Data Driven Grammatical Error Detection in Transcripts of Children's Speech

Eric Morley
CSLU
OHSU
Portland, OR 97239
morleye@gmail.com

Anna Eva Hallin
Department of Communicative Sciences and Disorders
New York University
New York, NY
ae.hallin@nyu.edu

Brian Roark
Google Research
New York, NY 10011
roarkbr@gmail.com

Abstract

We investigate grammatical error detection in spoken language, and present a data-driven method to train a dependency parser to automatically identify and label grammatical errors. This method is agnostic to the label set used, and the only manual annotations needed for training are grammatical error labels. We find that the proposed system is robust to disfluencies, so that a separate stage to elide disfluencies is not required. The proposed system outperforms two baseline systems on two different corpora that use different sets of error tags. It is able to identify utterances with grammatical errors with an F1-score as high as 0.623, as compared to a baseline F1 of 0.350 on the same data.

1 Introduction

Research into automatic grammatical error detection has primarily been motivated by the task of providing feedback to writers, whether they be native speakers of a language or second language learners. Grammatical error detection, however, is also useful in the clinical domain, for example, to assess a child's ability to produce grammatical language. At present, clinicians and researchers into child language must manually identify and classify particular kinds of grammatical errors in transcripts of children's speech if they wish to assess particular aspects of the child's linguistic ability from a sample of spoken language. Such manual annotation, which is called *language sample analysis* in the clinical field, is expensive, hindering its widespread adoption. Manual annotations may also be inconsistent, particularly between different research groups, which may be investigating different phenomena. Automated grammatical error detection has the potential to address both of these issues, being both cheap and consistent.

Aside from performance, there are at least two key requirements for a grammatical error detector to be useful in a clinical setting: 1) it must be able to handle spoken language, and 2) it must be trainable. Clinical data typically consists of transcripts of spoken language, rather than formal written language. As a result, a system must be prepared to handle disfluencies, utterance fragments, and other phenomena that are entirely grammatical in speech, but not in writing. On the other hand, a system designed for transcripts of speech does not need to identify errors specific to written language such as punctuation or spelling mistakes. Furthermore, a system designed for clinical data must be able to handle language produced by children who may have atypical language due to a developmental disorder, and therefore may produce grammatical errors that would be unexpected in written language. A grammatical error detector appropriate for a clinical setting must also be trainable because different groups of clinicians may wish to investigate different phenomena, and will therefore prefer different annotation standards. This is quite different from grammatical error detectors for written language, which may have models for different domains, but which are not typically designed to enable the detection of novel error sets.

We examine two baseline techniques for grammatical error detection, then present a simple data-driven technique to turn a dependency parser into a grammatical error detector. Interestingly, we find that the dependency parser-based approach massively outperforms the baseline systems in terms of identifying ungrammatical utterances. Furthermore, the proposed system is able to identify specific error codes, which the baseline systems cannot do. We find that disfluencies do not degrade performance of the proposed detector, obviating the need (for this task) for explicit disfluency detection. We also analyze the output of our system to see which errors it finds, and which it misses.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 980–989,
October 25-29, 2014, Doha, Qatar.

Code	Description	Example
[EO]	Overgeneralization errors	He falled [EO] .
[EW]	Other word level errors	He were [EW] looking .
[EU]	Utterance level errors	And they came to stopped .
[OM]	Omitted bound morpheme	He go [OM] .
[OW]	Omitted word	She [OW] running .

Table 1: Error codes proposed in the SALT manual. Note that in SALT annotated transcripts, [OM] and [OW] are actually indicated by '*' followed by the morpheme or word hypothesized to be omitted. When treating codes (other than [EU]) as tags, they are attached to the previous word in the string.

Finally, we evaluate our detector on a second set of data with a different label set and annotation standards. Although our proposed system does not perform as well on the second data set, it still outperforms both baseline systems. One interesting difference between the two data sets, which does appear to impact performance, is that the latter set more strictly follows SALT guidelines (see Section 2.1) to collapse multiple errors into a single label. This yields transcripts with a granularity of labeling somewhat less amenable to automation, to the extent that labels are fewer and can be reliant on non-local context for aggregation.

