Precision Isn't Everything: A Hybrid Approach to Grammatical Error Detection

Michael Heilman and **Aoife Cahill** and **Joel Tetreault**
Educational Testing Service
660 Rosedale Road
Princeton, NJ 08541, USA
{mheilman,acahill,jtetreault}@ets.org

Abstract

Some grammatical error detection methods, including the ones currently used by the Educational Testing Service's e-rater system (Attali and Burstein, 2006), are tuned for precision because of the perceived high cost of false positives (i.e., marking fluent English as ungrammatical). Precision, however, is not optimal for all tasks, particularly the HOO 2012 Shared Task on grammatical errors, which uses F-score for evaluation. In this paper, we extend e-rater's preposition and determiner error detection modules with a large-scale n-gram method (Bergsma et al., 2009) that complements the existing rule-based and classifier-based methods. On the HOO 2012 Shared Task, the hybrid method performed better than its component methods in terms of F-score, and it was competitive with submissions from other HOO 2012 participants.

1 Introduction

The detection of grammatical errors is a challenging problem that, arguably, requires the use of both linguistic knowledge (e.g., in the form of rules or complex features) and large corpora for statistical learning. Additionally, grammatical error detection can be applied in various scenarios (e.g., automated essay scoring, writing assistance, language learning), many of which may benefit from task-specific adaptation or tuning. For example, one might want to take a different approach when detecting errors for the purpose of providing feedback than when detecting errors to evaluate the quality of writing in an essay. Thus, it seems desirable to take a flexible approach to grammatical error detection that incorporates multiple, complementary techniques.

In this paper, we extend the preposition and determiner error detection modules currently used in the Educational Testing Service's e-rater automated essay scoring system (Attali and Burstein, 2006) for the HOO 2012 Shared Task on grammatical errors (§2). We refer to this set of modules from e-rater as our "base system" (§3). While the base system uses statistical methods to learn models of grammatical English, it also leverages substantial amounts of linguistic knowledge in the form of various hand-coded filters and complex syntactic features. The base system is also tuned for high precision at the expense of recall in order to avoid a high rate of potentially costly false positives (i.e., frequent marking of correct English sentences as ungrammatical).

We apply the pre-existing base system without modifications but complement it with a large-scale n-gram method (§5) based on work by Bergsma et al. (2009). The n-gram method employs very little linguistic knowledge and instead relies almost exclusively upon corpus statistics. We also tune the resulting hybrid system with labeled training data in order to maximize the primary evaluation metric used in the HOO 2012 Shared Task: balanced F-score, or F_1 (§6). We find that the tuned hybrid system improves upon the recall and F-score of the base system. Also, in the HOO 2012 Shared Task, the hybrid system achieved results that were competitive with other submitted grammatical error detection systems (§7).

2 Task Definition

In this section, we provide a brief overview of the HOO 2012 Shared Task (Dale et al., 2012). The task focuses on prepositions and determiners only, distinguishing the following error types: preposition selection errors (coded "RT" in the data), extraneous prepositions ("UT"), missing prepositions ("MT"), determiner selection errors ("RD"), extraneous determiners ("UD"), and missing determiners ("MD").

For training and testing data, the shared task uses short essays from an examination for speakers of English as a foreign language. The data includes gold standard human annotations identifying preposition and determiner errors. These errors are represented as **edits** that transform an ungrammatical text into a grammatical one. Edits consist of start and end offsets into the original text and a correction string that should replace the original text at the specified offsets. The offsets differ by error type: word selection errors include just the word, extraneous word errors include an extra space after the word so that a blank will result in an appropriate amount of whitespace, and missing word errors specify spans of length zero.[1]

There are three subtasks: detection, recognition, and correction. Each is evaluated according to precision, recall, and F-score according to a set of gold standard edits produced by human annotation. While the correction subtask requires both correct character offsets and appropriate corrections, the detection and recognition subtasks only consider the offsets. Detection and recognition are essentially the same, except that detection allows for loose matching of offsets, which permits mismatches between the extraneous use (e.g., UT) and word selection (e.g., RT) error types. For our submission to the shared task, we chose to tune for the detection subtask, and we also chose to avoid the correction task entirely since the interface to the pre-existing base system did not give us access to possible corrections.

