Automatic Extraction of Learner Errors in ESL Sentences Using Linguistically Enhanced Alignments

Mariano Felice **Christopher Bryant** **Ted Briscoe**
ALTA Institute
Computer Laboratory
University of Cambridge Cambridge, UK
{mf501, cjb255, ejb1}@cl.cam.ac.uk

Abstract

We propose a new method of automatically extracting learner errors from parallel English as a Second Language (ESL) sentences in an effort to regularise annotation formats and reduce inconsistencies. Specifically, given an original and corrected sentence, our method first uses a linguistically enhanced alignment algorithm to determine the most likely mappings between tokens, and secondly employs a rule-based function to decide which alignments should be merged. Our method beats all previous approaches on the tested datasets, achieving state-of-the-art results for automatic error extraction.

1 Introduction

Within the field of Machine Translation (MT), one of the first steps of data processing is to align a source sentence with a target sentence. This is necessary because we want to determine which tokens and phrases in the source language map to which equivalent tokens or phrases in the target language. As this would be extremely time consuming to do manually, several tools, such as GIZA++ (Och and Ney, 2003), have been made available to do this automatically.

Within the related field of Grammatical Error Correction (GEC), we similarly want to align a source sentence with a target sentence to map errors to corrections (sometimes referred to as 'correction detection'; see example in Table 1). However, unlike in MT, the source and target sentences in GEC are in the *same* language and so a majority of tokens match. This means alignment is comparatively more straightforward and so it is more feasible to annotate texts manually, rather than automatically. In fact two of the largest publicly available GEC datasets, the First Certificate in English (FCE) corpus (Yannakoudakis et al., 2011) and the National University of Singapore Corpus of Learner English (NUCLE) (Dahlmeier and Ng, 2012), were aligned and annotated manually.

We	took	a	**guide**	tour	**on**		**center**	**city**	.
We	took	a	**guided**	tour	**of**	**the**	**city**	**center**	.

Table 1: A sample alignment between an original uncorrected sentence and its corrected version.

Nevertheless, automatic alignment of GEC data still has several advantages over manual alignment, not least because the latter is slow, laborious work. This is especially important for datasets that do not always contain explicit alignments, such as Lang-8 (Mizumoto et al., 2011), or GEC system output that needs to be aligned to the original uncorrected sentence.

Another important benefit of an automatic alignment is that it tends to be more consistent than a human alignment. For example, within both the FCE and NUCLE, strings such as *has eating* are inconsistently corrected as [*has* → *was*] or [*has eating* → *was eating*] even though they fundamentally equate to the same thing. In fact, the latter seems less desirable given the token *eating* does not actually change. A similar case is [*has eating* → *was eaten*], which is inconsistently realised either as one edit, as above,

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers,
pages 825–835, Osaka, Japan, December 11-17 2016.

or two edits: [*has* → *was*] and [*eating* → *eaten*]. Ultimately, it seems desirable to regularise such edits and hence reduce ambiguity in the data. If all datasets are treated in the same way, this would also make them fully compatible with each other.

Finally, automatic alignment can also simplify the annotation of new data. For instance, Sakaguchi et al. (2016) recently claimed that forcing annotators to annotate grammatical errors within the confines of an error scheme often led to unnatural sounding sentences and that unconstrained editing correlated more with human judgements. As such, if we no longer ask humans to explicitly mark edit boundaries in new data, we would need to extract this information automatically. This is particularly useful for English as a Second Language (ESL) teaching, where teachers could edit text freely and then let a computer delimit the edits.

2 Background

There is very little previous work on automatic alignment of sentences for GEC. The first attempt was made by Swanson and Yamangil (2012), who built a system to align sentences and then classify the non-match tokens type for the purposes of ESL feedback. In particular, they used the well-known Levenshtein distance to align the sentences and then classified any non-matches according to the FCE error scheme (Nicholls, 2003) using a maximum entropy classifier.

One complication noted by Swanson and Yamangil is that edits do not necessarily consist of just a single token. For instance, reordering errors (e.g. [*only can* → *can only*]) or errors involving phrasal verbs (e.g. [*look at* → *watch*]) necessarily consist of more than one token on at least one side of the edit. The Levenshtein distance, however, only aligns individual tokens and so some alignments must be merged in order to obtain multi-token edits. Swanson and Yamangil hence experimented with some basic merging strategies and found that simply merging all adjacent non-match alignments most closely approximated human alignments.