2 Background

2.1 Systematic Analysis of Language Transcripts (SALT)

The Systematic Analysis of Language Transcripts (SALT) is the de facto standard for clinicians looking to analyze samples of natural language. The SALT manual includes guidelines for transcription, as well as three types of annotations, of which two are relevant here: *maze* annotations, and *error codes*.[1]

Mazes are similar to what is referred to as 'disfluencies' in the speech literature. The SALT manual defines mazes as "filled pauses, false starts, repetitions, reformulations, and interjections" (Miller et al., 2011, p. 6), without defining any of these terms. Partial words, which are included and marked in SALT-annotated transcripts, are also included in mazes. Mazes are delimited by parentheses, and have no internal structure, unlike disfluencies annotated following the Switchboard guidelines (Meteer et al., 1995), which are commonly followed by the speech and language processing communities. An example maze annotation would be: "He (can not) can not get up."

The SALT manual proposes the set of error codes shown (with examples) in Table 1, but research groups may use a subset of these codes, or augment them with additional codes. For example, the SALT-annotated Edmonton Narrative Norms Instrument (ENNI) corpus (Schneider et al., 2006) rarely annotates omitted morphemes ([OM]), instead using the [EW] code. Other SALT-annotated corpora include errors that are not described in the SALT manual. For example the CSLU ADOS corpus (Van Santen et al., 2010) includes the [EX] tag for extraneous words, and the Narrative Story Retell corpus (SALT Software, 2014b) uses the code [EP] to indicate pronominal errors (albeit inconsistently, as many such errors are coded as [EW] in this corpus). We note that the definitions of certain SALT errors, notably [EW] and [EU], are open to interpretation, and that these codes capture a wide variety of errors. For example, some of the errors captured by the [EW] code are: pronominal case and gender errors; verb tense errors; confusing 'a' and 'an'; and using the wrong preposition.

The SALT guidelines specify as a general rule that annotators should not mark utterances with more than two omissions ([OM] or [OW]) and/or word-level errors (ex [EW], [EP]) (SALT Software, 2014a). Instead, annotators are instructed to code such utterances with an utterance-level error ([EU]). How strictly annotators adhere to this rule affects the distribution of errors, reducing the number of word-level errors and increasing the number of utterance-level errors. Following this rule also increases the variety of errors captured by the [EU] code. The annotations in different corpora, including ENNI and NSR, vary in how strictly they follow this rule, even though this is not mentioned in the the published descriptions of

[1] SALT also prescribes annotation of bound morphemes and clitics, for example -ed in past tense verbs. We preprocess all of the transcripts to remove bound morpheme and clitic annotations.

these corpora.

2.2 Grammatical Error Detection

The most visible fruits of research into grammatical error detection are the spellchecking and grammar checking tools commonly included with word processors, for example Microsoft Word's grammar checker. Although developed for handling written language, many of the techniques used to address these tasks could still be applicable to transcripts of speech because many of the same errors can still occur. The earliest grammaticality tools simply performed pattern matching (Macdonald et al., 1982), but this approach is not robust enough to identify many types of errors, and pattern matching systems are not trainable, and therefore cannot be adapted quickly to new label sets. Subsequent efforts to create grammaticality classifiers and detectors leveraged information extracted from parsers (Heidorn et al., 1982) and language models (Atwell, 1987). These systems, however, were developed for formal written English produced by well-educated adults, as opposed to spoken English produced by young children, particularly children with suspected developmental delays.

There have been a few investigations into techniques to automatically identify particular constructions in transcripts of spoken English. Bowden and Fox (2002) proposed a rule-based system to classify many types of errors made by learners of English. Although their system could be used on either transcripts of speech, or on written English, they did not evaluate their system in any way. Caines and Buttery (2010) use a logistic regression model to identify the zero-auxiliary construction (e.g., 'you going home?') with over 96% accuracy. Even though the zero-auxilliary construction is not necessarily ungrammatical, identifying such constructions may be useful as a preprocessing step to a grammaticality classifier. Caines and Buttery also demonstrate that their detector can be integrated into a statistical parser yielding improved performance, although they are vague about the nature of the parse improvement (see Caines and Buttery, 2010, p. 6).

