Toward More Precision in Correction of Grammatical Errors

Dan Flickinger
Center for the Study of
Language and Information
Stanford University
danf@stanford.edu

Jiye Yu
Center for the Study of
Language and Information
Stanford University
jyu2009@stanford.edu

Abstract

We describe a system for detecting and correcting instances of a small class of frequently occurring grammatical error types in a corpus of essays which have been manually annotated for these errors. Our system employs a precise broad-coverage grammar of English which has been augmented with a set of *mal-rules* and *mal-entries* to explicitly license certain types of erroneous expressions. The derivation tree produced by a parser using this grammar identifies the location and type of an error in an ill-formed sentence, enabling a post-processing script to use the tree and the inventory of error types to delete and/or insert tokens in order to produce a corrected version of the original sentence.

1 Overview

As a participating group in the 2013 CoNLL Shared Task on Grammatical Error Correction, we adapted an existing system for error detection in a simpler closed-vocabulary domain to meet the additional demands of accommodating an open vocabulary and producing corrections for the errors identified. The training and test data for this shared task are from the NUCLE corpus (Dahlmeier et al., 2013), which consists of about one million words of short essays written by relatively competent English language learners. Each sentence has been manually annotated to identify and correct a wide range of grammatical and stylistic error types, though the shared task focused only on correcting instances of five of these types. Following standard procedure for such shared tasks, the organizers supplied most of the annotated data as a development corpus, and held out a 1381-sentence test corpus which was used for the evaluation of system output.

2 Resources and Method

The system developed for this task is an extension of an existing language-processing engine used to identify grammatical errors in short sentences and paragraphs written by elementary school students as part of the automated Language Arts and Writing course included in the EPGY (Education Program for Gifted Youth) course offerings (Suppes et al., 2012). This error detection engine consists of a grammar, a parser, and a post-processing script that interprets the error codes in the derivation tree for each parsed sentence. Both the grammar and the parser are open-source resources developed and distributed as part of the DELPH-IN consortium (*www.delph-in.net*). We use the English Resource Grammar, described below, which we have augmented with both rules and lexical entries that license instances of certain error types, using the *mal-rule* approach of (Schneider and McCoy, 1998), adapted and extended for the ERG as described in (Bender et al., 2004). For parsing each sentence with this grammar, we use the relatively efficient PET parser (Callmeier, 2002), along with a parse-ranking method based on a model trained on a manually disambiguated treebank, so far consisting only of parses of well-formed sentences. In addition to using the manually constructed 37,000-word lexicon included in the ERG, we accommodate unknown words by mapping POS tags produced by TnT (Brants, 2000) to generic lexical entry types on the fly. The bottom-up chart parser then exhaustively applies the rules of the grammar to the lexical entries introduced by the tokens in the input sentence, producing a packed forest of analyses (derivations) ranked by likelihood, and then presents the most likely derivation for post-processing. The post-processor is a script which uses the derivation tree to identify the type and location of each error, and then takes appropriate action, which in the course is an instructional message to the student, and in this shared task is a corrected version of the original sentence.

2.1 English Resource Grammar

The English Resource Grammar used for this task (ERG: (Flickinger, 2000), (Flickinger, 2011)) is a broad-coverage grammar implementation which has been under continuous development since the mid-1990s at Stanford. As an implementation within the theoretical framework of Head-driven Phrase Structure Grammar (HPSG: (Pollard and Sag, 1994)), the ERG has since its inception encoded both morphosyntactic and semantic properties of English, in a declarative representation that enables both parsing and generation. While development has always taken place in the context of specific applications, primary emphasis in the ERG has consistently been on the linguistic accuracy of the resulting analyses, at some expense to robustness. Its initial use was for generation within the German-English machine translation prototype developed in the Verbmobil project (Wahlster, 2000), so constraining the grammar to avoid overgeneration was a necessary design requirement that fit well with the broader aims of its developers. Applications using the grammar since then have included automatic processing of e-commerce customer support email messages, a second machine translation system (LOGON: (Lnning et al., 2004)), and information extraction over the full English Wikipedia (Flickinger et al., 2010).