Building on this foundation, Xue and Hwa (2014) carried out an analysis of Swanson and Yamangil's work and found that approximately 70% of all errors in their error type classifier were the result of bad alignments (merged or otherwise). In order to improve on the simple *all-merge* alignment strategy, they hence trained a binary maximum entropy classifier to determine whether edits should be merged or not. They tested this merging classifier on several datasets, including NUCLE and the FCE, and reported improvements of between 5-10% for both alignment and classification compared to Swanson and Yamangil.

Despite these improvements, however, there is still a considerable margin between automatic and human edit annotation. In addition, both approaches require training on existing annotations, which vary across datasets and can often be inconsistent. Ultimately, training on different datasets leads to different results and so undermines any effort towards data standardisation.

3 Automatic Alignment

A high-quality alignment between an original and corrected sentence is crucial for deriving meaningful edits. Unfortunately, however, the most common method of aligning sentences is to use the Levenshtein distance, which only optimises in terms of insertions, deletions and substitutions. This means that, while optimal in terms of edit operation, the alignments do not take linguistic information into account and are hence not optimal in terms of human intuition (see Table 2 (a)). Human alignments, on the other hand, do make use of linguistic information, so we propose automatic alignments should do the same.

3.1 Damerau-Levenshtein

First, however, as noted by Xue and Hwa (2014), another limitation of Levenshtein is that it is unable to handle word order errors. For example, [*only can* → *can only*] is realised as [*only* → Ø], [*can* → *can*] and [Ø → *only*]; in other words, reorderings are treated as deletions followed by insertions of identical tokens. Since we also need to preserve word order errors in the data, we argue that the Damerau-Levenshtein distance is better suited for the task than standard Levenshtein because it allows for token transpositions.

```
function DL_distance_extended(a, b):
  declare d[0..length(a), 0..length(b)]
  for i := 0 to length(a) inclusive do
    d[i, 0] := i
  for j := 0 to length(b) inclusive do
    d[0, j] := j

  for i := 1 to length(a) inclusive do
    for j := 1 to length(b) inclusive do
      if a[i] = b[j] then
        d[i, j] := 0
      else
        d[i, j] := min(d[i-1, j  ] + del_cost(a[i]),
                       d[i  , j-1] + ins_cost(b[j]),
                       d[i-1, j-1] + sub_cost(a[i], b[j]))

      // Damerau-Levenshtein extension for multi-token transpositions
      k = 1
      while i > 1 and j > 1 and (i - k) >= 1 and (j - k) >= 1 and
            d[i-k, j-k] - d[i-k-1, j-k-1] > 0 do
        if sorted(lowercase(a[i-k:i+1])) = sorted(lowercase(b[j-k:j+1])) then
          d[i, j] := min(d[i, j], d[i-k, j-k] + trans_cost(a[i-k:i+1], b[j-k:j+1])
          break
        k += 1

  return d[length(a), length(b)]
```

Listing 1: Damerau-Levenshtein distance allowing for transpositions of arbitrary length.

As the majority of word order errors in NUCLE and FCE data tend to only involve two tokens, this implies that the standard Damerau-Levenshtein distance, which is likewise only able to handle two-token transpositions, is generally sufficient for our purposes. Nevertheless, while it might seem acceptable to ignore the longer word order errors, this ultimately means they will be broken up into smaller and less meaningful edits which will increase the overall number of false positives and false negatives in the alignment.

To overcome this problem, we extend the Damerau-Levenshtein distance to allow for transpositions of arbitrary length, as shown in Listing 1. This is achieved by traversing a diagonal back from the current cell in the cost matrix and looking for a source sequence that would match the target sequence in any order. The cost of a transposition of length n is defined as $n - 1$, which is compatible with the original definition.

3.2 Linguistically Motivated Alignment

In an effort to incorporate linguistic information into the alignment, we replaced the substitution cost in Damerau-Levenshtein with the function shown in Listing 2. In this function, we set the cost to 0 if the original and corrected tokens differ only in case (e.g. [*the* → *The*]), otherwise, the substitution cost is the sum of sub-costs for lemma, part of speech and character differences. Each of these sub-costs is defined as follows:

lemma cost: 0 if tokens share the same lemma or derivationally related form (e.g. 'met' and 'meeting'), otherwise 0.499.

part-of-speech cost: 0 if tokens share the same part of speech, otherwise 0.25 if both tokens are content words (adjectives, adverbs, nouns or verbs) and 0.5 in all other cases.

character cost: the proportion of character mismatches between 0 and 1, computed as the character-level Damerau-Levenshtein distance between the tokens divided by the length of their alignment.