[1]The offsets for extraneous word errors prior to punctuation, a relatively rare occurrence, include a space before the word rather than after it. Our script for converting our system's output into the HOO 2012 format did not account for this, which may have decreased recognition performance slightly.

3 Base System

As our base system, we repurpose a complex system designed to automatically score student essays (both native and non-native and across a wide range of competency levels). The system is also used to give feedback to essay writers, so precision is favored over recall. There are three main modules in the essay-scoring system whose purpose it is to detect preposition and determiner errors (as they are defined in that system). Many of the details have been reported previously (Chodorow and Leacock, 2000; Han et al., 2004; Han et al., 2006; Chodorow et al., 2007; Tetreault and Chodorow, 2008), so here we will only give brief summaries of these modules.

It is important to note that this system was run without modification. That is, no training of new models or tuning was carried out specifically for the shared task. In addition, for the two statistical modules, we only had access to the final, boolean decisions about whether an error is present or not at a particular location in text. That is, we did not have access to confidence scores, and so task-specific tuning for F-score was not an option.

3.1 Preposition Error Detection

The base system detects incorrect and extraneous prepositions (Chodorow et al., 2007; Tetreault and Chodorow, 2008). Tetreault and Chodorow (2008) reports approximately 84% precision and 19% recall on both error types combined when evaluating the system on manually annotated non-native text.

3.1.1 Incorrect Prepositions

The module to detect incorrectly used prepositions consists of a multiclass logistic regression (i.e., "Maximum Entropy") model of grammatical usage, along with heuristic pre- and post- filters. The module works by extracting a set of features from the "context" around a preposition, generating a distribution over possible prepositions using the model of grammatical usage, and then flagging an error if the difference in probability between the text's original preposition and an alternative preposition exceeds a certain threshold. The probability for any correction also needs to exceed another minimum threshold. For this work, we used the pre-existing, manually-set thresholds.

A pre-filter prevents any contexts that contain spelling errors from being submitted to the logistic regression model. The motivation for this is that the NLP components that provide the features for the model are unreliable on such data, and since the systems favors precision over recall, no attempt is made to correct prepositions where the system cannot rely on the accuracy of those features.

The logistic regression model of correct preposition usage is trained on approximately 82 million words from the San Jose Mercury News[2] and texts for 11th to 12th grade reading levels from the MetaMetrics Lexile corpus, resulting in 7 million preposition contexts. The model uses 25 types of features: words and part-of-speech tags around the existing preposition, head verb (or noun) in the preceding VP (or NP), head noun in the following NP, among others. NPs and VPs were detected using chunking rather than full parsing, as the performance of statistical parsers on erroneous text was deemed to be too poor.

A post-filter rules out certain candidates based on the following heuristics: (1) if the suggested correction is an antonym of the original preposition (e.g., *from* vs *to*), it is discarded; (2) any correction of the benefactive *for* is discarded when the head noun of the following NP is human (detected as a WordNet hyponym of *person* or *group*).

3.1.2 Extraneous Prepositions

Heuristics are applied to detect common occurrences of extraneous prepositions in two scenarios: (1) accidentally repeated prepositions (e.g., *with with*) and (2) insertion of unnecessary prepositions in plural quantifier constructions (e.g., *some **of** people*).

3.2 Determiner Error Detection

There are two separate components that detect errors related to determiners. The first is a filter-based model that detects determiner errors involving number and person agreement. The second is a statistical system that supplements the rule-based system and detects article errors.

[2]The San Jose Mercury News is available from the Linguistic Data Consortium (catalog number LDC93T3A).

3.2.1 Filter-based system

The filter-based system combines unsupervised detection of a set of possible errors (Chodorow and Leacock, 2000) with hand-crafted filters designed to reduce this set to the largest subset of correctly flagged errors and the smallest possible number of false positives. Chodorow and Leacock (2000) found that low-frequency bigrams (sequences of two lexical categories with a negative log-likelihood) are quite reliable predictors of grammatical errors. Text is tagged and chunked, and filters that detect likely cases of NP-internal agreement violations are applied. These filters will mark, for example, a singular determiner followed by a plural noun head and vice versa, or a number disagreement between a numeral and the noun it modifies. This system has the ability to take advantage of linguistic knowledge, which contributes to its ability to detect errors with high precision.