Hassanali and Liu (2011) conducted the first investigation into grammaticality detection and classification in both speech of children, and speech of children with language impairments. They identified 11 types of errors, and compared three types of systems designed to identify the presence of each type of error: 1) rule based systems; 2) decision trees that use rules as features; and 3) naive Bayes classifiers that use a variety of features. They were able to identify all error types well ($F1 > 0.9$ in all cases), and found that in general the statistical systems outperformed the rule based systems. Hassanali and Liu's system was designed for transcripts of spoken language collected from children with impaired language, and is able to detect the set of errors they defined very well. However, it cannot be straightforwardly adapted to novel error sets.

Morley et al. (2013) evaluated how well the detectors proposed by Hassanali and Liu could identify utterances with SALT error codes. They found that a simplified version of one of Hassanali and Liu's detectors was the most effective at identifying utterances with any SALT error codes, although performance was very low (F1=0.18). Their system uses features extracted solely from part of speech tags with the Bernoulli Naive Bayes classifier in Scikit (Pedregosa et al., 2012). Their detector may be adaptable to other annotation standards, but it does not identify which errors are in each utterance; it only identifies which utterances have errors, and which do not.

2.3 Redshift Parser

We perform our experiments with the redshift parser[2], which is an arc-eager transition-based dependency parser. We selected redshift because of its ability to perform disfluency detection and dependency parsing jointly. Honnibal and Johnson (2014) demonstrate that this system achieves state-of-the-art performance on disfluency detection, even compared to single purpose systems such as the one proposed by Qian and Liu (2013). Rasooli and Tetreault (2014) have developed a system that performs disfluency detection and dependency parsing jointly, and with comparable performance to redshift, but it is not publicly available as of yet.

Redshift uses an averaged perceptron learner, and implements several feature sets. The first feature set, which we will refer to as ZHANG is the one proposed by Zhang and Nivre (2011). It includes 73 templates that capture various aspects of: the word at the top of the stack, along with its

[2]Redshift is available at `https://github.com/syllog1sm/redshift`. We use the version in the `experiment` branch from May 15, 2014.

leftmost and rightmost children, parent and grandparent; and the word on the buffer, along with its leftmost children; and the second and third words on the buffer. Redshift also includes features extracted from the Brown clustering algorithm (Brown et al., 1992). Finally, redshift includes features that are designed to help identify disfluencies; these capture rough copies, exact copies, and whether neighboring words were marked as disfluent. We will refer to the feature set containing all of the features implemented in redshift as FULL. We refer the reader to Honnibal and Johnson (2014) for more details.

3 Data, Preprocessing, and Evaluation

Our investigation into using a dependency parser to identify and label grammatical errors requires training data with two types of annotations: dependency labels, and grammatical error labels. We are not aware of any corpora of speech with both of these annotations. Therefore, we use two different sets of training data: the Switchboard corpus, which contains syntactic parses; and SALT annotated corpora, which have grammatical error annotations.

3.1 Switchboard

The Switchboard treebank (Godfrey et al., 1992) is a corpus of transcribed conversations that have been manually parsed. These parses include EDITED nodes, which span disfluencies. We preprocess the Switchboard treebank by removing all partial words as well as all words dominated by EDITED nodes, and converting all words to lowercase. We then convert the phrase-structure trees to dependencies using the Stanford dependency converter (De Marneffe et al., 2006) with the basic dependency scheme, which produces dependencies that are strictly projective.

3.2 SALT Annotated Corpora

We perform two sets of experiments on the two SALT-annotated corpora described in Table 2. We carry out the first set of experiments on on the Edmonton Narrative Norms Instrument (ENNI) corpus, which contains 377 transcripts collected from children between the ages of 3 years 11 months and 10 years old. The children all lived in Edmonton, Alberta, Canada, were typically developing, and were native speakers of English.

After exploring various system configurations, we evaluate how well our method works when it is applied to another corpus with different annotation standards. Specifically, we train and test our technique on the Narrative Story Retell (NSR) corpus (SALT Software, 2014b), which contains 496 transcripts collected from typically developing children living in Wisconsin and California who were between the ages of 4 years 4 months and 12 years 8 months old. The ENNI and NSR corpora were annotated by two different research groups, and as Table 2 illustrates, they contain a different distribution of errors. First, ENNI uses the [EW] (other word-level error) tag to code both overgeneralization errors instead of [EO], and omitted morphemes instead of [OM]. The [EU] code is also far more frequent in ENNI than NSR. Finally, the NSR corpus includes an error code that does not appear in the ENNI corpus: [EP], which indicates a pronominal error, for example using the wrong person or case. [EP], however, is rarely used.