At present, the ERG consists of a rich hierarchy of types encoding regularities both in the lexicon and in the syntactic constructions of English. The lexicon contains 40,000 manually constructed lexeme entries, each assigned to one of 975 lexical types at the leaves of this hierarchy, where the types encode idiosyncracies of subcategorization, modification targets, exceptional behavior with respect to lexical rules, etc. The grammar also includes 70 derivational and inflectional rules which apply to these lexemes (or to each other's outputs) to produce the words as they appear in text. The grammar provides 225 syntactic rules which admit either unary or binary phrases; these include a relatively small number of highly schematic rules which license ordinary combinations of heads with their arguments and their modifiers, and a rather larger number of construction-specific rules both for frequently occurring phrase types such as coordinate structures or appositives, and for phrase types that occur with markedly differing frequencies in verious corpus domains, such as questions or vocatives. Statistical models trained on manually annotated treebanks are used both in parsing (Toutanova et al., 2005) and in generation (Velldal, 2008) to rank the relative likelihoods of the outputs, in order to address the issue of disambiguation which is central to the use of any broad-coverage grammar for almost any task.

2.2 Mal-rule example

Each of the hand-coded mal-rules added to the standard ERG is a variant of a rule needed to analyse well-formed English input. A simple example of a mal-rule is given below, expressed in the attribute-value representation for an HPSG rule; this unary rule licenses a noun phrase headed by a singular count noun but lacking its normally obligatory article, as for the NP *black cet* in *That dog chased black cat*. Here the single daughter in this noun phrase (the HD-DTR) is a nominal phrase still seeking an obligatory specifier (the article or determiner in a well formed noun phrase), where the head noun is a singular count noun (non-divisible). The SYNSEM value in the rule discharges that obligatory specifier requirement just as the normal unary rule for bare plural noun phrases does, and supplies the necessary implicit quantifier in the semantics of the phrase.

$$\begin{bmatrix} \text{SYNSEM} & \begin{bmatrix} \text{LOCAL} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \boxed{1}\textit{noun} \\ \text{VAL} & \begin{bmatrix} \text{SPR} & \langle\, \rangle \\ \text{COMPS} & \langle\, \rangle \end{bmatrix} \end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \text{RELS} & \langle\, \textit{quant\_rel} \,\rangle \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{HD-DTR} & \begin{bmatrix} \text{SYNSEM} & \begin{bmatrix} \text{LOCAL} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \boxed{1} \\ \text{VAL} & \begin{bmatrix} \text{SPR} \langle \begin{bmatrix}\text{OPT} -\end{bmatrix} \rangle \\ \text{COMPS} \langle\, \rangle \end{bmatrix} \end{bmatrix} \\ \text{AGR} & \begin{bmatrix} \text{PN} & \text{3sing} \\ \text{DIV} & - \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Mal-rule for bare singular NP

2.3 Error types in the task

Of the five error types used in the shared task, four were already included in the grammar as used in the EPGY course, involving errors with articles/determiners, number on nouns, subject-verb agreement, and verb form. For the task, we added mal-rules and mal-entries to analyze a subset of errors of the fifth type, which involve incorrect use of prepositions. Within the ERG, each of the five error types is associated with multiple mal-rules or mal-entries, each licensing one specific error configuration, such as a mal-rule to accommodate the omission of an obligatory determiner for a noun phrase headed by a singular count noun, or a mal-entry for the unwanted use of *the* with a proper name.

Most of these grammar-internal error identifiers correspond to a simple adjustment for correction in the original sentence, such as the insertion or deletion of a particular token, or a change to the inflection of a particular noun or verb. However, for some errors, several candidate corrections are triggered by the error identifier, so the post-processing script must select the most suitable of these correction candidates. The most frequent correction illustrating this ambiguity is for singular count noun phrases missing the determiner, such as *black cat* in *we admired black cat.*, where the correction might be *the black cat*, *a black cat*, or *black cats*. Lacking a rich discourse representation of the context surrounding the error, we employ an N-gram based ranking approach to choose among the three alternatives, where the post-processor currently calls the Microsoft N-gram online resource (Wang et al., 2011).

Since the development and test data is presented as pre-tokenized input with one token per line in familiar CoNLL format, we also employ an offline script which converts a file of this format into one which has a single sentence per line, preserving the tokenization of the CoNLL file, and it is this one-sentence-per-line file which is processed by the correction script, which in turn calls the parser and applies the post-processing steps to its output.