3.2.2 Statistical model

In addition to the hand-crafted filters described above, there is a statistical component that detects incorrect, missing and extraneous articles (Han et al., 2004; Han et al., 2006). This component consists of a multiclass logistic regression that selects an appropriate article for every NP from *a, an, the*, or ϵ. This model is trained on 31.5 million words of diverse genres from the MetaMetrics Lexile corpus (from 10th to 12th grade reading levels), or 8 million NP contexts. Again, NPs were determined by chunking. The model includes various features: words and POS tags around and within the NP, NP head information including the countability of the head noun (estimated automatically from large corpora), etc.

In a cross-validation experiment, the model achieved approximately 83% accuracy on well-edited text. In an experiment evaluated on non-native learner text, the model achieved approximately 85% agreement with human annotators.

4 Task-Specific Heuristic Filtering

There is not a one-to-one mapping between the definitions of determiner and preposition errors as used in the HOO data set and the definitions used in our base system. For example, our base system marks errors involving *every*, *many* and other quantifiers as determiner errors, while these are not marked in the current HOO 2012 Shared Task data.

To ensure that our system was aligned with the HOO 2012 Shared Task, we automatically extracted lists of the most frequently occurring determiners and prepositions in the HOO training data. Any RT, UT, RD or UD edit predicted for a word not in those lists is automatically discarded. In the training data, this resulted in the removal of 4 of the 463 RT errors and 98 of the 361 RD errors detected by the base system.

5 Large-scale n-Gram Models

In order to complement the high-precision base system and increase recall, we incorporate a large scale n-gram model into our full system. Specifically, we adapt the SUMLM method from Bergsma et al. (2009). SUMLM creates confusion sets for each preposition token in an input text and uses the Google Web 1T 5-gram Corpus to score each item in the confusion set.[3] We extend SUMLM to support determiners, extraneous use errors, and missing word errors.

Consider the case of preposition selection errors. For a preposition token at position i in an input sentence $\mathbf{w}$, we compute the following score for each possible alternative v, using Eq. 1.[4]

$$\mathrm{s}(\mathbf{w}, i, v) = \frac{\sum\limits_{n=2\ldots5} \sum\limits_{x \in \mathrm{G}(\mathbf{w},i,n,v)} \log(\mathrm{count}(x))}{|\mathrm{G}(\mathbf{w}, i, n, v)|} \tag{1}$$

The function $\mathrm{G}(\mathbf{w}, i, n, v)$ returns the set of n-grams in $\mathbf{w}$ that include the word at position i and replace that word, w_i, with v. For example, if $\mathbf{w} =$ *Mary and John went at the store to buy milk*, $n = 4$, $i = 4$, and $v =$ *to*, then $\mathrm{G}(\mathbf{w}, i, n, v)$ returns the following 4-grams:

- *and John went to*
- *John went to the*
- *went to the store*
- *to the store to*

The expression $\log(\mathrm{count}(x))$ is the natural logarithm of the number of times the n-gram x occurred in the corpus.[5] $|\mathrm{G}(\mathbf{w}, i, n, v)|$ is the number of n-gram count lookups, used to normalize the scores. Note that this normalization factor is not included in the original SumLM. When v is an alternative preposition not near the beginning or end of a sentence, $|\mathrm{G}(\mathbf{w}, i, n, v)| = 14$ since there are 14 n-gram count lookups in the numerator. Or, for example, if $i = 0$, indicating that the preposition occurs at the beginning of the sentence, $|\mathrm{G}(\mathbf{w}, i, n, v)| = 4$.[6]

Next, we compute the ratio of the score of each alternative to the score for the original, using Eq. 2.

$$\mathrm{r}(\mathbf{w}, i, v) = \frac{\mathrm{s}(\mathbf{w}, i, v)}{\mathrm{s}(\mathbf{w}, i, w_i)} \tag{2}$$

We then identify the best scoring alternative, requiring that its score be higher than the original (i.e., $\mathrm{r}(\mathbf{w}, i, v) > 1$). The procedure is the same for determiners, except, of course, that the set of alternatives includes determiners rather than prepositions.

