An Extended Sequence Tagging Vocabulary for Grammatical Error Correction

Stuart Mesham[♠] **Christopher Bryant**[♢] **Marek Rei**[♡♢] **Zheng Yuan**[♣♢]
[♠]Department of Computer Science, University of Cambridge
[♢]The ALTA Institute, Department of Computer Science, University of Cambridge
[♡]Department of Computing, Imperial College London
[♣]Department of Informatics, King's College London
sm2613@cantab.ac.uk, christopher.bryant@cl.cam.ac.uk
marek.rei@imperial.ac.uk, zheng.yuan@kcl.ac.uk

Abstract

We extend a current sequence-tagging approach to Grammatical Error Correction (GEC) by introducing specialised tags for spelling correction and morphological inflection using the SymSpell and LemmInflect algorithms. Our approach improves generalisation: the proposed new tagset allows a smaller number of tags to correct a larger range of errors. Our results show a performance improvement both overall and in the targeted error categories. We further show that ensembles trained with our new tagset outperform those trained with the baseline tagset on the public BEA benchmark.

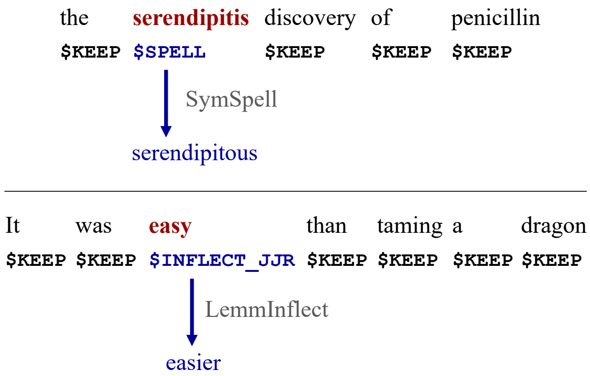

Figure 1: Our model applied to two inputs. Beneath each word is the tagger's output. Arrows denote transformations by SymSpell and LemmInflect respectively.

1 Introduction

Current approaches to Grammatical Error Correction (GEC) fall under two broad categories: sequence-to-sequence and sequence-tagging. The former treats GEC as a machine-translation problem, "translating" from error-containing to error-free language (Yuan and Briscoe, 2016; Schmaltz et al., 2017; Junczys-Dowmunt et al., 2018; Grundkiewicz et al., 2019; Yuan et al., 2019; Rothe et al., 2021). By contrast, sequence-tagging approaches tag each input word with an edit operation such that applying the operations produces the corrected output (Yannakoudakis et al., 2017; Awasthi et al., 2019; Omelianchuk et al., 2020; Tarnavskyi et al., 2022). The basic operations include keeping a word unchanged, deleting a word, and inserting new words (Awasthi et al., 2019; Malmi et al., 2019).

One advantage of sequence-tagging over sequence-to-sequence approaches is computational efficiency: the former do not require expensive auto-regressive decoding,[1] and currently achieve competitive performance using smaller models (Tarnavskyi et al., 2022; Rothe et al., 2021). However, current sequence-tagging approaches require manual linguistic efforts to curate language-specific edit tags (Yuan et al., 2021). For example, Awasthi et al. (2019) introduce rule-based morphological inflection tags, like replacing the "-ing" suffix with "-ion" (e.g. completing → completion). Omelianchuk et al. (2020) introduce a wider range of operations including verb-form and noun-number changes. For verb-form inflections, they use a dictionary to map between verb forms.[2]

In this paper, we focus on a sequence-tagging approach. We extend the approach of Omelianchuk et al. (2020) by introducing more general transformation tags (Figure 1). Specifically, we introduce:

- A tag for correcting spelling errors.
- Inflection tags capable of a broader range of inflections than the tags introduced by Omelianchuk et al. (2020).

These modifications allow a broader range of errors to be handled by a smaller number of transformation tags, which simplifies the sequence tagging

[1] Malmi et al. (2019) show that sequence-taggers can be orders of magnitude faster than comparable seq-to-seq models at inference time.

[2] https://github.com/gutfeeling/word_forms/blob/master/word_forms/en-verbs.txt

Findings of the Association for Computational Linguistics: EACL 2023, pages 1608–1619
May 2-6, 2023 ©2023 Association for Computational Linguistics

problem, as well as improves the generalisation of the GEC system. Our results show that our modifications improve the system's performance on the BEA-2019 development and test sets. Our code and model weights are publicly available.[3]

2 Methods

We extend the system described by Omelianchuk et al. (2020) by adding new tags to the model's output vocabulary and modifying the inference and dataset preprocessing code to support our new tags. Our new tags perform spelling correction and morphological inflection and are described in Sections 2.3 and 2.4 below. We evaluate our tagset using the RoBERTa (Liu et al., 2019), DeBERTa (He et al., 2021b), DeBERTaV3 (He et al., 2021a), ELECTRA (Clark et al., 2020) and XLNet (Yang et al., 2019) encoders, as well as an ensemble of three encoders (see Section 2.6).

2.1 Model and Training Procedure

Our work builds on GECToR from Omelianchuk et al. (2020), which follows the sequence tagging approach to GEC. We use the same sequence tagger architecture: a pre-trained transformer encoder with two separate "tagging" and "detection" heads. We also follow the same multi-phase training procedure using the synthetic PIE Corpus (Awasthi et al., 2019), NUCLE (Dahlmeier et al., 2013), FCE (Yannakoudakis et al., 2011), Lang-8 (Mizumoto et al., 2011; Tajiri et al., 2012) and W&I + LOCNESS (Bryant et al., 2019) English datasets.

