}$	$60.8^{\pm0.9}$	$56.8^{\pm1.1}$	$35.3^{\pm1.3}$	p: 34.5 r: 28.7 $31.4^{\pm1.4}$	6:46 h	107.7 M
	PERIN – labeled edge	$57.1^{\pm2.0}$	$54.8^{\pm1.6}$	$55.2^{\pm1.1}$	$33.1^{\pm0.4}$	p: 35.7 r: 26.4 $30.3^{\pm0.5}$	7:16 h	109.6 M
	PERIN – opinion-tuple	$56.0^{\pm0.6}$	$64.2^{\pm1.7}$	$51.7^{\pm2.8}$	$42.1^{\pm0.8}$	p: 44.3 r: 30.1 $35.8^{\pm0.6}$	6:43 h	108.1 M
	Frozen PERIN – opinion-tuple	$42.0^{\pm3.8}$	$48.1^{\pm1.7}$	$46.6^{\pm2.6}$	$28.1^{\pm2.2}$	p: 24.3 r: 20.8 $22.2^{\pm1.5}$	0:37 h	23.1 M
DSU	PERIN – node-centric	$0.0^{\pm0.0}$	$41.5^{\pm4.3}$	$40.3^{\pm2.6}$	$27.2^{\pm2.0}$	p: 33.4 r: 16.9 $22.4^{\pm1.3}$	2:31 h	107.7 M
	PERIN – labeled edge	$0.0^{\pm0.0}$	$46.5^{\pm1.8}$	$41.9^{\pm3.4}$	$28.4^{\pm2.7}$	p: 33.2 r: 17.8 $23.1^{\pm2.0}$	2:37 h	109.6 M
	PERIN – opinion-tuple	$12.0^{\pm11.0}$	$50.9^{\pm4.7}$	$42.6^{\pm3.9}$	$34.9^{\pm4.1}$	p: 39.5 r: 22.6 $28.6^{\pm3.5}$	2:30 h	108.1 M
	Frozen PERIN – opinion-tuple	$0.0^{\pm0.0}$	$42.7^{\pm4.8}$	$35.9^{\pm3.3}$	$26.0^{\pm3.3}$	p: 29.1 r: 16.3 $20.3^{\pm2.0}$	0:22 h	23.1 M

Table 7: Development scores of all our models from the main section of this paper. SF_1 scores are extended by the average precision and recall values. We also show the runtime of a single model and the number of trainable parameters.

Dataset		Span F_1			Sent. graph	
		H.	T.	E.	NSF_1	SF_1
MPQA	original	44.7	51.3	45.7	25.4	15.0
	new data	49.3	56.9	47.6	30.5	18.9
	Δ	+4.6	+5.6	+1.9	+5.1	+4.9
DSU	original	21.0	22.6	35.2	24.0	21.0
	new data	26.8	33.6	36.4	22.9	18.0
	Δ	+5.8	+11.0	+1.3	−1.1	−3.0

Table 8: Results comparing the XLM-R dependency model on the original **MPQA** and **DSU** data, and the new data.

for the comparable baseline model and show the performance differences in Table 8.

C Bootstrap Significance Testing

In order to see whether the performance differences for the experiments are significant, we do bootstrap significance testing Berg-Kirkpatrick et al. (2012), combining two variations. First, we resample the test sets with replacement from all 5 runs together, $b = 1\,000\,000$ times, setting the threshold at $p = 0.05$. Additionally, we test each pair out of the 5×5 combinations for all runs, resampling the test set with replacement $b = 100\,000$ times, setting the threshold again at $p = 0.5$. When one system is significantly better in 15 out of the 25 comparisons, and additionally significantly better in the first joint test, we finally mark it as significantly better.

D Results on development data

To make any future comparison of our approach easier, we show the development scores of all reported models in Table 7.

E Training details

Generally, we follow the training regime described in the original PERIN paper (Samuel and Straka, 2020). The trainable parameters are updated with the AdamW optimizer (Loshchilov and Hutter, 2019), and their learning rate is linearly warmed-up for the first 10% of the training to improve stability, and then decayed with a cosine schedule. The XLM-R parameters are updated with a lower learning rate and higher weight decay to improve generalization; its lower also use an increasingly lower learning rate (Howard and Ruder, 2018). Similarly to PERIN, we freeze the embedding parameters for increased efficiency and regularization. Following the finding by Zhang et al. (2021), we use small learning rates and fine-tune for a rather long time to increase the training stability. Unlike the authors of PERIN, we did not find any benefits from a dynamic scaling of loss weights (Chen et al., 2018), so we simply set all loss weights to constant 1.0.

