Automating Language Sample Analysis

Eric Morley

April 19, 2016

School of Medicine

Oregon Health & Science University

CERTIFICATE OF APPROVAL

This is to certify that the PhD dissertation of

Eric Abel Morley

has been approved

[redacted]

Committee chair — Jan van Santen

[redacted]

Adviser

[redacted]

Member — Kyle Gorman

[redacted]

Member — Xubo Song

[redacted]

External member — Joel Tetreault

Acknowledgments

Working on this dissertation, and learning the skills I needed to do so, would not have been possible without the many people who I've been fortunate enough to have support me along the way. First and foremost I would like to thank my adviser Brian Roark for his unending support and patience throughout this process. My other advisers, Jan van Santen, Kyle Goreman, and Joel Tetreault have also been integral to my success, and I feel lucky to have worked with them.

My life has taken many turns during the long journey from beginning graduate school to defending this dissertation, with new people coming into my life, and others leaving it. Throughout this time I've depended heavily upon others for support, and without them I don't think I would have completed this. So thank you to my family: Mom, Dad, Andrew, Nana, Pop, and Jenny. To my friends, who were there when I most needed them: Dan, Trevor and Jen, Sanjaya, David, Steve, Mike, Matt, Bruckner, Sarah's C and J, Karan, Everett, Ryan, and last, but not least, Anna, who I have had the pleasure of collaborating with on research. Thank you to my DWRT teammates for the camaraderie, pushing me in workouts, and keeping me sane. Joe and Denise VanLeuven, thank you for all the warmth you have showed me. I am glad you are still a part of my life. Thank you to everyone whose path has crossed mine during this time: even if they didn't stay intertwined, I'm thankful for the time when they were.

The CSLU community has changed greatly during my time there. Some of the professors I've learned so much from are still there, in particular Alex Kain and Steven Bedrick. Others, like Richard Sproat, have moved on, usually to Google. I feel lucky that I was at CSLU when I was, and that I had the chance to learn from and work with so many talented people. Also included in those are my fellow graduate students, especially Masoud, Mahsa, Emily, and Ethan. Finally, thank you to Pat for holding the whole operation together. CSLU has been so flexible with me: letting me move, first to Michigan, then to New York, and giving me time off to gain professional experience. This flexibility has been so valuable to me, so thank you all for affording me it.

The industry experience I have gained during graduate school has provided motivation, technical skills, and some degree of freedom that have helped me finish this dissertation. In particular, I'd like to thank Jason Brenier, who has been, and continues to be, a wonderful mentor. I'd also like to thank some of the people I've had the pleasure of working with over the years: Kunjan, Ken, Anna, Brant, Ralph, Dave, and all of my colleagues at RedOwl. I've learned a lot from you, and hope to continue to do so in the future.

Contents

Chapter 1

Introduction

Many techniques developed in Natural Language Processing (NLP), a field of research concerned with using computers to process human languages, have moved from research labs to consumer products in recent years. Some well-known products backed by these technologies include the grammar checker in Microsoft Word, Google translate, and speech-driven digital assistants such as Apple's Siri. Natural language processing is also being used by businesses in various ways, for example in fraud detection, legal investigations, and marketing. This thesis develops extant and novel techniques to apply NLP to the automation of *language sample analysis*.

Language sample analysis (LSA, not to be confused with latent semantic analysis, which is not discussed in this thesis) is the practice of eliciting, transcribing, and analyzing samples of spoken language. At present, LSA is used for a variety of purposes, including research into language development and developmental disorders (e.g. autism and language impairments), and less commonly, for assessing a child's language, or evaluating the effectiveness of remediative efforts. LSA is typically used by researchers, and occasionally by practicing speech-

language pathologists, although tasks such as transcription and annotation may be performed by trained assistants. The language analyzed typically comes from children, many of whom are suspected of having developmental disorders that impair their ability to use language. Although there are various frameworks for LSA, they all involve transcribing what the child said, and then annotating the transcripts. Researchers have recently been advocating the use of LSA to complement structured instruments (essentially tests, see Chapter 2) because some phenomena, including ones which may be informative for diagnosis, are hard to capture using structured instruments alone (Tager-Flusberg et al., 2009).

