Neural Sequence-Labelling Models for Grammatical Error Correction

Helen Yannakoudakis, Marek Rei, Øistein E. Andersen and Zheng Yuan

The ALTA Institute
Computer Laboratory
University of Cambridge
{hy260,mr472,oa223,zy249}@cl.cam.ac.uk

Abstract

We propose an approach to $N$-best list re-ranking using neural sequence-labelling models. We train a compositional model for error detection that calculates the probability of each token in a sentence being correct or incorrect, utilising the full sentence as context. Using the error detection model, we then re-rank the $N$ best hypotheses generated by statistical machine translation systems. Our approach achieves state-of-the-art results on error correction for three different datasets, and it has the additional advantage of using only a small set of easily computed features that require no linguistic input.

1 Introduction

Grammatical Error Correction (GEC) in non-native text attempts to automatically detect and correct errors that are typical of those found in learner writing. High precision and good coverage of learner errors is important in the development of GEC systems. Phrase-based Statistical Machine Translation (SMT) approaches to GEC have attracted considerable attention in recent years as they have been shown to achieve state-of-the-art results (Felice et al., 2014; Junczys-Dowmunt and Grundkiewicz, 2016). Given an ungrammatical input sentence, the task is formulated as “translating” it to its grammatical counterpart. Using a parallel dataset of input sentences and their corrected counterparts, SMT systems are typically trained to correct all error types in text without requiring any further linguistic input. To further adapt SMT approaches to the task of GEC and tackle the paucity of error-annotated learner data, previous work has investigated a number of extensions, ranging from the addition of further features into the decoding process (Felice et al., 2014) via re-ranking the SMT decoder’s output (Yuan et al., 2016) to neural-network adaptation components to SMT (Chollampatt et al., 2016a).

In this paper, we propose an approach to $N$-best list re-ranking using neural sequence-labelling models. $N$-best list re-ranking allows for fast experimentation since the decoding process remains unchanged and only needs to be performed once. Crucially, it can be applied to any GEC system that can produce multiple alternative hypotheses. More specifically, we train a neural compositional model for error detection that calculates the probability of each token in a sentence being correct or incorrect, utilising the full sentence as context. Using the error detection model, we then re-rank the $N$ best hypotheses generated by the SMT system. Detection models can be more fine-tuned to finer nuances of grammaticality and acceptability, and therefore better able to distinguish between correct and incorrect versions of a sentence.

Our approach achieves state-of-the-art results on GEC for three different datasets, and it has the additional advantage of using only a small set of easily computed features that require no linguistic information, in contrast to previous work that has utilised a large set of features in a supervised setting (Hoang et al., 2016; Yuan et al., 2016).

2 Previous work

The first approaches to GEC primarily treat the task as a classification problem over vectors of contextual lexical and syntactic features extracted from a fixed window around the target token. A large body of work has investigated error-type-specific models, and in particular models targeting preposition and article errors, which are among the most frequent ones in non-native English learner writing (Chodorow et al., 2007; De Felice and Pul-
man, 2008; Han et al., 2010; Tetreault et al., 2010; Han et al., 2006; Tetreault and Chodorow, 2008; Gamon et al., 2008; Gamon, 2010; Rozovskaya and Roth, 2010; Rozovskaya et al., 2012; Dale and Kilgarriff, 2011; Leacock et al., 2014). Core components of one of the top systems in the CoNLL 2013 and 2014 shared tasks on GEC (Ng et al., 2013, 2014) include Averaged Perceptron classifiers, native-language error correction priors in Naïve Bayes models, and joint inference frameworks capturing interactions between errors (e.g., noun number and verb agreement errors) (Rozovskaya et al., 2012, 2014, 2011; Rozovskaya and Roth, 2011). The power of the classification paradigm comes from its ability to generalise well to unseen examples, without necessarily requiring error-annotated learner data (Rozovskaya and Roth, 2016).

One of the first approaches to GEC as an SMT task is the one by Brockett et al. (2006), who generate artificial data based on hand-crafted rules to train a model that can correct countability errors. Dahlmeier and Ng (2011) focus on correcting collocation errors based on paraphrases extracted from parallel corpora, while Dahlmeier and Ng (2012a) are the first to investigate a discriminatively trained beam-search decoder for full-sentence correction, focusing on five different error types: spelling, articles, prepositions, punctuation insertion, and noun number. Yoshimoto et al. (2013) utilise SMT to tackle determiner and preposition errors, while Yuan and Felice (2013) use POS-factored, phrase-based SMT systems, trained on both learner and artificially generated data to tackle determiner, preposition, noun number, verb form, and subject-verb agreement errors. The SMT approach has better capacity to correct complex errors, and it only requires parallel corrected sentences as input.

