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.

1 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 (Hirschcovitch 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.

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

1Available from https://competitions.codalab.org/competitions/33556.
Node-centric representation

Labeled-edge representation

Opinion-tuple representation

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

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

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 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.
The significance difference between our two best approaches, determined by bootstrap testing (see Appendix C).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Span F1</th>
<th>Exp.</th>
<th>Sent. graph</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>⊕</td>
<td></td>
<td>⊕</td>
</tr>
<tr>
<td>NoRec</td>
<td>Barnes et al. (2021)</td>
<td>60.4</td>
<td>54.8</td>
<td>55.5</td>
</tr>
<tr>
<td></td>
<td>Peng et al. (2021)</td>
<td>63.6</td>
<td>55.3</td>
<td>56.1</td>
</tr>
<tr>
<td></td>
<td>PERIN – node-centric</td>
<td>60.7±1.8</td>
<td>51.8±2.5</td>
<td>54.2±0.9</td>
</tr>
<tr>
<td></td>
<td>PERIN – labeled edge</td>
<td>64.0±1.5</td>
<td>52.3±4.2</td>
<td>56.1±2.7</td>
</tr>
<tr>
<td></td>
<td>PERIN – opinion-tuple</td>
<td>65.1±2.5</td>
<td>*58.3±1.5</td>
<td>*60.7±1.1</td>
</tr>
<tr>
<td>EU</td>
<td>Barnes et al. (2021)</td>
<td>60.5</td>
<td>64.0</td>
<td>72.1</td>
</tr>
<tr>
<td></td>
<td>Peng et al. (2021)</td>
<td>65.8</td>
<td>71.0</td>
<td>76.7</td>
</tr>
<tr>
<td></td>
<td>PERIN – node-centric</td>
<td>58.9±1.1</td>
<td>63.5±1.5</td>
<td>73.9±0.6</td>
</tr>
<tr>
<td></td>
<td>PERIN – labeled edge</td>
<td>57.6±2.5</td>
<td>64.9±0.8</td>
<td>72.5±1.9</td>
</tr>
<tr>
<td></td>
<td>PERIN – opinion-tuple</td>
<td>64.2±2.5</td>
<td>67.4±0.8</td>
<td>73.2±1.2</td>
</tr>
<tr>
<td>CA</td>
<td>Barnes et al. (2021)</td>
<td>37.1</td>
<td>71.2</td>
<td>67.1</td>
</tr>
<tr>
<td></td>
<td>Peng et al. (2021)</td>
<td>46.2</td>
<td>74.2</td>
<td>71.0</td>
</tr>
<tr>
<td></td>
<td>PERIN – node-centric</td>
<td>56.1±3.0</td>
<td>69.8±0.4</td>
<td>70.5±0.5</td>
</tr>
<tr>
<td></td>
<td>PERIN – labeled edge</td>
<td>60.8±5.1</td>
<td>70.8±1.9</td>
<td>72.5±0.8</td>
</tr>
<tr>
<td></td>
<td>PERIN – opinion-tuple</td>
<td>48.0±3.9</td>
<td>72.5±0.7</td>
<td>68.9±0.2</td>
</tr>
<tr>
<td>MPQA</td>
<td>Barnes et al. (2021)</td>
<td>46.3</td>
<td>49.5</td>
<td>46.0</td>
</tr>
<tr>
<td></td>
<td>Peng et al. (2021)</td>
<td>47.9</td>
<td>50.7</td>
<td>47.8</td>
</tr>
<tr>
<td></td>
<td>PERIN – node-centric</td>
<td>58.4±2.3</td>
<td>60.3±2.0</td>
<td>55.8±1.5</td>
</tr>
<tr>
<td></td>
<td>PERIN – labeled edge</td>
<td>53.6±1.2</td>
<td>53.4±1.9</td>
<td>53.4±1.1</td>
</tr>
<tr>
<td></td>
<td>PERIN – opinion-tuple</td>
<td>55.7±1.7</td>
<td>*64.0±0.6</td>
<td>53.5±1.2</td>
</tr>
<tr>
<td>DSU</td>
<td>Barnes et al. (2021)</td>
<td>37.4</td>
<td>42.1</td>
<td>45.5</td>
</tr>
<tr>
<td></td>
<td>Peng et al. (2021)</td>
<td>50.0</td>
<td>44.8</td>
<td>43.7</td>
</tr>
<tr>
<td></td>
<td>PERIN – node-centric</td>
<td>31.4±5.6</td>
<td>35.0±1.6</td>
<td>35.1±2.2</td>
</tr>
<tr>
<td></td>
<td>PERIN – labeled edge</td>
<td>32.5±6.8</td>
<td>38.0±3.7</td>
<td>36.2±2.5</td>
</tr>
<tr>
<td></td>
<td>PERIN – opinion-tuple</td>
<td>42.2±4.6</td>
<td>40.6±2.7</td>
<td>39.3±2.5</td>
</tr>
</tbody>
</table>