We trained our models on a single Nvidia P100 with 16GB RAM, the runtimes are given in Table 7. We made five runs from different seeds for each reported value to better estimate the expected error. The hyperparameter configurations for all runs follow, please consult the released code for more details and context: https://github.com/censored/for-review.

General hyperparameters

```
batch_size = 16
beta_2 = 0.98
char_embedding = True
char_embedding_size = 128
decoder_learning_rate = 6.0e-4
decoder_weight_decay = 1.2e-6
dropout_anchor = 0.4
dropout_edge_label = 0.5
dropout_edge_presence = 0.5
dropout_label = 0.85
dropout_transformer = 0.25
dropout_transformer_attention = 0.1
dropout_word = 0.1
encoder = "xlm-roberta-base"
encoder_freeze_embedding = True
encoder_learning_rate = 6.0e-6
encoder_weight_decay = 0.1
epochs = 200
focal = True
freeze_bert = False
hidden_size_ff = 4 * 768
hidden_size_anchor = 256
hidden_size_edge_label = 256
hidden_size_edge_presence = 256
layerwise_lr_decay = 0.9
n_attention_heads = 8
n_layers = 3
query_length = 1
pre_norm = True
```

NoReC node-centric hyperparameters

```
graph_mode = "node-centric"
query_length = 2
```

NoReC labeled-edge hyperparameters

```
graph_mode = "labeled-edge"
query_length = 2
```

NoReC opinion-tuple hyperparameters

```
graph_mode = "opinion-tuple"
```

NoReC frozen opinion-tuple hyperparameters

```
graph_mode = "opinion-tuple"
freeze_bert = True
batch_size = 8
decoder_learning_rate = 1.0e-4
dropout_transformer = 0.5
epochs = 50
```

EU node-centric hyperparameters

```
graph_mode = "node-centric"
query_length = 2
n_layers = 0
```

EU labeled-edge hyperparameters

```
graph_mode = "labeled-edge"
query_length = 2
n_layers = 0
```

EU opinion-tuple hyperparameters

```
graph_mode = "opinion-tuple"
n_layers = 0
```

EU frozen opinion-tuple hyperparameters

```
graph_mode = "opinion-tuple"
freeze_bert = True
n_layers = 0
epochs = 50
```

CA node-centric hyperparameters

```
graph_mode = "node-centric"
query_length = 2
n_layers = 0
```

CA labeled-edge hyperparameters

```
graph_mode = "labeled-edge"
query_length = 2
n_layers = 0
```

CA opinion-tuple hyperparameters

```
graph_mode = "opinion-tuple"
n_layers = 0
```

CA frozen opinion-tuple hyperparameters

```
graph_mode = "opinion-tuple"
freeze_bert = True
n_layers = 0
epochs = 50
```

MPQA node-centric hyperparameters

```
graph_mode = "node-centric"
decoder_learning_rate = 1.0e-4
query_length = 2
```

MPQA labeled-edge hyperparameters

```
graph_mode = "labeled-edge"
decoder_learning_rate = 1.0e-4
query_length = 2
```

MPQA opinion-tuple hyperparameters

```
graph_mode = "opinion-tuple"
```

MPQA frozen opinion-tuple hyperparameters

```
graph_mode = "opinion-tuple"
freeze_bert = True
batch_size = 8
decoder_learning_rate = 1.0e-4
dropout_transformer = 0.5
epochs = 50
```

DSU node-centric hyperparameters

```
graph_mode = "node-centric"
decoder_learning_rate = 1.0e-4
query_length = 2
```

DSU labeled-edge hyperparameters

```
graph_mode = "labeled-edge"
decoder_learning_rate = 1.0e-4
query_length = 2
```

DSU opinion-tuple hyperparameters

```
graph_mode = "opinion-tuple"
```

DSU frozen opinion-tuple hyperparameters

```
graph_mode = "opinion-tuple"
freeze_bert = True
batch_size = 8
decoder_learning_rate = 1.0e-4
dropout_transformer = 0.5
epochs = 50
```