UW-Stanford System Description for AESW 2016 Shared Task on Grammatical Error Detection

Dan Flickinger
Stanford University

Michael Goodman
University of Washington

Woodley Packard
University of Washington

Abstract

This is a report on the methods used and results obtained by the UW-Stanford team for the Automated Evaluation of Scientific Writing (AESW) Shared Task 2016 on grammatical error detection. This team developed a symbolic grammar-based system augmented with manually defined *mal-rules* to accommodate and identify instances of high-frequency grammatical errors. System results were entered both for the probabilistic estimation track, where we ranked second, and for the Boolean decision track, where we ranked fourth.

1 Introduction

Over the past several years, a series of shared tasks have been organized to foster research on the automatic detection of grammatical errors in compositions from a variety of genres. In this year's task, the organizers invited participants to "analyze the linguistic characteristics of scientific writing to promote the development of automated writing evaluation tools that can assist authors in writing scientific papers. The task is to predict whether a given sentence requires editing to ensure its 'fit' within the scientific writing genre." This Automated Evaluation of Scientific Writing (AESW) Shared Task 2016 is described in more detail, along with descriptions of the seven teams and their system results, in Daudaravicius et al. (2016).

This paper provides a description of the system adapted for this task by a team of collaborators from the University of Washington and from Stanford University, The system is based on a symbolic grammar augmented with a set of manually constructed *mal-rules* (Schneider and McCoy, 1998; Bender et al., 2004) designed to license and identify ungrammatical or stylistically deprecated sentence properties in running text. We used an efficient parser to analyze the training and development sets repeatedly with successively refined versions of augmented grammar, producing for each sentence a derivation which recorded any use of mal-rules, thus triggering the "needs editing" flag relevant for the shared task. In the end, we applied the best-performing of these grammar versions to the test data in order to produce the results submitted for scoring.

2 Resources and methods

Our basic approach to this task has much in common with the one used in the Stanford system participating in the 2013 CoNLL Shared Task on Grammatical Error Correction (Flickinger and Yu, 2013), again employing a current version of the English Resource Grammar (ERG: Flickinger (2000), Flickinger (2011)) and a task-specific inventory of mal-rules, but this time using a more efficient parser (ACE: `moin.delph-in.net/AceTop`). As with the earlier task, we used the most likely derivation licensed by the grammar and produced by the parser for each sentence, to identify any use of one or more of these mal-rules in the analysis.

For the Boolean decision track, we predicted an error if the top-ranked analysis has an error or does not exist, if the probability of using a mal-rule was at least 1%, or if this sentence was the most likely

Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications, pages 105–111,
San Diego, California, June 16, 2016.

sentence in the paragraph to require a mal-rule. For the probabilistic track, our estimator also included several parameters hand-tuned on the training data.

2.1 Existing resources

Most of the components used in the UW-Stanford system were drawn from the inventory of resources developed and maintained by members of the DELPH-IN (Deep Linguistic Processing with HPSG) consortium (`www.delph-in.net`). These components include the ACE parser, the ERG, the Redwoods treebank (Oepen et al., 2004), and a set of Python libraries for the processing of DELPH-IN components' data (`github.com/delph-in/pydelphin`).

For the grammar, we used a modified version of the "1214" release of the ERG, with substantially the same core linguistic coverage, but with the important addition of a separate set of rules and lexical entries dubbed *mal-rules*, to explicitly characterize frequently occurring ill-formed phrasal constructions and words observed in the data for this shared task. This extended grammar enabled full analyses of roughly 92% of the sentences in each of the training and development data sets provided for the task. Of the 8% of the sentences not covered by this grammar, nearly half failed to parse due to reaching externally imposed (though generous) resource limits before yielding any analysis. Lacking any guidance from the grammar, all of these unparsable sentences were tagged by the system as in need of editing, for the purposes of scoring.

