The Utility of Manual and Automatic Linguistic Error Codes for Identifying Neurodevelopmental Disorders[*]

Eric Morley, Brian Roark and Jan van Santen
Center for Spoken Language Understanding, Oregon Health & Science University
morleye@gmail.com, roarkbr@gmail.com, vansantj@ohsu.edu

Abstract

We investigate the utility of linguistic features for automatically differentiating between children with varying combinations of two potentially comorbid neurodevelopmental disorders: autism spectrum disorder and specific language impairment. We find that certain manual codes for linguistic errors are useful for distinguishing between diagnostic groups. We investigate the relationship between coding detail and diagnostic classification performance, and find that a simple coding scheme is of high diagnostic utility. We propose a simple method to automate the pared down coding scheme, and find that these automatic codes are of diagnostic utility.

1 Introduction

In Autism Spectrum Disorders (ASD), language impairments are common, but not universal (American Psychiatric Association, 2000). Whether these language impairments are distinct from those in Specific Language Impairment (SLI) is an unresolved issue (Williams et al., 2008; Kjelgaard and Tager-Flusberg, 2001). Accurate and detailed characterization of these impairments is important not only for resolving this issue, but also for diagnostic practice and remediation.

Language ability is typically assessed with structured instruments ("tests") that elicit brief, easy to score, responses to a sequence of items. For example, the CELF-4 includes nineteen multi-item subtests with tasks such as object naming, word definition, reciting the days of the week, or repeating sentences (Semel et al., 2003). Researchers are beginning to discuss the limits of structured instruments in terms of which language impairments they tap into and how well they do so, and are advocating the potential benefits of *language sample analysis* – analyzing natural language samples – to complement structured assessment, specifically for language assessment in ASD where pragmatic and social communication issues are paramount yet are hard to assess in a conventional test format (e.g. Tager-Flusberg et al. 2009). However, language sample analysis faces two labor-intensive steps: transcription and detailed coding of the transcripts.

To illustrate the latter, consider the Systematic Analysis of Language Transcripts (SALT) (Miller and Chapman, 1985; Miller et al., 2011), which is the de-facto standard choice by clinicians looking to code elicited language samples. SALT comprises a scheme for coding transcripts of recorded speech, together with software that tallies these codes, computes scores describing utterance length and error counts, and compares these scores with normative samples. SALT codes indicate bound morphemes, edits (which are referred to in the clinical literature as 'mazes'), and several types of errors in transcripts of natural language, e.g., omitted or inappropriate words.

Although this has not been formally documented, our experience with SALT coding has shown that the codes vary in terms of: 1) difficulty of manual coding – e.g., relatively subtle pragmatic errors versus overgeneralization or marking bound morphemes;

[*]This research was supported in part by NIH NIDCD award R01DC012033 and NSF award #0826654. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not reflect the views of the NIH or NSF. Thanks to Emily Prud'hommeaux for useful discussion on this topic and help with the data.

2) utility for identifying particular disorders; and 3) difficulty of automating the code. This raises an important question: Is there a combination of codes that jointly discriminate well between relevant diagnostic groups, and at the same time are either easy to code manually or can in principle be automated? This paper explores, first, how well the various manual SALT codes classify certain diagnostic groups; and, second, whether we can automate manual codes that are of diagnostic utility. Our goal is limited: it is not the automation of all SALT codes, but the automation of those that in combination are of high diagnostic utility. Automating all SALT codes is substantially more challenging; yet, we note that even when some of these codes do not aid in classifying groups, they nevertheless may be of importance for developing remediation strategies for individual children. We are particularly interested in the impact of Autism in addition to language impairments for the utility of particular SALT codes.

The diagnostic groups are carefully chosen to be pairwise matched either on language abilities or on autism symptomatology, thus enabling a precise, "surgical" determination of the degrees to which SALT codes reflect language-specific vs. autism-specific factors. Specifically, the groups include children with ASD with language impairment (ALI); ASD with no language impairment (ALN); SLI alone; and typically developing (TD), which is strictly defined to exclude any neurodevelopmental disorder. The TD and ALN groups, as well as the ALI and SLI groups, are matched on language and overall cognitive abilities, while the ALN and ALI groups are matched on autism symptomatology but not on language and overall cognitive abilities; all groups are matched on chronological age.