	ENNI		NSR	
	Words	Utts	Words	Utts
Train	360,912	44,915	103,810	11,869
Dev.	45,504	5,614	12,860	1,483
Test	44,996	5,615	12,982	1,485
% with error		13.2		14.3

(a) Word and utterance counts

	ENNI	NSR
[EP]	0	20
[EO]	0	495
[EW]	4,916	1,506
[EU]	3,332	568
[OM]	10	297
[OW]	766	569
Total	9,024	3,455

(b) Error code counts

Table 2: Summary of ENNI and NSR Corpora. There can be multiple errors per utterance. Word counts include mazes.

We preprocess the ENNI and NSR corpora to reconstruct surface forms from bound morpheme annotations (ex. 'go/3S' becomes 'goes'), partial words, and non-speech sounds. We also either excise manually identified mazes or remove maze annotations, depending upon the experiment.

3.3 Evaluation

Evaluating system performance in tagging tasks on manually annotated data is typically straight-

Evaluation Level:	ERROR			UTTERANCE
	Individual error codes			Has error?
Gold error codes:	[EW]	[EW]		Yes
Predicted error codes:	[EW]		[OW]	Yes
Evaluation:	TP	FN	FP	TP

Figure 1: Illustration of UTTERANCE and ERROR level evaluation
TP = true positive; FP = false positive; FN = false negative

forward: we simply compare system output to the gold standard. Such evaluation assumes that the best system is the one that most faithfully reproduces the gold standard. This is not necessarily the case with applying SALT error codes for three reasons, and each of these reasons suggests a different form of evaluation.

First, automatically detecting SALT error codes is an important task because it can aid clinical investigations. As Morley et al. (2013) illustrated, even extremely coarse features derived from SALT annotations, for example a binary feature for each utterance indicating the presence of any error codes, can be of immense utility for identifying language impairments. Therefore, we will evaluate our system as a binary tagger: each utterance, both in the manually annotated data and system output either contains an error code, or it does not. We will label this form of evaluation as UTTERANCE level.

Second, clinicians are not only interested in how many utterances have an error, but also which particular errors appear in which utterances. To address this issue, we will compute precision, recall, and F1 score from the counts of each error code in each utterance. We will label this form of evaluation as ERROR level. Figure 1 illustrates both UTTERANCE and ERROR level evaluation. Note that the utterance level error code [EU] is only allowed to appear once per utterance. As a result, we will ignore any predicted [EU] codes beyond the first.

Third, the quality of the SALT annotations themselves is unknown, and therefore evaluation in which we treat the manually annotated data as a gold standard may not yield informative metrics. Morley et al. (2014) found that there are likely inconsistencies in maze annotations both within and across corpora. In light of that finding, it is possible that error code annotations are somewhat inconsistent as well. Furthermore, our approach has a critical difference from manual annotation: we perform classification one utterance at a time, while manual annotators have access to the context of an utterance. Therefore certain types of errors, for example using a pronoun of the wrong gender, or responding ungrammatically to a question (ex. 'What are you doing?' 'Eat.') will appear grammatical to our system, but not to a human annotator. We address both of these issues with an in-depth analysis of the output of one of our systems, which includes manually re-coding utterances out of context.

4 Detecting Errors in ENNI

4.1 Baselines

We evaluate two existing systems to see how effectively they can identify utterances with SALT error codes: 1) Microsoft Word 2010's grammar check, and 2) the simplified version of Hassanali and Liu's grammaticality detector (2011) proposed by Morley et al. (2013) (mentioned in Section 2.2). We configured Microsoft Word 2010's grammar check to look for the following classes of errors: negation, noun phrases, subject-verb agreement, and verb phrases (see `http://bit.ly/1kphUHa`). Most error classes in grammar check are not relevant for transcribed speech, for example capitalization errors or confusing *it's* and *its*; we selected classes of errors that would typically be indicated by SALT error codes.