3 An example

We illustrate our method with a simple example sentence, to show each step of the process. Consider the analysis in Figure 1 of the following sentence taken from the test corpus:

In supermarkets monitors is needed because we have to track thieves .

The parser is called with this sentence as input, constructs a packed forest of all candidate analyses licensed by the grammar, and identifies the most likely analysis as determined by a general-purpose statistical model trained only on analyses of well-formed sentences. A more detailed view of the parse tree in Figure 1 is the bracketed derivation tree given in (2). Each line of the derivation identifies the syntactic construction, lexical rule, or lexical entry used to build each constituent, and shows its token span, and for the leaf nodes, the lexical entry, its type (after the slash), and the surface form of that word in the input sentence. The boldface identifier on the first line of the derivation tree shows that this analysis contains at least one erroneous constituent, which a perusal of the tree locates as the other boldface identifier, *be_c_is_rbst*, for the mal-entry for *is* that licenses a mismatch in subject-verb agreement.

(2) Derivation tree view of Fig. 1:

```
hd-aj_scp_c 0 11 [ root_robust_s ]
 flr-hd_nwh-nc-pp_c 0 5
  hd-cmp_u_c 0 2
   in/p_np_i-reg 0 1 "in"
   hdn_bnp_c 1 2
    n_pl_olr 1 2
     supermarket_n1/n_-_c 1 2
      "supermarkets"
  sb-hd_nmc_c 2 5
   hdn_bnp_c 2 3
    n_pl_olr 2 3
     monitor_n1/n_-_c 2 3 "monitors"
   hd-cmp_u_c 3 5
    be_c_is_rbst 3 4  "is"
    hd_xaj-int-vp_c 4 5
     hd_optcmp_c 4 5
      v_pas_odlr 4 5
       need_v1/v_np 4 5 "needed"
 hd-cmp_u_c 5 11
  because/p_cp_s 5 6 "because"
  sb-hd_nmc_c 6 11
   hdn_bnp-qnt_c 6 7
    we/n_-_pr-we 6 7 "we"
   hd-cmp_u_c 7 11
    v_n3s-bse_ilr 7 8
     have_to1/v_vp_ssr 7 8 "have"
    hd-cmp_u_c 8 11
     to_c_prop/cm_vp_to 8 9 "to"
     hd-cmp_u_c 9 11
      v_n3s-bse_ilr 9 10
       track_v1/v_np* 9 10 "track"
      hdn_bnp_c 10 11
       period_plr 10 11
        n_pl_olr 10 11
         thief_n1/n_-_c 10 11 "thieves."
```

The correction script finds this mal-entry identifier in the derivation tree, notes its token position, and determines from the identifier that the required correction consists of a simple token substitution, replacing the surface token *is* with *are*. Since no other errors are present in the derivation tree, the script then records in the corpus output file the corrected sentence with only the one alteration from its original form.

Of course, a derivation tree will often identify multiple errors, and for some error types may require that multiple tokens be modified for a sin-

