CBNU System for SIGMORPHON 2019 Shared Task 2: a Pipeline Model

Uygun Shadikhodjaev and Jae Sung Lee
Department of Computer Science
Chungbuk National University
Cheongju City, Chungbuk Province, South Korea
ushadikhodjaev@gmail.com, jasonlee@cbnu.ac.kr

Abstract

In this paper we describe our system for morphological analysis and lemmatization in context, using a transformer-based sequence to sequence model and a biaffine attention based BiLSTM model. First, a lemma is produced for a given word, and then both the lemma and the given word are used for morphological analysis. We also make use of character level word encodings and trainable encodings to improve accuracy. Overall, our system ranked fifth in lemmatization and sixth in morphological accuracy among twelve systems, and demonstrated considerable improvements over the baseline in morphological analysis.

1 Introduction

In this paper we present our neural network architecture that we have used for the SIGMORPHON 2019 shared task 2 (McCarthy et al., 2019). We use two models by pipelining them in the sequence of operations. Our approach is based on the idea that lemmatization is an *m-to-n* mapping task where given a word of *m* characters we need to produce its lemma consisting of *n* characters. Unlike lemmatization, morphological analysis calls for a different approach where given a sentence consisting of *m* words, we need to choose one label from a fixed set of labels for each word. Hence, morphological analysis/tagging is a classification task for an input sequence.

	Word	Lemma	MSD
1	these	these	PL;DET
2	guys	guy	N;PL
3	were	be	PST;IND;V;FIN
4	fantastic	fantastic	ADJ
5	!	!	_

Table 1: Sample data of SIGMORPHON 2019 Shared Task 2

2 Task and Dataset

There are two tasks in SIGMORPHON 2019 and we chose task 2. The idea of the task is simple: the input is a sentence made of words and the output is a lemma and morphosyntactic description (MSD) for each word. Table 1 shows sample data for task 2: the first column is the input, the second is the lemma, and the last is the MSD for each word. There may be a difference in the result if a lemma is used as an additional input for MSD tagging. Our experiments showed improved performance when a lemma was incorporated.

The dataset consists of initial 98 datasets of more than 60 distinct languages, and additional nine surprise languages/datasets that were added later. Some of the datasets consist of languages that are not widespread in terms of their usage and amount of available training data. For example, Akkadian has only 80 sentences in training data, and other low-resource languages similarly have small numbers of sentences: Amharic has 859, Bambara 820, Buryat 741, Cantonese 520, etc. On the other hand, Russian SynTagRus and Czech PDT respectively have 49,511 and 70,330 sentences in their training data. In addition to

Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology, pages 19–24

having less training data, some of the low-resource languages also do not have pre-trained word vectors. In such cases, we use other related languages' word vectors as a substitute, as will be discussed later.

3 Model

The baseline model (Malaviya et al., 2019) provided by the task organizers approaches task 2 by first finding a MSD tag for a given word and incorporating that information in lemmatization. Given a sequence of words w, a sequence of morphological tags m, and a sequence of lemmas l, they define their model as:

$$p(l, m \mid w) = p\,(l|m, w)p(m|w) \qquad (1)$$

This illustrates the importance of MSD tags in the lemmatization process. However, lemmatization can be done effectively even without consideration of morphological tags. Therefore, our approach flips the order of operations: we first find the lemma for a given word and input the original sentence with the generated lemma to the MSD tagger. Equation 2 summarizes this idea:

$$p(m, l \mid w) = p\,(m|l, w)p(l|w) \qquad (2)$$

Overall, given the nature of the required tasks, an *m-to-n* sequence to sequence model for lemmatization and a label classifier model for morphological analysis are used. The two models are trained separately and pipelined as shown in Figure 1. As an example, when given an initial sentence "these guys are fantastic!", we lemmatize each input word as "these guy be fantastic!" We then input the derived lemmas and the original input to the MSD tagger. At the end, we obtain MSD tag for each input word.