2.2 Baseline Tagset

GECToR's tagset includes the basic edit tags, `$KEEP`, `$DELETE`, `$REPLACE_{t}` and `$APPEND_{t}`, which respectively leave the word unchanged, delete the word, replace the word with another word t, and append t after the input word.

The tagset also contains a set of more complex grammatical transformation or "g-transform" tags. These include case, agreement (singular/plural), verb-form and merge/split transformations. For example, there is a tag to transform a verb into its past-tense equivalent. The verb-form transformations are performed using a dictionary. Omelianchuk et al. (2020, Table 9) provide a full list of the transformations and their descriptions.

2.3 Spelling Correction Tag

GECToR corrects spelling errors using its vocabulary of `$REPLACE_{t}` tags. This limits its ability to generalise to unseen or rare spelling errors for two reasons. The first is that GECToR can only correct misspellings of words which appear in its output vocabulary. The second is that for each word, there are many possible misspellings that the model must learn to associate with the corrected form.

To remedy this, we introduce a new `$SPELL` tag for spelling correction. When this tag is predicted during inference, we use SymSpell[4] to produce the corrected version of the input word (see Section A.1 for details). We hypothesise that this improves generalisation because the sequence tagger need only detect spelling errors, and the corrections are performed by SymSpell. SymSpell can handle a variety of misspellings of each word and can correct words from a dictionary much larger than the output vocabulary of the sequence tagger.

2.4 Inflection Tags

We introduce inflection tags of the form `$INFLECT_{POS}` where POS denotes the Penn Treebank POS tag of the desired form of the input word. When an inflection tag is predicted at inference time, the input word is inflected to the target POS specified in the tag. The inflection is achieved using the software modules spaCy[5] and LemmInflect[6]. LemmInflect first attempts to use a dictionary for the inflection. If the input word is not in LemmInflect's dictionary, the inflection is performed using a rule-based approach (see Section A.2 for details).

Our inflection tags offer two main advantages over GECToR's dictionary-based verb transformations. The first is that they are not limited to verbs, but rather can be used for any inflected part of speech.[7] The second is that words which do not appear in LemmInflect's dictionary can still be handled using a rule-based approach (see Section A.2). We also note that GECToR's singular/plural transformation tag only adds or removes an "-s" from the end of the input word, making it unable to handle less trivial cases such as inflecting "activity" to its plural "activities". By contrast, our system

[3] https://github.com/StuartMesham/gector_experiment_public

[4] https://github.com/wolfgarbe/SymSpell#single-word-spelling-correction

[5] https://spacy.io

[6] https://github.com/bjascob/LemmInflect

[7] In English, the inflected parts of speech are adjectives, adverbs, nouns and verbs.

Model	BEA-2019 dev precision	recall	$F_{0.5}$	BEA-2019 test precision	recall	$F_{0.5}$ $(\bar{x} \pm \sigma)$
DeBERTa$_{5K}^{(L)}$ basetags	68.13	38.12	58.86	77.89	56.72	72.47 ± 0.56
DeBERTa$_{5K}^{(L)}$ `$SPELL`	68.37	**39.03**	59.40	77.96	**57.67**	72.82 ± 0.49
DeBERTa$_{5K}^{(L)}$ `$INFLECT`	68.73	38.43	59.33	77.72	57.23	72.51 ± 0.93
DeBERTa$_{5K}^{(L)}$ `$SPELL` + `$INFLECT`	**69.75**	38.97	**60.20**	**78.45**	57.44	**73.09 ± 0.72**
ensemble basetags	73.25	37.17	61.32	83.47	55.64	75.87 ± 0.20
ensemble `$SPELL`	73.54	37.76	61.79	**83.72**	**56.28**	**76.26 ± 0.37**
ensemble `$INFLECT`	73.89	37.35	61.80	83.71	55.68	76.06 ± 0.43
ensemble `$SPELL` + `$INFLECT`	**74.19**	**38.16**	**62.39**	83.59	56.23	76.17 ± 0.38
DeBERTa$_{10K}^{(L)}$ $\bigoplus$ RoBERTa$_{10K}^{(L)}$ $\bigoplus$ XLNet$_{5K}^{(L)}$ (Tarnavskyi et al., 2022)	70.32	34.62	58.30	84.44	54.42	76.05
RoBERTa$_{5K}^{(L)}$ (KD) (Tarnavskyi et al., 2022)	-	-	-	80.70	53.39	73.21
T5 xxl (Rothe et al., 2021)	-	-	-	-	-	75.88
ESC (Qorib et al., 2022)	**73.63**	**40.12**	**63.09**	**86.65**	**60.91**	**79.90**

Table 1: A table showing BEA-2019 development and test set scores. The top section shows our models with varying tagsets using the DeBERTa$_{5K}^{(L)}$ encoder. The middle section shows the results for our ensemble models with varying tagsets. In the table, "ensemble" denotes the encoders DeBERTa$_{5K}^{(L)}$ $\bigoplus$ ELECTRA$_{5K}^{(L)}$ $\bigoplus$ RoBERTa$_{5K}^{(L)}$. Finally, the bottom section shows models from related work. The model labelled "(KD)" was trained using Tarnavskyi et al. (2022)'s knowledge distillation procedure. The results in the top and middle sections are averaged over 6 seeds, and the standard deviation, σ, of the test $F_{0.5}$ is shown.

applies the full dictionary and rule-based procedure to singular/plural transformations. In summary, our inflection tags handle a broader range of transformations than GECToR's transformation tags. We hypothesise that this improves generalisation.