Regarding our algorithmic approach, we note that automatic detection of relatively subtle errors may be exceedingly difficult, but perhaps such subtle errors are less critical for diagnosis than more obvious ones. Most prior work in grammaticality detection in spoken language has focused on specialized detectors (e.g., Caines and Buttery 2010; Hassanali and Liu 2011), such as mis-use of particular verb constructions rather than coarser detectors for the presence of diverse classes of errors. We demonstrate that these specialized error detectors can break down when confronted with real world dialogue, and that in general, the features in these detectors restricts their utility in detecting other sorts of errors.

We implement a detector to automatically extract coarse SALT codes from an uncoded transcript. This detector only depends upon part of speech tags, as opposed to the parse features that are often used in grammaticality detectors. In most cases, these automatically extracted codes enable us to distinguish between diagnostic groups more effectively than do features that can be extracted trivially from an uncoded transcript.

As far as we know, researchers have not previously considered the utility of grammatical error codes to identify ASD or SLI. Prudhommeaux and Rouhizadeh (2012), however, found that automatically extracted pragmatic features are useful for identifying children with ASD, among children both with and without SLI. Gabani et al. (2009) found that features derived from language models are useful for distinguishing between children with and without a language impairment, both in monolingual English speakers, and in children who are bilingual in English and Spanish.

Improving the characterization of a child's language impairments is a prerequisite to developing a sound plan for language training and education for that child. This paper presents a step in the direction of effective automated analysis of linguistic samples that can provide useful information even in the face of comorbid disorders such as ASD and SLI.

2 Systematic Analysis of Language Transcripts

Here we give an overview of what SALT requires of transcriptions, and of SALT coding. The approach has been in wide use for nearly 30 years (Miller and Chapman, 1985), and now also exists as a software package[1] providing transcription and coding support along with tools for aggregating statistics for manual codes over the annotated corpora and comparing with age norms. The SALT software is not the focus of this investigation, so we do not discuss it further.

2.1 Basic Transcription

We apply the automated methods to what will be called *basic transcripts*. Key for this concept is that, first, these transcripts do not require linguistic expertise and thus can be performed by standard transcription services; and, second, that – as we shall

[1]http://www.saltsoftware.com/

see – useful features can be automatically computed from them.

Following the SALT guidelines, a basic transcript should indicate: the speaker of each utterance, partial words (or stuttering), overlapping speech, unintelligible words, and non-speech sounds. It should be verbatim, regardless of whether a child's utterance contains neologisms (novel words) or grammatical errors (for example 'I goed' should be written as such).

A somewhat subtle issue is that SALT prescribes that the basic transcript be broken into *communication units* (which in this paper will be synonymous with *utterance*). Communication units are defined as "a main clause with all its dependent clauses" (Miller et al., 2011). One reason for defining utterance boundaries with communication units, rather than turns or sentences, is that in addition to this being standard practice in language sample analysis, doing so does not reward children for making long, but rather simple statements, nor does it penalize children for being interrupted. To illustrate the first point, the utterance "I like apples, and bananas, and pears, and oranges, and grapes." is one sentence long, but has five communication units (one at each comma). If the sentence were used as the basic unit, the utterance would indicate the same level complexity as the obviously more intricate "for the past three years we have lived in an apartment". In the basic transcript, each communication unit should be terminated by one of the following punctuation marks: '?' if it is a question, '^' if the speaker was interrupted, '>' if the speaker abandoned the utterance, and '.' in all other cases. Thus, the above example would be transcribed as "C: I like apples. . . . C: and grapes."

2.2 Markup

There are three broad categories of SALT codes: indicators of 1) certain bound morphemes, 2) *edits* (discussed below), and 3) errors.

Morphology The following inflectional suffixes must be coded according to the SALT guidelines: plural -s (/S), possessive -'s (/Z), possessive plural -s' (/S/Z), past tense -ed (/ED), 3rd person singular -s (/3S), progressive -ing (/ING). The following clitics must also be delimited with a '/', provided the resulting root is unmodified in the surface form: n't, 't, 'd, 're, 's, 've. Since these morphemes are only indicated if the root is unmodified in the surface form, "won't" will remain unsegmented because 'wo' is not the root; "can't" will be segmented "can/'T" and "don't" will be segmented "do/N'T", so as to preserve their respective roots. Nominal or verbal forms with any of the preceding suffixes or clitics are written as the base form with the code appended, for example *hitting* → *hit/ING*, *bases* → *base/S*.

Edits Edits consist of filler words such as 'like', 'um' and 'uh', false starts, and revisions. There may be multiple edits in a single utterance, as well as multiple adjacent edits. Edits are indicated by parentheses, for example: "(And they like) and she (like) faint/3S." Note that in the SALT manual, and the language sample analysis literature, edits are referred to as *mazes*. We use the term *edit* here because this is the more widely used term for this phenomenon in natural language processing.