3.1 Lemmatizer

Our lemmatizer is a sequence to sequence model and is based on an encoder-decoder architecture using Google's transformer (Vaswani et al., 2017). Lemmatization is a similar task to translation, where an input sequence is mapped to an output sequence of a different length. Therefore, our approach is justified by the model's robust performance in neural machine

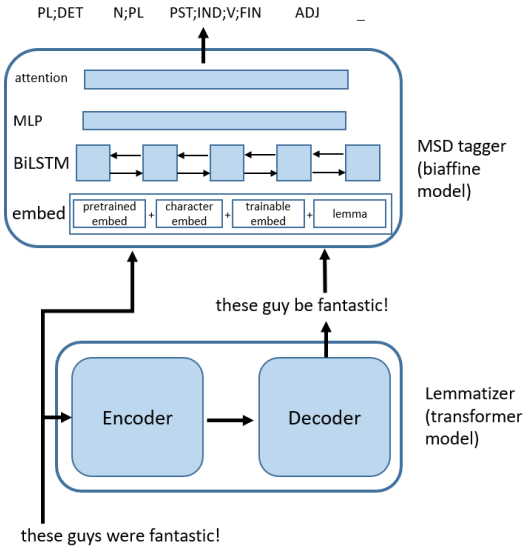

Figure 1: Pipeline Model

translation, particularly for WMT 2014 English-to-German and WMT 2014 EN-FR datasets. An informal leaderboard at http://nlpprogress.com demonstrates that the best performing teams use a transformer architecture for their encoder-decoder architecture (cf. Edunov et al., 2018, Wu et al., 2019).

A more formal leaderboard for the GLUE benchmark (Wang et al. 2018) consists of tasks that mainly use the encoder part of the encoder-decoder architecture. Therefore, the tasks of the GLUE benchmark are not directly comparable with lemmatization, but even in this case, at least the top 10 performers use BERT (Devlin et al., 2018), which uses a transformer encoder architecture (cf. Liu et al., 2019, Keskar et al., 2019).

The specific code for lemmatization is taken from the tensor2tensor library [1] version 1.13.4 with some modification added for our task. We chose the built-in hyperparameter configuration of *transformer_tiny*. The input and the output is a sequence of characters and no pre-trained embedding is used. One word is input at a time, and thus no consideration is taken of context words. For instance, in the mentioned example, the encoder input is "t h e s e" as a sequence of characters and the decoder output is "t h e s e". Likewise, "g u y s" and "g u y", "w e r e" and "b e", etc. are input and output one by one. Overall, the number of attention layers or heads is 4 as opposed to 8 in the original paper and hence it

[1] https://github.com/tensorflow/tensor2tensor

requires less computational power without substantial loss in the accuracy. The model performs quite well and with this basic setup was ranked fifth among 12 participating systems.

3.2 MSD tagger

The task of morphological analysis uses the output of lemmatization after pipelining it. Furthermore, MSD tagging is very similar to another well researched NLP task: head-dependent relation labelling in dependency parsing. Like head-dependent relation labelling, an MSD tag of a word is dependent on the word itself and its position within the sentence. As an example, let's consider two sentences: "I live in an apartment" and "I like live music". Even though "live" occurs in both sentences, the label we attach is dependent on the context. In other words, context words and the word itself determine its MSD tag. Therefore, we use the modified dependency parser reported by Dozat et al. (2017), which is based on Kiperwasser et al. (2016). The original model won in the CoNLL 2017 shared task (Nivre et al. 2017a, Nivre et al. 2017b) and its subsequent modifications won in the CoNLL 2018 shared task (Zeman et al., 2018, Che et al., 2018). Unlike dependency parsing, for the morphological analysis it is not necessary to find the head of a word. Therefore, we amend the dependency parser by Dozat et al. (2017) and use only the model's head-dependent relation labeling functionality for the MSD tagging.

The model's input is an elementwise addition of four embeddings for an input word. We then pass the vector representation for each input word through BiLSTM layers with subsequent multilayer perceptron (MLP) and biaffine attention layers. The MSD tagging assigns a tag to each word while the dependency parsing assigns a tag to a relation between a pair of words. In the latter case, even though we need to tag a relation between a pair of words, each word needs a label. Furthermore, information from two words only is not enough and the parser has to attend actually to the whole context to assign the correct label. Therefore, we need attention over all input words in the dependency parsing and we leave this feature for the MSD tagger too.

The optimization is done by the Adam optimizer (Knigma and Ba 2014). We trained the model until there were no improvements after 5000 steps. The number of BiLSTM layers was