2.5 Preprocessing

To incorporate our `$SPELL` tag into the training data, we take data preprocessed with Omelianchuk et al. (2020)'s code, and for each instance of a `$REPLACE_{t}` tag, we apply SymSpell to the input word. If SymSpell produces the correct output, t, we change the `$REPLACE_{t}` tag to a `$SPELL` tag. Otherwise, we leave the `$REPLACE_{t}` tag unchanged.

For the inflection tags, we first modify Omelianchuk et al. (2020)'s preprocessing code by removing existing tags which perform inflections.[8] Then, similar to our process for the `$SPELL` tag, for each instance of a `$REPLACE_{t}` tag, we attempt to inflect the input word to obtain the target word t and, if successful, change the tag to an `$INFLECT_{POS}` tag. Otherwise, we leave the tag unchanged. For details about this process, we refer the reader to the relevant script in our repository.[9]

2.6 Ensembling

To create ensemble models, we use the span-based voting procedure of Tarnavskyi et al. (2022). Their system takes the corrected output of each model, compares it with the input text, and extracts edit spans of the same type (insert, delete, or replace). In an ensemble of k models, spans predicted by at least $k-1$ models are included in the output of the ensemble.

Our particular combination of encoders was chosen on the BEA-2019 development set by searching over all possible combinations of three models from the set of individual models we trained with the `$SPELL` + `$INFLECT` tagset.

3 Results

We report the span-based precision, recall and $F_{0.5}$ scores on the BEA-2019 development and test sets (Bryant et al., 2019) using the ERRANT scorer (Bryant et al., 2017).[10] The term "basetags" indicates the tagset proposed by Omelianchuk et al.

[8] We remove tags g-8 to g-29 (Omelianchuk et al., 2020, Table 9).

[9] See the `lemminflect_preprocess.py` script in the `utils` directory of our repository.

[10] https://github.com/chrisjbryant/errant

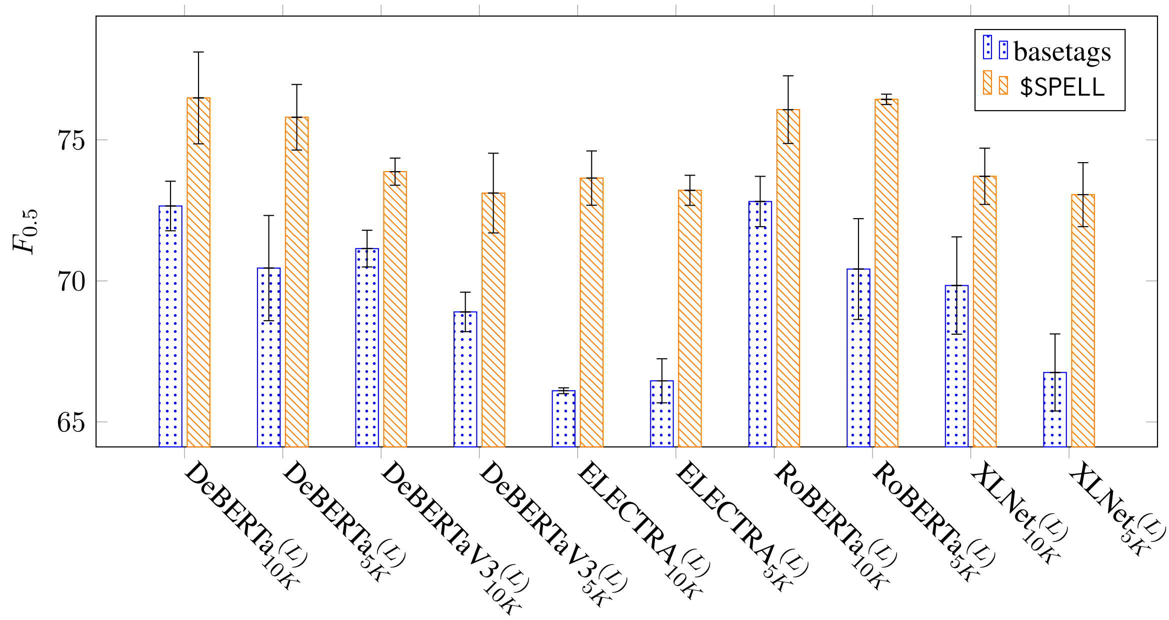

Figure 2: A bar graph showing the BEA-2019 development set $F_{0.5}$ scores for the "spelling" error category for different encoders, tagsets and vocabulary sizes. Specifically, the `$SPELL` and basetags tagsets and vocabulary sizes of 5k and 10k. Each bar represents the mean score over three training runs with different seeds. The error bars show the standard deviations of the scores.

(2020), and `$SPELL` and `$INFLECT` denote our proposed tagsets containing the spelling and inflection tags respectively. `$SPELL` + `$INFLECT` denotes tagsets containing both the spelling and inflection tags. We adopt the model and tagset size notation of Tarnavskyi et al. (2022) which, for example, denotes a DeBERTa-large model using a 5k vocabulary size as DeBERTa$^{(L)}_{5K}$.

Table 1 shows the scores of our models on the BEA-2019 development and test sets. Of the three encoders chosen for our ensemble, DeBERTa$^{(L)}_{5K}$ had the highest mean development set score when using `$SPELL` + `$INFLECT` tagset, and is thus shown in Table 1.[11]

For the DeBERTa$^{(L)}_{5K}$ encoder, on both the development and test sets, the `$SPELL` and `$INFLECT` tagsets provide an improvement over the basetags tagset, and the `$SPELL` + `$INFLECT` tagset provides a larger improvement. Similarly, for the ensemble models, on the development set, the `$SPELL` and `$INFLECT` tagsets show an improvement over the basetags tagset, and the `$SPELL` + `$INFLECT` tagset obtains the highest score. However, on the test set, the `$SPELL` tagset scores the highest.

[11]See Section A.6 for the results of the other encoders, and Section A.8 for CoNLL-2014 results.

3.1 Target Error Categories

Figures 2 and 3 show BEA-2019 development set scores in the ERRANT error categories (Bryant et al., 2017, Table 2) targeted by the `$SPELL` and `$INFLECT` tagsets respectively. The former targets only the "spelling" error category, and the latter targets categories related to inflection.[12] In Figure 2 we observe substantial performance improvements in the spelling category for all models. Figure 3 shows a smaller improvement in the target error categories of the `$INFLECT` tagset for all models except XLNet$^{(L)}_{10K}$.

4 Discussion

In general, the `$SPELL` and `$INFLECT` tagsets both improve performance over the baseline tagset. The results of Section 3.1 show that the tagsets improve performance in their respective targeted error categories. This indicates that our modifications were successful.

In the results showing all error categories (Table 1), the inclusion of many non-targeted categories reduces the weighting of targeted categories, resulting in smaller apparent differences between models. For the ensemble models, the `$SPELL`

[12]Specifically, the ADJ:FORM, MORPH, NOUN:INFL, NOUN:NUM, VERB:FORM, VERB:INFL, VERB:SVA and VERB:TENSE categories.

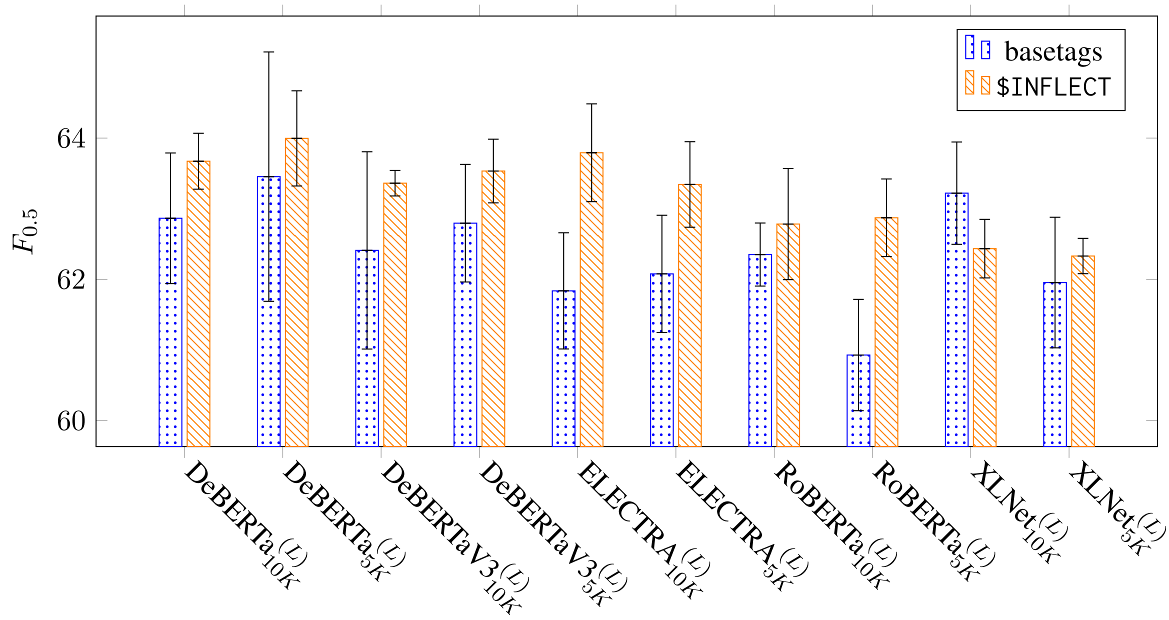

Figure 3: A bar graph showing the BEA-2019 development set $F_{0.5}$ scores for inflection-related errors for different encoders, tagsets and vocabulary sizes. Specifically, the `$INFLECT` and basetags tagsets and vocabulary sizes of 5k and 10k. Each bar represents the mean score over three training runs with different seeds. The error bars show the standard deviations of the scores.

tagset obtains a higher test score than the `$SPELL` + `$INFLECT` tagset. This is contrary to our expectation that the combination of our modifications should provide a cumulative improvement. It is also unexpected that the ranking of the ensemble models on the development and test sets differs.

Differences in error-type frequencies in the development and test sets do not provide an explanation, since the frequency of spelling errors is lower in the test set than in the development set, and the frequencies of the error types which the `$INFLECT` tagset most impacts[13] are higher in the test set than in the development set (Bryant et al., 2019, Table 4). We therefore hypothesise that this unexpected pattern is an artefact of the variation between different random seeds.

5 Conclusions

We have motivated and described new tags for spelling correction and morphological inflection. These tags are capable of correcting a broader range of errors than previous tags, thereby improving generalisation. Our results show that the new tags improve performance both in the targeted error categories and overall for both single-encoder models and ensembles.

Our findings ultimately show there is great scope for improving GEC sequence-labelling model performance by introducing tags capable of correcting more general and possibly complex classes of errors.

Finally, we believe our results are of immediate value to practitioners building GEC applications since they offer improved performance without the use of seq-to-seq models which can require orders of magnitude more computation at inference time.