Two state-of-the-art systems in the 2014 CoNLL shared task on correction of all errors regardless of type use SMT systems: Felice et al. (2014) use a hybrid approach that includes a rule-based and an SMT system augmented by a large web-based language model and combined with correction-type estimation to filter out error types with zero precision. Junczys-Dowmunt and Grundkiewicz (2016) investigate parameter tuning based on the MaxMatch ($M^2$) scorer, the shared-task evaluation metric (Dahlmeier and Ng, 2012b; Ng et al., 2014), and experiment with different optimisers and interactions of dense and sparse features.

Susanto et al. (2014) and Rozovskaya and Roth (2016) explore combinations of SMT systems and classifiers, the latter showing substantial improvements over the CoNLL state of the art. Chollampatt et al. (2016a) integrate a neural network joint model that has been adapted using native-language-specific learner text as a feature in SMT, while Chollampatt et al. (2016b) integrate a neural network global lexicon model and a neural network joint model to exploit continuous space representations of words rather than discrete ones, and learn non-linear mappings. Yuan and Briscoe (2016) present a Neural Machine Translation (NMT) model and propose an approach that tackles the rare-word problem in NMT. Yuan et al. (2016) and Mizumoto and Matsumoto (2016) employ supervised discriminative methods to re-rank the SMT decoder’s $N$-best list output based on language model and syntactic features respectively. Hoang et al. (2016) also exploit syntactic features in a supervised framework, but further extend their approach to generate new hypotheses. Our approach is similar in spirit, but differs in the following aspects: inspired by the work of Rei and Yannakoudakis (2016) who tackle error detection rather than correction within a neural network framework, we develop a neural sequence-labelling model for error detection to calculate the probability of each token in a sentence as being correct or incorrect; using the error detection model, we propose a small set of features that require no linguistic processing to re-rank the $N$ best hypotheses. We evaluate our approach on three different GEC datasets and achieve state-of-the-art results, outperforming all previous approaches to GEC.

3 Datasets

We use the First Certificate in English (FCE) dataset (Yannakoudakis et al., 2011), and the NUS Corpus of Learner English (NUCLE) (Dahlmeier et al., 2013) that was used in the CoNLL GEC shared tasks. Both datasets are annotated with the language errors committed and suggested corrections from expert annotators. The former consists of upper-intermediate learner texts written by speakers from a number of different native language backgrounds, while the latter consists of essays written by advanced undergraduate university students.
students from an Asian language background. We use the public FCE train/test split, and the NUCLE train/test set used in CoNLL 2014 (the test set has been annotated by two different annotators).

We also use the publicly available Lang-8 corpus (Mizumoto et al., 2012; Tajiri et al., 2012) and the JHU Fluency-Extended GUG corpus (JFLEG) (Napoles et al., 2017). Lang-8 contains learner English from lang-8.com, a language-learning social networking service, which has been corrected by native speakers. JFLEG is a newly released corpus for GEC evaluation that contains fluency edits to make the text more native-like in addition to correcting grammatical errors, and contains learner data from a range of proficiency levels.

We use Lang-8 and the FCE and CoNLL training sets to train our neural sequence-labelling model, and test correction performance on JFLEG, and the FCE and CoNLL test sets. For JFLEG, we use the 754 sentences on which Napoles et al. (2017) have already benchmarked four leading GEC systems. As our development set, we use a subset of the FCE training data.

4 Neural sequence labelling

We treat error detection as a sequence labelling task and assign a label to each token in the input sentence, indicating whether it is correct or incorrect in context. These binary gold labels can be automatically extracted from the manual error annotation available in our data (see Section 3). Similarly to Rei and Yannakoudakis (2016), we construct a bidirectional recurrent neural network for detecting writing errors. The system receives a series of tokens [w1...wT] as input, and predicts a probability distribution over the possible labels for each token.