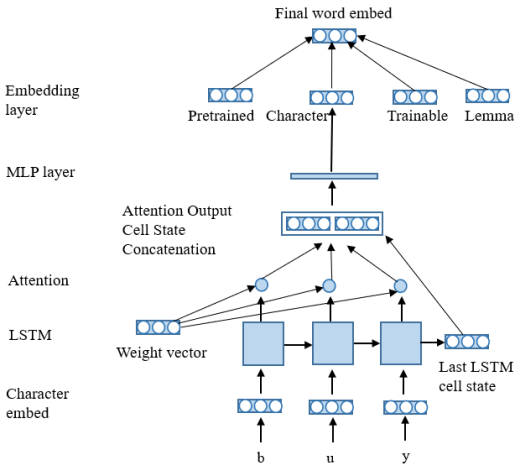

Figure 2: Character level embedding

three and the dimension of each LSTM cell as well as the word vector was 100 (300 when fastText [2] is used). We mainly used pre-trained embeddings of words from the CoNLL 2017 shared task (Nivre et al. 2017a, Nivre et al. 2017b) trained on word2vec (Mikolov et al., 2013). For Akkadian, Amharic, and Japanese we used fastText (Bojanowski et al., 2017). Interestingly, using the pre-trained word vector of Dutch from the CoNLL 2017 shared task demonstrates better performance than the Afrikaans pre-trained word vector of fastText for Afrikaans-AfriBooms treebank. Similar results were observed for some other datasets and therefore we used fastText only for the mentioned languages. At the same time, using the word vector for a related language is also in the spirit of cross-lingual learning transfer from a resource-rich to a resource-lean language (Ruder et al., 2017).

For each word, there are four embeddings, which are summed elementwise: pre-trained, trainable, character level, and lemma. Trainable embeddings are vectors that are initialized randomly and then trained as the training proceeds. Likewise, lemma vectors are also initialized randomly. The process of character level embedding generation is more involved and is based on the character level word representation by Cao and Rei (2016). Character level embeddings are a sequence of characters that pass through unidirectional LSTM cells (Hochreiter and Schmidhuber, 1997) and are then

[2] https://fasttext.cc/

summed after the conventional attention layer (Bahdanau et al., 2015). Figure 2 summarizes this process.

4 Results

After experiments with different hyperparameter settings, we were able to choose optimal settings, as was described earlier. Table 2 summarizes the results of lemmatization and MSD tagging by the sequence to sequence transformer model and the biaffine attention based BiLSTM model.

Our choice of lemmatization followed by an MSD tagging was an important step for increasing MSD tag accuracy. Although, a full-scale ablation