Table 2: Experiments comparing the PERIN model with previous results. Bold numbers indicate the best result for the main SF1 metric in each dataset. * marks significant difference between our two best approaches, determined by bootstrap testing (see Appendix C).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Span F1</th>
<th>Exp.</th>
<th>Sent. graph</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>⊕</td>
<td></td>
<td>⊕</td>
</tr>
<tr>
<td>NoRec</td>
<td>XLM-R dependency</td>
<td>58.5</td>
<td>49.9</td>
<td>58.5</td>
</tr>
<tr>
<td></td>
<td>Frozen PERIN</td>
<td>48.3</td>
<td>51.9</td>
<td>57.9</td>
</tr>
<tr>
<td>EU</td>
<td>XLM-R dependency</td>
<td>50.0</td>
<td>60.3</td>
<td>70.0</td>
</tr>
<tr>
<td></td>
<td>Frozen PERIN</td>
<td>55.5</td>
<td>58.5</td>
<td>68.8</td>
</tr>
<tr>
<td>CA</td>
<td>XLM-R dependency</td>
<td>24.9</td>
<td>67.7</td>
<td>67.3</td>
</tr>
<tr>
<td></td>
<td>Frozen PERIN</td>
<td>*39.8</td>
<td>69.2</td>
<td>66.3</td>
</tr>
<tr>
<td>MPQA</td>
<td>XLM-R dependency</td>
<td>49.3</td>
<td>*56.9</td>
<td>47.6</td>
</tr>
<tr>
<td></td>
<td>Frozen PERIN</td>
<td>44.0</td>
<td>49.0</td>
<td>46.6</td>
</tr>
<tr>
<td>DSU</td>
<td>XLM-R dependency</td>
<td>26.8</td>
<td>33.6</td>
<td>36.4</td>
</tr>
<tr>
<td></td>
<td>Frozen PERIN</td>
<td>13.8</td>
<td>37.3</td>
<td>33.2</td>
</tr>
</tbody>
</table>

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).

The benefits of using the PERIN model are not as clear as those of using the XLM-R model. The XLM-R model outperforms the PERIN model in terms of span F1 and Exp. The XLM-R model also has a higher Sent. graph score in most cases, indicating better semantic graph encoding.

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


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 S\textsubscript{F}1 score when converting the gold sentiment graphs to bi-lexical dependency graphs and back – an inherent upper bound for any dependency parser.

### 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 S\textsubscript{F}1 score when converting the gold sentiment graphs to bi-lexical dependency graphs and back – an inherent upper bound for any dependency parser.