Every token w_t is first mapped to a token representation x_t, which is also optimised during training. These embeddings are composed together into context-specific representations using a bidirectional LSTM (Hochreiter and Schmidhuber, 1997):

\[
\overline{h}_t = \text{LSTM}(\overline{x}_t, \overline{h}_{t-1}) \quad (1)
\]

\[
\overline{h}_t = \text{LSTM}(\overline{x}_t, \overline{h}_{t+1}) \quad (2)
\]

\[
h_t = [\overline{h}_t; \overline{h}_t] \quad (3)
\]

where \(\overline{x}_t\) is the token representation at position \(t\), \(\overline{h}_t\) is the hidden state of the forward-moving LSTM, \(\overline{h}_t\) is the hidden state of the backward-moving LSTM, and \(h_t\) is the concatenation of both hidden states. A feedforward hidden layer with tanh activation is then used to map the representations from both directions into a more suitable combined space, and allow the model to learn higher-level features:

\[
d_t = \tanh W_d h_t \quad (4)
\]

where \(W_d\) is a weight matrix. Finally, a softmax output layer predicts the label distribution for each token, given the input sequence:

\[
P(y_t|w_1...w_T) = \text{softmax} W_o d_t \quad (5)
\]

where \(W_o\) is an output weight matrix.

We also make use of the character-level architecture proposed by Rei et al. (2016), allowing the model to learn morphological patterns and capture out-of-vocabulary words. Each individual character is mapped to a character embedding and a bidirectional LSTM is used to combine them together into a character-based token representation. This vector \(m\), constructed only from individual characters, is then combined with the regular token embedding \(x_t\) using an adaptive gating mechanism:

\[
z = \sigma \left( W_{z1} \cdot \tanh \left( W_{z2} x_t + W_{z3} m \right) \right) \quad (6)
\]

\[
\tilde{x}_t = z \cdot x_t + (1 - z) \cdot m \quad (7)
\]

where \(W_{z1}, W_{z2}\), and \(W_{z3}\), are weight matrices, \(z\) is a dynamically calculated gating vector, and \(\tilde{x}_t\) is the resulting token representation at position \(t\).

We optimise the model by minimising cross-entropy between the predicted label distributions and the annotated labels. In addition to training the error detection objective, we make use of a multi-task loss function and train specific parts of the architecture as language models. This provides the model with a more informative loss function, while also encouraging it to learn more general compositional features and acting as a regulariser (Rei, 2017). First, two extra hidden layers are constructed:

\[
\overline{m}_t = \tanh \overline{W}_m \overline{h}_t \quad (8)
\]
Figure 1: Error detection network architecture that is repeated for all the words in a sentence (illustration for the word “cat”).

\[
\hat{m}_t = \tanh \hat{W}_m \hat{h}_t \tag{9}
\]

where \(\hat{W}_m\) and \(\hat{W}_m\) are direction-specific weight matrices, used for connecting a forward or backward LSTM hidden state to a separate layer. The surrounding tokens are then predicted based on each hidden state using a softmax output layer:

\[
P(w_{t+1}|w_1...w_t) = \text{softmax} \hat{W}_q \hat{m}_t \tag{10}
\]

\[
P(w_{t-1}|w_t...w_T) = \text{softmax} \hat{W}_q \hat{m}_t \tag{11}
\]

During training, the following cost function is minimised, which combines the error detection loss function with the two language modeling objectives:

\[
E = - \sum_{t=1}^{T} \log P(y_t|w_1...w_T)
- \gamma \sum_{t=1}^{T-1} \log P(w_{t+1}|w_1...w_t) \tag{12}
- \gamma \sum_{t=2}^{T} \log P(w_{t-1}|w_1...w_T)
\]

where \(\gamma\) is a weight that controls the importance of language modeling in relation to the error detection objective. Figure 1 shows the error detection network architecture.

4.1 Experimental settings

All digits in the text are replaced with the character ‘0’. Tokens that occur less than 2 times in the training data share an out-of-vocabulary (OOV) token embedding, whereas the character-level component still operates over the original tokens. The model hyperparameters are tuned based on F\(_{0.5}\) on the FCE development set (Section 3) and \(\gamma\) is set to 0.1.\(^1\) The model is optimised using Adam (Kingma and Ba, 2015), and training is stopped when F\(_{0.5}\) does not improve on the development set over 5 epochs. Token representations have size 300 and are initialised with pre-trained word2vec embeddings trained on Google News (Mikolov et al., 2013). The character representations have size 50 and are initialised randomly. The LSTM hidden layers have size 200 for each direction.