ACE is a modern unification-based chart parser, using an agenda-driven variant of the CKY algorithm. Efficiency is enhanced by aggressive ambiguity packing (Oepen and Carroll, 2000) and selective unpacking (Carroll and Oepen, 2005), and disambiguation is performed using a maximum entropy ranker trained on the Redwoods treebank. ACE's integrated preprocessing and support for clustered computing made it easy to use for parsing the large bodies of text in the training and development data.

Since for this task we only wanted the most likely analysis for each sentence, we used the best available statistical parse selection model for the ERG, one trained using the Redwoods treebank, a manually annotated corpus of some 1.4 million words of text across multiple genres. Crucially, and to the disadvantage of our use for this task, the treebank does not contain any analyses using the mal-rules which of course figure prominently in the derivations of many of the sentences in the AESW corpus. We hypothesize that the system could have been improved in both precision and recall if the parse selection model had contained information about the relative likelihoods of these mal-rules, but a methodical test of this hypothesis will have to await further study.

2.2 Error Discovery

In order to focus our efforts we performed a series of basic searches over the training corpus. Table 1 shows the number of deletes (not followed by inserts), inserts (not preceded by deletes), and substitutions (delete followed by insert) in the corpus. With the training set consisting of about 1.2 million sentences, roughly one of every three sentences contains one or more error corrections.

Error Type	Count	% Sentences
del-only	124,919	9.03%
ins-only	267,465	17.89%
del-ins	341,108	28.68%
total	733,492	39.24%

Table 1: Basic error types: unique counts and percentage of sentences affected

Table 2 shows the ten most frequent token-specific insertions, deletions, and substitutions made by the editors in the training corpus. Changes to commas, hyphens, and colons together comprise more than half of all of the edits in the corpus in each of the three types of changes, and the frequency of these is reflected in the choices of phrasal and lexical mal-rules added to the grammar for this task. Changes to the English articles *the* and *a*, while much less frequent than punctuation edits, are the next most significant category. The substitution of *section* to *Section* is the most frequent instance of a class of capitalization edits which collectively account for roughly 5.91% of substitutions. Edits in this category are highly dependent on their context (e.g., the initial capital is generally preferred in *Section _REF_*, but not in *this section*).

~~,~~	40.09%	<ins>,</ins>	52.94%	<del> </del><ins>-</ins>	13.03%
<del>the </del>	9.19%	<ins>the </ins>	12.66%	~~-~~<ins> </ins>	5.10%
~~:~~	8.96%	<ins>.</ins>	4.78%	~~,~~<ins>;</ins>	1.92%
~~-~~	3.14%	<ins>a </ins>	4.03%	~~.~~<ins>,</ins>	0.74%
~~.~~	1.94%	<ins>:</ins>	2.87%	~~section~~<ins>Section</ins>	0.66%
~~'s~~	1.84%	<ins>that </ins>	2.14%	~~:~~<ins>.</ins>	0.60%
<del>that </del>	1.49%	<ins>and </ins>	1.92%	~~the~~<ins>a</ins>	0.59%
<del>a </del>	1.45%	<ins>an </ins>	0.78%	~~a~~<ins>the</ins>	0.56%
~~'~~	1.29%	<ins>is </ins>	0.63%	~~which~~<ins>that</ins>	0.54%
~~"~~	0.91%	<ins>of </ins>	0.63%	~~is~~<ins>are</ins>	0.54%

Table 2: Top ten deletions, insertions, and substitutions

2.3 Symbolic methods

The system relies on the analyses licensed by the grammar to identify sentences containing an error, by finding, in the most likely analysis of an ill-formed sentence, one or more occurrences of mal-rules that permit specific types of ungrammatical constituents. For example, in (1) we show the analysis produced for the sentence "Another important point to note is the problem of unknown SNR situation." where the rule used to admit the phrase *unknown SNR situation* is a mal-rule (`hdn_bnp_c_rbst`) recording the omission of an obligatory article for the noun phrase.

(1) Sample derivation tree with mal-rule

```
(sb-hd_mc_c
 (sp-hd_n_c
  (another "another")
  (hd-cmp_u_c
   (aj-hdn_norm_c
    (hd_optcmp_c
     (j_tough_dlr (important_a2 "important")))
    (n_sg_ilr (point_n2 "point")))
   (hd-cmp_u_c
    (to_c_prop "to")
    (hd_xcmp_c
     (v_n3s-bse_ilr (note_v1 "note"))))))
 (hd-cmp_u_c
  (be_id_is "is")
  (sp-hd_n_c
   (the_1 "the")
   (hd-cmp_u_c
    (n_sg_ilr (problem_n1 "problem"))
    (hd-cmp_u_c
     (of_prtcl "of")
     (hdn_bnp_c_rbst
      (aj-hdn_norm_c
       (j_j-un_dlr
        (v_j-nb-pas-tr_dlr
         (v_pas_odlr (know_v1 "unknown"))))
       (np-hdn_cpd_c
        (hdn_bnp-pn_c
         (n_sg_ilr (generic_proper_ne "SNR")))
        (w_period_plr
         (n_sg_ilr
          (situation_n1 "situation."))))))))))))
```

In order to identify frequently occurring error phenomena as candidates to motivate the addition of mal-rules for this task, we carried out a manual analysis and classification of the types of insertion and deletion editing operations annotated in the training and development corpora. As each good candidate emerged, we added a corresponding mal-rule to the grammar, and evaluated the resulting behavior of the system on sample data sets drawn from both training and development. Not all of the rules survived in the final system, either because a particular rule interacted poorly with the rest of the grammar, or because the annotations in the corpus for that phenomenon showed more variation than the rule anticipated. In the final system, the grammar included thirteen phrasal mal-rules, seventeen lexical mal-rules, and about 350 robust lexical entries. Examples of each are given in (2).

(2) Sample mal-rules added to the ERG

a. Syntactic: Comma-spliced sentences

   `cl-cl_runon-cma_c_rbst`

   Multi-biometrics may address the problem of nonuniversality, e.g., in a speaker recognition system, the individuals who cannot speak cannot be enrolled.

b. Lexical: Subject-verb agreement mismatch

   `third_sg_fin_v_rbst`

   In what follow, this letter investigates mixed synchronization of fractional-order Lorenz-like system

c. Lexical entry: Missing obligatory direct object

   `allow_v1_rbst`

   The LHC will be a top quark factory, allowing to study several of its properties in great detail.

A large number of the editorial annotations in the AESW corpora addressed errors in the use of punctuation marks, and thus several of the mal-rules were added to the grammar to permit ill-formed or stylistically deprecated uses of commas and hyphens. For example, several of the phrasal coordination rules were adapted as mal-rules to identify the missing final comma in a multi-part conjoined structure, as in the phrase *such as _MATH_, _MATH_ or _MATH_*.

The process of refining the choices and definitions of these mal-rules included a frequently repeated cycle of manual inspection of the corpus annotations, modifications to the mal-rule inventory, parsing of three 1000-item samples taken from the training and development data sets, and examination of the resulting precision and recall performance of the system on these sample sets.

2.4 Statistical methods

The ERG nearly always produces multiple candidate analyses for a given input. Since not all analyses are equally likely, the ERG supplies a maximum entropy model which defines a probability distribution over the set of analyses of any given input:

$$P(a|I) = \frac{1}{Z} e^{\sum_{f \in a} \lambda_f}$$

$$Z = \sum_{b \in \text{analyses}(I)} e^{\sum_{f \in b} \lambda_f}$$

By virtue of ambiguity packing, the parser is able to represent vast sets of analyses in a compact form known as a packed forest. Typically, a user of the ERG wants a single parse tree out, and to this end the ACE parser is capable of efficiently selecting the single tree from the packed forest without enumerating the rest of the analyses (selective unpacking). For the task at hand, however, we were interested not in the complete structure of the best tree, but in whether or not it used any mal-rules. This Boolean quantity can be evaluated directly on the top-ranked tree, but its expected value with respect to the probability distribution defined by the maximum entropy model can also be evaluated over the entire packed forest.

To compute this expectation, we defined a *root symbol*[1] which matches[2] all and only trees that include mal-rules. These are conceptually similar to unary rules which can only appear at the root of the derivation tree. The parser computes the normalization factor Z_r for the set of analyses[3] dominated by each root symbol, using an algorithm similar to the inside algorithm for PCFGs, but keeping track of grandparent contexts as required by the maximum entropy model in a fashion similar to that used by the selective unpacking algorithm. The expected value of the mal-rule indicator (i.e. the probability that a tree drawn according to the maximum entropy distribution uses mal-rules) is then:

$$P(\text{mal-rules}|I) = \frac{Z_{\text{mal}}(I)}{Z_{\text{mal}}(I) + Z_{\text{ordinary}}(I)}$$

Thus we have two signals for use in producing the output value for the shared task: the Boolean indicator for the top-ranked tree:

$$B_{\text{top}}(I) = \begin{cases} 1 & \text{best tree uses mal-rules} \\ 0 & \text{best tree does not use mal-rules} \end{cases}$$

and the real-valued expectation of that indicator:

$$E(B|I) = P(\text{mal-rules}|I)$$

We use both of these signals in both the Boolean and probabilistic tracks. Specifically, our Boolean estimator for an input I in a paragraph P is:

$$\text{output}_{\text{boolean}}(I) = \begin{cases} 1 & \text{no parse was found} \\ 1 & B(I) = 1 \\ 1 & E(B|I) > 0.01 \\ 1 & E(B|I) = \max_{S \in P} E(B|S) \\ 0 & \text{otherwise} \end{cases}$$

That is, we predict an error if the top-ranked tree has an error or does not exist, if the probability of using a mal-rule was at least 1%, or if this sentence was

[1] In fact, we have multiple root symbols for mal-rule trees and multiple root symbols for ordinary trees.

[2] The fact that a mal-rule has been used is passed up through the feature structures by the grammar.

[3] Actually, the factor computed can in some cases include probabilities for trees which are not fully-consistent with all of the constraints in the grammar, depending on the aggressiveness of the ambiguity packing optimizations employed; however, we hypothesize that this does not have a large effect on the accuracy of the system.

the most likely sentence in the paragraph to require a mal-rule.

For the probabilistic track, our estimator included several parameters hand-tuned on the training and development data. An obvious approach would be to directly use $P(\text{mal-rules}|I)$ as our probabilistic estimate, but this does not produce very good results, because the task's evaluation metric is a kind of F-score on errors, rather than a balanced measure over the entire data set. Indeed, we noticed that results are penalized for ever guessing a probability less than 0.5, and indeed always guessing 0.63 yields an F-score that outperforms all but 2 of the participating teams. This is arguably a deficiency in the probabilistic track evaluation metric. Instead of using $P(\text{mal-rules}|I)$ directly, we applied a simple nonlinear transformation:

$$\text{output}_{\text{probabilistic}}(I) =$$

$$\begin{cases} 0.75 & \text{no parse was found} \\ 0.72 + 0.1 \cdot (E(B) - 0.5) & B = 1 \\ 0.70 + 0.12 \cdot E(B)^{0.2} & \substack{B=0 \text{ and} \\ E(B)>0.01} \\ 0.70 & \text{otherwise} \end{cases}$$

The various "magic" numbers in these formulae were determined by manual search on the training and development data.

3 Results

Roughly one in three sentences in the training corpus was annotated with an error. The probabilistic track and the Boolean track both used varieties of F-score over error sentences as the evaluation metric. Our system tended to be somewhat conservative about identifying errors; as a result, we found that skewing our outputs towards recall in exchange for some precision improved F-score, although accuracy suffered substantially.

3.1 Probabilistic track

Our system ranked second of eight entrants in F-score in the probabilistic track. The probabilistic F-score was defined as the harmonic mean of two related quantities, dubbed precision and recall, defined as follows:

$$P_{\text{prob}} = 1 - \frac{1}{n} \sum_{\substack{i \\ \pi_i > 0.5}} (\pi_i - G_i)^2$$

$$R_{\text{prob}} = 1 - \frac{1}{m} \sum_{\substack{i \\ G_i = 1}} (\pi_i - G_i)^2$$

F-scores ranged from 0.6925 to 0.8311 on the evaluation data, with our system achieving 0.7849. Table 3 shows the relative F-score of the entrants on the official evaluation data, alongside the correlation coefficient between their outputs and the gold standard.

Team	F-Score	Correlation
1	0.8311	0.0600
UW-SU	0.7849	0.2471
2	0.7581	0.2690
3	0.7419	0.4043
4	0.7224	0.1298
5	0.7220	0.1666
6	0.6926	0.4173
7	0.6925	0.3516

Table 3: Results for the probabilistic track

It is interesting (and a bit concerning) to note that the F-score metric defined for the task has a strong negative correlation (-0.60) with the more intuitively interpretable correlation coefficient. As mentioned above, we modified the intuitively more appropriate estimator $E(B|I)$ to fit the F-score metric better. Table 4 shows the result of this modification, on the development data (as well as the baseline of always guessing 0.63):

Estimator	F-Score	Correlation
$P = E(B\|I)$	0.5644	0.2414
$P = \text{output}_{\text{probabilistic}}$	0.7902	0.2602
$P = 0.63$	0.7756	–

Table 4: Comparison of estimators (probabilistic track)

As can be seen, rearranging our results with a simple nonlinear transformation had almost no effect on the correlation with gold scores, but improved our F-score tremendously. We wonder what effect a similar simple transformation might have for a team like NTNU-YZU or Knowlet, whose correlation with gold substantially exceeds ours.

3.2 Boolean track

Our Boolean track system ranked number four of nine entrants. The evaluation metric for the binary

track was the F-score for identifying error sentences, defined in a normal way. The relative performance of the nine entrants on the official evaluation data is shown in Table 5.

Team	Prec	Rec	F-score
1	0.5444	0.7413	0.6278
2	0.5025	0.7785	0.6108
3	0.4482	0.7279	0.5548
UW-SU	0.4145	0.8201	0.5507
5	0.3851	0.9241	0.5436
6	0.3765	0.9480	0.5389
7	0.3960	0.6970	0.5051
8	0.6717	0.3805	0.4858
9	0.6241	0.3685	0.4634

Table 5: Results for the Boolean track

This evaluation metric, while somewhat different in focus from a simple accuracy measure, did not exhibit the disconcerting behaviors observed with the probabilistic track metric; for instance, the correlation coefficient between F-score and system-to-gold correlation was 0.31, which is far less alarming than the -0.60 observed in the probabilistic track.

Again, we found that using the raw grammar output (i.e. the B indicator variable described above) was less effective in terms of the evaluation metric than the thresholded and transformed $\text{output}_{\text{boolean}}$, as demonstrated in Table 6 (computed over the development data):

Estimator	F-Score	Accuracy
B	0.5411	0.6333
$\text{output}_{\text{boolean}}$	0.5854	0.5322
no errors	–	0.6111
all errors	0.5600	0.3889

Table 6: Comparison of estimators (Boolean track)

The raw B estimator achieves much higher accuracy than the transformed $\text{output}_{\text{boolean}}$, but somewhat lower F-score—lower in fact than the baseline of guessing that every sentence contains an error.

4 Discussion

We encountered several challenges while developing and tuning our system for this task, both in the consistency of the error annotations in the corpus, and in the grammatically lossy method of substituting the single tokens *_MATH_*, *_MATHDISP_*, etc. in place of formulaic sequences of tokens. It is not surprising that a corpus of this size would reflect inconsistencies due to the use of multiple "annotators" (editors), and also some number of overlooked errors due to the complexity and scale of the editorial task. Yet for our rule-based approach to error detection, judging the benefit of a newly added mal-rule was not easy, since it might perform as intended on the data, but fail to have a positive impact on the final F-score because of a substantial number of missing or inconsistent error annotations. For example, an error was often signaled for adjective-noun compounds used as modifiers if there was no hyphen connecting the two tokens, as with *second order derivatives* vs. the corrected *second-order derivatives*; however, in the training set, we find the following frequency counts for the three different patterns:

" second order "	636
" second<del> </del><ins>-</ins>order "	710
" second-order "	952

Clearly, the dominant intended pattern is for these adjective-noun compounds to be hyphenated, but a significant percentage of occurrences without the hyphen were left unedited for many such compounds, and this made the calculation of whether or not to globally impose the regularity via mal-rule a murky one. We saw similar wide variation in error annotation for other hyphenation conventions, such as with the insertion or deletion of hyphen with the *non-* prefix as in *non-linear*.

A second set of challenges to our grammar-based method arose from effects of the quite reasonable and useful decision to replace certain complex expressions in the text with single placeholder tokens, such as *_MATH_* for mathematical expressions, or *_CITE_* for citations. Because these tokens could stand in for a variety of grammatically distinct phrase types, such as singular or plural noun phrases, declarative clauses, or numerical adjectives, the parsing task could quickly become costly when explicitly representing these ambiguities for each occurrence of each such placeholder token, particularly when a sentence included a series of such terms. A more useful representation might have been to preserve the original literal string as markup

on the token that replaced it.

5 Conclusions

The grammar-based method we adopted for this task gave us quite fine-grained control over the types of errors that our system would attend to, but many of the more linguistically interesting error phenomena occurred with low enough frequency in the training and development corpora that their accurate identification had little noticeable effect on system scoring, given the preponderance of punctuation-oriented edits. Overall, we remain optimistic about the utility of the kind of grammar-based approach we adopted here, when applied to real-world grammar-checking where fully consistent execution of a particular set of editorial principles should be welcomed by the scientific writer.

References

Emily M. Bender, Dan Flickinger, Stephan Oepen, Annemarie Walsh, and Timothy Baldwin. 2004. Arboretum. Using a precision grammar for grammar checking in CALL. In *Proceedings of the InSTIL Symposium on NLP and Speech Technologies in Advanced Language Learning Systems*, Venice, Italy, June.

John Carroll and Stephan Oepen. 2005. High efficiency realization for a wide-coverage unification grammar. In Robert Dale and, editor, *Proceedings of the 2nd International Joint Conference on Natural Language Processing*, Springer Lecture Notes in Computer Science. Jeju, Republic of Korea.

Vidas Daudaravicius, Rafael E. Banchs, Elena Volodina, and Courtney Napoles. 2016. A report on the automatic evaluation of scientific writing shared task. In *Proceedings of the Eleventh Workshop on Innovative Use of NLP for Building Educational Applications*, San Diego, CA, USA, June. Association for Computational Linguistics.

Dan Flickinger and Jiye Yu. 2013. Toward more precision in correction of grammatical errors. In *Proceedings of the 17th Conference on Natural Language Learning*, pages 68–73, Sofia, Bulgaria.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6 (1) (Special Issue on Efficient Processing with HPSG):15 – 28.

Dan Flickinger. 2011. Accuracy vs. robustness in grammar engineering. In Emily M. Bender and Jennifer E. Arnold, editors, *Language from a Cognitive Perspective: Grammar, Usage, and Processing*, pages 31–50. Stanford: CSLI Publications.

Stephan Oepen and John Carroll. 2000. Ambiguity packing in constraint-based parsing – practical results. In *Proceedings of NAACL 2000*, pages 162–169, Seattle, USA.

Stephan Oepen, Daniel Flickinger, Kristina Toutanova, and Christopher D. Manning. 2004. LinGO Redwoods. A rich and dynamic treebank for HPSG. *Journal of Research on Language and Computation*, 2(4):575 – 596.

David Schneider and Kathleen McCoy. 1998. Recognizing syntactic errors in the writing of second language learners. In *Proceedings of Coling-ACL 1998*, pages 1198 – 1204, Montreal.