Note that these baseline systems can only give us an indication of whether there is an error in the utterance or not; they do not provide the specific error tags that mimic the SALT guidelines. Hence we evaluate just the UTTERANCE level performance of the baseline systems on the ENNI development and test sets. These results are given in the top two rows of each section of Table 3. We apply these systems to utterances in two conditions: with mazes (i.e., disfluencies) excised; and with unannotated mazes left in the utterances. As can be seen in Table 3, the performance Microsoft Word's grammar checker degrades severely when

(a)

Him [EW] (can not) can not get up .

(b)

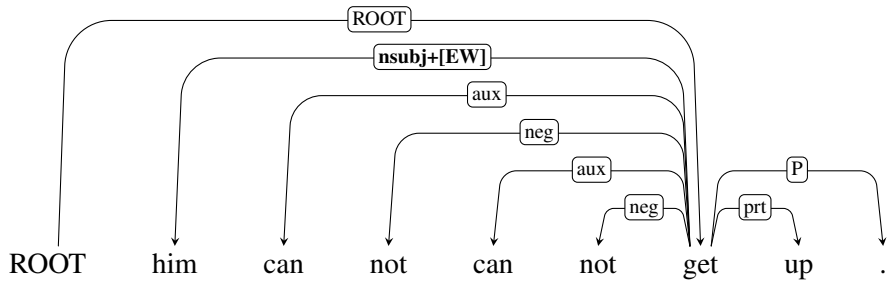


Figure 2: (a) SALT annotated utterance; mazes indicated by parentheses; (b) Dependency parse of same utterance parsed with a grammar trained on the Switchboard corpus and augmented dependency labels. We use a corpus of parses with augmented labels to train our grammaticality detector.

mazes are not excised, but this is not the case for the Morley et al. (2013) detector.

4.2 Proposed System

Using the ENNI corpus, we now explore various configurations of a system for grammatical error code detection. All of our systems use redshift to learn grammars and to parse. First, we train an initial grammar G_0 on the Switchboard treebank (Godfrey et al., 1992) (preprocessed as described in Section 3.1). Redshift learns a model for part of speech tagging concurrently with G_0. We use G_0 to parse the training portion of the ENNI corpus. Then, using the SALT annotations, we append error codes to the dependency arc labels in the parsed ENNI corpus, assigning each error code to the word it follows in the SALT annotated data. Figure 2 shows a SALT annotated utterance, as well as its dependency parse augmented with error codes. Finally, we train a grammar G_{Err} on the parse of the ENNI training fold that includes the augmented arc labels. We can now use G_{Err} to automatically apply SALT error codes: they are simply encoded in the dependency labels. We also apply the [EW] label to any word that is in a list of overgeneralization errors[3].

We modify three variables in our initial trials on the ENNI development set. First, we change the proportion of utterances in the training data that contain an error by removing utterances.[4] Doing so allows us to alter the operating point of our system in terms of precision and recall. Second, we again train and test on two versions of the ENNI corpus: one which has had mazes excised, and the other which has them present (but not annotated). Third, we evaluate two feature sets: ZHANG and FULL.

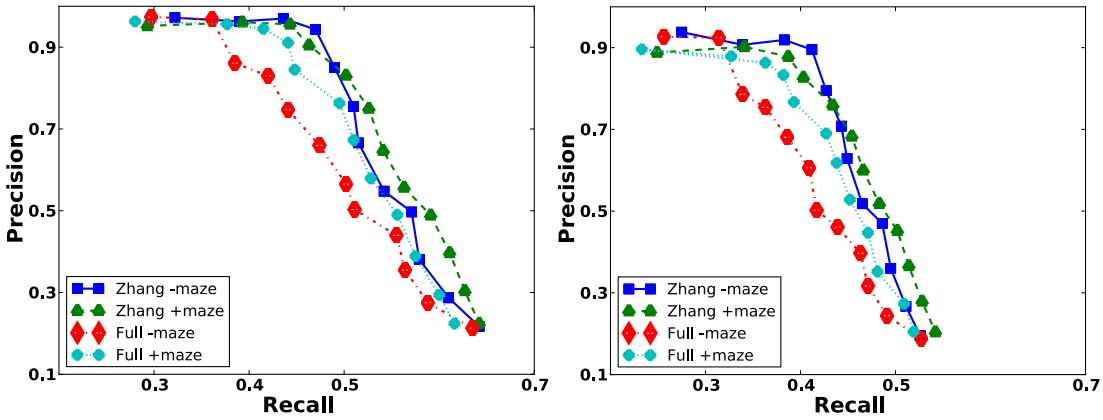
The plots in Figure 3 show how the performances of our systems at different operating points vary, while Table 3 shows the performance of our best system configurations on the ENNI development and test sets. Surprisingly, we see that neither the choice of feature set, nor the presence of mazes has much of an effect on system performance. This is in strong contrast to Microsoft Word's grammar check, which is minimally effective when mazes are included in the data. The Morley et al. (2013) system is robust to mazes, but still performs substantially worse than our proposed system.