To extend the method from Bergsma et al. (2009) for extraneous prepositions and determiners, we simply set v to be a blank and sum over $j = 3 \ldots 5$ instead. $|\mathrm{G}(\mathbf{w}, i, n, v)|$ will then be 12 instead of 14, since bigrams from the original sentence, which become unigrams when replacing w_i with a blank, are excluded.

To identify positions at which to flag selection or extraneous use errors, we simply scan for words that match an item in our sets of possible prepositions and determiners. To extend the method for missing

[3]The Google Web 1T 5-gram Corpus is available from the Linguistic Data Consortium (catalog number LDC2006T13). We plan to test other corpora for n-gram counts in future work.

[4]The n-gram approach considers all of the following words to be prepositions: *to*, *of*, *in*, *for*, *on*, *with*, *at*, *by*, *as*, *from*, *about*, *up*, *over*, *into*, *down*, *between*, *off*, *during*, *under*, *through*, *around*, *among*, *until*, *without*, *along*, *within*, *outside*, *toward*, *inside*, *upon*, *except*, *onto*, *towards*, *besides*, *beside*, and *underneath*. It considers all of the following words to be determiners: *a*, *an*, and *the*. The sets of possible prepositions and determiners for the base system are not exactly the same. Part of speech tags are not used in the n-gram system except to identify insertion points for missing prepositions and determiners.

[5]We use the TrendStream system (Flor, 2012) to retrieve n-gram counts efficiently.

[6]Our n-gram counts do not include start- or end-of-sentence symbols. Also, all n-grams are case-normalized with numbers replaced by a special symbol.

Algorithm 1 `tune`($\mathbf{W}$, $\mathbf{y}$, $\hat{\mathbf{y}}$ α, α_{min}):
The hill-climbing algorithm for optimizing the n-gram method's penalty parameters $\mathbf{q}$. $\mathbf{W}$ consists of the training set texts. $\hat{\mathbf{y}}$ is a set of candidate edits. $\mathbf{y}$ is a set of gold standard edits. α is an initial step size, and α_{min} is a minimum step size.

```
$\mathbf{q}_{allbest} \leftarrow \mathbf{0}$
$\text{score}_{allbest} \leftarrow \texttt{eval}(\mathbf{q}_{allbest}, \mathbf{W}, \mathbf{y}, \hat{\mathbf{y}})$
while $\alpha > \alpha_{min}$ do
  $\text{score}_{best} \leftarrow -\infty$
  $\mathbf{q}_{best} \leftarrow \mathbf{q}_{allbest}$
  for $\mathbf{q}_{tmp} \in \texttt{perturb}(\mathbf{q}_{best}, \alpha)$ do
    $\text{score}_{tmp} \leftarrow \texttt{eval}(\mathbf{q}_{tmp}, \mathbf{W}, \mathbf{y}, \hat{\mathbf{y}})$
    if $\text{score}_{tmp} > \text{score}_{best}$ then
      $\mathbf{q}_{best} \leftarrow \mathbf{q}_{tmp}$
      $\text{score}_{best} \leftarrow \text{score}_{tmp}$
    end if
  end for
  if $\text{score}_{best} > \text{score}_{allbest}$ then
    $\mathbf{q}_{allbest} \leftarrow q_{best}$
    $\text{score}_{allbest} \leftarrow \text{score}_{best}$
  else
    $\alpha \leftarrow 0.5 * \alpha$
  end if
end while
return $\mathbf{q}_{allbest}$
```

word errors, however, we apply a set of heuristics to identify potential insertion points.[7]

6 Tuning

The n-gram approach in §5 generates a large number of possible edits of different types. In this section, we describe how we filter edits using their scores and how we combine them with edits from the base system (§3).

As described above, for an alternative v to be considered as a candidate edit, the value of $\text{r}(\mathbf{w}, i, v)$ in Eq. 2 must be greater than a threshold of 1, indicating that the alternative scores higher than the original word. However, we observed low precision during development when including all candidate edits and decided to penalize the ratios. Bergsma et al. (2009) discuss raising the threshold, which has a similar effect. Preliminary experiments indicated that different edits (e.g., extraneous preposition edits and preposition selection edits) should have different penalties, and we also want to avoid edits with overlapping spans. Thus, for each location with one or more candidate edits, we select the best according to Equation 3 and filter out the rest.

$$v^* = \arg\max_v \quad \text{r}(\mathbf{w}, i, v) - \text{penalty}(w_i, v) \tag{3}$$

$\text{penalty}(w_i, v)$ is a function that takes the current word w_i and the alternative v and returns one of 6 values: q_{RT} for preposition selection, q_{UT} for extraneous prepositions, q_{MT} for missing prepositions, q_{RD} for determiner selection, q_{UD} for extraneous determiners, and q_{MD} for missing determiners.