6 Future Work

We used SymSpell in its context-free configuration when correcting spelling errors. We chose this method because of its speed and simplicity, however, better performance could likely be obtained by switching to a context-sensitive spelling correction algorithm.

Although our experiments demonstrate a performance improvement over the results of Tarnavskyi et al. (2022), other recent work has demonstrated further performance improvements (Lai et al., 2022; Qorib et al., 2022). Our contribution is orthogonal to these, and so future work could investigate whether using our tagset for the sequence tagger used by Lai et al. (2022) or using our models in the ensemble described by Qorib et al. (2022) would yield further improvements.

[13]Specifically the NOUN:NUM, VERB:FORM and VERB:SVA error types. See Section A.7 for details.

Limitations

The results obtained have high variance with respect to the random seed used (see Appendix Figures 5 and 6). Due to compute limitations, we were unable to run more seeds to better observe the distributions of development and test scores.

The generalised tags we experimented with are also somewhat language specific, as, for example, the `$INFLECT` tagset will not be beneficial to a language with little or no morphology.

Ethics Statement

This work is conducted in accordance with the ACM Code of Ethics.[14] In this section we comment on the topics of privacy, safety and accessibility, as we believe they are particularly relevant to the development and use of our system.

Privacy

Since machine learning systems can reveal sensitive information about their training data, it is important to consider privacy concerns relating to the development and use of such systems. The training data for our system originates from two primary sources: publicly available text and essays collected from examinations and online error correction services. The PIE Corpus is derived from publicly available texts (Awasthi et al., 2019). The Lang-8 and Write & Improve essays are collected in accordance with the services' respective privacy policies. The FCE dataset is anonymised before use (Yannakoudakis et al., 2011). Privacy-related information is not documented for the NUCLE and LOCNESS datasets.

Safety

Automated GEC systems have the potential to change the meaning of the input text. Therefore, the systems described in this work should be applied with caution. In scenarios where miscommunication is dangerous, the system should only be used as an aid for the manual correction of text, rather than a fully automated system.

Accessibility

The development of our system required compute-intensive model training and data preprocessing.[15] This cost may be prohibitive for some research groups or potential users. We make our trained models, hyperparameters and source code publicly available to alleviate this issue and increase the accessibility of our developments.

Acknowledgements

This work was primarily funded by the Skye Foundation and Cambridge Trust.

[14] https://www.acm.org/code-of-ethics

[15] See Section A.5 for details.

References

Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. 2019. Parallel iterative edit models for local sequence transduction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4260–4270, Hong Kong, China. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75, Florence, Italy. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations*.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572, Montréal, Canada. Association for Computational Linguistics.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner English: The NUS corpus of learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31, Atlanta, Georgia. Association for Computational Linguistics.

Fred J Damerau. 1964. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 252–263, Florence, Italy. Association for Computational Linguistics.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021a. DeBERTaV3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021b. DeBERTa: Decoding-enhanced BERT with disentangled attention. In *9th International Conference on Learning Representations*.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 595–606, New Orleans, Louisiana. Association for Computational Linguistics.

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

Shaopeng Lai, Qingyu Zhou, Jiali Zeng, Zhongli Li, Chao Li, Yunbo Cao, and Jinsong Su. 2022. Type-driven multi-turn corrections for grammatical error correction. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3225–3236, Dublin, Ireland. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations*.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5054–5065, Hong Kong, China. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning SNS for automated Japanese error correction of second language learners. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 147–155, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA → Online. Association for Computational Linguistics.

Muhammad Qorib, Seung-Hoon Na, and Hwee Tou Ng. 2022. Frustratingly easy system combination for grammatical error correction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1964–1974, Seattle, United States. Association for Computational Linguistics.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. 2021. A simple recipe for multilingual grammatical error correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 702–707, Online. Association for Computational Linguistics.

Allen Schmaltz, Yoon Kim, Alexander Rush, and Stuart Shieber. 2017. Adapting sequence models for sentence correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2807–2813, Copenhagen, Denmark. Association for Computational Linguistics.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for ESL learners using global context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 198–202, Jeju Island, Korea. Association for Computational Linguistics.

Maksym Tarnavskyi, Artem Chernodub, and Kostiantyn Omelianchuk. 2022. Ensembling and knowledge distilling of large sequence taggers for grammatical error correction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3842–3852, Dublin, Ireland. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189, Portland, Oregon, USA. Association for Computational Linguistics.

Helen Yannakoudakis, Marek Rei, Øistein E. Andersen, and Zheng Yuan. 2017. Neural sequence-labelling models for grammatical error correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2795–2806, Copenhagen, Denmark. Association for Computational Linguistics.

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–386, San Diego, California. Association for Computational Linguistics.

Zheng Yuan, Felix Stahlberg, Marek Rei, Bill Byrne, and Helen Yannakoudakis. 2019. Neural and FST-based approaches to grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 228–239, Florence, Italy. Association for Computational Linguistics.

Zheng Yuan, Shiva Taslimipoor, Christopher Davis, and Christopher Bryant. 2021. Multi-class grammatical error detection for correction: A tale of two systems. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8722–8736, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

A Appendix

A.1 SymSpell

SymSpell is an open-source spelling correction system. It is initialised with a dictionary of correct words and their frequency in some sample of English text. Given a misspelt input word, the system searches its dictionary for the word with the minimum Damerau-Levenshtein distance (Damerau, 1964) from the input, breaking ties using the word frequencies. A parameter n limits the maximum number of edits allowed. If the dictionary contains no words within n Damerau-Levenshtein edits of the input, the system reports that the input could not be corrected.