4.2 Error detection performance

Rei and Yannakoudakis (2016)’s error detection framework uses token-level embeddings, bidirectional LSTMs for context representation, and a multi-layer architecture for learning more complex features. They train their model on the public FCE training set, and benchmark their results on the FCE and CoNLL test sets (Baseline LSTM\(_{FCE}\)). We also train and test our detection model on the same data and evaluate the effectiveness of our approach (LSTM\(_{FCE}\)). In Table 1, we can see that our architecture achieves a higher performance on both FCE and CoNLL, and particularly for FCE (7% higher F\(_{0.5}\)) and CoNLL test annotation 2 (around 2% higher F\(_{0.5}\)). When we use a larger training set that also includes the CoNLL training data and the public Lang-8 corpus (see Section 3), performance improves even further (LSTM), particularly for CoNLL test annotation 1 (at least 8% higher F\(_{0.5}\) compared to LSTM\(_{FCE}\)). We use this model in the experiments reported in the following sections.

5 Statistical machine translation

SMT attempts to identify the 1-best correction hypothesis \(c^*\) of an input sentence \(s\) that maximises the following:

\[
c^* = \arg \max_c \ p_{lm}(c)p(s|c) \tag{13}
\]

\(^1\)Lower \(\gamma\) values tend to give better error detection results as this essentially prioritises the error detection objective.
A Language Model (LM) is used to estimate the correction hypothesis probability \( p_{LM}(c) \) from a corpus of correct English, and a translation model to estimate the conditional \( p(s|c) \) from a parallel corpus of corrected learner sentences. State-of-the-art SMT systems are phrase-based (Koehn et al., 2003) in that they use phrases as “translation” units and therefore allow many-to-many “translation” mappings. The translation model is decomposed into a phrase-translation probability model and a phrase re-ordering probability model, and the 1-best correction hypothesis is of the following log-linear form (Och and Ney, 2002):

\[
c^* = \arg\max_c \exp \sum_{i=1}^{K} \lambda_i h_i(c, s)
\]  

(14)

where \( h \) represents a feature function (e.g., phrase-translation probability) and \( \lambda \) the feature weight.

In this work, we employ two SMT systems: Yuan et al. (2016)

\(^2\) and Junczys-Dowmunt and Grundkiewicz (2016). We apply our re-ranking approach to each SMT system’s N-best list using features derived from the neural sequence-labelling model for error detection described in the previous section, improve each of the SMT systems, and achieve state-of-the-art results on all three GEC datasets: FCE, CoNLL and JFLEG.

### 5.1 N-best list re-ranking

For each SMT system, we generate the list of all the 10 best candidate hypotheses. We then use the following set of features (tuned on the FCE development set, see Section 3) to assign a score to each candidate, and determine a new ranking for each SMT model:

**Sentence probability:** Our error detection model outputs a probability indicating whether a token is likely to be correct or incorrect in context. We therefore use as a feature the overall sentence probability, calculated based on the probability of each of its tokens being correct:

\[
\sum_{w} \log P(w)
\]

Levenshtein distance: We first use Levenshtein distance (LD) to identify which tokens in the original/source sentence have been corrected by the candidate hypothesis. We then identify the tokens that our detection model predicts as incorrect (i.e., the probability of being incorrect is greater than 0.5). These give us two different sets of annotations for the source sentence: tokens in the source sentence that the candidate hypothesis identifies as incorrect; and tokens in the source sentence that the error detection model identifies as incorrect. We then convert these annotations to binary sequences – i.e., 1 if the token is identified as incorrect, and 0 otherwise – and use as a feature the LD between those binary representations. More specifically, we would like to select the candidate sentence that has the smallest LD from the binary sequence created by the detection model: \( \frac{1}{LD} \)

**True and false positives:** Given the binary sequences described above, we also use as a feature the ratio of true positives (TP) to false positives (FP) by treating the error detection model as the “gold standard”. Specifically, we count how many times the candidate hypothesis agrees or not with the detection model on the tokens identified as incorrect: \( \frac{TP}{FP} \)

We use a linear combination of the above three scores together with the overall score (i.e., original rank) given by the SMT system (we do not include any other SMT features) to re-rank each SMT system’s 10-best list in an unsupervised way. The new 1-best correction hypothesis \( c^* \) is then the one that maximises:

\[
c^* = \arg\max_c \sum_{i=1}^{K} \lambda_i h_i(c)
\]  

(15)

---

### Table 1: Token-level error detection performance of our detection models (LSTM\textsubscript{FCE} and LSTM) on FCE and the two CoNLL 2014 test set annotations.

<table>
<thead>
<tr>
<th>System</th>
<th>FCE test set</th>
<th>CoNLL test set annotation 1</th>
<th>CoNLL test annotation 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
<td>F 0.5</td>
</tr>
<tr>
<td>Baseline LSTM\textsubscript{FCE}</td>
<td>46.10</td>
<td>28.50</td>
<td>41.10</td>
</tr>
<tr>
<td>LSTM\textsubscript{FCE}</td>
<td>58.88</td>
<td>28.92</td>
<td>48.48</td>
</tr>
<tr>
<td>LSTM</td>
<td>79.10</td>
<td>21.19</td>
<td>51.14</td>
</tr>
</tbody>
</table>

\(^2\) Yuan et al. (2016) propose a supervised N-best list re-ranking approach; however, we only use their baseline SMT system.
where $h$ represents the score assigned to candidate hypothesis $c$ according to feature $i$; $\lambda$ is a parameter that controls the effect feature $i$ has on the final ranking; and $K = 4$ as we have four different features (three features presented in this section, plus the original score output by the SMT system). $\lambda$s are tuned on the FCE development set and are set to 1, except for the sentence probability feature which has $\lambda = 1.5$.

6 Evaluation

We evaluate the effectiveness of our re-ranking approach on three different datasets: FCE, CoNLL 2014 and JFLEG. We report $F_{0.5}$ using the shared task’s $M^2$ scorer (Dahlmeier and Ng, 2012b), and GLEU scores (Napoles et al., 2015). The latter is based on a variant of BLEU (Papineni et al., 2002) that is designed to reward correct edits and penalise ungrammatical ones. As mentioned in Section 5, we re-rank the 10-best lists of two SMT systems: Yuan et al. (2016) (CAMB16 SMT) and Junczyz-Dowmunt and Grundkiewicz (2016) (AMU16 SMT). The results are presented in Table 2.

We replicate the AMU16 SMT system to obtain the 10-best output, and report results using this version (AMU16 SMT (replicated)). Compared to the original results on CoNLL reported in their paper (AMU16 SMT (reported)), we obtain slightly lower performance.\footnote{\textsuperscript{4}} We can see that AMU16 SMT is the current state of the art on CoNLL, with an $F_{0.5}$ of 49.49. On the other hand, CAMB16 SMT generalises better on FCE and JFLEG: 52.90 and 52.44 $F_{0.5}$ respectively. The lower performance of AMU16 SMT can be attributed to the fact that it is tuned for the CoNLL shared task.

The current state of the art on FCE is a neural machine translation system, CAMB16 SMT (Yuan and Briscoe, 2016), which is also the best model on JFLEG in terms of GLEU. The rest of the baselines we report are: Rozovskaya and Roth (2016), who explore combinations of SMT systems and classifiers (VT16 SMT + classifiers); Chollampatt et al. (2016a), who integrate a neural network joint model that has been adapted using native-language-specific learner text as a feature in SMT (NUS16 SMT + NNJM); and Hoang et al. (2016), who perform supervised $\lambda$-best list re-ranking using a large set of features, and further extend their approach to generate new hypotheses (NUS16 SMT + re-ranker).\footnote{\textsuperscript{5}}