If the value for $\text{r}(\mathbf{w}, i, v^*) - \text{penalty}(w_i, v^*)$ does not exceed 1, we exclude it from the output.

We tune the vector $\mathbf{q}$ of all the penalties to optimize our objective function (F-score, see §7) on the training set using the hill-climbing approach described in Algorithm 1. The algorithm initializes the parameter vector to all zeros, and then iteratively evaluates candidate parameter vectors that result from taking positive and negative steps of size α in each direction (steps with negative penalties are skipped). The best step is taken if it improves the current score, according to the `eval` function, which returns the training set F-score after filtering based on the current parameters.[8] This process proceeds until there is no improvement. Then, the step size α is halved, and the whole process is repeated. The algorithm proceeds as such until the step size becomes lower than a specified minimum α_{min}.

When merging edits from the base system and the n-gram approach, the hybrid system always prefers edits from the base system if any edit spans overlap, equivalent to including them in Eq. 3 and assigning them a penalty of $-\infty$.[9] Note that the set of predicted edits $\mathbf{y}$ passed as input to the `tune` algorithm

[7]The heuristics are based on those used in Gamon (2010) (personal communication).

[8]Our implementation of the tuning algorithm uses the HOO 2012 Shared Task's `evalfrag.py` module to evaluate the F-score for the error detection subtask.

[9]If the base system produces overlapping edits, we keep them all. If there are overlapping edits from the n-gram system that have the same highest value for the penalized score in Equation 3 and do not overlap with any base system edits, we keep them all.

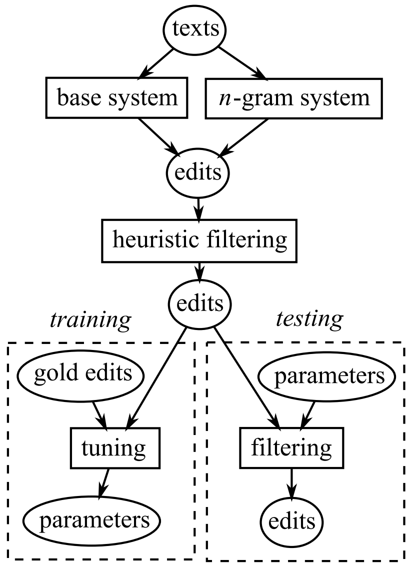

Figure 1: The architecture of the hybrid system. Different steps are discussed in different parts of the paper: "base system" in §3, "n-gram system" in §5, "heuristic filtering" in §4, and "tuning" and "filtering" in §6.

	run	**P**	**R**	**F**
base	0	52.63	17.66	26.45
n-gram	–	25.87	37.75	30.70
hybrid	1	33.59	37.97	35.65
hybrid$_{indep}$	2	24.88	44.15	31.82
UI	8	37.22	43.71	40.20

Table 1: Precision, recall, and F-score for the combined preposition and determiner error detection subtask for various methods, before participant-suggested revisions to the gold standard were applied. All values are percentages. Official run numbers are shown in the "run" column. The "n-gram" run was not part of our official submission. For comparison, "UI" is the submission, from another team, that achieved the highest detection F-score in the HOO 2012 Shared Task.

includes edits from both the base and n-gram methods.

Figure 1 illustrates the processes of training and of producing test output from the hybrid system.

7 Results

Table 1 presents results for the HOO 2012 detection subtask, including errors of all types. The results here, reproduced from Dale et al. (2012), are prior to applying participant-suggested revisions to the set of gold standard edits.[10] We include four variations of our approach: the base system (§3, labeled "base"); the n-gram system (§5, labeled "n-gram") by itself, tuned without edits from the base system; the hybrid system, tuned with edits from the base system ("hybrid"); and a variation of the hybrid system ("hybrid$_{indep}$") with the penalties tuned independently, rather than jointly, to maximize F-score for detection of each error type. For comparison, we also include the best performing run for the detection subtask in terms of F-score (labeled "UI").