### 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...
Table 7: Development scores of all our models from the main section of this paper. SF1 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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Span F1</th>
<th>Target</th>
<th>Exp.</th>
<th>Sent. graph</th>
<th>Runtime</th>
<th># Params</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Holder</td>
<td>Target</td>
<td>Exp.</td>
<td>SF1↑</td>
<td>SF1↑</td>
<td></td>
</tr>
<tr>
<td>NoRec</td>
<td>PERIN – node-centric</td>
<td>54.9±4.3</td>
<td>52.7±2.0</td>
<td>57.4±1.5</td>
<td>44.8±1.8</td>
<td>36.4</td>
<td>108.9 M</td>
</tr>
<tr>
<td></td>
<td>PERIN – labeled edge</td>
<td>59.4±2.8</td>
<td>52.0±2.3</td>
<td>57.5±2.7</td>
<td>44.4±1.7</td>
<td>37.6</td>
<td>109.5 M</td>
</tr>
<tr>
<td></td>
<td>PERIN – opinion-tuple</td>
<td>59.2±1.3</td>
<td>59.6±1.3</td>
<td>61.5±1.0</td>
<td>49.4±1.0</td>
<td>45.8</td>
<td>108.1 M</td>
</tr>
<tr>
<td></td>
<td>Frozen PERIN – opinion-tuple</td>
<td>50.1±2.5</td>
<td>53.8±1.6</td>
<td>59.4±1.0</td>
<td>44.0±0.6</td>
<td>42.5</td>
<td>23.1 M</td>
</tr>
<tr>
<td></td>
<td>PERIN – node-centric</td>
<td>57.1±2.1</td>
<td>68.7±1.5</td>
<td>69.9±1.0</td>
<td>61.1±1.1</td>
<td>56.8</td>
<td>87.6 M</td>
</tr>
<tr>
<td></td>
<td>PERIN – labeled edge</td>
<td>51.2±4.7</td>
<td>66.1±2.1</td>
<td>66.0±1.0</td>
<td>59.4±1.2</td>
<td>57.4±1.2</td>
<td>88.2 M</td>
</tr>
<tr>
<td></td>
<td>PERIN – opinion-tuple</td>
<td>57.3±3.0</td>
<td>65.1±2.3</td>
<td>68.6±3.0</td>
<td>59.9±1.0</td>
<td>56.5±0.6</td>
<td>89.9 M</td>
</tr>
<tr>
<td></td>
<td>Frozen PERIN – opinion-tuple</td>
<td>57.0±4.0</td>
<td>61.1±3.2</td>
<td>65.1±3.9</td>
<td>55.5±2.9</td>
<td>51.9±2.4</td>
<td>0.06 M</td>
</tr>
<tr>
<td>EU</td>
<td>PERIN – node-centric</td>
<td>57.1±2.0</td>
<td>73.8±2.5</td>
<td>74.2±1.6</td>
<td>68.4±2.6</td>
<td>60.3</td>
<td>87.6 M</td>
</tr>
<tr>
<td></td>
<td>PERIN – labeled edge</td>
<td>48.9±4.3</td>
<td>72.1±0.9</td>
<td>72.6±1.1</td>
<td>67.1±1.6</td>
<td>61.6</td>
<td>88.2 M</td>
</tr>
<tr>
<td></td>
<td>PERIN – opinion-tuple</td>
<td>46.1±3.0</td>
<td>74.4±1.0</td>
<td>72.9±0.5</td>
<td>68.4±1.5</td>
<td>61.6</td>
<td>89.9 M</td>
</tr>
<tr>
<td></td>
<td>Frozen PERIN – opinion-tuple</td>
<td>48.1±6.4</td>
<td>65.5±1.8</td>
<td>69.2±5.5</td>
<td>62.2±2.7</td>
<td>56.5</td>
<td>0.07 M</td>
</tr>
<tr>
<td>MPQA</td>
<td>PERIN – node-centric</td>
<td>58.2±1.3</td>
<td>60.8±0.9</td>
<td>56.8±1.1</td>
<td>35.3±1.3</td>
<td>34.2±2.8</td>
<td>107.7 M</td>
</tr>
<tr>
<td></td>
<td>PERIN – labeled edge</td>
<td>57.1±2.0</td>
<td>54.8±1.6</td>
<td>55.2±1.1</td>
<td>33.1±0.4</td>
<td>33.2±1.2</td>
<td>109.6 M</td>
</tr>
<tr>
<td></td>
<td>PERIN – opinion-tuple</td>
<td>56.0±0.6</td>
<td>64.2±1.7</td>
<td>51.7±2.8</td>
<td>42.1±0.8</td>
<td>44.3±3.3</td>
<td>108.1 M</td>
</tr>
<tr>
<td></td>
<td>Frozen PERIN – opinion-tuple</td>
<td>42.0±8.8</td>
<td>48.1±1.7</td>
<td>46.6±2.8</td>
<td>35.2±2.8</td>
<td>26.9±4.2</td>
<td>0.37 M</td>
</tr>
<tr>
<td>DSU</td>
<td>PERIN – node-centric</td>
<td>0.0±0.0</td>
<td>41.5±4.3</td>
<td>40.3±2.6</td>
<td>27.2±2.0</td>
<td>33.9±2.4</td>
<td>107.7 M</td>
</tr>
<tr>
<td></td>
<td>PERIN – labeled edge</td>
<td>0.0±0.0</td>
<td>46.5±1.8</td>
<td>41.9±3.4</td>
<td>28.4±2.7</td>
<td>33.2±2.3</td>
<td>109.6 M</td>
</tr>
<tr>
<td></td>
<td>PERIN – opinion-tuple</td>
<td>12.0±1.0</td>
<td>50.9±4.7</td>
<td>42.6±2.9</td>
<td>34.8±3.4</td>
<td>39.5±2.6</td>
<td>108.1 M</td>
</tr>
<tr>
<td></td>
<td>Frozen PERIN – opinion-tuple</td>
<td>0.0±0.0</td>
<td>42.7±4.8</td>
<td>35.9±3.3</td>
<td>36.0±3.3</td>
<td>29.1±0.7</td>
<td>0.22 M</td>
</tr>
</tbody>
</table>

Table 8: 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 gener-

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 \times 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.
alization; 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`
- `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`
- `epochs = 50`
MPQA node-centric hyperparameters

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

MPQA labeled-edge hyperparameters

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

MPQA opinion-tuple hyperparameters

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

MPQA frozen opinion-tuple hyperparameters

```python
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

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

DSU labeled-edge hyperparameters

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

DSU opinion-tuple hyperparameters

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

DSU frozen opinion-tuple hyperparameters

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