The *de facto* standard system for LSA is the Systematic Analysis of Language Transcripts, which is commonly referred to as SALT (Miller and Chapman, 1985). SALT includes conventions on transcription and annotation. It also includes a piece of software to compute summary statistics from one's own transcripts, and compare these with ones computed from included sets of reference transcripts. We refer to these sets of reference transcripts as the *SALT corpora*, and we investigate the consistency of annotations in the SALT corpora throughout this thesis. We demonstrate in Chapters 5 and 6 that the annotation standards, along with the consistency of the annotations themselves, vary widely between different SALT corpora. The fact that these corpora are not annotated consistently calls into question how useful they are for comparison. Variable standards and consistency of annotations are critical issues for researchers and clinicians who compare their own transcripts to the SALT corpora, whether with the SALT software or in publications.

LSA presents two particular challenges to clinicians, particularly when compared with structured instruments: it is time consuming, and in practice, it appears to be difficult to perform consistently.[1] In this thesis, we will address

[1] The creators of SALT, however, insist that it is both quick to perform, and that annotations are reliable (ex. Miller et al., 2011, chap. 1).

both of these issues by introducing trainable, automated systems based on techniques from NLP that are able to perform two critical steps of LSA: annotating mazes (a simplified version of what are referred to as *disfluencies* in the NLP and linguistics literature), and annotating grammatical errors. These two annotations are illustrated below, with mazes indicated by parentheses, while the grammatical error annotation, indicating an overregularized past tense form ('goed'), is in brackets:

(1.1) (I uh) I (uh) goed [EO] there.

Because the maze and grammatical error detection systems are automated, they can reduce or eliminate the need for humans to annotate data. Furthermore, the same system can be applied to multiple sets of transcripts; although these systems will not annotate the data perfectly, they may be able to do so more consistently than human annotators in certain cases. A reduction in annotation variability leads to more reliable annotations, which is a critical step in improving quality of care and outcomes (Ransom et al., 2005).

Automatically annotating transcripts for LSA may seem like a straightforward task, similar to others in NLP: after selecting an annotation system, just use existing techniques to automate its various components. In practice, however, the annotations used for LSA vary quite substantially between groups, and we explore many of these differences in this thesis. As a result, every component of an automated system to annotate transcripts for LSA must be trainable. This requirement, as well as several other characteristics of the data analyzed in LSA, prevent existing techniques in grammatical error detection from being used for LSA. In this thesis we propose and compare several systems for grammatical error detection, all of which are trainable. In particular, we evaluate systems based on classifiers, taggers, dependency parsers, and random walks to perform error detection in Chapter 6.

LSA is a widely-known, but rarely used technique, and little is known on what can and cannot be gleaned from analyzing transcripts with techniques falling under the LSA umbrella. In this thesis, we address this deficit first by investigating whether certain features derived from transcripts with manual LSA-style annotations can be used to predict scores on widely used structured instruments for language assessment, which, compared to LSA, are straightforward to administer and evaluate. We also investigate whether these features can identify the presence of two potentially comorbid developmental disorders: specific language impairment, and autism. When features derived from manually annotated transcripts prove useful, we compare their effectiveness to the same features derived from transcripts annotated using the automated systems we propose. These investigations can be found in Chapter 7.

The contributions of this thesis are:

1. We adapt an existing system for disfluency detection to maze detection. We show that the modified system's performance on maze detection is comparable to the original system's performance on disfluency detection. (Chapter 5)

2. We propose several novel techniques to explore the consistency of maze annotations in the SALT corpora, and we identify some major inconsistencies. We find that these inconsistencies are severe enough to prevent us from training a model for maze detection that is appropriate for all corpora. (Chapter 5)

3. We propose several automated, trainable systems to identify certain SALT grammatical error codes, specifically those which can be identified by looking at an utterance in isolation. Some of the systems we propose outperform existing systems for similar tasks. In particular, we find that

two systems, one based on conditional random fields, and another based on dependency parsing, perform particularly well. (Chapter 6)

4. We identify major inconsistencies in the quality of error code annotations in the SALT corpora. These inconsistencies include: different corpora using different sets of error codes; particular error codes being used differently in different corpora; and a high rate of unannotated errors in certain corpora. (Chapter 6)

5. We investigate whether features derived from transcripts with manual SALT annotations can predict scores on structured instruments. When they are able to do so, we evaluate whether automatically applied SALT annotations are as informative. We find that features derived from SALT annotations, whether manually or automatically produced, have some predictive power. (Chapter 7)