Error codes The exact set of error codes used depends upon the clinician's needs and the errors of interest. Here we consider several key errors outlined in the SALT manual. These error codes and examples are shown in Table 1. Some of these codes describe precise classes of errors, for example [EO] or [OW], but others do not. For example, [EW] can describe using the wrong verb, tense, preposition or pronoun (in terms of case, person or gender), as well as other errors. Note that [EU] (and [EC]) error codes can occur in grammatical utterances. The [EU] code marks utterances that are ungrammatical for reasons not captured by the other error codes, for example severe problems with word order, or utter-

Table 1: SALT error codes and examples

Code	Meaning	Example	Count in Corpus
[EC]	Inappropriate response	Did you help yourself stop? Mom[EC].	9
[EO]	Overgeneralization	Yeah, cuz I almost saw/ED[EO] one.	229
[EW]	Error word	I play/ED of[EW] the cat.	1,456
[EU]	Utterance-level error	You can see it very hard because it/'S under my hair[EU].	532
[EX]	Extraneous word	Would you like to be[EX] fall down?	322
[OM]	Omitted morpheme	The cat eat[OM] fish.	881
[OW]	Omitted word	He [OW] going now.	770

ances which are simply nonsensical, as in Table 1.

3 Evaluation of Manual Codes

In this section we use features extracted from SALT-coded transcripts for classification. We consider two different types of features: baseline features, which are easily derived from a basic transcript; and features derived from SALT codes. We investigate these features to determine which SALT codes are most worth automating for classification.

3.1 Data

Our data is a collection of 144 transcripts of the Autism Diagnostic Observation Schedule (ADOS), which is a semi-structured task that includes an examiner and a child (Lord et al., 2002). *Semi-structured* means that the examiner carries out a sequence of rigorously specified activities, but her prompts and questions are not scripted verbatim for all of them. Detailed guidelines exist for scoring the ADOS, but these are not considered in the current paper. All transcripts have been manually coded with SALT codes, described in Table 1.

Subjects ranged in age between 4 and 8 years and were required to be intelligible, to have a full-scale IQ of greater than 70, and to have a mean length of utterance (MLU) of at least 3. Diagnoses of ASD and of SLI followed standard procedures, and were based on clinical consensus in accordance to diagnostic criteria outlined in the DSM-IV (American Psychiatric Association, 2000). Furthermore, ASD diagnosis required ADOS and Social Communication Questionnaire scores (SCQ) (Berument et al., 1999) to meet conventional thresholds. Diagnosis of SLI required a CELF Core Language Score of at least 1 standard deviation below the mean, in addition to exclusion of ASD.

Children were partitioned into pairs of groups matched on certain key measures. Table 2 shows these pairs and what they were matched on. The individuals were selected from the initial pool of all participants using the algorithm proposed by van Santen et al. (2010), in which, for a given pair of groups, children are iteratively removed from each group until there is no significant difference (at $p < 0.02$) on any measure on which we want the pair to be matched. We combined some groups into composite groups: ASD (ALI and ALN), nASD (SLI and TD), LN ('language normal': ALN and TD), and LI ('language impaired': ALI and SLI).

Group 1		Group 2		
Group	N	Group	N	Matched on
ALI	25	ALN	21	Age, ADOS, SCQ
ALI	24	SLI	19	Age, NVIQ, VIQ
ALN	25	TD	27	Age, NVIQ, VIQ
ASD	48	nASD	61	Age
LN	61	LI	39	Age
SLI	15	TD	38	Age

Table 2: Matched measures for paired groups (ADOS = ADOS score, NVIQ = non-verbal IQ, VIQ = verbal IQ)

3.2 Features

The term "feature" will be used to refer to instances of various classes of SALT codes as well as to instances of other events that can be trivially extracted from the basic transcripts but do not involve SALT codes (e.g, the ratio of 'uh' to 'um'). We distinguish between five levels of features, enumerated in Table 3, that vary in the number and complexity of codes required. This ranges from the baseline features that require no manual codes to SALT-5 features that require full SALT coding. We consider two normalized variants of each feature: one normalized by the number of utterances spoken by the child, and the other normalized by the number of words spoken by the child (except for TKCT). The ratios OCRAT and UMUHRAT are never normalized. Each feature level includes all features on lower levels. Finally, to make our investigation into feature combinations more tractable, we do not consider combining two different normalizations of the same feature.

3.3 Classification

We perform six classification tasks in our investigation, according to the paired groups in Table 2: ALI/ALN; ALI/SLI; ALN/TD; ASD/nASD; LN/LI; and SLI/TD. We extract various features from the ADOS transcripts, and then classify the children in a leave-pair-out (LPO) schema (Cortes et al., 2007) using the scikit logistic regression classifier with default parameters (Pedregosa et al., 2011). For LPO analysis, we iterate over all possible pairs that contain one positive and one negative instance (i.e. children with different diagnoses), training on all other instances, and testing on that pair. We count a trial as a success if the classifier assigns a higher probability of being positive to the positive instance than to the negative instance. We then divide the number of successes by the number of pairs to get an unbiased estimate of the area under the receiver operating curve (AUC) (Airola et al., 2011). AUC is

Group	Feature	Description
Baseline	CEOLP	# of times examiner speaks while child is talking
	ECOLP	# of times child speaks while examiner is talking
	INCCT	Incomplete word count
	OCRAT	Ratio of open- to closed-class words
	TKCT	Token count
	TPCT	Type count
	UMUHRAT	Ratio of 'uh' to 'um'
	UINTCT	Unintelligible word count
SALT-1	All baseline features +	
	MPCT	Morpheme count
	EDITCT	Edit count
SALT-2	All SALT-1 features +	
	NERRUTT	Number of utterances with any SALT error codes
SALT-3	All SALT-2 features +	
	ERRCT	Count of SALT error codes
SALT-4	All SALT-3 features +	
	UTLERRCT	Count of utterance level errors (EC / EU)
	WDLERRCT	Count of word level errors (all other error codes)
SALT-5	All SALT-4 features +	
	XCT	Count of individual error codes (X=EC, EO, . . . ; see Table 1)

Table 3: Features by Level

the probability that the classifier will assign a higher score to a randomly chosen positive example than to a randomly chosen negative example.

3.4 Determining Relevant Features

We use a t-test based criterion as a simple way to determine which features to investigate for each classification task. For a given classification task, we perform a t-test for independent samples on each feature under both normalization schemes (if appropriate). We retain a feature for investigation if that feature is significantly different between the two groups at the $\alpha = 0.10$ level. If a particular feature varies significantly between groups under both normalization schemes, we retain the version that has the larger T-statistic. For the sake of brevity, we do not report all of the features that varied between groups here, but this data is available upon request from the authors.

3.5 Initial Feature Ablation

We perform feature ablation to see which features are most useful for performing each classification task. Figure 1 shows the maximum performance (in terms of AUC) over all subsets of features at each feature level (on the x-axis) on each of the six diagnostic classification tasks. Missing values for a particular level of features for any comparison indicate that no features in that level that passed the t-test based criterion for the two groups being compared.

Figure 1 illustrates two important points. First, classification difficulty depends heavily on the pair that is being compared. For example, the AUC for ALI/SLI is at most 0.723 (SALT-5), while the AUC for SLI/TD reaches 0.982 (SALT-5). This is not surprising, as some pairs, most notably SLI/TD, differ widely in coarse measures of language ability (such as non-verbal IQ), while other pairs, including ALI/SLI, do not. Second, in many of the tasks, SALT-derived features are of high utility, but the biggest gain in classification performance comes with SALT-2, which is a count of the number of sentences containing any SALT error code. In fact, for all but one classification task (ASD/nASD), the AUC achieved with SALT-2 is at least 96% of the maximum AUC. Furthermore, the best feature set using SALT-2 features for most of these tasks is either the NERRUTT feature alone, or in the case of ALI/SLI, NERRUTT and TPCT. These results lead us to conclude that the most important SALT-derived feature to code is NERRUTT.

Perhaps surprisingly, Figure 1 also shows that for ALN/TD and SLI/TD, performance at SALT-1 is lower than the baseline. There are two reasons for this, which we explain in turn: 1) the SALT-1 feature set must include a feature that is less useful than those in the optimal baseline feature set, and 2) the classifier will not ignore this feature. MPCT must be included in SALT-1 for both pairs, because the only

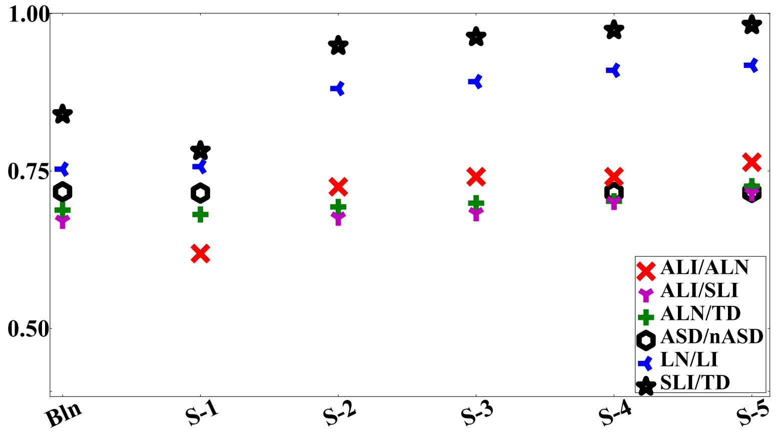

Figure 1: Maximum classification performance (AUC) at different feature levels (Bln=Baseline, S-N=SALT-N)

other SALT-1 feature, EDITCT, does not vary significantly between either ALN/TD or SLI/TD. Furthermore, MPCT is highly correlated with TKCT, yet TKCT is not in the best baseline feature set for either of these pairs. Therefore, the SALT-1 feature set is required to include a feature that is less useful than the most useful ones in the baseline set, which results in lower performance. Once MPCT is included in the SALT-1 feature set, the logistic regression classifier will not ignore it by assigning it a zero coefficient. This is because MPCT distinguishes between groups, and because the classifier is trained at each round of LPO classification to maximize the likelihood of the training data, rather than the AUC estimate provided by LPO classification.

3.6 Counting Specific Error Codes

The single feature in SALT-2, NERRUTT, counts how many utterances spoken by the child contain at least one SALT error code. Some of these heterogenous errors, for example overgeneralization errors ([EO]), should be straightforward to identify automatically. Automatically identifying others, for example utterances that are inappropriate in context ([EC]), would be more difficult. Therefore, before automating the extraction of NERRUTT, we should see which errors most need to be identified, and which can safely be ignored. To do this, we repeat our LPO classification procedure on various tasks using SALT-2 features.

We perform the following procedure to identify the most diagnostically informative errors: for each subset s of SALT error codes, 1) compute the feature NERRUTTSUBSET by counting the number of utterances that contain any of the errors in s; then 2) perform the LPO diagnostic classification task using NERRUTTSUBSET as the only feature. The results of this experiment are in Table 4. The '% Max' column shows classification performance when a particular subset of error codes were counted, relative to the maximum performance yielded by any subset of error codes for that particular task. We exclude the ALN/TD and ASD/nASD tasks from this experiment because NERRUTT does not improve performance on these tasks. This is perhaps unsurprising, because SALT codes were designed to be diagnostic of SLI, not ASD.

We find that in all tasks, ignoring certain error codes raises performance. These results also show that it is not necessary, and indeed not ideal, to identify utterances containing any SALT code. Identifying utterances that contain any of the following three codes is sufficient to achieve at least 97% of the maximum AUC enabled by counting any subset of SALT codes: [EW], [OM], [OW]. For clarity, NERRUTTMOD is the count of utterances that contain any of those three SALT codes.

Table 4: AUC from Counting Subsets of Errors

Classification	Errors Counted	AUC	% Max
ALI/ALN	EW, OM	0.762	100
	EW, OM, OW	0.739	97
	all	0.724	93
ALI/SLI	EW, OM	0.715	100
	EW, OM, OW	0.704	98
	all	0.676	95
LN/LI	EW, OM, OW	0.901	100
	all	0.881	98
SLI/TD	OM, OW	0.984	100
	EW, OM, OW	0.970	99
	all	0.951	97

3.7 Robustness of NErrUttMod feature to noise: a simulation experiment

We will consider two general ways of automatically extractingNErrUttMod. The first way is to build a detector to identify utterances that contain at least one relevant error. The second way is to make detectors for the each relevant error, then combine the output of these detectors. It is unlikely that any error detector will perform perfectly. Prior to investigation of automation strategies, we would like to get an idea of how much such errors will affect diagnostic classification performance. To this end, we investigate how well we can perform the diagnostic classification tasks when noise is deliberately introduced into the NErrUttMod values via simulation.

We consider two scenarios. In the first, we assume a single error detector will be used to extract NErrUttMod. We take each manually coded utterance, then randomly change whether or not that sentence is counted as having an error to simulate different precision and recall levels of the automated NErrUttMod extractor. We repeat this procedure 100 times for each classification task, and then examine the mean AUC over all trials. In the second scenario, we assume a detector for each error code that counts a sentence as having an error any time one of the detectors fires. We randomly corrupt the detection of each error code considered in NErrUttMod in turn to simulate different precision and recall levels of each individual error detector. We assume perfect detection of all errors not being randomly corrupted. Again, we repeat this procedure 100 times for each classification task, and consider the mean AUC over all trials.

In both experiments, and in all classification tasks, we find that the NErrUttMod feature is extremely robust to noise. For example, finding the NErrUttMod feature with a single detector with a precision/recall of 0.1/0.3 enables SLI/TD classification with an average AUC of 0.975, as compared to the maximum AUC of 0.984, enabled by a perfect detector. When we use a cascaded detector to corrupt each of the two errors counted in NErrUttMod for classifying SLI/TD, so long as one error is detected perfectly, the other error only needs to be detected with precision and recall of 0.1 to enable a classification AUC within 0.02 of the maximum.

The extreme robustness of this feature may appear surprising, but it is easily explained by the data. The mean value of NErrUttMod for the SLI group is 7.8 times the mean value of this feature for the TD group. So long as there is a correlation between the true value of NErrUttMod and the estimated value, as we have assumed in this experiment, then the estimated value is bound to be of utility in classification. This bodes well for the utility of automation, even for a difficult task of discovering some of the relatively subtle errors coded in SALT.

4 Automatic Feature Extraction

4.1 Evaluating Hassanali and Liu's System

Hassanali and Liu developed two grammaticality detectors that they used to identify ungrammatical utterances in transcriptions of speech from children both with and without language impairments (Hassanali and Liu, 2011). They tested their grammaticality detectors on the Paradise corpus, which consists of conversations with children elicited during an investigation of *otitis media*, a hearing disorder. They present both a rule-based and a statistical grammaticality detector. Both detectors consist of sub-detectors for the errors shown in Table 5. The rule-based and statistical detectors perform well, with the statistical detector outperforming the rule-based one (F1=0.967 vs. 0.929). The statistical detector, however, requires each error identified by any of the sub-detectors to be manually identified in the training data.

We reimplement both the rule based and statistical detectors proposed by Hassanali and Liu, and apply it to our data, with three modifications. The first two are minor: 1) we substitute the Charniak-Johnson reranking parser (2005) for Charniak's original parser (Charniak, 2000), and 2) we use the scikit multinomial naive bayes classifier (Pedregosa et al., 2011) instead of the one in WEKA (Hall et al., 2009). The third difference is that we use these detectors to identify SALT error codes rather than the errors these classifiers were originally built to detect. The mapping of the original errors to SALT error codes is given in Table 5. To clarify, if we are training the 'Missing Verb' detector, then any utterance with an [OW] code is taken to be a positive example. This issue does not present itself with the rule-based detector because it is not trained. Note that the two verb agreement features may correspond to either [EW] or [OM] SALT codes. For example, 'you does' would be [EW] because of the otiose 3^{rd} per-

Error	SALT code
Misuse of -ing participle	[EW]
Missing copulae	[OW]
Missing verb	[OW]
Subject-auxilliary agreement	[EW]
Subject-verb agreement	[EW]/[OM]
Missing infinitive ‘to’	[OW]

Table 5: Error detectors proposed by Hassanali and Liu

son singular suffix, while ‘he do’ would be an [OM] because it is missing that same suffix.

Hassanali and Liu’s error detectors perform poorly on our data. Table 6 reports the performance of their detectors detecting utterances with various error codes. Five of the six statistical error detectors that Hassanali and Liu proposed are unable to identify any of the errors in our data. The‘misuse of -ing participle’ detector, however, is an exception, and its performance detecting the analogous error code [EW], using 10-fold cross validation is, shown in Table 6. To detect the two pairs of error codes, [EW][OM] and [OM][OW], and all three relevant error codes ([EW][OM][OW]), we use the appropriate rule based detectors. For example, to detect utterances with either [EW] or [OM] errors, we pool the detectors for the analogous error codes: ‘misuse of -ing participle’, ‘subject-auxilliary agreement’, and ‘subject-verb agreement’.