To increase the likelihood of aligning derivationally related forms, we lemmatise each token as if it were an adjective, adverb, noun and verb. We do this because if we only lemmatise for a single part of speech, then we might overlook certain derivationally related words. For example, while the lemma of

```
function substitution(a, b):
  if lowercase(a) = lowercase(b) then
    return 0
  else
    return lemma_cost(a, b) + pos_cost(a, b) + char_cost(a, b)
```

Listing 2: Our linguistically motivated token substitution function.

(a)	This	wide	spread	propaganda	benefits	only	to	the	companys	.
	This	widespread	publicity	only	benefits	their	companies			.
(b)	This	wide	spread	propaganda	benefits	only	to	the	companys	.
	This	widespread		publicity	only	benefits		their	companies	.

Table 2: Differences between (a) standard Levenshtein and (b) linguistically-enriched Damerau-Levenshtein alignment.

the verb ‘met’ is ‘meet’, the lemma of the noun ‘meeting’ is ‘meeting’, which suggests these words are not related. By also lemmatising ‘meeting’ as a verb however, we find that the two tokens do share a common lemma, ‘meet’, which instead correctly suggests they are related and should align. Ultimately, we consider two tokens to be derivationally related if their respective sets of lemmas intersect.

The sub-costs are also set in such a way that the overall substitution function always yields values in the $[0, 2)$ range. Keeping the cost asymptotic to 2 is important to enforce a preference for substitutions over insertions and deletions (both set to 1); this is why we use a lemma cost of 0.499 instead of 0.5. We tried different combinations of these costs, provided they met this condition, but did not find significant differences in the results.

By incorporating all this additional linguistic information into the cost, we improve the likelihood that tokens with a similar etymology, spelling or function will align. This is better than the simple surface matching used by the standard token-level Levenshtein distance and hence, we argue, results in more natural, human-like alignments (see Table 2 (b)). The final alignment is retrieved by collecting the operations that make up the optimal path in the cost matrix. Given that the cost is now dependent upon a variable function, it is often the case that there is just a single optimal alignment.

3.3 Data

We evaluated our improved alignment algorithm using the public FCE (Yannakoudakis et al., 2011), NUCLE corpus (Dahlmeier et al., 2013), and CoNLL test sets (Ng et al., 2013; Ng et al., 2014). While the CoNLL data is available in a pretokenised format, the FCE data is not, and so to keep things comparable, we only worked with the untokenized CoNLL data.

It should be noted that processing each of these datasets in a standard way is not at all straightforward. For example, unlike the CoNLL data, the FCE contains nested edits; e.g. [*entery* → *entry* → *entrance*] indicates a spelling error followed by a replacement noun error. Similarly, the NUCLE corpus can be quite noisy and it is not uncommon for annotators to select entire paragraphs or even essays as edits with comments such as “Rewrite in the 3rd person”, which, in the context of edit extraction, should definitely be ignored.

In addition to these, a more general problem concerns converting character level edit spans into token level edit spans; there is no guarantee that a human-annotated character span will map exactly to a token span, which has consequences for token-based processing and evaluation. Similarly, some edits change sentence boundaries, which subsequently makes aligning original sentences with corrected sentences a lot more complicated, especially when there are multiple annotators. Ultimately, we refer the reader to Bryant and Felice (2016) for more information about the challenges involved in processing these datasets and for details about how we overcame them in our implementation.

Having preprocessed the data, we used spaCy[1] v0.101.0 to tokenize (words and sentences), part of

[1] https://spacy.io/

Dataset	**Sents**	**Edits**
CoNLL 2013	1,375	3,415
CoNLL 2014 (0)	1,314	2,397
CoNLL 2014 (1)	1,319	3,331
NUCLE	55,963	43,832
FCE-test	2,715	4,776
FCE-train	31,022	48,026

Table 3: Basic statistics of the datasets we use. CoNLL 2014 was annotated by two annotators who changed different sentence boundaries.