\begin{table}[h]
\centering
\begin{tabular}{|l|c|c|c|c|c|c|c|}
\hline
          & FCE test set &          & CoNLL test set &          & JFLEG & \hline
          & P    & R    & $F_{0.5}$ & GLEU & P    & R    & $F_{0.5}$ & GLEU & P    & R    & $F_{0.5}$ & GLEU \\
\hline
Baseline &       &       &          &       &       &       &          &       &       &       &          &       \\
CAMB16 SMT & 63.27 & 31.95 & 52.90 & 70.15 & 45.39 & 21.82 & 37.33 & 64.90 & 65.56 & 29.12 & 52.44 & 46.10 \\
\hline
Our work &       &       &          &       &       &       &          &       &       &       &          &       \\
CAMB16 SMT + LSTM & 65.03 & 32.45 & 54.15 & 70.72 & 49.58 & 21.84 & 39.53 & 65.68 & 65.86 & 30.56 & 53.50 & 46.74 \\
CAMB16 SMT + LSTM\textsubscript{amb} & 64.25 & 36.13 & 55.60 & 71.76 & 51.09 & 25.30 & 42.44 & 66.42 & 65.41 & 32.97 & 54.66 & 47.72 \\
Oracle & 80.53 & 49.62 & 71.60 & 78.54 & 68.77 & 35.90 & 58.13 & 70.42 & 73.45 & 38.03 & 61.92 & 50.64 \\
\hline
Baseline &       &       &          &       &       &       &          &       &       &       &          &       \\
AMU16 SMT (reported) & 46.94 & 13.75 & 31.66 & 63.73 & 61.15 & 27.84 & 49.34 & 68.23 & 69.22 & 18.56 & 44.77 & 41.98 \\
AMU16 SMT (replicated) &       &       &          &       &       &       &          &       &       &       &          &       \\
\hline
Our work &       &       &          &       &       &       &          &       &       &       &          &       \\
AMU16 SMT (replicated) + LSTM & 40.67 & 17.36 & 32.06 & 63.57 & 58.79 & 30.63 & 49.66 & 68.26 & 60.68 & 22.65 & 45.43 & 42.65 \\
AMU16 SMT (replicated) + LSTM\textsubscript{amb} & 43.34 & 19.88 & 35.07 & 64.78 & 59.88 & 32.16 & 51.08 & 68.69 & 64.12 & 25.06 & 48.88 & 43.26 \\
Oracle & 71.54 & 26.69 & 53.54 & 69.52 & 76.47 & 35.97 & 62.41 & 71.18 & 79.10 & 27.47 & 57.49 & 45.00 \\
\hline
Other baselines &       &       &          &       &       &       &          &       &       &       &          &       \\
VT16 SMT + classifiers &       &       &          &       &       &       &          &       &       &       &          &       \\
NUS16 SMT-NNJM &       &       &          &       &       &       &          &       &       &       &          &       \\
NUS16 SMT + re-ranker &       &       &          &       &       &       &          &       &       &       &          &       \\
CAMB16 SMT &       &       &          &       &       &       &          &       &       &       &          &       \\
\hline
\end{tabular}
\caption{Using the neural sequence-labelling model for error detection (‘+ LSTM’ or ‘+ LSTM\textsubscript{amb}’) to re-rank the 10-best lists of two SMT systems – Yuan et al. (2016) (CAMB16 SMT) and Junczyz-Dowmunt and Grundkiewicz (2016) (AMU16 SMT).}
\end{table}
When using our LSTM detection model to re-rank the 10-best list (+ LSTM), we can see that performance improves across all three datasets for both SMT systems. \( F_{0.5} \) performance of CAMB16\textsubscript{SMT} on FCE improves from 52.90 to 54.15, on CoNLL from 37.33 to 39.53, and on JFLEG from 52.44 to 53.50 (the latter demonstrating that the detection model also helps with fluency edits). This improved result is also better than the state of the art CAMB16\textsubscript{SMT} on FCE.\(^6\) When looking at AMU16\textsubscript{SMT}, we can see that re-ranking (+ LSTM) further improves the best result on CoNLL from 49.34 (replicated) to 49.66 \( F_{0.5} \), and there is a similar level of improvement for both FCE and JFLEG.

As a further experiment, we re-train our error detection model on the same training data as CAMB16\textsubscript{SMT} (+ LSTM\textsubscript{camb}). More specifically, we use the Cambridge Learner Corpus (CLC) (Nicholls, 2003), a collection of learner texts of various proficiency levels, written in response to exam prompts and manually annotated with the errors committed (around 2M sentence pairs). In Table 2, we can see that the detection model further improves performance across all datasets and SMT systems. Compared to just doing SMT with CAMB16\textsubscript{SMT}, re-ranking improves \( F_{0.5} \) from 52.90 to 55.60 on FCE (performance increases further even though CAMB16\textsubscript{SMT}’s training set includes a large set of FCE data), from 37.33 to 42.44 on CoNLL, and from 52.44 to 54.66 on JFLEG. The largest improvement is on CoNLL (5%), which is likely because CoNLL is not included in the training set. AMU16\textsubscript{SMT}(replicated) is specifically tuned for CoNLL; nevertheless, the detection model also improves \( F_{0.5} \) on CoNLL from 49.34 to 51.08. Re-ranking using a small set of detection-based features produces state-of-the-art results on all three datasets (we note that CAMB16\textsubscript{SMT} generalises better across all).

We next run ablation tests to investigate the extent to which each feature contributes to performance. Results obtained on the FCE test set after excluding each of the features of the ‘CAMB16\textsubscript{SMT} + LSTM\textsubscript{camb}’ re-ranking system are presented in Table 3. Overall, all features have a positive effect on performance, though the sentence probability feature does have the biggest impact: its removal is responsible for a 1.47 and 1.11 decrease of \( F_{0.5} \) and GLEU respectively. A similar pattern is observed on the other datasets too.

<table>
<thead>
<tr>
<th>Ablated feature</th>
<th>( F_{0.5} )</th>
<th>GLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>55.60</td>
<td>71.76</td>
</tr>
<tr>
<td>Sentence probability</td>
<td>54.13</td>
<td>70.65</td>
</tr>
<tr>
<td>Levenshtein distance</td>
<td>55.42</td>
<td>71.78</td>
</tr>
<tr>
<td>True/false positives</td>
<td>55.14</td>
<td>71.75</td>
</tr>
</tbody>
</table>

Table 3: Ablation tests on the FCE test set when removing one feature of the re-ranking system at a time.

6We note that CAMB16\textsubscript{SMT} outperforms the re-ranking approach by Yuan et al. (2016).

6.1 Oracle

To calculate an upper bound per SMT system per dataset, we calculate character-level LD between each candidate hypothesis in the 10-best list and the gold corrected sentence. We then calculate an oracle score by selecting the candidate hypothesis that has the smallest LD. Essentially the oracle is telling us the maximum performance that can be obtained with the given 10-best list on each dataset. For datasets for which we have more than one annotation available, we select the oracle that gives the highest \( F_{0.5} \).

In Table 2, we can see that, overall, CAMB16\textsubscript{SMT} has a higher oracle performance compared to AMU16\textsubscript{SMT}. More specifically, the maximum attainable \( F_{0.5} \) on FCE is 71.60, on CoNLL 58.13, and on JFLEG 61.92. This shows empirically that the 10-best list has great potential and should be exploited further. AMU16\textsubscript{SMT} has a lower oracle performance overall, though again this can be attributed to the fact that it is specifically tuned for CoNLL.

6.2 \( N \)-best list size

Next, we examine performance as the \( N \)-best list varies in size, ranging from 1 to 10 (Table 4). We observe a positive effect: the larger the size, the better the model for all datasets. \( F_{0.5} \) does not seem to have reached a plateau with \( n < 10 \), which suggests that increasing the size of the list further can potentially lead to better results. We do, however, observe that large improvements are obtained when increasing the size from 1 to 3, suggesting that, most of the time, better alternatives are identified within the top 3 candidate hypotheses. This, however, is not the case for the oracle \( \text{Foracle}_{0.5} \), which consistently increases as \( n \) gets larger.
6.3 Error type performance

In Table 5, we can see example source sentences, together with their corrected counterparts (reference), 1-best candidates by CAMB16SMT and 1-best candidates by CAMB16SMT + LSTM\textsubscript{camb}. Re-ranking seems to fix errors such as subject–verb agreement (“the Computer help” to “the computer helps”) and verb form (“I recommend you to visit” to “I recommend visiting”). In this section, we perform an analysis of performance per type to get a better understanding of where the strength of the re-ranking detection model comes from.

Until recently, GEC performance per error type was only analysed in terms of recall, as system output is not annotated. Recently, however, Bryant et al. (2017) proposed an approach to automatically annotating GEC output with error type information, which utilises a linguistically-enhanced alignment to automatically extract the edits between pairs of source sentences and their corrected counterparts, and a dataset-independent rule-based classifier to classify the edits into error types. Human evaluation showed that the predicted error types were rated as “Good” or “Acceptable” 95% of the time. We use their publicly available code to analyse per-error-type performance.

Table 4 presents the performance for a subset of error types that are affected the most before and after re-ranking CAMB16SMT on the FCE test set. The error types are interpreted as follows: 

- Missing error;
- Replace error;
- Unnecessary error.

The largest improvement is observed in replacement errors referring to possessive nouns (R:NOUN:POSS) and verb agreement (R:VERB:SVA); and in unnecessary errors referring to adverbs (U:ADV), determiners (U:DET), pronouns (U:PRON), and verb tense (U:VERB:TENSE).