There are three factors that may explain the poor performance observed with most of Hassanali and Liu’s error detectors when used with our data. The first is that the three SALT codes we try to detect ([EW], [OM], and [OW]) capture a wider variety of errors than the six in Hassanali and Liu’s system. This could account for the low recall. Second, there are many utterances in our data that Hassanali and Liu’s system would label an error, but which are not marked with any SALT error codes. For example, if the examiner asks the child what she is doing, ‘eating spaghetti’ is a faultless response, even though it is missing both the subject and auxiliary verb. Such utterances may account for the low precision. Finally, most of Hassanali and Liu’s sub-detectors depend upon features describing the presence or absence of specific structures in the parses of the input. The exception to this is the statistical ‘misuse of -ing participle’ detector, which uses part of speech (POS) tag bigrams and skip bigrams as features. It should come as no surprise then that the ‘misuse of -ing participle’ is the most robust of these detectors. Indeed,

System	Codes Detected	P	R	F1
Hassanali & Liu	[EW]†	0.074	0.218	0.110
	[EW][OM]*	0.049	0.277	0.083
	[OM][OW]*	0.028	0.191	0.049
	All three*	0.066	0.354	0.111
POS-tag feature-based classifier	[EW]	0.074	0.218	0.110
	[OM]	0.070	0.191	0.103
	[OW]	0.064	0.210	0.099
	[EW][OM]	0.102	0.269	0.148
	[OM][OW]	0.102	0.269	0.148
	All three	0.127	0.308	0.180

Table 6: Performance on automatic detection of utterances with certain error codes using Hassanali and Liu’s detectors, and general POS-tag-feature-based classifier. † = ‘misuse of -ing participle’, statistical; * = rule-based

in what follows, we make use of general POS-tag features (tag n-gram and skip n-grams) as they do in this detector, for a general purpose detector not targeted specifically at this particular construction, but rather to detect the presence of arbitrary given sets of error tags.

4.2 Automatic SALT error code detection

We compare three types of automatic error code detectors: 1) *individual* error code detectors; 2) *pair* detectors, each of which detects a pair of error codes included in NERRUTTMOD, following Table 4; and 3) a *generic* detector that identifies any utterance containing any of the following SALT codes: [EW], [OM], or [OW]. We investigate four different features, all of which are easily derived from the basic transcript: bigrams and skip bigrams of words, and POS tags. We use POS tags extracted from the output of the Charniak-Johnson reranking parser (2005) (also used in our reimplementation of Hassanali and Liu’s detectors) for simplicity. We use the Bernoulli Naive Bayes classifier in scikit with the default settings (Pedregosa et al., 2011).

We find that the word features do not aid classification in any condition, and that using both bigrams and skip bigrams of POS tags improves on using either alone. We report the performance of the three types of error detectors in Table 6. These results are from 10-fold cross-validation using POS tag bigrams and skip bigrams as features. Note that the general POS-tag-feature-based classifier uses the same features as Hassanali and Liu’s statistical ‘misuse of -ing participle’ detector, which is why the performance for detecting [EW] error codes alone

	Manual features		Automatic extraction SALT-2 features			
	Baseline	SALT-2	Baseline θ		Optimized θ	
Diagnoses	AUC	AUC	θ	AUC	θ	AUC
ALI/ALN	0.619[†]	0.723	0.5	0.611	0.94	0.676
ALI/SLI	0.562	0.686	0.5	0.632	0.99	0.671
LN/LI	0.755	0.881	0.5	0.801	0.50	0.801
SLI/TD	0.840	0.951	0.5	0.805	0.99	0.840

[†] SALT-1; no significantly different baseline features

Table 7: Diagnostic classification AUC using automatically extracted NERRUTTMOD

is identical between the two systems.

The generic error detector yields higher performance than either the individual or pair error detectors. Coding training data for the generic detector is simpler than doing so for the others because it only involves a single round of binary coding.

4.3 Diagnostic Classification

We repeat the LPO diagnostic classification tasks using the automatically extracted NERRUTTMOD feature. We recompute NERRUTTMOD for each speaker at each iteration, training on all data except for the two speakers in the test pair, and the speaker whose NERRUTTMOD feature we are predicting. The results from this task are shown in Table 7.

As can be seen in Table 7, diagnostic classification performance using the automatically extracted the NERRUTTMOD feature is markedly lower than when we extracted this feature from manual codes. However, raising the probability threshold θ at which utterances are counted as containing an error from its default value of 0.5, improves diagnostic classification performance for all but one pair (LN/LI). This is because increasing the probability threshold at which we count an utterance as having an error improves in NERRUTTMOD detection. For example, in the ALI/SLI group, using the default $\theta = 0.5$, and a leave-one-out scenario, we can automatically extract NERRUTTMOD with a precision/recall score of 0.19/0.47. When we increase θ to 0.99, the precision and recall become 0.23/0.24. Even though there is a massive drop in recall, the improvement in precision is able to boost diagnostic classification performance.

In all but one pair (SLI/TD), the automatically extracted NERRUTTMOD feature improves classification over the baseline, even though the NERRUTTMOD extractor performs poorly in terms of intrinsic evaluation, with an F1 score of 0.180. These results are in line with the experiments performing diagnostic classification with an artificially noisy NERRUTTMOD feature (see Section 3.7). These results also demonstrate that the automatically extracted values of NERRUTTMOD are sufficiently correlated with the true values of this feature to be of some diagnostic utility.

5 Conclusions

We have found that the SALT codes provide useful information for distinguishing between certain diagnostic groups, but not all of them. Specifically, and not surprisingly given SALT's focus on language disorders and not generally on atypical language use characteristic of ASD, adding SALT-derived features to baseline features added little to ASD/nASD, ALI/SLI, or ALN/TD classification accuracy, but added substantially to SLI/TD, ALI/ALN, and LN/LI classification accuracy. Furthermore, we found that a simplified coding schema is almost as useful as the complete one for differentiating between these groups. Finally, we have proposed a simple method to automatically extract a variant of the most useful SALT-derived feature, NERRUTTMOD, which is a count of sentences that contain any of three types of errors (omitted morphemes or words, and generic word-level errors). Although this feature's utility degrades when extracted automatically, it still has considerable discriminative value.

In future work, we will investigate the utility of more sophisticated features for extracting NERRUTTMOD and other SALT-derived features. We will also investigate the utility of other linguistic features, for example parse structure, for the diagnostic classification task. Finally, we will also consider whether we can perform the diagnostic classification task more effectively using cascaded binary classifiers (for example language impaired vs. language normal), as opposed to having a classifier for every diagnostic pair.

References

Antti Airola, Tapio Pahikkala, Willem Waegeman, Bernard De Baets, and Tapio Salakoski. 2011. An experimental comparison of cross-validation techniques for estimating the area under the roc curve. *Computational Statistics & Data Analysis*, 55(4):1828–1844.

American Psychiatric Association. 2000. *DSM-IV-TR: Diagnostic and Statistical Manual of Mental Disorders.* American Psychiatric Publishing, Washington, DC, 4th edition.

Sibel Kazak Berument, Michael Rutter, Catherine Lord, Andrew Pickles, and Anthony Bailey. 1999. Autism screening questionnaire: diagnostic validity. *The British Journal of Psychiatry*, 175(5):444–451.

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. Association for Computational Linguistics.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180. Association for Computational Linguistics.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139. Morgan Kaufmann Publishers Inc.

Corinna Cortes, Mehryar Mohri, and Ashish Rastogi. 2007. An alternative ranking problem for search engines. In *Proceedings of WEA-2007, LNCS 4525*, pages 1–21. Springer-Verlag.

Keyur Gabani, Melissa Sherman, Thamar Solorio, Yang Liu, Lisa M Bedore, and Elizabeth D Pena. 2009. A corpus-based approach for the prediction of language impairment in monolingual english and spanish-english bilingual children. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 46–55. Association for Computational Linguistics.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. 2009. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

K. Hassanali and Y. 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. Association for Computational Linguistics.

Margaret M Kjelgaard and Helen Tager-Flusberg. 2001. An investigation of language impairment in autism: Implications for genetic subgroups. *Language and cognitive processes*, 16(2-3):287–308.

Catherine Lord, Michael Rutter, PC DiLavore, and Susan Risi. 2002. *Autism diagnostic observation schedule: ADOS.* Western Psychological Services.

J. Miller and R. Chapman. 1985. Systematic analysis of language transcripts. *Madison, WI: Language Analysis Laboratory.*

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.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Emily Prudhommeaux and Masoud Rouhizadeh. 2012. Automatic detection of pragmatic deficits in children with autism. In *Proceedings of the 3rd Workshop on Child, Computer and Interaction (WOCCI 2012).*

Eleanor Messing Semel, Elisabeth Hemmersam Wiig, and Wayne Secord. 2003. *Clinical evaluation of language fundamentals.* The Psychological Corporation, A Harcourt Assessment Company, Toronto, Canada, fourth edition.

Helen Tager-Flusberg, Sally Rogers, Judith Cooper, Rebecca Landa, Catherine Lord, Rhea Paul, Mabel Rice, Carol Stoel-Gammon, Amy Wetherby, and Paul Yoder. 2009. Defining spoken language benchmarks and selecting measures of expressive language development for young children with autism spectrum disorders. *Journal of Speech, Language and Hearing Research*, 52(3):643.

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.

David Williams, Nicola Botting, and Jill Boucher. 2008. Language in autism and specific language impairment: Where are the links? *Psychological Bulletin*, 134(6):944.