Alignment	*Lev*	*Lev*	*DL*
Reference	*Gold*	*Gold-Min*	*Gold-Min*
CoNLL 2013	49.17	62.29	70.51
CoNLL 2014 (0)	51.09	60.40	66.81
CoNLL 2014 (1)	48.41	62.98	69.16
FCE-test	58.52	63.30	72.45

Table 4: Table showing how minimised edits in the reference (Gold-Min) and our linguistically enriched Damerau-Levenshtein algorithm (DL) perform against standard Levenshtein (Lev) and unmodified references (Gold). All scores are F_1.

speech (POS) tag and lemmatise each sentence. The basic statistics of each processed dataset are shown in Table 3.

3.4 Alignment Experiments

To establish a baseline, we simply compared a standard Levenshtein alignment against the human alignments of the CoNLL 2013, CoNLL 2014 (each annotator individually) and FCE test sets (Table 4). The results show that Levenshtein alone does not perform particularly well at this task and is only able to achieve F_1 scores of about 50%.

In addition to improving alignment quality, another aim of our work is to attempt to standardise edit annotation. As mentioned previously, human annotations sometimes include tokens that do not change; e.g. [*has eating* → *was eating*]. Automatic alignments will never match these human edits, however, because any token that is common to both sides will be considered a match and hence not part of an edit. This is undesirable, so we also minimised the gold human reference edits by recursively removing tokens that were common to both sides of the edit from the right and left hand sides. This also removes edits that annotators detected, but were unable to correct. Results showing the effect of this minimisation, as well as of comparing Levenshtein against our linguistically enhanced Damerau-Levenshtein approach, are also shown in Table 4.

The first thing to notice about these results is that the scores for the minimised gold reference are typically substantially higher than for the unmodified gold reference. In the case of CoNLL 2014 (1), using the minimised gold reference even shows an improvement of almost 15% F_1. While the increase in score is less pronounced for FCE-test, at just under 5% F_1, this nevertheless demonstrates the high degree of variability in the way GEC data is annotated and that it is highly desirable to standardise annotations. In light of this result, we only use minimised edit spans in all subsequent experiments.

Comparing Levenshtein against our own approach, we again see a significant improvement, with scores greater than 70% F_1 on some datasets. This is significant, because these results are in spite of the fact that we fail to match all multi-token edits given that we do not yet merge any alignments.

4 Alignment Merging

The output of the automatic alignment is a list of individual token-level operations that map the original sentence to the corrected sentence in terms of insertions, deletions, substitutions and transpositions. An example of this is shown in Table 5. Each of these operations involves one token at most in either sentence, except in the case of transpositions, which involve 2 or more tokens in both sentences.

In most cases, these individual operations already represent a complete error: [*propaganda* → *publicity*] is a word choice error, [*benefits only* → *only benefits*] is a word order error and [*companys* → *companies*] is a noun inflection error. Other errors, however, may involve more than one single operation. For example, the correction [*wide spread* → *widespread*] in Table 5 involves two individual operations: a substitution ([*wide* → *widespread*]) and a deletion ([*spread* → Ø]).

The statistics in Table 6 show that most errors in NUCLE and FCE-train involve only a single token

M	S	D	S	T	D	S	S	M
This This	wide widespread	spread	propaganda publicity	benefits only only benefits	to	the their	companys companies	. .

Table 5: Individual operations obtained from automatic alignment: (M)atch, (I)nsertion, (D)eletion, (S)ubstitution and (T)ransposition.

Orig:Corr Token	**NUCLE**		**FCE-train**	
Edit Size	**Cum. Freq.**	**%**	**Cum. Freq.**	**%**
1:1	17,580	41.23%	23,833	51.51%
0:1	24,823	58.22%	32,875	71.06%
1:0	30,599	71.77%	37,743	81.58%
0–2:0–2	36,868	86.47%	43,593	94.22%
0–3+:0–3+	42,636	100.00%	46,265	100.00%

Table 6: Distribution of edits in minimised gold NUCLE and FCE-train according to the number of tokens on either side of the edit; e.g. there are 17,580 instances of 1:1 token substitutions in NUCLE. Total edits are lower than in Table 3 because edit minimisation causes some edits to disappear.

