Supplementary material

Steffen Eger[†‡], **Johannes Daxenberger**[†], **Iryna Gurevych**[†‡]
[†]Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universitt Darmstadt
[‡]Ubiquitous Knowledge Processing Lab (UKP-DIPF)
German Institute for Educational Research and Educational Information
http://www.ukp.tu-darmstadt.de

1 Experimental Setup: Detailed Descriptions

Pre-trained word embeddings: The sequence tagging systems, including the multi-task learners, as well as the neural dependency parsers can be initialized with pre-trained word embeddings. For our experiments, we chose Glove embeddings (Pennington et al., 2014) of different sizes (50, 100, and 200), the syntactic embeddings of Komninos and Manandhar (2016), and the "structured skip n-gram" model of Ling et al. (2015).

Hyperparameter optimization: Hyperparameter optimization is an art in itself and often makes the difference between state-of-the-art results or subpar performance (Wang et al., 2015). Finding good parametrizations for neural networks—such as size of the hidden units or number of hidden layers—is often a very challenging problem. For the dependency parsers as well as for the sequence taggers T in the $\texttt{STag}_T$ framing, we performed random hyperparameter optimization (Bergstra and Bengio, 2012), running systems 20 times with hyperparameters randomly chosen within pre-defined ranges, and then averaged this ensemble of 20 systems. These ranges were:[1]

- BiLSTM tagger in MTL setup: hidden layers of size 150 and 50 dimensional embedding layers (always using 50-dimensional Glove embeddings); the system was trained for 15 iterations and the best model on development set was chosen. All other hyperparameters at their defaults.

- BiLSTM-CNN-CRF tagger: one hidden layer of size in $\{125, 150, 200, 250\}$, randomly drawn; training was stopped when performance on development set did not improve for 5 iterations. All other hyperparameters at their defaults. Embeddings randomly chosen from the above-named pre-trained word embeddings, with a preference for 50 dimensional Glove embeddings.

For LSTM-ER, we ran the system with 50-dimensional Glove embeddings, which yielded better results than other embeddings we tried, and no further tuning. This is because, as outlined, the system already performs regularization techniques such as entity pre-training and scheduled sampling, which we did not implement for any of the other models. In addition, the system took considerably longer for training, which made it less suitable for ensembling.

For the neural parsers, our chosen hyperparameters can be read off from the accompanying scripts on our github. We trained the non-neural parsers with default hyperparameters.

Practical issues As outlined in the data section, our data has a particular structure, but the models we investigate are not guaranteed to yield outputs that agree with these conditions (unlike, e.g., ILP models where such constraints can be enforced). For example, the taggers T in the $\texttt{STag}_T$ framing do not need to produce a tree structure, nor do they need to produce legitimate B, I, O labeling—e.g., in BIO labeling, an "I" may never follow an "O". Likewise, while the parsers are guaranteed to output trees, the labeling they produce need not be consistent with our data. For example, an argumentative token may be predicted to link to a non-argumentative unit. Throughout, we observe very few such violations—that is, the systems tend to produce output consistent with the structures on which they were trained. Still, for such violations, we implemented simple and innocuous post-processing rules.

For the $\texttt{STag}_T$ systems, we corrected the fol-

[1] In all cases for the neural networks, we chose a development set of roughly 10% of the training set.

lowing:

(1) Invalid BIO structure, i.e., "I" follows "O".

(2) A predicted component is not homogeneous: for example, one token is predicted to link to the following argument component, while another token within the same component is predicted to link to the preceding argument component.

(3) A link goes 'beyond' the actual text, e.g., when a premise is predicted to link to another component at 'too large' distance $|d|$.

In case (1), we corrected "I" to "B". In case (2), we chose the majority labeling within the predicted component. In case (3), we link the component to the maximum permissible component; e.g., when a premise links to a claim at distance 3, but the last component in the document has distance 2, we link the premise to this claim. We applied (1), (2), and (3) in order. For $\texttt{STag}_{\texttt{BLCC}}$ this correction scheme led to 61 out of 29537 tokens changing their labeling in the test data (0.20%) on essay level and 69 on the paragraph level. For $\texttt{STag}_{\texttt{BL}}$ there were on average many more corrections. For example, 1373 (4.64%) tokens changed their labeling in the $\mathcal{Y}$-3:$\mathcal{Y}_C$-3 setting described in Table 2. This is understandable because a standard BiLSTM tagger makes output predictions independently; thus, more BIO, etc., violations can be expected.

For the parsers, we additionally corrected when (4) they linked to a non-argumentative unit at index i_n. In this case, we would re-direct the faulty link to the "closest" component in the vicinity of i_n (measured in absolute distance). Again, we applied (1) to (4) in order. For the LSTM-Parser, this led to 1224 corrections on token level (4.14%). While this may seem as leading to considerable improvements, this was actually not the case; most of our 'corrections' did not improve the measures reported—e.g., token level accuracy decreased, from 57.17% to 55.68%. This indicates that a better strategy might have been to re-name the non-argumentative unit to an argumentative unit.

For LSTM-ER, when a source component is predicted to relate to several targets (something which is always incorrect for our data), we connect the source to its closest target (and no other targets), measured in absolute distance. This is in agreement with the distributional properties of d sketched in Figure 2, which prefers shorter distances over longer ones.

Links to code used

We used the following code for our experiments: BLCC (https://github.com/XuezheMax/LasagneNLP); MTL BL (https://bitbucket.org/soegaard/mtl-cnn/src); LSTM-ER (https://github.com/tticoin/LSTM-ER); LSTM-Parser (https://github.com/clab/lstm-parser); Kiperwasser parser (https://github.com/elikip/bist-parser); Mate parser (https://code.google.com/archive/p/mate-tools/wikis/ParserAndModels.wiki); MST parser (http://www.seas.upenn.edu/~strctlrn/MSTParser/MSTParser.html). The results for the ILP model were provided to us by the first author of Stab and Gurevych (2016).

2 Error Analysis

We conduct some more error analysis, focussing on the three best models ILP, LSTM-ER and $\texttt{STag}_{\texttt{BLCC}}$.

Which component *types* are particularly difficult to detect? Table 1 investigates F1-scores for component segmentation+classification. In this case, there are seven classes: $\{\mathrm{B},\mathrm{I}\} \times \{\mathrm{C},\mathrm{MC},\mathrm{P}\} \cup \{\mathrm{O}\}$. We observe that the O class is particularly easy, as well as I-P. These two are the most frequent labels in the data and are thus most robustly estimated. While all systems are more troubled predicting the beginning of a claim than its continuation (this is often due to difficulty of predicting the inclusion or omission of discourse markers as illustrated above), major claims follow a reverse trend. Further analysis reveals that claims are often mistaken for premises and vice versa, and major claims for claims or—to a lesser degree—for premises. The mismatch between claims and premises is sometimes due to misleading introductory phrases such as "*Consequently*," which often imply conclusions (and hence claims), but sometimes also give reasons—i.e., premises—for other claims or premises.

We also note that the ILP model is substantially worse than the two LSTMs in all cases except for I-P on the component segmentation+classification task.

A major source of errors for *relations* is that either of their arguments (the two components) do not match exactly or approximately. When they do

	ILP	Paragraph LSTM-ER	Paragraph $\mathtt{STag}_{\mathbf{BLCC}}$	Essay LSTM-ER	Essay $\mathtt{STag}_{\mathbf{BLCC}}$
B-C	51.89	59.09	50.00	56.54	53.35
I-C	57.74	76.09	72.46	69.67	72.72
B-MC	76.56	80.64	78.26	77.15	73.80
I-MC	55.76	58.59	50.11	59.84	54.37
B-P	62.77	77.48	74.62	73.40	75.31
I-P	88.60	88.24	87.14	86.20	83.63
O	85.74	89.08	89.52	86.65	88.81
F1	68.56	75.62	71.76	72.93	72.66

Table 1: F1 scores in % for component segmentation+classification. Last row is macro-F1 score.

match, errors are mostly a mismatch between actual Attack/Against vs. predicted Support/For relations. Support/For relations are the vast majority in the PE data (94% and 82%, respectively). In rare cases, the two arguments have been correctly identified but their types are wrong (e.g. premise and claim while the gold components are claim and major claim, respectively).

References

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* 13:281–305.

Alexandros Komninos and Suresh Manandhar. 2016. Dependency based embeddings for sentence classification tasks. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. pages 1490–1500. http://aclweb.org/anthology/N/N16/N16-1175.pdf.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 1299–1304. http://www.aclweb.org/anthology/N15-1142.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Isaac Persing and Vincent Ng. 2016. End-to-end argumentation mining in student essays. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 1384–1394. http://www.aclweb.org/anthology/N16-1164.

Marina Sokolova and Guy Lapalme. 2009. A systematic analysis of performance measures for classification tasks. *Information Processing & Management* 45(4):427–437. https://doi.org/10.1016/j.ipm.2009.03.002.

Christian Stab and Iryna Gurevych. 2016. Parsing argumentation structures in persuasive essays. *arxiv preprint https://arxiv.org/abs/1604.07370, under review* .

Lidan Wang, Minwei Feng, Bowen Zhou, Bing Xiang, and Sridhar Mahadevan. 2015. Efficient hyper-parameter optimization for NLP applications. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 2112–2117. http://aclweb.org/anthology/D/D15/D15-1253.pdf.

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)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 1640–1649. http://www.aclweb.org/anthology/P13-1161.