4.3 Error Analysis

We now examine the errors produced by our best performing system for data in which mazes are present. As shown in Table 3, when we apply our system to ENNI-development, the UTTERANCE P/R/F1 is 0.831 / 0.502 / 0.626 and the ERROR P/R/F1is 0.759 / 0.434 / 0.552. This system's performance detecting specific error codes is shown in Table 4. We see that the recall of [EU] errors is quite low compared with the recall for [EW] and [OW] errors. This is not surprising, as human annotators may need to leverage the context of an utterance to identify [EU] errors, while our system makes predictions for each utterance in isolation.

[3]The list of overgeneralization errors was generously provided by Kyle Gorman

[4]Of course, we never modify the development or test data.



(a) UTTERANCE level evaluation

(b) ERROR level evaluation

Figure 3: SALT error code detection performance at various operating points on ENNI development set

System	Eval type	Mazes Excised P	R	F1	Mazes Present P	R	F1
Development							
MS Word	UTT	0.843	0.245	0.380	0.127	0.063	0.084
Morley et al. (2013)	UTT	0.407	0.349	0.376	0.343	0.321	0.332
Current paper	UTT	0.943	0.470	0.627	0.831	0.502	0.626
	ERR	0.895	0.412	0.564	0.759	0.434	0.552
Test							
MS Word	UTT	0.824	0.209	0.334	0.513	0.219	0.307
Morley et al. (2013)	UTT	0.375	0.328	0.350	0.349	0.252	0.293
Current Paper	UTT	0.909	0.474	0.623	0.809	0.501	0.618
	ERR	0.682	0.338	0.452	0.608	0.360	0.452

Table 3: Baseline and current paper systems' performance on ENNI. Evaluation is at the UTTERANCE (UTT) level except for the current paper's system, which also presents evaluation at the ERROR (ERR) level.

Error Code	P	R	F1
EU	0.639	0.193	0.297
EW	0.832	0.582	0.685
OW	0.680	0.548	0.607

Table 4: ERROR level detection performance for each code (system trained on ENNI; 30% error utterances; ZHANG feature set; with mazes)

We randomly sampled 200 utterances from the development set that have a manually annotated error, are predicted by our system to have an error, or both. A speech-language pathologist who has extensive experience with using SALT for research purposes in both clinical and typically developing populations annotated the errors in each utterance. She annotated each utterance in isolation so as to ignore contextual errors. We compare our annotations to the original annotations, and system performance using our annotations and the original annotations as different gold standards. The results of this comparison are shown in Table 5.

Comparing our manual annotations to the original annotations, we notice some disagreements. We suspect there are two reasons for this. First, unlike the original annotators, we annotate these utterances out of context. This may explain why we identify far fewer utterance level error [EU] codes than the original annotators (20 compared with 67). Second, we may be using different criteria for each error code than the original annotators. This is an inevitable issue, as the SALT guidelines do not provide detailed definitions of the error codes, nor do individual groups of annotators. To illustrate, the "coding notes" section of

Tag	Gold	Gold Count	Disagreement	P	R	F1
[EU]	Original	67	52	0.500	0.149	0.230
	Revised	20		0.450	0.333	0.383
[EW]	Original	137	27	0.859	0.533	0.658
	Revised	126		0.800	0.540	0.645
[OW]	Original	16	13	0.667	0.275	0.480
	Revised	15		0.444	0.267	0.333