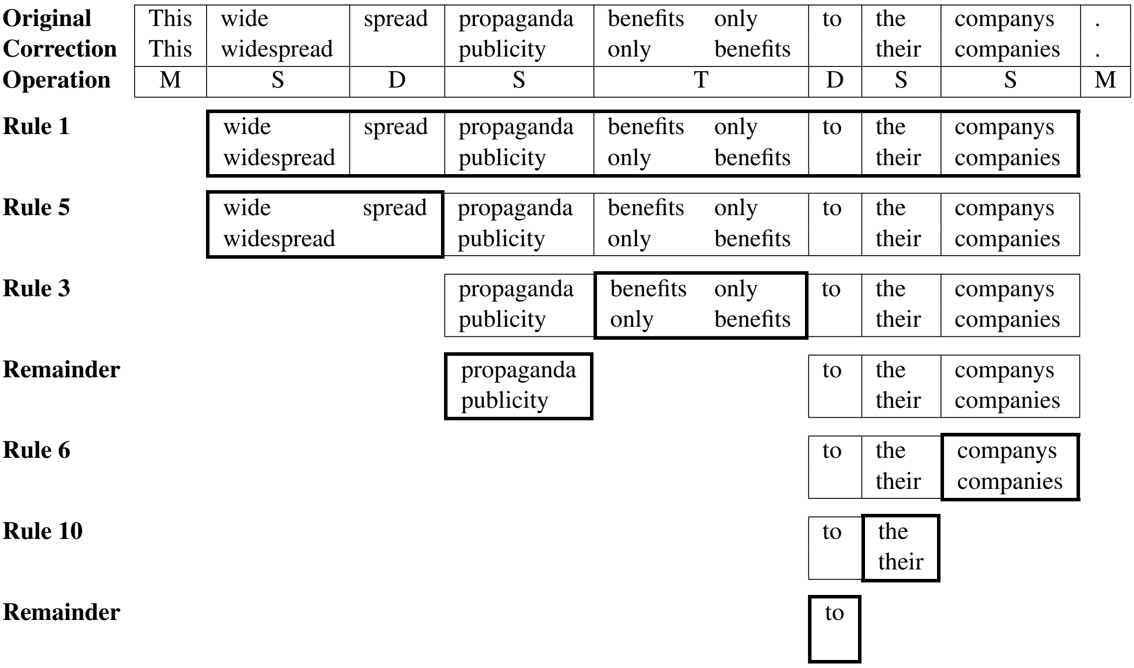
on either side of an edit (i.e. 0:1, 1:0 or 1:1), and so a simple *all-split* strategy that merges nothing is likely to cover most of these edits. In fact this explains why the results in Table 4 are so high; just using Damerau-Levenshtein is equivalent to the *all-split* setting. Nevertheless, multi-token edits still form an important class of learner errors and so we should attempt to handle them.

4.1 Merging Rules

In order to improve performance and capture multi-token edits, we hence implemented a recursive rule-based merging function. First, we analysed the relationship between human annotations and how they mapped to alignment operations in NUCLE and FCE-train. For example, we found that the most common multi-token errors involved phrasal verbs, such as [*look at* → *watch*]; possessive nouns, such as [*friends* → *friend 's*]; or orthographic changes, such as [*wide spread* → *widespread*]. Second, we wrote rules to merge or separate alignments based on these observed patterns.

The complete list of rules and their priority is as follows:

1. Any match operation (M) breaks a sequence into sub-sequences that are processed individually, e.g. MDDSMMTMSI is split into DDS, T and SI.
2. Any operation that involves punctuation and is followed by a token that changes case is merged, e.g. [, → .] + [*we* → *We*] becomes [, *we* → . *We*].
3. Transpositions are returned as individual edits, e.g. [*only can* → *can only*].
4. Any operation that involves a possessive suffix is merged with any previous operations, e.g. [*freinds* → *friend*] + [∅ → *'s*] becomes [*freinds* → *friend 's*].
5. Operations that add or remove whitespace between tokens are merged, even if they have unmatched apostrophes, e.g. [*sub* → *subway*] + [*way* → ∅] = [*sub way* → *subway*].
6. Substitutions between very similar tokens ($>$ 70% character matches) are returned as individual edits, e.g. [*writting* → *writing*], unless they have the same POS as the previous token, e.g. [*eated* → *have eaten*]; the verb phrase would be split without this exception.
7. Substitutions preceded by another substitution are returned as individual edits.
8. Any combination of operations that involves at least one content word is merged, e.g. [*On* → *In*] + [*the* → ∅] + [*other* → ∅] + [*hand* → *addition*] = [*On the other hand* → *In addition*].
9. Consecutive operations that involve tokens with the same part of speech are merged, e.g. [*because* → ∅] + [*of* → *for*] = [*because of* → *for*].
10. Any determiner at the end of a sequence is returned as an individual edit.