Treebanks	lemma acc.	lemma Leven.	morph acc.	morph F1
Afri. AfriBooms	98.49	0.03	98.45	98.66
Akk.PISANDUB	67.82	0.89	82.18	81.77
Amharic-ATT	99.91	0.00	88.19	92.41
A. Greek-Perseus	94.48	0.14	81.75	91.58
A. Greek-PROIEL	96.75	0.08	83.82	93.86
Arabic-PADT	94.16	0.16	93.22	96.32
Arabic-PUD	85.29	0.42	79.16	91.27
Armenian-ArmTDP	94.34	0.11	76.80	84.91
Bambara-CRB	83.90	0.30	92.79	94.74
Basque-BDT	95.75	0.10	87.63	92.80
Belarusian-HSE	89.81	0.19	58.67	65.26
Breton-KEB	92.54	0.19	87.36	90.13
Bulgarian-BTB	96.56	0.09	96.02	98.00
Buryat-BDT	89.23	0.26	80.48	82.93
Cantonese-HK	100	0.00	90.00	87.40
Catalan-AnCora	97.20	0.05	96.19	97.71
Chinese-CFL	99.76	0.00	91.49	90.37
Chinese-GSD	99.98	0.00	94.60	94.42
Coptic-Scriptorium	89.95	0.21	94.81	95.93
Croatian-SET	95.14	0.09	88.64	94.64
Czech-CAC	98.22	0.05	91.76	96.86
Czech-CLTT	98.41	0.03	90.01	94.98
Czech-FicTree	97.89	0.04	91.49	95.6
Czech-PDT	98.08	0.03	89.88	95.84
Czech-PUD	93.06	0.12	76.17	89.38
Danish-DDT	94.86	0.08	95.52	96.96
Dutch-Alpino	97.37	0.05	96.45	97.18
Dutch-LassySmall	96.45	0.07	96.38	97.00
English-EWT	97.31	0.08	95.82	97.01
English-GUM	97.09	0.05	95.46	96.54
English-LinES	97.87	0.04	96.34	97.16
English-ParTUT	97.30	0.05	94.75	95.56
English-PUD	94.90	0.07	93.43	94.95
Estonian-EDT	95.76	0.09	93.08	96.45
Faroese-OFT	88.28	0.22	81.08	88.28
Finnish-FTB	95.87	0.09	92.55	95.59
Finnish-PUD	89.09	0.23	88.52	93.32
Finnish-TDT	95.68	0.10	93.62	96.22
French-GSD	97.56	0.04	96.76	97.98
French-ParTUT	95.81	0.07	93.10	96.54
French-Sequoia	97.32	0.05	96.27	98.13
French-Spoken	97.17	0.06	97.25	97.31
Galician-CTG	97.00	0.04	97.94	97.73
Galician-TreeGal	94.05	0.08	92.74	95.58
German-GSD	97.11	0.06	86.05	93.73
Gothic-PROIEL	96.62	0.09	82.33	91.77
Greek-GDT	95.98	0.08	93.24	97.26
Hebrew-HTB	96.83	0.06	95.84	97.22
Hindi-HDTB	96.40	0.04	91.05	96.65
Hungarian-Szeged	95.19	0.09	88.11	94.63
Indonesian-GSD	99.50	0.01	90.17	93.15
Irish-IDT	91.24	0.20	82.40	88.35
Italian-ISDT	96.82	0.07	96.81	98.05
Italian-ParTUT	96.34	0.09	96.08	97.59
Italian-PoSTWITA	95.26	0.11	95.12	96.33
Italian-PUD	94.14	0.13	93.32	96.40
Japanese-GSD	98.13	0.02	97.74	97.46
Japanese-Modern	96.94	0.04	96.74	96.74
Japanese-PUD	97.46	0.03	97.88	97.65
Komi_Zyrian-IKDP	80.47	0.30	53.12	42.98
Komi_Zyrian-Lattice	84.07	0.38	57.14	65.07
Korean-GSD	93.19	0.12	95.87	95.25
Korean-Kaist	95.57	0.07	96.71	96.30
Korean-PUD	97.96	0.04	91.02	93.99
Kurmanji-MG	91.40	0.17	79.48	87.13
Latin-ITTB	97.44	0.06	93.32	96.62
Latin-Perseus	91.16	0.19	78.68	88.54
Latin-PROIEL	96.51	0.08	87.99	95.16
Latvian-LVTB	95.77	0.07	91.60	95.10
Lithuanian-HSE	86.42	0.30	56.03	57.49
Marathi-UFAL	74.25	0.65	47.43	59.40
Naija-NSC	99.93	0.00	94.94	93.17
North_Sami-Giella	91.96	0.16	87.04	91.90
Norwegian-Bokmaal	97.83	0.03	95.81	97.40
Norwegian-Nynorsk	97.74	0.04	94.87	96.60
N.NynorskLIA	97.51	0.04	93.03	94.29
OCS-PROIEL	96.51	0.08	83.44	91.82
Persian-Seraji	96.27	0.17	97.06	97.70
Polish-LFG	95.66	0.08	92.19	96.23
Polish-SZ	94.99	0.09	89.17	94.58
Portuguese-Bosque	95.13	0.08	93.39	96.48
Portuguese-GSD	87.82	0.25	96.91	97.14
Rom.-Nonstandard	93.40	0.14	91.91	95.60
Romanian-RRT	95.53	0.09	96.85	97.99
Russian-GSD	95.89	0.07	88.91	94.20
Russian-PUD	90.72	0.16	79.88	90.15
Russian-SynTagRus	96.97	0.06	93.28	96.46
Russian-Taiga	89.86	0.22	76.53	84.11
Sanskrit-UFAL	61.81	0.92	33.17	46.19
Serbian-SET	96.42	0.07	91.76	95.34
Slovak-SNK	96.24	0.07	89.24	94.68
Slovenian-SSJ	96.38	0.06	91.56	95.27
Slovenian-SST	93.79	0.13	83.44	90.24
Spanish-AnCora	97.69	0.04	96.64	97.98
Spanish-GSD	98.31	0.03	93.97	96.78
Swedish-LinES	95.37	0.08	92.57	96.24
Swedish-PUD	91.65	0.12	92.66	95.43
Swedish-Talbanken	96.56	0.05	96.66	98.16
Tagalog-TRG	83.78	0.54	72.97	79.70
Tamil-TTB	91.52	0.23	79.13	88.48
Turkish-IMST	96.34	0.07	87.37	91.37
Turkish-PUD	85.13	0.37	81.89	89.47
Ukrainian-IU	95.42	0.09	88.70	94.23
Upper_Sorbian-UFAL	90.66	0.14	66.95	76.85
Urdu-UDTB	94.25	0.08	79.06	92.21
Vietnamese-VTB	99.93	0.00	91.79	90.69
Yoruba-YTB	98.84	0.01	91.47	92.05
Mean	**94.07**	**0.12**	**88.09**	**91.84**
Median	**95.87**	**0.08**	**91.76**	**95.16**
Mean – baseline by the task organizers	**94.17**	**0.13**	**72.18**	**86.25**
Median – baseline by the task organizers	**95.92**	**0.08**	**76.40**	**89.45**