Table 5: System performance using ERROR level evaluation on 200 utterances selected from ENNI-dev using original and revised annotations as gold standard

	UTTERANCE level			ERROR level		
System	P	R	F1	P	R	F1
ENNI-trained	0.310	0.124	0.178	0.157	0.057	0.084
NSR-trained	0.243	0.249	**0.277**	0.150	0.195	0.170
MS Word	0.561	0.171	0.261	–	–	–
Morley et al. (2013)	0.250	0.281	0.264	–	–	–
NSR ∪ MS Word	0.291	0.447	0.353	–	–	–
NSR ∪ Morley et al. (2013)	0.297	0.387	0.336	–	–	–
All 3	0.330	0.498	**0.397**	–	–	–

Table 6: Error detection performance on NSR-development, mazes included

the description of the ENNI corpus[5] only lists the error codes that were used consistently, but does not describe how to apply them. These findings illustrate the importance of having a rapidly trainable error code detector: research groups will be interested in different phenomena, and therefore will likely have different annotation standards.

5 Detecting Errors in NSR

We apply our system directly to the NSR corpus with mazes included. We use the same parameters set on the ENNI corpus in Section 4.2. We apply the model trained on ENNI to NSR, but find that it does not perform very well as illustrated in Table 6. These results further underscore the need for a trainable error code detector in this domain, as opposed to the static error detectors that are more common in the grammatical error detection literature.

We see in Table 6 that retraining our model on NSR data improves performance substantially (UTTERANCE F1 improves from 0.178 to 0.277), but not to the level we observed on the ENNI corpus. The Morley et al. (2013) system also performs worse when trained and tested on NSR, as compared with ENNI. When mazes are included, the performance of Microsoft Word's grammar check is higher on NSR than on ENNI (F1=0.261 vs 0.084), but it it still yields the lowest performance of the three systems. We find that combining our proposed system with either or both of the baseline systems further improves performance.

The NSR corpus differs from ENNI in several ways: it is smaller, contains fewer errors, and uses a different set of tags with a different distribution from the ENNI corpus, as shown in Table 2. We found that the smaller amount of training data is not the only reason for the degradation in performance; we trained a model for ENNI with a set of training data that is the same size as the one for NSR, but did not observe a major drop in performance. We found that UTTERANCE F1 drops from 0.626 to 0.581, and ERROR F1 goes from 0.552 to 0.380, not nearly the magnitude drop in accuracy observed for NSR.

We believe that a major reason for why our system performs worse on NSR than ENNI may be that the ENNI annotations adhere less strictly to certain SALT recommendations than do the ones in NSR. The SALT guidelines suggest that utterances with two or more word-level [EW] and/or omitted word [OW] errors should only be tagged with an utterance-level [EU] error (SALT Software, 2014a). ENNI, however, has many utter-

[5] http://www.saltsoftware.com/salt/databases/ENNIRDBDoc.pdf

ances with multiple [EW] and [OW] error codes, along with utterances containing all three error codes. NSR has very few utterances with [EU] and other codes, or multiple [EW] and [OW] codes. The finer grained annotations in ENNI may simply be easier to learn.

6 Conclusion and Future Directions

We have proposed a very simple method to rapidly train a grammatical error detector and classifier. Our proposed system only requires training data with error code annotations, and is agnostic as to the nature of the specific error codes. Furthermore, our system's performance does not appear to be affected by disfluencies, which reduces the burden required to produce training data.

There are several key areas we plan to investigate in the future. First, we would like to explore different update functions for the parser; the predicted error codes are a byproduct of parsing, but we do not care what the parse itself looks like. At present, the parser is updated whenever it produces a parse that diverges from the gold standard. It may be better to update only when the error codes predicted for an utterance differ from the gold standard. Second, we hope to explore features that could be useful for identifying grammatical errors in multiple data sets. Finally, we plan to investigate why our system performed so much better on ENNI than on NSR.

Acknowledgments

We would like to thank the following people for valuable input into this study: Joel Tetreault, Jan van Santen, Emily Prud'hommeaux, Kyle Gorman, Steven Bedrick, Alison Presmanes Hill and others in the CSLU Autism research group at OHSU. This material is based upon work supported by the National Institute on Deafness and Other Communication Disorders of the National Institutes of Health under award number R21DC010033. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

References

Eric Atwell. 1987. How to detect grammatical errors in a text without parsing it. In Bente Maegaard, editor, *EACL*, pages 38–45, Copenhagen, Denmark, April. The Association for Computational Linguistics.

Mari I Bowden and Richard K Fox. 2002. A diagnostic approach to the detection of syntactic errors in english for non-native speakers. *The University of Texas–Pan American Department of Computer Science Technical Report*.

Peter F. Brown, Vincent J. Della Pietra, Peter V. de Souza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Andrew Caines and Paula Buttery. 2010. You talking to me?: A predictive model for zero auxiliary constructions. In *Proceedings of the 2010 Workshop on NLP and Linguistics: Finding the Common Ground*, pages 43–51.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

John J Godfrey, Edward C Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 517–520.

Khairun-nisa Hassanali and Yang Liu. 2011. Measuring language development in early childhood education: a case study of grammar checking in child language transcripts. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 87–95.

George E. Heidorn, Karen Jensen, Lance A. Miller, Roy J. Byrd, and Martin S Chodorow. 1982. The EPISTLE text-critiquing system. *IBM Systems Journal*, 21(3):305–326.

Matthew Honnibal and Mark Johnson. 2014. Joint incremental disfluency detection and dependency parsing. *TACL*, 2:131–142.

Nina H Macdonald, Lawrence T Frase, Patricia S Gingrich, and Stacey A Keenan. 1982. The writer's workbench: Computer aids for text analysis. *Educational psychologist*, 17(3):172–179.

Marie W Meteer, Ann A Taylor, Robert MacIntyre, and Rukmini Iyer. 1995. *Dysfluency annotation stylebook for the switchboard corpus*. University of Pennsylvania.

Jon F Miller, Karen Andriacchi, and Ann Nockerts. 2011. *Assessing language production using SALT software: A clinician's guide to language sample analysis*. SALT Software, LLC.

Eric Morley, Brian Roark, and Jan van Santen. 2013. The utility of manual and automatic linguistic error codes for identifying neurodevelopmental disorders. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 1–10, Atlanta, Georgia, June. Association for Computational Linguistics.

Eric Morley, Anna Eva Hallin, and Brian Roark. 2014. Challenges in automating maze detection. In *Proceedings of the First Workshop on Computational Linguistics and Clinical Psychology*, pages 69–77, Baltimore, Maryland, June.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2012. Scikit-learn: Machine learning in python. *CoRR*, abs/1201.0490.

Xian Qian and Yang Liu. 2013. Disfluency detection using multi-step stacked learning. In Lucy Vanderwende, Hal Daumé III, and Katrin Kirchhoff, editors, *HLT-NAACL*, pages 820–825, Atlanta, Georgia, USA, June. The Association for Computational Linguistics.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2014. Non-monotonic parsing of fluent umm i mean disfluent sentences. In Gosse Bouma and Yannick Parmentier, editors, *EACL*, pages 48–53, Gothenburg, Sweden, April. The Association for Computational Linguistics.

LLC SALT Software. 2014a. Course 1306: Transcription - Conventions Part 3. `http://www.saltsoftware.com/onlinetraining/section-page?OnlineTrainingCourseSectionPageId=76`. [Online; accessed 29-May-2104].

LLC SALT Software. 2014b. Narrative Story Retell Database. `http://www.saltsoftware.com/salt/databases/NarStoryRetellRDBDoc.pdf`. [Online; accessed 29-May-2104].

Phyllis Schneider, Denyse Hayward, and Rita Vis Dubé. 2006. Storytelling from pictures using the edmonton narrative norms instrument. *Journal of Speech Language Pathology and Audiology*, 30(4):224.

Jan PH Van Santen, Emily T Prud'hommeaux, Lois M Black, and Margaret Mitchell. 2010. Computational prosodic markers for autism. *Autism*, 14(3):215–236.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *ACL (Short Papers)*, pages 188–193, Portland, Oregon, USA, June. The Association for Computational Linguistics.