Original	This	wide	spread	propaganda	benefits	only	to	the	companys	.
Correction	This	widespread		publicity	only	benefits		their	companies	.
Operation	M	S	D	S	T		D	S	S	M
Rule 1		wide widespread	spread	propaganda publicity	benefits only	only benefits	to	the their	companys companies	
Rule 5		wide widespread	spread	propaganda publicity	benefits only	only benefits	to	the their	companys companies	
Rule 3				propaganda publicity	benefits only	only benefits	to	the their	companys companies	
Remainder				propaganda publicity			to	the their	companys companies	
Rule 6							to	the their	companys companies	
Rule 10							to	the their		
Remainder							to			

Figure 1: A step-by-step edit extraction example.

Each sequence of alignment operations between two sentences (e.g. MSDSTDSSM) is processed recursively using the above rules in a top-down fashion. Rules are applied in order, with priority relative to their position in the list. Every time an edit is returned by one of the rules, we process the remaining sub-sequences individually until they are exhausted or no more rules can be applied (see Figure 1). It should be noted that rules that iteratively grow the merge range of the alignment (e.g. #8) can be overridden by others with higher priority (e.g. #4), causing the remaining operations in the truncated subsequence to be reprocessed from scratch.

4.2 Merging Experiments

We evaluated our rule-based merging method on the CoNLL 2013, CoNLL 2014 (each annotator individually) and FCE test sets, and contrasted it against the following merging strategies:

all-split: All consecutive non-matches are split, e.g. DDSI → D, D, S, I.

all-merge: All consecutive non-matches are merged, e.g. DDSI → DDSI.

all-equal: All consecutive non-matches of the same operation are merged, e.g. DDSI → DD, S, I.

All of these methods were applied to the output of our enhanced Damerau-Levenshtein alignment described in the previous section. In addition to evaluating edit extraction against a minimised reference, we also evaluate error type classification on the merged output to replicate results for an end-to-end classification system. For this purpose, we retrained Xue and Hwa's publicly available implementation of their MaxEnt error type classifier[2] separately on NUCLE and FCE-train. This classifier was based on work by Swanson and Yamangil and is used in all classification experiments.

Results for both tasks are reported in Table 7 and reveal that our method achieves the best performance on all tasks and datasets. For edit extraction (i.e. merging), improvements in F_1 range between 4% and 12% over the second-best method (*all-merge*). We also observe that while *all-split* tends to have the highest number of TPs and lowest number of FNs, it also has the highest number of FPs, which shows how ignoring multi-token edits affects performance. In contrast, the *all-merge* strategy has the lowest number of FPs, but at the cost of also having the lowest number of TPs and highest number of FNs. This shows that each strategy has different strengths which our rule-based approach attempts to exploit.

[2] https://github.com/xuehuichao/correction_detector

Dataset	Method	Edit Extraction						Edit Extraction + Classification					
		TP	FP	FN	P	R	F_1	TP	FP	FN	P	R	F_1
CoNLL 2013	All-split	2715	1612	659	62.75	80.47	70.51	2088	2239	1286	48.26	61.89	54.23
	All-merge	2194	653	1180	77.06	65.03	70.54	1634	1213	1740	57.39	48.43	52.53
	All-equal	2417	1160	957	67.57	71.64	69.54	1856	1721	1518	51.89	55.01	53.40
	This work	2784	591	590	82.49	82.51	**82.50**	2072	1303	1302	61.39	61.41	**61.40**
CoNLL 2014 (0)	All-split	1858	1320	526	58.46	77.94	66.81	1267	1911	1117	39.87	53.15	45.56
	All-merge	1662	415	722	80.02	69.71	74.51	1062	1015	1322	51.13	44.55	47.61
	All-equal	1705	884	679	65.86	71.52	68.57	1130	1459	1254	43.65	47.40	45.45
	This work	1893	550	491	77.49	79.40	**78.43**	1242	1201	1142	50.84	52.10	**51.46**
CoNLL 2014 (1)	All-split	2635	1699	651	60.80	80.19	69.16	2052	2282	1234	47.35	62.45	53.86
	All-merge	2435	554	851	81.47	74.10	77.61	1791	1198	1495	59.92	54.50	57.08
	All-equal	2453	1174	833	67.63	74.65	70.97	1904	1723	1382	52.50	57.94	55.08
	This work	2866	598	420	82.74	87.22	**84.92**	2139	1325	1147	61.75	65.09	**63.38**
FCE-test	All-split	3660	1936	847	65.40	81.21	72.45	3073	2523	1434	54.91	68.18	60.83
	All-merge	3144	778	1363	80.16	69.76	74.60	2564	1358	1943	65.37	56.89	60.84
	All-equal	3373	1447	1134	69.98	74.84	72.33	2845	1975	1662	59.02	63.12	61.01
	This work	3861	739	646	83.93	85.67	**84.79**	3182	1418	1325	69.17	70.60	**69.88**