We initialise SymSpell using $n = 2$ and use the dictionary of approximately 83k English words included with SymSpell.[16] The dictionary is derived from the Spell Checker Oriented Word Lists[17] database and contains both British and American spelling variants. Word frequencies are obtained from the Google Books n-gram dataset.[18]

A.2 LemmInflect

LemmInflect is a software module which performs lemmatisation and inflection on English words. For example, we may want to inflect the singular present tense verb “runs” to its past tense form “ran”. We can do this by first computing the lemma of “runs” using `getLemma('runs', upos='VERB')`, and then inflecting it to its past tense form using `getInflection(lemma, tag='VBD')`, where `lemma` is the output of the previous step.[19] LemmInflect’s functions first attempt to use dictionaries to map between word forms. If the input does not appear in its dictionary, LemmInflect uses a classification model to determine which of a pre-defined set of morphing rules to apply (e.g. adding “-ed” to the input).

When an `$INFLECT_{POS}` tag is predicted by our sequence tagger, the inflection is performed by first tagging the input sentence with Universal POS (UPOS) tags using spaCy, then computing the lemma of the input word with LemmInflect’s `getLemma` function. Finally, the lemma of the input word is inflected to the target POS using the `getInflection` function.

[16] https://github.com/wolfgarbe/SymSpell/blob/master/SymSpell/frequency_dictionary_en_82_765.txt

[17] http://wordlist.aspell.net

[18] https://storage.googleapis.com/books/ngrams/books/datasetsv2.html

[19] The “upos” and “tag” arguments are the Universal POS tag (Nivre et al., 2020) of the input word and the Penn Treebank POS tag (Marcus et al., 1993) of the desired output respectively.

A.3 Training Details

We use a batch size of 256 in stages 1 and 2, and 128 in stage 3. During training, the model is evaluated on the development set every 10k steps in stage one, and every epoch in stages two and three. Training is stopped when the development set accuracy does not improve for three consecutive evaluations or a maximum number of training steps or epochs have been completed. The accuracy is computed as the combined tag-level accuracy of the detection and tagging heads. We use a maximum of 200k steps for stage one, and a maximum of 15 epochs for stages two and three. In our experiments, stages two and three never reach this maximum.

We use the cross entropy loss function[20] and the Adam optimiser (Kingma and Ba, 2015) with the default parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$).[21] We follow the learning rate schedule of (Omelianchuk et al., 2020). Specifically, we perform the first 20k steps and first 2 epochs of training stages one and two respectively with a learning rate of 10^{-3} and the encoder weights frozen.[22] After these respective points in training are reached, the encoder weights are unfrozen and the learning rate is decreased to 10^{-5}. In stage three, the encoder weights are never frozen and we only use a learning rate of 10^{-5}.

Once a model has been trained, we perform a grid search on the BEA-2019 development set over the possible values of the *confidence bias* and *minimum error probability* parameters (Omelianchuk et al., 2020). We later refer to these as the "inference tweak" parameters. For both parameters, we test values ranging from 0.0 to 0.9 inclusive, in increments of 0.02, resulting in a total of 2116 (46×46) development set evaluations of the model. We have included, in our public repository, the BEA-2019 development set scores for all of the parameter combinations tested, as well as the chosen parameters for each of the models.

A.4 Dataset Sizes and Splits

We use the same datasets for each training stage as Omelianchuk et al. (2020). We refer readers to Table 1 of their paper for statistics on each dataset's size and error frequencies. For stages 1 and 2, we combine the relevant datasets as described in their repository.[23] We generate a random split of each dataset into training and development sets, which contain 98% and 2% of the data respectively.[24] For stage 3, we use the pre-defined training, development and test sets of the W&I + LOCNESS dataset (Bryant et al., 2019).

Encoder	Parameters
DeBERTa-large	405M
DeBERTaV3-large	435M
ELECTRA-large	335M
RoBERTa-large	355M
XLNet-large	360M

Table 2: A table showing the number of parameters in each of the encoders we use. Note that these numbers do not include the weights of the detection and tagging heads which vary based on the vocabulary size used.

A.5 Model Size and Compute Requirements

We use the standard "large" configuration of each of our encoders. The number of parameters in each encoder is shown in Table 2.

Training took 15-20 hours per model with four NVIDIA A100 GPUs connected via NVLink, each with 80 GB of VRAM, using the HuggingFace PyTorch DistributedDataParallel trainer implementation. Our grid search over the inference tweak hyperparameters took 8-13 hours on one A100.

We did not perform detailed inference time experiments. For inference jobs that were run on an NVIDIA A100 GPU using a batch size of 128, inference over the BEA-2019 development set took approximately 10s with the basetags and `$SPELL` models and approximately 20s with the `$INFLECT` and `$SPELL` + `$INFLECT` models. We note that our implementation was not optimised for inference speed. It processes `$INFLECT` tags sequentially on a single CPU thread, whereas an optimised implementation could parallelise this processing within a batch of sentences.

This paper reports results from 156 models[25]

[20] `https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html`

[21] We use the PyTorch implementation of the AdamW optimiser (Loshchilov and Hutter, 2019) with the *weight decay* parameter set to zero, making it equivalent to the Adam optimiser.

[22] During this initial phase, only the weights of the prediction heads are updated.

[23] `https://github.com/grammarly/gector/blob/master/docs/training_parameters.md`

[24] The 98/2 training/development split was used by Omelianchuk et al. (2020). This is documented in the main README file in their repository.

[25] Figures 2-4 show the results from 120 models (5 encoders $\times$ 2 vocabulary sizes $\times$ 4 tagsets $\times$ 3 seeds) and Tables 1 and 4 required a further 36 models to be trained (3 encoders $\times$ 4

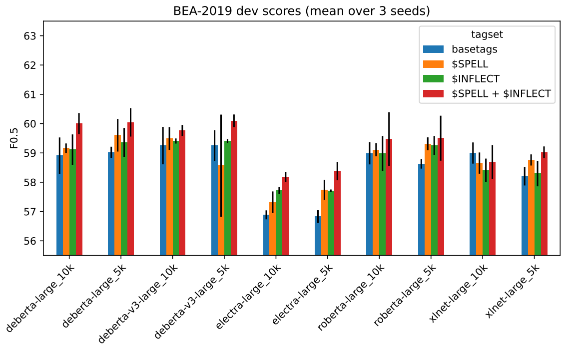

Figure 4: A bar graph showing the BEA-2019 development set $F_{0.5}$ scores of single models using different tagsets. Each bar represents the mean score over three training runs with different seeds. The black lines are error bars showing the standard deviations.

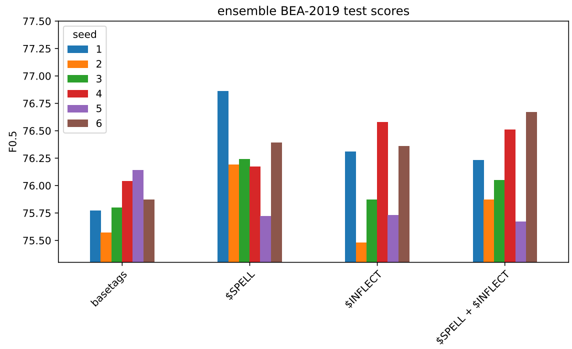

Figure 6: A bar graph showing the BEA-2019 test set $F_{0.5}$ scores of our ensemble models using different tagsets with six different random seeds. Each model is an ensemble of three encoders: DeBERTa$_{5K}^{(L)}$ $\bigoplus$ ELECTRA$_{5K}^{(L)}$ $\bigoplus$ RoBERTa$_{5K}^{(L)}$

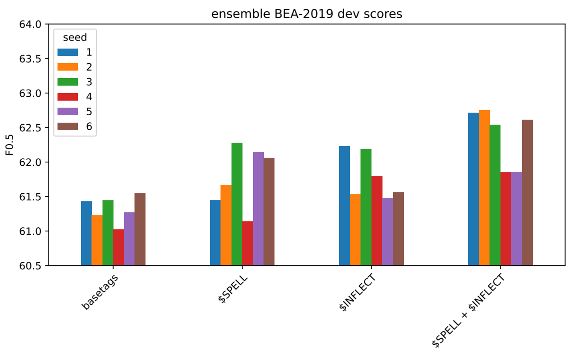

Figure 5: A bar graph showing the BEA-2019 development set $F_{0.5}$ scores of our ensemble models using different tagsets with six different random seeds. Each model is an ensemble of three encoders: DeBERTa$_{5K}^{(L)}$ $\bigoplus$ ELECTRA$_{5K}^{(L)}$ $\bigoplus$ RoBERTa$_{5K}^{(L)}$

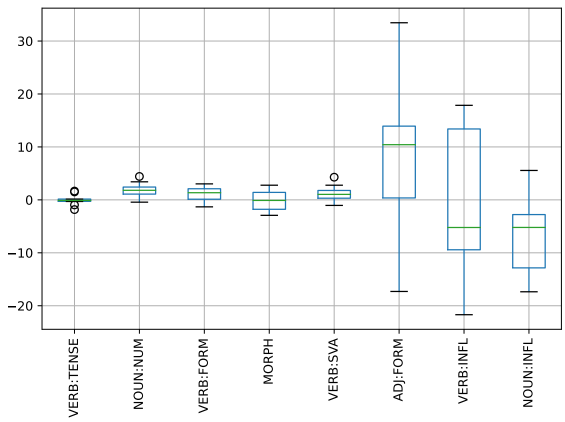

Figure 7: A box plot showing the change in BEA-2019 development set $F_{0.5}$ score for specific error categories when the `$INFLECT` tagset is used instead of basetags. Each result shows the distribution of deltas over 10 combinations of encoders and tagset sizes. For each such combination and tagset, we take the mean $F_{0.5}$ score over three seeds and subtract the `$INFLECT` mean from the basetags mean. The categories are ordered by frequency, decreasing from left to right.

which took approximately 12.5k GPU hours to train and tune. Before this, we used approximately 2k GPU hours for development and preliminary experiments with smaller models. Therefore in total, approximately 14.5k GPU hours were used in creating this paper.

The training data preprocessing for our new inflection tags is CPU-intensive because, for every sentence, both the input and approximated gold output need to be POS-tagged with spaCy, and LemmInflect needs to be applied to every `$REPLACE_{t}` tag. In our experiments, preprocessing the datasets for all three training stages took approximately 35 minutes on a dual-socket 76-core Intel(R) Xeon(R) Platinum 8368Q CPU @ 2.60GHz. This process was run for both the `$INFLECT` and `$SPELL + $INFLECT` tagsets.

tagsets × 3 seeds).

A.6 Additional Single Encoder and Ensemble Results

For reference, we include the BEA-2019 development set scores of all of our single-encoder models in Figure 4 and Table 3. These models were trained as part of our search process for the best combination of encoders for our ensemble.