6. We investigate whether features derived from transcripts with manual SALT annotations can predict the presence of autism or a language impairment. When they are able to do so, we evaluate whether automatically applied SALT annotations are as informative. We find that SALT-derived features are useful for predicting the presence of a language impairment, but not autism. (Chapter 7)

In short, we highlight some challenges in applying techniques from NLP to clinical data, and we propose several techniques for addressing these challenges. The primary challenge that we identify is variable annotation quality: as we investigate in detail, both mazes and grammatical errors are annotated differently in different corpora, even though these corpora are ostensibly annotated following the same standards. A secondary challenge is that systems to automatically produce SALT maze or error code annotations should be tunable in terms of

precision and recall. The systems we propose for maze and error detection are all trainable, and are therefore not tied to a particular annotation standard, and they are tunable, thus enabling them to be used for a wide variety of downstream purposes.

Finally, Table 1.1 contains a list of acronyms and initialisms used in this thesis.

ADOS	autism diagnostic observation schedule
ALI	autism with impaired language
ALN	autism language 'normal' (unimpaired)
ASD	autism spectrum disorders
CCC	Children's Communicative Checklist (Bishop and Volkmar, 2003)
CELF	clinical evaluation of language fundamentals (Semel et al., 2003, 2004)
CRF	conditional random field
LI	language impaired
LN	language 'normal' (unimpaired)
LSA	language sample analysis (never latent semantic analysis here)
M^3N	Maximum-margin Markov network
NLP	natural language processing
POS	part of speech
SLI	specific language impairment
TD	typically developing
VIQ	verbal intelligence quotient (IQ)

Table 1.1: Acronyms and initialisms used in this thesis

Chapter 2

Clinical Background

This thesis focuses on extending and developing techniques from Natural Language Processing to automate components of *language sample analysis* (LSA), which is a collection of techniques designed to aid in the identification of developmental disorders. LSA stands in contrast to *structured instruments*, which are easy to use and administer, but potentially limited in what sorts of linguistic phenomena they can elicit. LSA is also used in research characterizing the deficits in communication caused by various disorders. In this chapter we first present an overview of how language is assessed in a clinical setting. We then give a detailed overview of the SALT annotation system, which is the *de facto* standard annotation system for LSA, and which we focus on automating components of in Chapters 5 and 6. Finally, we turn our attention to two neurodevelopmental disorders that involve impaired communication: language impairments and autism spectrum disorders (ASD). These disorders are of particular interest to us both because of the ways in which they impair communication, and also because we have a corpus of manually transcribed and annotated transcripts collected from children with confirmed diagnoses of ASD and language impairments (which we

present in detail in Section 3.2).

2.1 Assessment of language

Assessment of a child's language is typically performed either for diagnostic purposes or for evaluating a child's linguistic skills longitudinally, for example to see whether a particular intervention has been effective. There are two broad categories of tools for assessing a child's language, which we discuss in turn: structured instruments, and LSA.. Both have advantages and disadvantages, although as we discuss below, structured instruments suffer from certain inherent disadvantages. On the other hand, some issues with LSA can be addressed with technology, and remaining ones may be addressable with more research.

Structured instruments elicit brief, easy to score, responses to a sequence of items. For example, the CELF-4 includes nineteen multi-item sub-tests with tasks such as object naming, word definition, reciting the days of the week, or repeating sentences (Semel et al., 2003).

Structured instruments for language assessment are widely used, and two major benefits of them are that they are reliable, and their results are easy to interpret. Nevertheless, their applicability is rather limited. First, such instruments are typically not applicable to populations that use non-standard dialects or second-language learners. For example, Ellsworth and Fuse (2008) noted that the 'formulating sentences' section of the CELF-4 is particularly inappropriate for speakers of African American Vernacular English (commonly referred to as AAVE). During this task, the child is presented with a picture, and the examiner gives the child a word to use in a sentence that she produces. The sentence is then scored as incorrect, partially correct, or correct. Some examples of sentences that are grammatical in AAVE, but which are deemed incorrect in the CELF-4 include:

1. There is children in the living room.

2. They playing a video game.

Even when used with their intended population, structured instruments cannot capture all aspects of language. For example, they do not capture pragmatic abilities or repetitiveness in conversation, both of which are relevant to ASD, because conversations cannot follow an exact script. As a result, there is a growing consensus among clinicians that LSA, which involves transcribing and analyzing samples of natural language, should be used to augment structured instruments. For example, Tager-Flusberg et al. (2009) recommend using LSA to analyze a child's phonological, grammatical and lexical abilities, as well as their pragmatic and communication skills. In particular, they note that pragmatic and communication skills are quite difficult to assess using structured instruments. They also note that transcription and coding standards need to be tailored to the phenomena of interest, and this is generally possible due to the flexible nature of systems for LSA such as SALT (described below in Section 2.2).