Table 7: Performance of different merging methods on the edit extraction and full error classification task. TP: true positives, FP: false positives, FN: false negatives, P: precision, R: recall.

Table 7 also reports performance on error classification, given edit extraction, revealing how each merging strategy affects automatic error type prediction. Results are consistent with edit extraction, although they are (expectedly) lower by an average 21.1% F_1. Improvements in F_1 between our method and the second-best range between 3.9% and 9.0%. While CoNLL 2014 (1) achieved the best result for edit extraction, FCE-test achieved the best result for error classification. This might be because the two datasets are annotated according to different error classification frameworks and the FCE annotation is more consistent than NUCLE annotations.

5 Discussion

It is worth stating that many of our reported results are actually an underestimate of true performance. This is because, despite gold reference minimisation, there is a high degree of variability in the way humans annotate this sort of data. For instance, as shown by Bryant and Ng (2015), human annotators often have very different perceptions of grammaticality and it is linguistically plausible that, for example, [*has eaten* → *was eating*] is annotated either as one edit (as above) or two edits ([*has* → *was*] + [*eaten* → *eating*]) by different, or even the same, annotators. This means the *all-split* merging strategy will never match the former while the *all-merge* merging strategy will never match the latter, even though the annotations fundamentally equate to the same thing. Due to this inconsistency, system performance will be underestimated regardless of which merge strategy you choose.

In contrast, we consider merge consistency a strength of our rule-based approach. Even if our alignment does not agree with the gold standard in some cases, at least the decision to merge or split is consistent across all similar cases. Our approach could hence be used to standardise ambiguous annotations where splits or merges are equally plausible.

Another strength of a rule-based approach is that it is easier to diagnose which rules are responsible for producing a given output sequence. This is in contrast with machine learning techniques, which additionally require feature engineering and retraining, where it is much more difficult to determine why certain edits were merged in a certain way. Nevertheless, considering only about 30% of all edits (at most) in any dataset require merging anyway, a rule-based approach seemed a more cost-effective solution.

We also investigated how our approach compared against previous approaches in terms of single-token edits and multi-token edits (Table 8). To produce comparable results for Xue and Hwa (X&H), we retrained their publicly available implementation of their MaxEnt merging classifier separately on NUCLE and FCE-train. In general, while our method tends to score slightly lower precision than the

Dataset	Method	Single-token edits			Multi-token edits		
		P	R	F_1	P	R	F_1
CoNLL 2013	S&Y	95.67	65.73	77.92	16.62	40.62	23.59
	X&H	90.34	73.13	80.82	24.53	48.75	32.64
	This work	88.76	87.80	**88.28**	51.00	31.87	**39.23**
CoNLL 2014 (0)	S&Y	95.79	68.59	79.94	24.11	51.52	**32.85**
	X&H	89.44	74.88	81.52	25.70	41.67	31.79
	This work	83.27	85.96	**84.60**	34.57	21.21	26.29
CoNLL 2014 (1)	S&Y	94.11	73.95	82.82	31.56	53.81	39.79
	X&H	90.71	80.87	85.51	38.06	52.38	44.09
	This work	86.32	93.64	**89.83**	71.43	40.48	**51.67**
FCE-test	S&Y	95.19	69.76	80.51	30.19	52.08	38.22
	X&H	82.98	83.90	83.44	66.27	52.72	58.72
	This work	88.35	91.00	**89.65**	74.06	50.16	**59.81**