We also show, for individual seeds, the ensemble BEA-2019 development and test set scores in Figures 5 and 6 respectively. This illustrates the variance in $F_{0.5}$ score over different random seeds.

encoder & tagset size	basetags	\$SPELL	\$INFLECT	\$SPELL + \$INFLECT
$\text{DeBERTa}_{10K}^{(L)}$	58.91 ± 0.62	59.16 ± 0.16	59.11 ± 0.52	60.00 ± 0.36
$\text{DeBERTa}_{5K}^{(L)}$	59.02 ± 0.19	**59.60 ± 0.56**	59.36 ± 0.49	60.04 ± 0.49
$\text{DeBERTaV3}_{10K}^{(L)}$	**59.25 ± 0.64**	59.49 ± 0.39	**59.41 ± 0.09**	59.77 ± 0.19
$\text{DeBERTaV3}_{5K}^{(L)}$	59.25 ± 0.53	58.56 ± 1.74	**59.41 ± 0.07**	**60.10 ± 0.22**
$\text{ELECTRA}_{10K}^{(L)}$	56.89 ± 0.15	57.32 ± 0.37	57.72 ± 0.12	58.17 ± 0.17
$\text{ELECTRA}_{5K}^{(L)}$	56.83 ± 0.22	57.74 ± 0.35	57.71 ± 0.05	58.38 ± 0.31
$\text{RoBERTa}_{10K}^{(L)}$	58.99 ± 0.38	59.11 ± 0.22	58.98 ± 0.59	59.47 ± 0.92
$\text{RoBERTa}_{5K}^{(L)}$	58.63 ± 0.16	59.31 ± 0.22	59.26 ± 0.32	59.50 ± 0.77
$\text{XLNet}_{10K}^{(L)}$	59.00 ± 0.36	58.65 ± 0.36	58.41 ± 0.40	58.69 ± 0.58
$\text{XLNet}_{5K}^{(L)}$	58.20 ± 0.31	58.76 ± 0.19	58.29 ± 0.43	59.02 ± 0.20

Table 3: A table showing BEA-2019 development set $F_{0.5}$ scores of single models using different tagsets and encoders. We show the mean and standard deviation of the scores over three training runs with different seeds.

	CoNLL-2014 test		
Model	precision	recall	$F_{0.5}$ $(\bar{x} \pm \sigma)$
$\text{DeBERTa}_{5K}^{(L)}$ basetags	76.70	42.73	66.16 ± 0.47
$\text{DeBERTa}_{5K}^{(L)}$ \$SPELL	**77.15**	**43.19**	**66.64 ± 0.40**
$\text{DeBERTa}_{5K}^{(L)}$ \$INFLECT	76.43	42.57	65.90 ± 0.49
$\text{DeBERTa}_{5K}^{(L)}$ \$SPELL + \$INFLECT	76.62	42.67	66.06 ± 0.44
ensemble basetags	80.70	41.25	67.72 ± 0.32
ensemble \$SPELL	**80.86**	**41.72**	**68.06 ± 0.43**
ensemble \$INFLECT	80.60	41.31	67.70 ± 0.54
ensemble \$SPELL + \$INFLECT	80.65	41.70	67.93 ± 0.40
$\text{DeBERTa}_{10K}^{(L)} \bigoplus \text{RoBERTa}_{10K}^{(L)} \bigoplus \text{XLNet}_{5K}^{(L)}$ (Tarnavskyi et al., 2022)	76.1	41.6	65.3
$\text{RoBERTa}_{5K}^{(L)}$ (KD) (Tarnavskyi et al., 2022)	74.40	41.05	64.0
T5 xxl (Rothe et al., 2021)	-	-	68.87
ESC (Qorib et al., 2022)	**81.48**	**43.78**	**69.51**

Table 4: A table showing CoNLL-2014 test set scores (using the M^2 scorer). The top section shows our models with varying tagsets using the $\text{DeBERTa}_{5K}^{(L)}$ encoder. The middle section shows the results for our ensemble models with varying tagsets. In the table, "ensemble" denotes the encoders $\text{DeBERTa}_{5K}^{(L)} \bigoplus \text{ELECTRA}_{5K}^{(L)} \bigoplus \text{RoBERTa}_{5K}^{(L)}$. Finally, the bottom section shows models from related work. The model labelled "(KD)" was trained using Tarnavskyi et al. (2022)'s knowledge distillation procedure. The results in the top and middle sections are averaged over 6 seeds, and the standard deviation, σ, of the test $F_{0.5}$ is shown.

A.7 Performance Analysis of Inflection-Related Error Categories

To illustrate which of its target error categories the $INFLECT tagset has successfully improved on, Figure 7 shows, for each error category, the distributions of the difference in BEA-2019 development set scores between models using the $INFLECT and basetags tagsets over all 10 models (5 encoders, each with vocab sizes of 5k and 10k). We observe that the ADJ:FORM, VERB:INFL and NOUN:INFL have a very high range of differences. This is expected because these three categories have frequencies of 11, 6 and 4 respectively in the development set. The small sample size makes it difficult to draw conclusions about these error categories. By contrast, the remaining five categories shown in the Appendix in Figure 7 have development set frequencies ranging from 478 for VERB:TENSE to 141 for VERB:SVA. Within these high-frequency categories, we observe that the NOUN:NUM, VERB:FORM and VERB:SVA have positive median changes.

A.8 CoNLL-2014 Results

For interested readers, we have included results on the CoNLL-2014 benchmark (Ng et al., 2014) in Table 4. The scores are computed with the M^2 scorer (Dahlmeier and Ng, 2012). In both the single and ensemble models, the $SPELL tagset performs best. However, these results should be interpreted with caution, since the model hyper-parameters were not tuned on the CoNLL-2014 development set.