In practice, clinicians use *semi-structured* instruments, which provide standardized situations and activities to elicit language, to collect language samples for analysis. For example, the Autism Diagnostic Observation Schedule (Lord et al., 2002) (discussed in Section 3.2.1) includes a task in which the child tells a story based on a picture book without any words, and a conversation about the child's friends. As far as we are aware, LSA is not performed with transcripts of arbitrary conversations or speech collected from a child.

Structured instruments may not capture as many aspects of language as semi-structured instruments and LSA, but the power of LSA comes at a cost: it is expensive to carry out, and interpreting the results of LSA is far more complicated than interpreting those of structured instruments. First, LSA must be carried out by trained annotators. Even optimistic assessments arguing that LSA is not

expensive still estimate approximately five minutes of transcription and analysis per minute of speech (Miller et al., 2011, p 7) under ideal circumstances (high quality recording, familiar circumstances, intelligible and fluent speech). Second, as we illustrate in Chapters 5 and 6, although there are corpora of annotated transcripts available for comparison, annotation standards vary widely. As a result, the validity of many metrics computed from these corpora as references or norms for comparison is suspect. Third, only some of the metrics used in LSA are well established, for example mean length of utterance in morphemes (MLUM) (see Gorman et al., 2015, for an overview). Other metrics, for example those related to mazes (similar to disfluencies in the linguistics and NLP literature, discussed in Section 2.2.3), are known to be informative, but annotation standards are not universal. Finally, the utility of error codes, and how to compare them across speakers, remains an open question, but one which we begin to address in Chapter 7.

2.2 SALT

The *Systematic Analysis of Language Transcripts* (SALT) system for LSA consists of four components, each of which we discuss in this chapter: 1) elicitation of language samples (Section 2.2.1), 2) transcription conventions (Section 2.2.2), 3) annotation conventions (Section 2.2.3), and 4) software analysis to analyze manually annotated transcripts, and compare them to databases of other transcripts (Section 2.2.4). Throughout this thesis, when we refer to 'standard' SALT annotations and conventions, we simply mean those described in the reference manual (Miller et al., 2011), or in the online training material (SALT Software, 2014d); all annotations and conventions not found in either of these sources will be referred to as 'non-standard'.

The SALT manual outlines several objectives for eliciting the optimum spoken

language sample (Miller et al., 2011, pp. 12-13):

1. Provide maximum information about the speaker's language
    - Vocabulary, syntax, semantics, and discourse
    - Structure and organization
    - Fluency, efficiency, and accuracy
2. Motivate the speaker to do their best talking
    - Age appropriate
    - Attentive listener or conversational partner
3. Identify speaker's oral language strengths and weaknesses within:
    - Community
    - School
    - Workplace
    - Family
4. For school-aged children, clearly demonstrate the student's difficulties with functional language regarding:
    - Classroom curriculum
    - State-wide oral language standards
    - Social language use
5. Optimize opportunity to interpret results
    - Follow the relevant elicitation protocol
    - Adhere to SALT transcription conventions

- Where possible, compare performance to typical speakers (SALT databases).

In this thesis, we distinguish between *transcription* and *annotation*, although in the SALT guidelines these tasks are referred to together as 'transcription'. Here, we use *transcription* to mean the act of writing down the child and examiner's speech, while *annotation* refers to the application of maze annotations, bound morpheme annotations, and grammatical error codes, none of which are standard in written language. Although most of the criteria above concern the elicitation step, we are primarily concerned with annotation. We include background on elicitation and transcription because we will use the results of these processes, *i.e.* the transcript, throughout the thesis.

2.2.1 Elicitation

SALT does not place any restrictions on the *elicitation* of language samples, or how one goes about collecting a language sample from a child. The SALT manual does, however, provide an overview of some relevant considerations. One major consideration is the task used to elicit language. The tasks used to elicit the SALT corpora are: conversation while playing (with toys or not); ordinary conversation; narrative tasks, which involve telling or retelling a story; and expository tasks, in which one presents a subject, for example describing how to play a favorite game. A related consideration is whether the task itself is appropriate for that child. For example, an expository task would almost certainly be too complicated for a 5 year old. Other considerations include how much language needs to be elicited, and whether there are comparable transcripts in the SALT corpora, should one wish to compare summary statistics from a transcript to the reference ones. We refer the reader to Chapter 2 of the SALT manual (Miller et al., 2011) and Rice et al. (2010) for more information on

elicitation. Nevertheless, elicitation is somewhat peripheral to our investigations, as we do not perform any ourselves.

2.2.2 Transcription

Transcription is the process of converting speech into text. In the case of SALT, it is critical to establish rigorous conventions for transcription because the transcripts are eventually analyzed quantitatively (see Section 2.2.4).

Words

In general, the SALT guidelines specify that words should be separated by spaces, but the spelling conventions themselves are left to the researcher. The guidelines note the importance of following the same spelling conventions in all transcripts. This is particularly important because the SALT software counts the number of word types, and variable spelling can inflate the word type count. For example, if the word three is transcribed both as 'three' and 'twee', presumably to reflect pronunciation, it will be counted as two tokens, which may not be appropriate.

Partial words are indicated by '*', although the SALT guidelines give no instructions on how to distinguish partial words from non-standard pronunciations, for example:

(1) C I have a po*.
E Potato?

The SALT software treats any string of letters delimited by spaces as a word. This can lead to inflated word and morpheme counts in some cases. For example, "I read The Chronicles of Narnia" has twice as many words as "I read Lolita", but this is simply due to the fact that one book title is longer than the other. The SALT guidelines instruct transcribers to join certain phrases words with underscores to count them as single words to avoid inflated word and

morpheme counts. In particular, the guidelines specify that proper names, titles, and repeated words or phrases should be joined, for example:

(2) C The_chronicles_of_Narnia.

(3) C Qaboos_bin_Said_Al_Said_of_Oman.

SALT marks single unintelligible words with X, unintelligible segments of unspecified length with XX, and unintelligible utterances with XXX:

(4) C She took the X to the park.

(5) C We XX dog.

Sound effects are indicated with '%'. The same convention is also used to mark "idiosyncratic forms", which the SALT manual defines as "immature productions which are consistent in reference to an object, person, or situation." (Miller et al., 2011, p 289):

(6) C The dog went %woof_woof.

(7) C I want a %coopa. NB: 'coopa' is an idiosyncratic form that means 'cookie'

Utterances

SALT requires transcribers to segment speech into *c-units*, each of which is "an independent clause with its modifiers" (Loban, 1976). In other words, a c-unit is an independent clause and all subordinate clauses. Utterances are written one per line, and each utterance is preceded by a single character indicating the speaker (C for child, E for examiner). For example:

(8) C I saw them when I left the house.

(9) C I do/n't know where they went.

Independent clauses linked with coordinating conjunctions (including 'so' in some cases) are counted as different C-units:

(10) C The frog was sit/ing on a lily pad.

(11) C And then it jumped in.

Yes/no responses are counted as single C-units:

(12) E Is that the Spanish teacher?
C No.
C That/'s the Science teacher.

Certain phrases, which SALT refers to as 'tags' (in bold below), are not counted as distinct C-units:

(13) C They miss/ed the bus, **did/n't they**?

(14) C He's gonna live with his dad, **I guess**.

If a C-unit contains a quote, all C-units within the quote *after* the first, are treated as separate C-units:

(15) C And he said "I/'m ready".
C "I want to go to the store now".

Finally, every utterance must end with one of the seven punctuation marks shown in Table 2.1. SALT does not provide other guidelines on punctuation use within utterances, although typically only commas and quotation marks are used.

We note that there are certain aspects of SALT transcription that are not relevant to this thesis, for example transcribing pauses and non-speech actions (ex. a child pointing to something). For more details on these, we refer the interested reader to the SALT manual and the SALT online training materials.

Mark	Description	Example
.	Statement/comment; not for abbreviations	C He left.
?	Question	C Where is she?
!	Surprise/exclamation	C There it is!
~	Intonation prompt	E And then you go to ~
^	Interrupted utterance	E What is^
		C A dog!
>	Abandoned utterance	C I was just >

Table 2.1: Utterance-final punctuation used in SALT

Speech is segmented into c-units in other corpora as well. The CHILDES style manual (MacWhinney, 2015), which provides the guidelines for annotating the eponymous collection of child's speech and language corpora, prescribes that utterances be split into c-units.[1] C-units are a sensible way of segmenting utterances in the CHILDES corpus, and one major reason for this is that c-units are particularly useful for computing informative mean length of utterance values. Nevertheless, c-units are not universal: the Switchboard corpus of telephone conversations between adults is typically segmented into conversational turns (Wheatley et al., 1995), although some have explored using alternative forms of utterance segmentation. For example, Deshmukh et al. (1998) explored using more acoustic cues to re-segment the Switchboard corpus to improve the performance of an acoustic model for speech recognition.

2.2.3 Annotations

SALT contains three broad categories of annotations: ones to delimit bound morphemes; *mazes*, which are typically referred to as disfluencies in the NLP

[1]The CHILDES manual claims that a c-unit "is defined as a main clause along with its dependent (subordinate or coordinate) clauses" (MacWhinney, 2015, p 57), and further specifies that "utterances [in CHILDES] should not include multiple main clauses", implying that they are not using standard c-units. In fact, the definition used in CHILDES, in which coordinate clauses are separate c-units, is no different from the the standard c-unit Loban (see 1970, p 9).

literature; and error codes, which are used to identify, and optionally classify, grammatical errors. Unlike morphological and maze annotations, there are some error codes used in the SALT corpora (and the CSLU ADOS corpus) that are not described in the SALT manual or online training materials. Again, we refer to these codes as *non-standard*, and those codes that are included in the official SALT resources as *standard* error codes.

SALT is not the only system for annotating disfluent speech, grammatical errors, or morphemes. In particular, the CHAT annotation guidelines (MacWhinney, 2015), which are used to annotate the CHILDES corpora of children's speech and language, provide annotations for all three of these categories. Interestingly, one of the SALT corpora, namely ENNI, was originally annotated with CHAT annotations, and then these were converted to SALT annotations. As noted below, the CHAT annotation guidelines are more thorough than SALT ones. In Chapters 5 and 6 we find that the quality of both the maze and error code annotations in the ENNI corpus is higher than in the other SALT corpora. It is quite possible that this is due to its having been annotated originally with CHAT, but since it is the only such corpus we cannot say so conclusively.

Morphological annotations

SALT prescribes the annotation of certain inflectional morphemes and clitics, all of which are shown in Table 2.2. These annotations are all delimited by a slash '/', with the dictionary form of the root word to the left of the slash, and the morpheme or clitic code to the right (ex. *babies* would be annotated as *baby/s*, not *babie/s*). In all cases, the bound morpheme must be visible in the surface form (ex. plural *-s* in *dogs*, but not in *mice*), and therefore irregular inflections do not receive any morphological annotations in SALT.

The morphemes annotated following SALT conventions are all visible in surface forms, which simplifies morphological annotations, particularly compared

Morpheme	Annotation	Notes	Example
Noun plural -s	/s	Not marked on words representing single entity	dogs → dog/s pants → pants mice → mice
Possessive -'s	/z	Not marked on possessive pronouns	dog's → dog/z hers → hers
Possessive plural -s'	/s/z	babies' → baby/s/z	
Past tense -ed	/ed	Only on main verbs, not past participles	cried → cry/ed is closed → closed
3^{rd} person singular -s	/3s	Irregular forms are not annotated	plays → play/3s does → does
Progressive -ing	/ing	Only for progressive construction	is cooking → cook/ing went jogging → jogging
Negative clitic -n't	/n't or /'t	Irregular forms (typically involving vowel changes) are not annotated	doesn't → does/n't won't → won't NB: can't → can/'t
Contracted 'be'	/'s, /'m, /'re		we're → we/'re
Contracted forms of 'will', 'would', 'have'	/'ll, /'d, /'ve		she'll → she/'ll
'Non-standard' contractions	/h's, /d's, /d'd, /'us	"the SALT database samples were not coded for these non-standard contractions" (SALT Software, 2014b)	It's → it/h's been What's → What/d's he do? What'd → What/d'd he do? Let's → Let/'us go.