We observe that the base and n-gram systems appear to complement each other well for this task: the base system achieved 26.45% F-score, and the n-gram system achieved 30.70%, while the hybrid system, with penalties tuned jointly, achieved 35.65%. Table 2 shows further evidence that the two systems have complementary performance. We calculate the overlap between each system's edits and the gold standard. We see that only a small number of edits are predicted by both systems (38 in total, 18 correct and 20 incorrect), and that the base system predicts 62 correct edits that the n-gram method does not predict, and similarly the n-gram method predicts 92 correct edits that the base system does not predict. The table also verifies that the base system exhibits high precision (only 68 false positives in total) while the n-gram system is tuned for higher recall (286 false positives).

Not surprisingly, when the n-gram method's penalties were tuned independently ("hybrid$_{indep}$") rather than jointly, the overall score was lower, at 31.82% F-score. However, tuning independently might be desirable if one were concerned with performance on specific error types or if macro-averaged F-score were the objective.

The hybrid system performed quite competitively

[10] After submitting our predictions for the shared task, we noticed a few minor implementation mistakes in our code related to the conversion of edits from the base system (§3) and the task-specific heuristic filtering (§4). We corrected them and retrained our system. The detection F-scores for the original and corrected implementations were as follows: 26.45% (original) versus 26.23% (corrected) for the base system, 30.70% (original) versus 30.45% (corrected) for the n-gram system, 35.65% (original) versus 35.24% (corrected) for the hybrid system, and 31.82% (original) versus 31.45% (corrected) for the hybrid$_{indep}$ system. Except for this footnote, all results in this paper are for the original system.

(1) All models had $\boxed{\text{a}}_{\text{UD}}$ very strange long shoes made from black skin …

(2) I think it is a great idea to organise this sort of festival because most $\boxed{\text{of}}_{\text{UT}}$ people enjoy it.

Figure 2: Examples of errors detected by the base system and missed by the n-gram models.

(3) We have to buy $\boxed{\text{for}}_{\text{UT}}$ some thing.

(4) I am $\boxed{\epsilon}_{\text{MD}}$ good diffender.

Figure 3: Examples of errors detected by the n-gram system and missed by the base model.

	$\in$ gold		$\notin$ gold	
	$\in$ base	$\notin$ base	$\in$ base	$\notin$ base
$\in$ n-gram	18	92	20	266
$\notin$ n-gram	62	276	48	—

Table 2: The numbers of edits that overlap in the hybrid system's output and the gold standard for the test set. The hybrid system's output is broken down by whether edits came from the base system (§3) or the n-gram method (§5). The empty cell corresponds to hypothetical edits that were in neither the gold standard or the system's output (e.g., edits missed by annotators), which we cannot count.

compared to the other HOO 2012 submissions, achieving the 3rd best results out of 14 teams for the detection and recognition subtasks. The performance of the "UI" system was somewhat higher, however, at 40.20% F-score compared to the hybrid system's 35.65%. We speculate that our hybrid system's performance could be improved somewhat if we also tuned the base system for the task.

8 Error Analysis

It is illustrative to look at some examples of edits that the base system correctly detects but the n-gram model does not, and vice versa. Figure 2 shows examples of errors detected by the base system, but missed by the n-gram system. Example (1) illustrates that the n-gram model has no concept of syntactic structure. The base system, on the other hand, carries out simple processing including POS tagging and chunking, and is therefore aware of at least some longer-distance dependencies (e.g., *a … shoes*). Example (2) shows the effectiveness of the heuristics based on quantifier constructions mentioned in §3.1.2. These heuristics were developed by developers familiar with the kinds of errors that language learners frequently make, and are therefore more targeted than the general n-gram method.

Figure 3 shows examples of errors detected by the n-gram system but missed by the base system. Example (3) shows an example of where the base system does not detect the extraneous preposition because it only searches for these in certain quantifier constructions. Example (4) contains a spelling error, which confuses the determiner error detection system. It has not seen the misspelling often enough to be able to reliably judge whether it needs an article or not before it, and so errs on the side of caution. When *diffender* is correctly spelled as *defender*, the base system does detect that there is a missing article in the sentence.