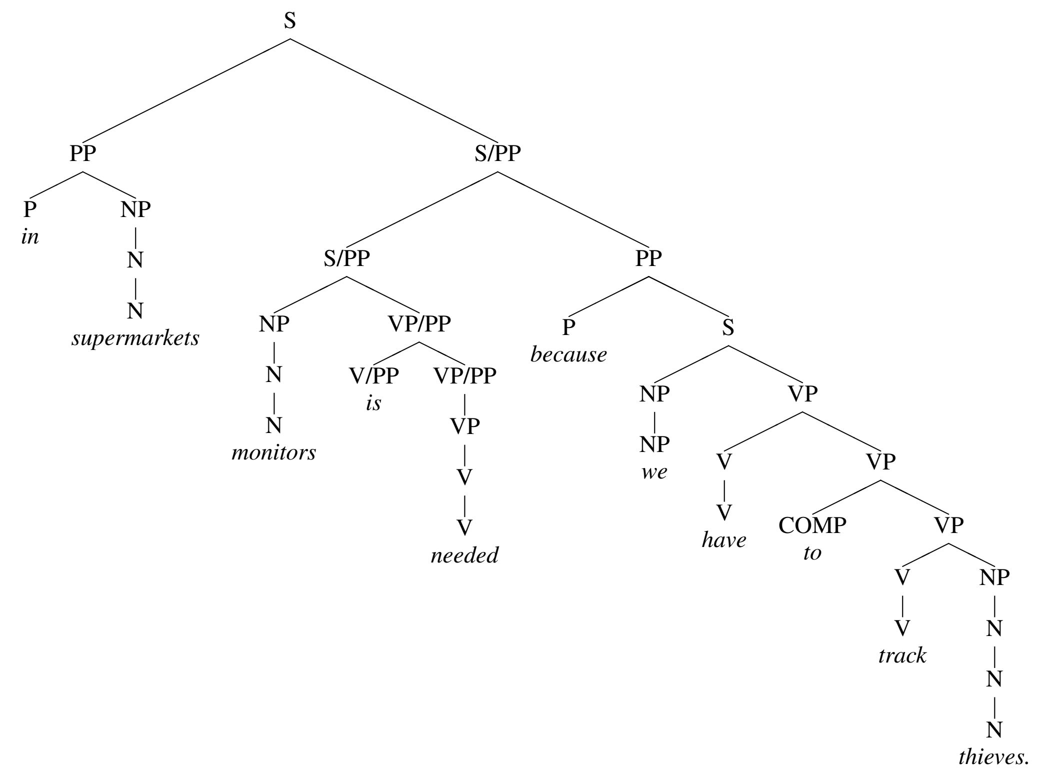

Figure 1: Sample parse tree produced with ERG

gle error, such as in the correction of *the equipments have arrived* to *the equipment has arrived*. Each mal-rule or mal-entry identifier is associated with a specific correction procedure defined in the correction script, and the script carries out these changes in a pre-determined order, for the relatively infrequent instances where the order of application matters. For simple alterations such as a change of number on nouns or verbs, we could have used the grammar-internal inflectional rule machinery, but found it more convenient to use existing Perl and Python modules for English word inflection.

4 Results and Discussion

During the development phase of the shared task, we adapted and refined our method using the first 5000 NUCLE sentences from the roughly 50,000-sentence development corpus. Since our focus in this task is on precision more than on recall, we carried out repeated detailed examinations of the correction procedure's outputs on the first 500 sentences. In comparing our system's proposed corrections with the 'gold' human annotations of errors for these 500, we found the following frequencies of mismatches between system and gold:

(3) Comparison of System and Gold on Dev-500

Alteration	# of Sentences
Both match	34
Missing gold	26
Differing correction	25
Wrong alteration	28

Examples of the missing gold annotations include (a) "ArtOrDet" errors such as the missing article for *habitable environment* in sentence 829-4-0 and for *password* in sentence 830-1-1; (b) "SVA" errors such as for the verb *increase* in sentence 831-3-8, and the verb *are* in sentence 840-4-2; and (c) "Nn" errors such as for the noun *equipments* appearing in sentence 836-1-0, or *evidences* in sentence 837-2-11.

These varying sources of mismatches made the automated scoring script used in the evaluation phase of the shared task (Dahlmeier and Ng, 2012) not so helpful during development, since it reported our system's precision as 28%, whereas the system is actually correct in more than 50% of the alterations it makes for these first 500 sentences of the development corpus.

This inconsistency in the gold annotations was less of an issue, but still present, in our system's precision measure in the evaluation phase of the shared task, as we found in studying the gold annotations distributed for the test data after the evaluation phase ended. The official scored results for the system output that we submitted are given in the table in (4).

(4) Official scoring of system output on test data

Precision	29.93%
Recall	5.86 %
F1	9.81 %

In examining the gold annotations for the 1381 sentences comprising the test corpus, we found 47 instances of genuine errors that were missing gold annotation, but that our system correctly identified and repaired. While this led to a somewhat lower precision measure, we acknowledge that compared with the total number of more than 1600 annotated corrections, this level of imperfection in the annotations was not seriously problematic for evaluation, and we view the official results in (4) as a reasonable measure of the system output we submitted for scoring.