Table 8: Performance of our proposal vs. previous methods in terms of single and multi-token edits. S&Y used Levenshtein with an *all-merge* strategy while X&H used Levenshtein with a MaxEnt merging classifier.

others in the single-token setting, it makes up for this with a much higher recall. In contrast, our method achieves a higher precision in the multi-token setting, but at the cost of a lower recall. Ultimately, however, our method increases overall performance in almost all cases, the exception being multi-token edits in CoNLL 2014 (0), which is known to be an inconsistent dataset. These results hence confirm that our rule-based merging strategy is superior to previous approaches.

In addition to a quantitative analysis, we also carried out an informal qualitative analysis of the errors made by our system. One source of errors involves tokens that are affected by more than one mistake; e.g. [*wide spraed* → *widespread*]. While our system includes a rule to merge adjacent alignments where the only difference is white space, this rule does not activate in the above case since one of the tokens also contains a misspelling. This consequently means the alignments are not merged and do not match the gold standard; such cases are difficult to handle.

Another issue is that reference minimisation is unable to deal with edits where identical tokens occur in the middle of an edit; e.g. [*can easily been* → *could easily be*]. As an automatic alignment will always consider *easily* a matched token, the remaining non-matches will become isolated and hence never merged. In this case, however, we would argue that our automatic alignment is more informative than the human alignment which needlessly includes a redundant token in the edit.

Finally, we provide a comparison between our method and the previous approaches by Swanson and Yamangil (S&Y) and Xue and Hwa (X&H) (Table 9). Our method achieves state-of-the-art performance on all tasks and datasets, with an average improvement over X&H of 6.0% F_1 for edit extraction and 4.1% F_1 when adding error classification.

6 Conclusion

We have presented a new method for extracting edits from a parallel original and corrected sentence pair based on a linguistically enhanced token alignment and rule-based merging component. Results on a number of GEC test sets show that our method outperforms all previous work on edit extraction and can also boost error classification performance.

Since we only use a few hand-coded rules, we do away with the complexity of machine learning solutions and are hence also able to annotate data much more consistently. This is particularly useful for standardising GEC datasets, which are often annotated using different guidelines.

Dataset	Method	Edit Extraction F_1	Edit Extraction + Classification F_1
CoNLL 2013	S&Y	70.42	52.85
	X&H	74.07	55.89
	This work	**82.50**	**61.40**
CoNLL 2014 (0)	S&Y	72.92	46.95
	X&H	74.25	49.15
	This work	**78.43**	**51.46**
CoNLL 2014 (1)	S&Y	76.39	56.18
	X&H	79.21	59.24
	This work	**84.92**	**63.38**
FCE-test	S&Y	73.59	59.80
	X&H	79.18	65.33
	This work	**84.79**	**69.88**

Table 9: Performance of our proposal vs. previous methods in an end-to-end edit extraction and classification task.

References

Christopher Bryant and Mariano Felice. 2016. Issues in preprocessing current datasets for grammatical error correction. Technical Report UCAM-CL-TR-894, University of Cambridge, Computer Laboratory, September.

Christopher Bryant and Hwee Tou Ng. 2015. How far are we from fully automatic high quality grammatical error correction? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 697–707, Beijing, China, July. Association for Computational Linguistics.

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, June. 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, June. Association for Computational Linguistics.

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. Asian Federation of Natural Language Processing.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel R. Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria. ACL.

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, USA. ACL.

Diane Nicholls. 2003. The cambridge learner corpus: Error coding and analysis for lexicography and ELT. In *Proceedings of the Corpus Linguistics 2003 conference*, pages 572–581.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Keisuke Sakaguchi, Courtney Napoles, Matt Post, and Joel Tetreault. 2016. Reassessing the goals of grammatical error correction: Fluency instead of grammaticality. *Transactions of the Association for Computational Linguistics*, 4:169–182.

Ben Swanson and Elif Yamangil. 2012. Correction detection and error type selection as an ESL educational aid. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 357–361, Montréal, Canada, June. Association for Computational Linguistics.

Huichao Xue and Rebecca Hwa. 2014. Improved correction detection in revised esl sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 599–604, Baltimore, Maryland, June. Association for Computational Linguistics.

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, June. Association for Computational Linguistics.