There were a small number of cases where dialect caused a mismatch between our system's error predictions and the gold standard. For example, *an hotel* is not marked as an error in the gold standard since it is correct in many dialects. However, it was always corrected to *a hotel* by our system. Our system also often corrected determiners before the noun *camp*, since in American Standard English it is more usual to talk about *going to Summer Camp* rather than *going to a/the Summer Camp*.

Although the task was to detect preposition and determiner errors in isolation, there was sometimes interference from other errors in the sentence. This impacted the task in two ways. Firstly, in a sentence with multiple errors, it was sometimes possible to correct it in multiple ways, not all of which involved preposition or determiner errors. For example, you could correct the phrase *a women* by either changing the *a* to *the*, deleting the *a* entirely or replacing *women* with *woman*. The last change would not fall under the category of determiner error, and so there was sometimes a mismatch between the corrections predicted by the system and the gold standard corrections. Secondly, the presence of multiple errors impacted the task when a gold standard correction depended on another error in the same sentence being corrected in a particular way. For example, you could correct *I'm really excited to read the book.* as *I'm really excited about reading the book.*, however if you add the preposition *about* without correcting *to read* this correction results in the sentence becoming even more ungrammatical than the original.[11]

9 Conclusion

In this paper, we have described a hybrid system for grammatical error detection that combines a pre-existing base system, which leverages detailed linguistic knowledge and produces high-precision output, with a large-scale n-gram approach, which relies almost exclusively on simple counting of n-grams in a massive corpus. Though the base system was not tuned at all for the HOO 2012 Shared Task, it performed well in the official evaluation. The two methods also complemented each other well: many of the predictions from one did not appear in the output of the other, and the F-score of the hybrid system was considerably higher than the scores for the individual methods.

Acknowledgments

We thank Martin Chodorow for discussions about the base system, Daniel Blanchard for help with running the base system, Nitin Madnani for discussions about the paper and for its title, and Michael Flor for the TrendStream system.

[11]Many of these cases were addressed in the revised version of the gold standard data, however we feel that the issue is a more general one and deserves consideration in the design of future tasks.

References

Yigal Attali and Jill Burstein. 2006. Automated Essay Scoring with e-rater V.2. *Journal of Technology, Learning, and Assessment*, 4(3). Available from http://www.jtla.org.

Shane Bergsma, Dekang Lin, and Randy Goebel. 2009. Web-Scale N-gram Models for Lexical Disambiguation. In *Proceedings of the 21st international joint conference on Artifical intelligence*, IJCAI'09, pages 1507–1512, Pasadena, California. Morgan Kaufmann Publishers Inc.

Martin Chodorow and Claudia Leacock. 2000. An Unsupervised Method for Detecting Grammatical Errors. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 140–147, Seattle, Washington. Association for Computational Linguistics.

Martin Chodorow, Joel Tetreault, and Na-Rae Han. 2007. Detection of Grammatical Errors Involving Prepositions. In *Proceedings of the Fourth ACL-SIGSEM Workshop on Prepositions*, pages 25–30, Prague, Czech Republic. Association for Computational Linguistics.

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. HOO 2012: A Report on the Preposition and Determiner Error Correction Shared Task. In *Proceedings of the Seventh Workshop on Innovative Use of NLP for Building Educational Applications*, Montreal, Canada. Association for Computational Linguistics.

Michael Flor. 2012. A fast and flexible architecture for very large word n-gram datasets. *Natural Language Engineering*, pages 1–33. doi:10.1017/S135132491100034.

Michael Gamon. 2010. Using Mostly Native Data to Correct Errors in Learners' Writing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 163–171, Los Angeles, California. Association for Computational Linguistics.

Na-Rae Han, Martin Chodorow, and Claudia Leacock. 2004. Detecting Errors in English Article Usage with a Maximum Entropy Classifier Trained on a Large, Diverse Corpus. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, pages 1625–1628, Lisbon, Portugal.

Na-Rae Han, Martin Chodorow, and Claudia Leacock. 2006. Detecting errors in English article usage by non-native speakers. *Natural Language Engineering*, 12:115–129. doi:10.1017/S1351324906004190.

Joel R. Tetreault and Martin Chodorow. 2008. The Ups and Downs of Preposition Error Detection in

ESL Writing. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 865–872, Manchester, UK. Coling 2008 Organizing Committee.