The LSTM architecture allows the network to learn advanced composition rules and remember dependencies over longer distances (e.g., R:VERB:SVA improves from 58.38 to 69.40). The network’s language modelling objectives allow it to learn better and more general compositional features (e.g., U:ADV improves from 13.51 to 22.73), while the character-level architecture facilitates modelling of morphological patterns (e.g., replacement errors referring to verb form (R:VERB:FORM) improve from 53.62 to 58.06). Between M, R, and U errors, the largest improvement is observed in U, for which there is at least 5% improvement in $F_{0.5}$.\footnote{U improves from 38.44 to 43.77; M from 43.43 to 45.40; R from 53.25 to 55.33.}

Overall, re-ranking improves $F_{0.5}$ across error types; however, there is a small subset that is

<table>
<thead>
<tr>
<th>N-best list</th>
<th>FCE test set</th>
<th>CoNLL test set</th>
<th>JFLEG</th>
</tr>
</thead>
<tbody>
<tr>
<td>F\textsubscript{0.5}</td>
<td>F\textsubscript{0.5}</td>
<td>F\textsubscript{0.5}</td>
<td>F\textsubscript{0.5}</td>
</tr>
<tr>
<td>1</td>
<td>52.90</td>
<td>52.90</td>
<td>37.33</td>
</tr>
<tr>
<td>2</td>
<td>54.28</td>
<td>62.10</td>
<td>39.87</td>
</tr>
<tr>
<td>3</td>
<td>54.96</td>
<td>65.93</td>
<td>41.00</td>
</tr>
<tr>
<td>4</td>
<td>55.18</td>
<td>68.05</td>
<td>41.83</td>
</tr>
<tr>
<td>5</td>
<td>55.33</td>
<td>69.47</td>
<td>42.12</td>
</tr>
<tr>
<td>6</td>
<td>55.43</td>
<td>70.24</td>
<td>42.49</td>
</tr>
<tr>
<td>7</td>
<td>55.48</td>
<td>70.74</td>
<td>42.60</td>
</tr>
<tr>
<td>8</td>
<td>55.47</td>
<td>71.00</td>
<td>42.63</td>
</tr>
<tr>
<td>9</td>
<td>55.51</td>
<td>71.27</td>
<td>42.65</td>
</tr>
<tr>
<td>10</td>
<td>55.60</td>
<td>71.60</td>
<td>42.44</td>
</tr>
</tbody>
</table>

Table 4: Re-ranking performance using LSTM\textsubscript{camb} as the N-best list varies in size from 1 to 10 for CAMB16SMT and its oracle.

Table 5: Source sentences along with gold corrections (reference), 1-best candidates by CAMB16SMT and by CAMB16SMT + LSTM\textsubscript{camb}.
negatively affected (Table 6, bottom part); for example, performance on missing errors referring to verb form (M:VERB:FORM) drops from 50.00 to 38.46, and on replace contraction errors (R:CONTR) from 50.00 to 27.78. Importantly, such an analysis allows us to examine the strengths and weaknesses of the models, which is key for the deployment of GEC systems.

7 Conclusion

To the best of our knowledge, no prior work has investigated the impact of detection models on correction performance. We proposed an approach to N-best list re-ranking using a neural sequence-labelling model that calculates the probability of each token in a sentence being correct or incorrect in context. Detection models can be more fine-tuned to finer nuances of grammaticality, and therefore better able to distinguish between correct and incorrect versions of a sentence. Using a linear combination of a small set of features derived from the detection model output, we re-ranked the N-best list of SMT systems and achieved state-of-the-art results on GEC on three different datasets. Our approach can be applied to any GEC system that produces multiple alternative hypotheses. Our results demonstrate the benefits of integrating detection approaches with correction systems, and how one can complement the other.

Acknowledgments

Special thanks to Christopher Bryant, Mariano Felice, and Ted Briscoe, as well as the anonymous reviewers for their valuable contributions at various stages.

References


Table 6: Error-type performance before and after re-ranking on the FCE test set (largest impact highlighted in bold; bottom part of the table displays negative effects on performance).


Na-Rae Han, Joel Tetreault, Soo-Hwa Lee, and Jin-Young Ha. 2010. Using error-annotated ESL data to develop an ESL error correction system. In Proceedings of LREC. Emi Izumi, Kiyotaka Uchimoto and Hitoshi Isahara.


Ippei Yoshimoto, Tomoya Kose, Kensuke Mitsuzawa, Keisuke Sakaguchi, Tomoya Mizumoto, Yuta Hayashibe, Mamoru Komachi, and Yuji Matsumoto. 2013. NAIST at 2013 CoNLL grammatical error correction shared task. In CoNLL Shared Task, pages 26–33.