Table 2: Test set scores

study was not performed due to time constraints, an experiment for MSD tagging without lemma on English-PUD and Korean-Kaist treebanks were performed. On both datasets, a decrease in accuracy was observed. For English-PUD's morph accuracy and F1 scores decreased by 1.18 and 0.43 percentage points, while Korean-Kaist's respective scores decreased by 7.50 and 8.41 percentage points. We conjecture that the larger decrease in Korean is due to its higher morphological complexity than English; a lemma itself is more important to find MSD tags for morphological rich languages.

In general, as more training data were available, higher scores were obtained in absolute terms. As an example, for Russian, among four available datasets (Russian-GSD, Russian-PUD, Russian-SynTagRus, and Russian-Taiga) Russian-SynTagRus was the largest, and its accuracy was best by all four metrics used.

Some languages have more MSD tags than others and therefore present another dimension for the task complexity. For instance, Czech-PDT treebank has 2895 unique MSD tags while English-EWT has only 179, i.e. 16 times less. This, therefore, partly affects the accuracy of the MSD tagger, where Czech-PDT treebank's morphological accuracy is 89.88% while English-EWT's is 95.82%.

While there is a lot of variance in the number of MSD tags among languages, most of the languages have around twenty to sixty characters in their alphabet. Hence, the number of characters in the alphabet does not seem to affect lemmatization. At the same time, Chinese uses distinct characters for each word and does not have word inflections. Despite having 3536 unique characters, Chinese-GSD treebank's lemma accuracy is 99.98%. It also has only 40 MSD tags due to the absence of inflections.

Overall, lemmatization appears to be a slightly easier task than MSD tagging, and in our case, incorporating lemma information in MSD tagging yielded more accurate results for the latter.

5 Conclusion

Our pipeline model has shown favorable results in SIGMORPHON Shared Task 2 and scored fifth and sixth place, respectively, for lemmatization and MSD tagging. For future work, it would be interesting to assess how incorporating the output of MSD tagging into lemmatization would affect lemma accuracy.

Acknowledgments

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (grant number: 2017R1D1A3B03035676).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*, San Diego, CA, USA.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146

Kris Cao and Marek Rei. A joint model for word embedding and word morphology. 2016. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 18–26, Berlin, Germany. Association for Computational Linguistics.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. 2018. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64, Brussels, Belgium. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. CoRR, abs/1810.04805.

Timothy Dozat, Peng Qi, and Christopher D. Manning. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. 2017. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language*

Processing, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. Neural Comput.,9(8):1735–1780.

Nivre Joakim, Agić Željko, Ahrenberg Lars, et al., 2017a, Universal Dependencies 2.0, LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University, http://hdl.handle.net/11234/1-1983.

Nivre Joakim, Agić Željko, Ahrenberg Lars, et al., 2017b, Universal Dependencies 2.0 – CoNLL 2017 Shared Task Development and Test Data, LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University, http://hdl.handle.net/11234/1-2184.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *CoRR*, abs/1901.11504.

Chaitanya Malaviya, Shijie Wu, and Ryan Cotterell. 2019. A simple joint model for improved contextual neural lemmatization. arXiv preprint arXiv:1904.02306v2.

Arya D. McCarthy, Ekaterina Vylomova, Shijie Wu, Chaitanya Malaviya, Lawrence Wolf-Sonkin, Garrett Nicolai, Christo Kirov, Miikka Silfverberg, Sebastian Mielke, Jeffrey Heinz, Ryan Cotterell, and Mans Hulden. 2019. The SIGMORPHON 2019 shared task: Crosslinguality and context in morphology. In *Proceedings of the 16th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, Florence, Italy. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*, San Diego, CA, USA.

Sebastian Ruder, Ivan Vulic, and Anders Sogaard. 2017. A survey of cross-lingual word embedding models. cite arxiv:1706.04902.

Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Unifying question answering and text classification via span extraction. *CoRR*, abs/1904.09286.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems* 30, pages 5998–6008. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. 2018. In *Proceedings of the 2018 EMNLP Workshop Blackbox NLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. In *International Conference on Learning Representations*.

Daniel Zeman, Jan Hajic, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium, Association for Computational Linguistics.