While comparing our system results with the gold test annotations after the evaluation phase ended, we have found and repaired several sources of undesirable behavior in the grammar and in our correction script, with the most significant being the revision of lexical entries for two compound nouns appearing with high frequency in the test corpus: *life expectancy* (91 occurrences) and *population aging/ageing* (40 occurrences). Our lexicon had erroneously identified *life expectancy* as countable, and the parser had wrongly analyzed *population aging* as a noun modified by a participle, analogous to *the person speaking*. A third frequently occurring error in the corpus was not so simple to correct in our grammar, namely the word *society* (95 occurrences), which is used consistently in the test corpus as an abstract noun often wrongly appearing with *the*. Since this noun can be used in a different sense (as an organization) where the article is appropriate, as in *the society of wealthy patrons*, we would need to find some other knowledge source to determine that in the domain of the test corpus, this sense is not used. Hence our system still fails to identify and correct the frequent and spurious *the* in *the society*.

With the small number of corrections made to our system's lexicon, and some minor improvements to the post-processing script, our system now produces output on the test corpus with an improved precision measure of 47.5%, and a more modest improvement in recall to 13.2%, for an F1 of 20.7%. Given the inconsistency of annotation in the development corpus, it is as yet difficult to evaluate whether these changes to our correction script will result in corresponding improvements in precision on unseen data.

5 Next steps

We see prospects for significant improvement using the method we are developing for the kind of automatic correction studied in this shared task. Many of the missteps that our correction procedure makes can be traced to imperfect parse selection from among the candidate analyses produced by the parser, and this could well be improved by creating a Redwoods-style treebank that includes both well-formed and ill-formed sentences for annotation, so the mal-rules and mal-entries get included in the ranking model trained on such a treebank. While our primary focus will continue to be on increased precision in the corrections the system proposes, we welcome the attention to recall that this task brings, and expect to work with hybrid systems that do more with large-scale corpora such as the English Wikipedia.

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.

Thorsten Brants. 2000. TnT - A statistical part-of-speech tagger. In *Proceedings of the 6th ACL Conference on Applied Natural Language Processing*, Seattle, WA.

Ulrich Callmeier. 2002. Preprocessing and encoding techniques in PET. In Stephan Oepen, Daniel Flickinger, J. Tsujii, and Hans Uszkoreit, editors, *Collaborative Language Engineering. A Case Study in Efficient Grammar-based Processing*. CSLI Publications, Stanford, CA.

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, Montreal, Canada.

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 *To appear in Proceedings of the 8th Workshop on Innovative Use of NLP for Building Educational Applications*, Atlanta, Georgia.

Dan Flickinger, Stephan Oepen, and Gisle Ytrestøl. 2010. WikiWoods. Syntacto-semantic annotation for English Wikipedia. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*, Valletta, Malta.

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.

Jan Tore Lnning, Stephan Oepen, Dorothee Beermann, Lars Hellan, John Carroll, Helge Dyvik, Dan Flickinger, Janne Bondi Johannessen, Paul Meurer, Torbjrn Nordgrd, Victoria Rosn, and Erik Velldal. 2004. LOGON. A Norwegian MT effort. In *Proceedings of the Workshop in Recent Advances in Scandinavian Machine Translation*, Uppsala, Sweden.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. The University of Chicago Press and CSLI Publications, Chicago, IL, and Stanford, CA.

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.

Patrick Suppes, Dan Flickinger, Elizabeth Macken, Jeanette Cook, and L. Liang. 2012. Description of the EPGY Stanford University online courses for Mathematics and the Language Arts. In *Proceedings of the International Society for Technology in Education*, San Diego, California.

Kristina Toutanova, Christoper D. Manning, Dan Flickinger, and Stephan Oepen. 2005. Stochastic HPSG parse selection using the Redwoods corpus. *Journal of Research on Language and Computation*, 3(1):83 – 105.

Erik Velldal. 2008. *Empirical Realization Ranking*. Ph.D. thesis, University of Oslo, Department of Informatics.

Wolfgang Wahlster, editor. 2000. *Verbmobil. Foundations of Speech-to-Speech Translation*. Springer, Berlin, Germany.

Kuansan Wang, Christopher Thrasher, Evelyne Viegas, Xiaolong Li, , and Paul Hsu. 2011. An overview of Microsoft Web N-gram corpus and applications. In *Proceedings of the 2011 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon.