Minimum Bayes' Risk Decoding for System Combination of Grammatical Error Correction Systems

Vyas Raina
ALTA Institute
Department of Engineering
University of Cambridge
vr313@cam.ac.uk

Mark Gales
ALTA Institute
Department of Engineering
University of Cambridge
mjfg@cam.ac.uk

Abstract

For sequence-to-sequence tasks it is challenging to combine individual system outputs. Further, there is also often a mismatch between the decoding criterion and the one used for assessment. Minimum Bayes' Risk (MBR) decoding can be used to combine system outputs in a manner that encourages better alignment with the final assessment criterion. This paper examines MBR decoding for Grammatical Error Correction (GEC) systems, where performance is usually evaluated in terms of edits and an associated F-score. Hence, we propose a novel MBR loss function directly linked to this form of criterion. Furthermore, an approach to expand the possible set of candidate sentences is described. This builds on a current max-voting combination scheme, as well as individual edit-level selection. Experiments on three popular GEC datasets and with state-of-the-art GEC systems demonstrate the efficacy of the proposed MBR approach. Additionally, the paper highlights how varying reward metrics within the MBR decoding framework can provide control over precision, recall, and the F-score in combined GEC systems. [1]

1 Introduction

Ensembling, the combination of system outputs, is a powerful technique in deep learning, exploiting diverse model capabilities for robust predictions. Though numerous methodologies exist for system combination (Ganaie et al., 2021), when there is only access to model outputs, many methods are inapplicable and thus the simplest method becomes the averaging of model outputs. However, for sequence-to-sequence (seq2seq) systems, such as summarization, machine translation, and grammatical error correction (GEC), output averaging is less straightforward. A further challenge with seq2seq tasks is the mismatch between the decoding and assessment criteria. Kumar and Byrne (2004) proposed the utilization of Minimum Bayes' Risk (MBR) decoding as a means to select an output that minimizes the theoretical risk according to a designated reward metric. We propose a novel variant of MBR decoding for GEC to allow for system combination and give better alignment with the assessment criteria.

The nature of a GEC task permits the use of MBR decoding within the "edit"-space. Each output sequence can be represented as a set of "edits" required to transform the input sequence into the output. Consequently, the selection of a single output sequence for GEC can be achieved through MBR decoding with a reward function defined on the set of edits, aligned with the edit-based F-score typically used in GEC assessment criteria. Beyond selection, an additional technique known as max-voting (Tarnavskyi et al., 2022) can be employed to combine different sets of edits. We propose an enhancement to the performance achieved through max-voting by treating the output sequences obtained from the combination as additional candidates for MBR decoding. Further, with a greedy MBR decoding algorithm, we explore the edit space to identify other candidate edit sets. Through experiments on three popular GEC datasets and use of state of the art GEC systems (Grammarly's GECToR (Omelianchuk et al., 2020)), we demonstrate that our MBR decoding approach in the edit space consistently leads to significant performance gains. Further, we also show that by selecting different reward metrics as part of the MBR decoding approach we can provide explicit control over precision, recall and the overall F-score used to assess GEC systems.

2 Related Work

Grammatical Error Correction: Early GEC systems using hand-crafted rules (Naber, 2003) were replaced by encoder-decoder architectures, using for example Recurrent Neural Networks (Cho

[1] Code available at: https://github.com/rainavyas/mbr_gec

Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 2: Short Papers), pages 105–112
November 1–4, 2023.

et al., 2014). Today, many state of the art GEC systems use Transformer-based (Vaswani et al., 2017) encoder-decoder architectures to perform the sequence-to-sequence GEC task (Kaneko et al., 2020; Chen et al., 2020; Kiyono et al., 2019; Lichtarge et al., 2020; Stahlberg and Kumar, 2020). However, LaserTagger (Malmi et al., 2019), the PIE model (Awasthi et al., 2019) and Grammarly's GECToR (Omelianchuk et al., 2020) are all able to achieve competitive performance using a sequence-to-edit structure for the overall sequence-to-sequence task, where a token can be tagged with edit operations. Once a set of tags have been defined, the edit operations can be applied to the input sequence to generate the grammatically correct output sequence. The GECToR system is particularly efficient at inference as it uses a Transformer encoder followed by softmax over linear layers for edit tag prediction, which is significantly faster than standard sequence-to-sequence GEC system decoders. Further, Wu et al. (2023) demonstrated that GECToR performs better than the most recent generative large language models, e.g. ChatGPT (Brown et al., 2020), which tend to over-correct, compromising on recall performance. Hence this work uses the GECToR model as its base GEC architecture.

System Combination for seqseq systems: Individual deep learning systems for classification tasks can be combined in many ways: stacking (Wolpert, 1992), negative correlation learning (Liu and Yao, 1999), max-voter schemes (Ju et al., 2018; Simonyan and Zisserman, 2014) or probability averaging (He et al., 2016; Raina et al., 2020; Szegedy et al., 2015). However, for generative language tasks such as GEC, where the output is a sequence of tokens, many traditional ensembling approaches are inapplicable. Sequence-level ensembling approaches, however, can address this by averaging conditional token level probabilities of multiple systems (Sennrich et al., 2015; Freitag et al., 2017; Malinin and Gales, 2021; Fathullah et al., 2021). However, this approach requires identical member architectures as well as access to the output probabilities of the predicted tokens. With the rising trend of limited black box access to large language models (e.g. ChatGPT (Liu et al., 2023)), system combination methods that only require the generated output sequences have practical benefit.

With access to only the output sequences from individual seq2seq systems, it is challenging to combine them into a single output. For automatic speech recognition, Sim et al. (2007) select a single output using a simple Minimum Bayes' Risk (MBR) decoding approach (Kumar and Byrne, 2004), where the aim is effectively to select the *most average*/representative output sequence. Similarly Manakul et al. (2023) use MBR to combine sequences for clinical document summarization. The MBR approach has also recently been applied to machine translation (Rosti et al., 2007a,b; Freitag et al., 2022; Müller and Sennrich, 2021; Zhang et al., 2022). For GEC systems, Tarnavskyi et al. (2022) propose a *max voting* scheme, where only edits predicted by the majority of individual systems are retained. We further improve GEC performance by applying MBR decoding to a sequence selection set augmented with sequences from max voting. We further enrich this selection space with a greedy search over edits.

3 Output Sequence Combination for GEC

A Grammatical Error Correction (GEC) system predicts a grammatically correct output sequence $\mathbf{y}$ from an input sequence, $\mathbf{x}$. With multiple different GEC system output sequence predictions, $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$, for the same input sequence, $\mathbf{x}$, it is challenging to combine them into a single, best sequence. It is useful to consider the *edit*-space, where a set of edits, $\mathbf{e}_n(\mathbf{x}, \mathbf{y}_n) = \{e_1, \ldots, e_{|\mathbf{e}_n|}\}$ can be used to represent each predicted output sequence, $\mathbf{y}_n$ [2]. A single edit in the edit set can be defined fully by an input token in $\mathbf{x}$ and an edit operation to apply (insertion, deletion or substitution). This section describes how Minimum Bayes' Risk decoding can be used in the edit-space to combine the different output sequences in $\mathcal{Y}$.

3.1 MBR decoding for GEC

MBR decoding aims to select the most representative output sequence, $\mathbf{y}^* \in \mathcal{Y}$. For GEC, we aim to maximise a reward score $\mathcal{R}$ in the edit-space that encourages better alignment with the final assessment metric,

$$\mathbf{y}^* = \arg\max_{\mathbf{y} \in \mathcal{Y}} \left\{ \mathbb{E}_{p(\tilde{\mathbf{y}}|x)} [\mathcal{R}(\tilde{\mathbf{e}}(\mathbf{x}, \tilde{\mathbf{y}}), \mathbf{e}(\mathbf{x}, \mathbf{y}))] \right\}, \tag{1}$$

where the reward score, $\mathcal{R}(\tilde{\mathbf{e}}, \mathbf{e})$, views $\tilde{\mathbf{e}}$ as reference edits and $\mathbf{e}$ as the hypothesis/predicted edits.

[2] Given an input sequence $\mathbf{x}$ and an output sequence $\mathbf{y}$ it is simple to create an edit set, using tools such as ERRANT (Bryant et al., 2017).

In practice, it is difficult to meaningfully estimate the posterior distribution, $p(\tilde{\mathbf{y}}|x)$ for each output sequence. Hence, we consider only similarly performing systems' output sequences, $\mathcal{Y}^{(c)} \in \mathcal{Y}$ to calculate the expectation of the reward and so we approximate each of these sequences as equiprobable,

$$\mathbf{y}^* \approx \arg\max_{\mathbf{y} \in \mathcal{Y}^{(s)}} \left\{ \frac{1}{|\mathcal{Y}^{(c)}|} \sum_{\tilde{\mathbf{y}} \in \mathcal{Y}^{(c)}} \mathcal{R}(\tilde{\mathbf{e}}(\mathbf{x}, \tilde{\mathbf{y}}), \mathbf{e}(\mathbf{x}, \mathbf{y})) \right\}, \tag{2}$$

where $\mathcal{Y}^{(s)}$ represents the set of possible output sequences we want to select from.

3.2 MBR decoding with edit voting

Inspired by Tarnavskyi et al. (2022) the different edit sets, $\{\mathbf{e_1}, \ldots, \mathbf{e}_N\}$ associated with the different output sequences, can be combined to create a single edit set, $\mathbf{e}^{(m)}$ containing all the individual edits present in at least m of the edit sets (i.e. m *votes*). This new combined edit set represents a new combined output sequence, $\mathbf{y}^{(m)}$. The MBR decoding approach of Equation 1 can now be applied by simply including the combined sequence in the set of sequences to select from, such that $\mathbf{y}^{(m)} \in \mathcal{Y}^{(s)}$. Note that the voting scheme can generate a maximum of N different combined sequences, with $\mathbf{e}^{(1)}$ being the union of all edit sets and $\mathbf{e}^{(N)}$ the intersection. Hence the selection space of sequences $\mathcal{Y}^{(s)}$ can be made richer with an extra N sequences.

3.3 Greedy MBR decoding for edit selection

Instead of augmenting the selection set $\mathcal{Y}^{(s)}$ with only a few sequences, it is useful to consider all possible edit sets. However, it is computationally infeasible to consider every possible edit set. Hence, this work proposes a practical, greedy method to increase the richness of the selection set. The minimal edit set is arguably the intersection of all edit sets, $\mathbf{e}^{(N)}$. In contrast the set of possible edits is given by the union set, $\mathbf{e}^{(1)}$. Hence, we can insert individual edits one by one from the union set to the intersection set. Every new edit insertion into the existing edit set represents a new output sequence $\mathbf{y}$ (that can be added to $\mathcal{Y}^{(s)}$). However, we only retain the edit insertions that give a new output sequence that increases the MBR expected reward, $\frac{1}{|\mathcal{Y}^{(c)}|} \sum_{\tilde{\mathbf{y}} \in \mathcal{Y}^{(c)}} \mathcal{R}(\tilde{\mathbf{e}}(\mathbf{x}, \tilde{\mathbf{y}}), \mathbf{e}(\mathbf{x}, \mathbf{y}))$ from Equation 2. This way we can efficiently search a richer selection set, $\mathcal{Y}^{(s)}$ of output sequences to find the best combined output sequence $\mathbf{y}^*$.

3.4 MBR reward score

Equation 1 uses a reward score $\mathcal{R}(\tilde{\mathbf{e}}, \mathbf{e})$ to perform MBR decoding. Careful selection of the reward score allows for control over the desired metric to optimise. We can for example aim to combine systems in a manner that encourages better edit *recall*,

$$\mathcal{R}^{(\text{rec})}(\tilde{\mathbf{e}}, \mathbf{e}) = \frac{|\tilde{\mathbf{e}} \cap \mathbf{e}|}{|\tilde{\mathbf{e}}|}. \tag{3}$$

Conversely, it may be desirable to have a system with high precision,

$$\mathcal{R}^{(\text{prec})}(\tilde{\mathbf{e}}, \mathbf{e}) = \frac{|\tilde{\mathbf{e}} \cap \mathbf{e}|}{|\mathbf{e}|}. \tag{4}$$

However, it is usually desirable to have a GEC system with a good combination of precision and recall, as measured by a F-k score,

$$\mathcal{R}^{(\text{f\{k\}})}(\tilde{\mathbf{e}}, \mathbf{e}) = \frac{(1+k^2)|\tilde{\mathbf{e}} \cap \mathbf{e}|}{|\tilde{\mathbf{e}}|k + |\mathbf{e}|}. \tag{5}$$

As the precision is more important than recall for GEC systems, this work aligns the reward metric with the F0.5 score. The Jaccard Similarity reward metric is also explored as an alternative in Appendix A.

4 Experiments

4.1 Experimental setup

We evaluate performance of the combined systems on three popular grammatical error correction corpora. **First Certificate in English (FCE)** corpus (Yannakoudakis et al., 2011) is a subset of Cambridge Learner Corpus (OpenCLC, 2019) made up of written examinations for general and business English of candidates from 86 different mother tongues, consisting of 2,720 test sentences. **Building Education Applications 2019 (BEA-19)** (Bryant et al., 2019) offers a test set of 4477 sentences, sourced from essays written by native and non-native English students. **Conference on Computational Natural Language Learning 2014 (CoNLL-14)** (Ng et al., 2014) test set consists of 1312 sentences sourced from 50 essays written by 25 non-native English speakers. Three different state of the art GECToR models are used as the individual systems to be combined [3]. Each system uses a different Transformer encoder (bert (b),

[3]GECToR model Weights: https://github.com/grammarly/gector#pretrained-models

roberta (r) or xlnet (x)). Table 1 gives the performance of these individual systems [4].

Model	conll	bea	fce
b	56.15 $\binom{61.75}{41.19}$	65.41 $\binom{67.33}{58.71}$	49.66 $\binom{54.47}{36.68}$
r	56.82 $\binom{61.99}{42.59}$	68.21 $\binom{70.21}{61.21}$	49.86 $\binom{53.47}{39.28}$
x	56.77 $\binom{61.74}{42.95}$	68.00 $\binom{69.89}{61.36}$	50.52 $\binom{53.49}{41.00}$

Table 1: F0.5 and (precision, recall) performance for individual GECToR systems

4.2 Results

MBR decoding (Equation 2) is applied in the edit-space for the three individual GECToR systems' outputs (b,r,x). Here, as the systems have similar performance (equiprobable posterior assumption valid), we let the selection set and the set of sequences to calculate the expected reward be the same $\mathcal{Y}^{(\mathrm{s})} = \mathcal{Y}^{(\mathrm{c})} = \{b, r, x\}$. Table 2 compares the different reward functions, $\mathcal{R}$, when applying MBR decoding. Selection with precision (Equation 6) and F0.5 (Equation 5) oriented reward metrics give a significant increase in performance over the individual systems in Table 1. Although the recall reward (Equation 3) does not increase F0.5 performance, it does significantly increase recall performance. This demonstrates that a simple application of MBR decoding can be used to combine individual systems to improve performance and selection of the reward function gives specific control over precision and recall of the combined system.

Reward	conll	bea	fce
$\mathcal{R}^{(\mathrm{rec})}$	55.13 $\binom{57.76}{46.66}$	64.67 $\binom{64.59}{64.99}$	48.74 $\binom{50.12}{43.90}$
$\mathcal{R}^{(\mathrm{prec})}$	**59.78** $\binom{69.38}{34.48}$	**70.87** $\binom{75.93}{55.96}$	**52.35** $\binom{60.13}{34.50}$
$\mathcal{R}^{(\mathrm{f05})}$	59.71 $\binom{66.42}{42.53}$	69.95 $\binom{72.95}{60.07}$	52.05 $\binom{56.61}{39.36}$

Table 2: MBR with $\mathcal{Y}^{(\mathrm{c})} = \mathcal{Y}^{(\mathrm{s})} = \{b, r, x\}$.

Section 3.2 describes how MBR decoding can be applied to systems combined by a voting scheme in the edit space. Table 3 shows the performance of systems combined with voting, where an individual edit requires m votes (from b,r or x edit system predictions) to be included in the combined edit set, $\mathbf{e}^{(m)}$ to form the single combined sequence $\mathbf{y}^{(m)}$. Note here that $\mathbf{e}^{(1)}$ is the union set and $\mathbf{e}^{(3)}$ is the intersection and so these sequences encourage either a higher recall or precision respectively. Table 4 shows the impact of MBR decoding where all the separate voting sets $(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \mathbf{y}^{(3)})$ are included in the selection set, $\mathcal{Y}^{(\mathrm{s})} = \{b, r, x, \mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \mathbf{y}^{(3)}\}$. Note that we maintain the same set of sequences for the expected reward calculation, $\mathcal{Y}^{(\mathrm{s})} = \{b, r, x\}$ to ensure the equiprobable posterior assumption holds [5]. It is evident that a richer selection set allows for even greater improvements in model performance for precision and F0.5 reward MBR decoding.

System	conll	bea	fce
$\mathbf{y}^{(1)}$	47.13 $\binom{48.02}{43.89}$	55.94 $\binom{55.03}{59.91}$	41.76 $\binom{42.38}{39.43}$
$\mathbf{y}^{(2)}$	60.58 $\binom{68.41}{41.54}$	71.82 $\binom{75.86}{59.22}$	52.73 $\binom{58.16}{38.38}$
$\mathbf{y}^{(3)}$	59.60 $\binom{77.30}{31.10}$	72.96 $\binom{84.32}{47.41}$	52.50 $\binom{67.05}{28.01}$

Table 3: Voting combination, $\mathbf{y}^{(m)}$ (m votes).

Reward	conll	bea	fce
$\mathcal{R}^{(\mathrm{rec})}$	53.99 $\binom{55.99}{47.23}$	63.81 $\binom{63.47}{65.25}$	48.18 $\binom{49.29}{44.20}$
$\mathcal{R}^{(\mathrm{prec})}$	60.24 $\binom{76.59}{32.50}$	**73.42** $\binom{83.40}{49.66}$	**53.51** $\binom{66.74}{29.85}$
$\mathcal{R}^{(\mathrm{f05})}$	**60.43** $\binom{67.94}{41.90}$	70.84 $\binom{74.48}{59.25}$	52.71 $\binom{57.83}{38.92}$

Table 4: MBR with $\mathcal{Y}^{(\mathrm{c})} = \{b, r, x\}$ and $\mathcal{Y}^{(\mathrm{s})} = \{b, r, x, \mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \mathbf{y}^{(3)}\}$.

Finally, as described in Section 3.3, MBR decoding can be performed over a richer edit selection space by greedily adding individual edits to the intersection edit set, $\mathbf{e}^{(3)}$ from the union edit set, $\mathbf{e}^{(1)}$. Experiments revealed (Appendix B) that allowing for all edits to be included from the union set can significantly increase the risk of poor insertions, compromising performance. Hence, instead we only consider edits from $\mathbf{e}^{(2)}$ to be added to the intersection set $\mathbf{e}^{(3)}$. Table 5 demonstrates that MBR decoding over this richer set of sequences can give better performance (CoNLL) than MBR with voting, but does not always give the best performance (BEA and FCE have better performance in Table 4). This is perhaps because the expected reward over the individual systems (b,r,x) is not necessarily perfectly aligned with the final F0.5 score relative to the true reference edits used in evaluation and thus over-optimisation of the selection set for MBR decoding does not help performance for some datasets.

[4] GEC performance for CoNLL and FCE is measured using the ERRANT tool (Bryant et al., 2017). Note that CoNLL is often evaluated with a different scorer in other papers. BEA is evaluated using the online submission portal: https://codalab.lisn.upsaclay.fr/competitions/4057

[5] Experiments with an alternative set of sequences for $\mathcal{Y}^{(\mathrm{c})}$ are in Appendix C

Reward	conll	bea	fce
$\mathcal{R}^{(\text{rec})}$	$61.06^{(69.50}_{41.11)}$	$72.20^{(76.87}_{58.08)}$	$52.94^{(58.87}_{37.73)}$
$\mathcal{R}^{(\text{prec})}$	$59.65^{(76.58}_{30.79)}$	$\mathbf{72.76}^{(84.18}_{47.16)}$	$52.62^{(67.61}_{27.89)}$
$\mathcal{R}^{(\text{f05})}$	$\mathbf{61.08}^{(69.71}_{40.85)}$	$72.34^{(77.16}_{57.19)}$	$\mathbf{53.00}^{(59.07}_{37.57)}$

Table 5: MBR with $\mathcal{Y}^{(\text{c})} = \{b, r, x\}$ and greedy search for $\mathcal{Y}^{(\text{s})}$.

5 Conclusions

The combination of sequence-to-sequence grammatical error correction (GEC) systems is challenging. There is also often a mismatch between the decoding criterion and assessment criterion used for GEC systems. This work demonstrates that a novel Minimum Bayes' Risk (MBR) decoding approach within the edit-space can give an effective system combination method that aligns better with the assessment criteria. We further showed that enhancing the selection space to encompass sequences formulated by max-voting over individual edits can further improve system performance. Moreover, the employment of a greedy search strategy, guided by an MBR reward function, can result in performance gains for the combined system. Crucially, the choice of a reward function in the MBR framework gives users the ability to optimize desired characteristics of the combined GEC system, such as precision, recall or the F-score.

6 Limitations

This work explored how MBR decoding can be used to combine individual GEC systems, as well as align the combined system's performance to the edit-based F-score used to assess GEC systems. Experiments were performed with Grammarly's GECToR based systems. It would be useful to extend these experiments to other state of the art GEC systems. Although these other systems are not as efficient as GECToR due to the use of an auto-regressive Transformer decoder (as opposed to GECToR's encoder only structure), it is still meaningful to understand how these systems react to MBR decoding used for system combination. This is particularly relevant as generative large language models are increasingly used for standard natural language tasks.

7 Ethics Statement

This work reports on an efficient method to combine individual GEC system outputs in a manner that better aligns with assessment and improve performance. There are no perceived ethical risks associated with this work.

8 Acknowledgements

This paper reports on research supported by Cambridge University Press & Assessment (CUP&A), a department of The Chancellor, Masters, and Scholars of the University of Cambridge.

References

Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. 2019. Parallel iterative edit models for local sequence transduction. *CoRR*, abs/1910.02893.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *CoRR*, abs/2005.14165.

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.

Mengyun Chen, Tao Ge, Xingxing Zhang, Furu Wei, and Ming Zhou. 2020. Improving the efficiency of grammatical error correction with erroneous span detection and correction. *CoRR*, abs/2010.03260.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Yassir Fathullah, Mark J.F. Gales, and Andrey Malinin. 2021. Ensemble distillation approaches for grammatical error correction. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2745–2749.

Markus Freitag, Yaser Al-Onaizan, and Baskaran Sankaran. 2017. Ensemble distillation for neural machine translation. *arXiv preprint arXiv:1702.01802*.

Markus Freitag, David Grangier, Qijun Tan, and Bowen Liang. 2022. High quality rather than high model probability: Minimum Bayes risk decoding with neural metrics. *Transactions of the Association for Computational Linguistics*, 10:811–825.

Mudasir A. Ganaie, Minghui Hu, Mohammad Tanveer, and Ponnuthurai N. Suganthan. 2021. Ensemble deep learning: A review. *CoRR*, abs/2104.02395.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Cheng Ju, Aurélien Bibaut, and Mark van der Laan. 2018. The relative performance of ensemble methods with deep convolutional neural networks for image classification. *Journal of Applied Statistics*, 45(15):2800–2818.

Masahiro Kaneko, Masato Mita, Shun Kiyono, Jun Suzuki, and Kentaro Inui. 2020. Encoder-decoder models can benefit from pre-trained masked language models in grammatical error correction. *CoRR*, abs/2005.00987.

Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. 2019. An empirical study of incorporating pseudo data into grammatical error correction. *CoRR*, abs/1909.00502.

Shankar Kumar and William Byrne. 2004. Minimum Bayes-risk decoding for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 169–176, Boston, Massachusetts, USA. Association for Computational Linguistics.

Jared Lichtarge, Chris Alberti, and Shankar Kumar. 2020. Data weighted training strategies for grammatical error correction. *CoRR*, abs/2008.02976.

Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, Zihao Wu, Dajiang Zhu, Xiang Li, Ning Qiang, Dingang Shen, Tianming Liu, and Bao Ge. 2023. Summary of chatgpt/gpt-4 research and perspective towards the future of large language models.

Yong Liu and Xin Yao. 1999. Ensemble learning via negative correlation. *Neural networks*, 12(10):1399–1404.

Andrey Malinin and Mark Gales. 2021. Uncertainty estimation in autoregressive structured prediction. In *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.

Potsawee Manakul, Yassir Fathullah, Adian Liusie, Vyas Raina, Vatsal Raina, and Mark Gales. 2023. Cued at probsum 2023: Hierarchical ensemble of summarization models.

Mathias Müller and Rico Sennrich. 2021. Understanding the properties of minimum Bayes risk decoding in neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 259–272, Online. Association for Computational Linguistics.

Daniel Naber. 2003. A rule-based style and grammar checker.

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.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem N. Chernodub, and Oleksandr Skurzhanskyi. 2020. Gector - grammatical error correction: Tag, not rewrite. *CoRR*, abs/2005.12592.

OpenCLC. 2019. Open cambridge learner english corpus.

Vyas Raina, Mark J.F. Gales, and Kate M. Knill. 2020. Universal adversarial attacks on spoken language assessment systems. In *Interspeech 2020*. ISCA.

Antti-Veikko Rosti, Necip Fazil Ayan, Bing Xiang, Spyros Matsoukas, Richard Schwartz, and Bonnie Dorr. 2007a. Combining outputs from multiple machine translation systems. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 228–235, Rochester, New York. Association for Computational Linguistics.

Antti-Veikko Rosti, Spyros Matsoukas, and Richard Schwartz. 2007b. Improved word-level system combination for machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 312–319, Prague, Czech Republic. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

K. C. Sim, W. J. Byrne, M. J. F. Gales, H. Sahbi, and P. C. Woodland. 2007. Consensus network decoding for statistical machine translation system combination. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, volume 4, pages IV–105–IV–108.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Felix Stahlberg and Shankar Kumar. 2020. Seq2Edits: Sequence transduction using span-level edit operations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5147–5159, Online. Association for Computational Linguistics.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

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.

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

David H Wolpert. 1992. Stacked generalization. *Neural networks*, 5(2):241–259.

Haoran Wu, Wenxuan Wang, Yuxuan Wan, Wenxiang Jiao, and Michael Lyu. 2023. Chatgpt or grammarly? evaluating chatgpt on grammatical error correction benchmark.

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.

Yidan Zhang, Yu Wan, Dayiheng Liu, Baosong Yang, and Zhenan He. 2022. Rmbr: A regularized minimum bayes risk reranking framework for machine translation.

A Jaccard Similarity as MBR Reward

Section 3.4 proposes three different reward functions, $\mathcal{R}$ to guide the MBR decoding process (Equation 1) to better align with desired assessment criteria. Here, we consider the Jaccard Similarity as an alternative reward function that can combine precision and recall properties,

$$\mathcal{R}^{(\text{jac})}(\tilde{\mathbf{e}}, \mathbf{e}) = \frac{|\tilde{\mathbf{e}} \cap \mathbf{e}|}{|\tilde{\mathbf{e}} \cup \mathbf{e}|}. \tag{6}$$

Table 6 gives the performance of systems combined with MBR decoding using the Jaccard reward function.

$\mathcal{Y}^{(s)}$	conll	bea	fce
$\{b, r, x\}$	$59.22^{(65.39}_{43.01)}$	$69.15^{(71.55}_{60.96)}$	$51.56^{(55.71}_{39.73)}$
$\{b, r, x, \mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \mathbf{y}^{(3)}\}$	$59.21^{(65.50}_{42.76)}$	$69.26^{(71.87}_{60.48)}$	$51.63^{(55.96}_{39.45)}$
Greedy	$60.94^{(69.84}_{40.37)}$	$72.37^{(77.29}_{57.68)}$	$52.92^{(59.09}_{37.33)}$

Table 6: GEC system performance with Jaccard Similarity reward function for MBR decoding. In all settings, $\mathcal{Y}^{(s)} = \mathcal{Y}^{(c)} = \{b, r, x\}$

Comparing to results in the main experiments (Section 4.2), it can be seen that using the Jaccard similarity reward gives similar behaviour but slightly worse performance than the F0.5 reward function used for MBR decoding. This is perhaps expected because both metrics encourage good precision and recall, but the final GEC systems are assessed using the F0.5 score. Hence, the Jaccard similarity reward offers a worse approximation to the final assessment metric than an explicit F0.5 reward in MBR decoding.

B Greedy MBR Decoding Selection Space

Section 3.3 describes an approach where MBR decoding can be used to greedily search over an edit space between the intersection edit set, $\mathbf{e}^{(3)}$ and the union edit set, $\mathbf{e}^{(1)}$ to find a combined edit set that as per the expected reward in the MBR algorithm should give better performance. Results in the main paper in Table 5 search the edit space between the intersection set, $\mathbf{e}^{(3)}$ and $\mathbf{e}^{(2)}$. Table 7 shows that it is sensible to not continue searching for all edits in the union set, $\mathbf{e}^{(1)}$, as searching the entire space compromises performance. This is perhaps due to the increased noise added into the system by potentially including spurious edits from the union set.

Reward	conll	bea	fce
$\mathcal{R}^{(\text{rec})}$	$53.22\binom{56.48}{43.24}$	$63.82\binom{65.34}{58.38}$	$47.73\binom{50.77}{38.52}$
$\mathcal{R}^{(\text{prec})}$	$59.03\binom{76.58}{30.79}$	$72.58\binom{84.27}{46.67}$	$52.56\binom{67.71}{27.73}$
$\mathcal{R}^{(\text{f05})}$	$58.42\binom{67.33}{38.19}$	$69.66\binom{75.05}{54.11}$	$50.42\binom{57.33}{34.02}$

Table 7: Greedy MBR decoding performance with edit search from intersection edit set, $\mathbf{e}^{(3)}$ to the union edit set, $\mathbf{e}^{(1)}$. By considering all possible edits in the union set, we can reduce performance. Hence in the main paper we limit edits to be between $\mathbf{e}^{(2)}$ and $\mathbf{e}^{(3)}$.

C Alternative Expected Reward Set, $\mathcal{Y}^{(\mathbf{c})}$

Equation 1 for MBR decoding can be simplified to equation 2, where we make the assumption that every sequence $\mathbf{y} \in \mathcal{Y}^{(\text{c})}$ used to calculate the expected reward is equiprobable (i.e. the posterior distribution is the same). We justify this assumption in the main paper by considering only similarly performing systems to form the set of sequences over which the expected reward is calculated: $\mathcal{Y}^{(\text{c})} = \{b, r, x\}$. It is interesting consider a situation where we violate/test this equiprobable posterior assumption by considering different possible sequence sets for $\mathcal{Y}^{(\text{c})}$ and observing the impact on performance after MBR decoding system combination. Table 8 reports the performance of MBR decoding with different output sequence sets, $\mathcal{Y}^{(\text{c})}$ used to calculate the expected reward. In comparison to the equivalent results in the main paper in Table 2, it is evident that a deviation from $\mathcal{Y}^{(\text{c})} = \{b, r, x\}$ does not compromises performance. This demonstrates that it is possible to diverge from the *similar performing system* constraint to validate the equiprobable posterior assumption to generate good combined systems using MBR decoding.

Reward	conll	bea	fce
$\mathcal{R}^{(\text{prec})}$	$59.83\binom{69.43}{38.52}$	$70.81\binom{75.81}{56.02}$	$52.40\binom{60.15}{34.57}$
$\mathcal{R}^{(\text{f05})}$	$59.96\binom{67.76}{41.07}$	$70.44\binom{74.21}{58.55}$	$52.09\binom{57.89}{37.19}$

Table 8: Impact of changing the set of sequences, $\mathcal{Y}^{(\text{c})}$ used to calculate the expected reward when using MBR decoding for system combination. We let $\mathcal{Y}^{(\text{c})} = \{b, r, x, \mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \mathbf{y}^{(3)}\}$. In all settings we maintain the same selection set, $\mathcal{Y}^{(\text{s})} = \{b, r, x\}$.