Table 2.2: SALT Morpheme Annotations

to the ones used in CHAT. Compared to SALT, CHAT distinguishes between far more morphemes: in addition to all of the bound morphemes and clitics included in SALT, CHAT also prescribes the annotation of adjective suffixes (ex. -er, -est), participial suffixes (-en, -ing), derivational affixes (un-, -ly, -er, over-, pro-, etc.) and others (MacWhinney, 2015, pp 114-115). CHAT also has conventions for annotating irregular inflections, which is explicitly missing from SALT: mouse is coded with a plural annotation in CHAT, and similarly 'was' is to be annotated with both past tense and first-or-third-person-singular annotations. In SALT, neither of these words would receive any morphological annotations.

Maze annotations

The SALT manual defines *mazes* as "filled pauses, false starts, repetitions, and reformulations" (Miller et al., 2011, p 288). More information about each of these categories is given in the online training material from SALT Software, specifically in Course 1304. It is critical to note that none of the SALT materials provides a definition of any of the constituents of mazes, for example reformulations, nor do they reference a more rigorous definition of any of them. Partial words are always included in mazes except for when they occur at the end of an utterance. Finally, all contiguous spans of words in mazes are annotated with a single set of parentheses. Chapter 5 deals extensively with mazes, including detecting them automatically, and the quality of maze annotations in the SALT corpora.

Repetitions and Revisions The online SALT training course states that any part of an utterance that is repeated or revised should be annotated as a maze. Annotators are instructed to consider the earlier instances of repeated words as mazes. In terms of the Switchboard disfluency annotations (Meteer et al., 1995, p 24) (and described in Section 4.5.1), maze annotations capture reparanda and interregna, but not repairs or interruption points. Here are examples of properly

annotated revisions and repetitions (taken from SALT Software (2014b)):

(16) C And (you you) you can come.

(17) C (And it almost and it) and it almost took her.

(18) C (That/'s) this is the one.

Filled Pauses SALT provides a fixed list of filled pauses: ah, eh, er, hm, hmm, uh, um. Words can be added to this list manually so that they are always identified as a filled pause when they occur inside a maze, and individual tokens can optionally be marked with `[FP]` to indicate that it is being used as a filled pause, as in the utterance (from SALT Software 2014b):

(19) C And then (he like[FP]) he said he was (like[FP]) sorry.

Filled pauses must be included within a maze unless it is "spoken as an affirmation, negation or interrogation" (SALT Software, 2014b). This is illustrated in the following example (from SALT Software 2014b), in which the examiner says 'um' to encourage the child to continue, or presumably as what the online materials refer to as an 'interrogation':

(20) C We bake/ed cookie/s.
E **Um**.
C Then we ate all of them.

The SALT annotations are far simpler than the Switchboard annotations (discussed in Section 4.5.1), which distinguish between filled pauses (indicated with *{F ...}*), explicit editing terms (*{E ...}*) among other non-sentence elements, and revisions and restarts. All of these distinctions are collapsed in SALT maze annotations.

Stuttering in the middle of a word Stuttering in the middle of a word is delimited with underscores, with the repeated material included as partial words. The material included as partial words must always be annotated as within a maze; the rest of the word may or may not be in a maze, depending upon the context. For example (from SALT Software 2014b):

(21) C He ate a green ap_ (p* p*) _ple.

(22) C He ate (an ap_ p* p* _ple) a green apple.

As discussed in Section 4.5, the Switchboard corpus uses a much richer system of annotations for difluencies, which are quite similar to mazes, although much more rigorously defined. The CHAT manual also includes guidelines for annotating disfluencies (MacWhinney, 2015, pp 73-75). These guidelines are more thorough than the cursory ones included in SALT, and the annotation system itself is far more complicated, distinguishing between repetitions, 'retracing', 'multiple retracings', 'reformulations', and other sorts of disfluencies. In sum, the SALT guidelines provide the simplest schema for annotating disfluent speech. As we shall see in Chapter 5, however, even this simple schema is not applied consistently by different research groups, suggesting that either quality control may be an issue, or that the guidelines themselves should be made more rigorous.

Error codes

In no circumstances is the language transcribed in SALT transcripts 'corrected': grammatical errors are written as they are produced. The SALT materials provide suggested *error codes* to annotate grammatical errors. We present the set of error codes included in the SALT online training material SALT Software (2014b) in Table 2.3, along with examples of how they are used. For consistency with the other error codes, we represent the omitted morpheme and word errors with `[OM]` and `[OW]`, respectively throughout this thesis. Unlike the '*' notation,

Code	Explanation	Examples
[EO:_]	Inflectional overgeneralization; includes correction	C He falled [EO:fell].
[EU]	Utterance level errors; also replaces >2 omission/[EW] errors	C And they came to stopped. [EU]
[EW:_]	Word-level error; may include correction	C She is a [EW:an] athlete. He is a [EW] sleep/ing.
/*	Omitted morpheme	C The car go/*3s fast.
*	Omitted word	C Give it *to me.

Table 2.3: Error codes in SALT Software (2014b)

but in line with the other SALT error codes, our notation does not require annotators to posit a correction.

It is important to note that adding error codes to a transcript is not a requirement for SALT transcription, and thus error codes are not present in every valid SALT-annotated transcript. For example, the GillamNT corpus, described in Section 3.1.4 has only four error codes in the entire corpus.

The set of error codes in Table 2.3 is by no means the definitive set of error codes: researchers may elect to use only a subset of these, or to create their own error codes. For example, the CSLU ADOS corpus, described in Section 3.2 contains the following non-standard error codes: [EC] for 'inappropriate response' ('Do you like ice cream?' 'Fight [EC]'), [EX] for extraneous word ('I showed the [EX] him'), and [EP] for pronominal errors ('Him [EP] left.').

The most relevant alternative schema for coding grammatical errors is the one described in the CHAT manual (MacWhinney, 2015, pp 104-109) as it is the only one of which we are aware that was designed for errors found in spoken language, as opposed to written language. The CHAT manual defines codes that capture errors at many different levels of language, including phonological, morphological, semantic, and utterance-level errors. Furthermore, the CHAT

error codes are far more specific than the SALT error codes. For example, rather than using a generic word-level error code ([EW]), CHAT distinguishes between case errors ('I saw he'), superfluous morphemes ('ranned' for 'ran' or 'going' for 'go'), and otiose plural morphemes ('kniveses') among many others. Another major difference between the CHAT and SALT standards for error codes is that users of CHILDES are not encouraged to create new error codes, while users of SALT are, should the need arise. From a computational perspective, the CHAT standard would appear to be far preferable to SALT: the errors are more specific and better defined, which in turn should make them easier to learn. Although we do not investigate CHAT-annotated data in this thesis, future work should investigate how effectively CHAT error codes can be automated, particularly in comparison to SALT error codes.[2]

2.2.4 Analysis and comparison

The SALT system for LSA includes software can be used to analyze a transcript by computing various summary statistics. These summary statistics are found by counting occurrences in manually annotated SALT transcripts. The SALT software can also select a reference sample of transcripts from the SALT corpora, and then derive these same summary statistics so that the transcript of interest can be compared to this reference. These samples can be selected using various criteria, including task, age, gender, and grade level. Here we provide a brief overview of some of the statistics computed by the SALT software. For more details we refer the reader to Miller et al. (2011). We do not reference the SALT software elsewhere in this thesis, but we do note that the variable standards and quality of annotations (in particular maze and error code annotations) in the

[2]We note that the ENNI corpus was originally annotated following the CHAT standards, but we only consider the SALT-annotated version here. In Chapter 6 we find that the quality of the error code annotations in the ENNI corpus is substantially higher than those in any of the other SALT corpora. Perhaps this could be a result of it originally being annotated following the more thorough CHAT standards.

SALT corpora undoubtedly affect these reference statistics.

The reference statistics computed by the SALT software that are relevant to this thesis are:

- Transcript length in utterances and words
- Mean length of utterance (MLU) in words and morphemes (from morphological annotations described in Section 2.2.3
- Token and type count, type-token ratio
- Percentage of intelligible utterances (those excluding X, XX and XXX tokens)
- Maze statistics: utterances with mazes, number of mazes, number of words in mazes, percentage of all words in mazes
- Error statistics: omitted words and morphemes, word-level error codes, utterance-level error codes

For the transcript of interest, the SALT software simply computes a raw count or ratio. If the transcript is compared to transcripts from a reference SALT corpus, then for each of the reference statistics above, the software reports the mean, minimum, maximum, and standard deviation, taken over the selected transcripts in the SALT corpus. We now turn our attention to the disorders that impair language, as LSA in general, and SALT in particular, is not simply a task in NLP, but rather a tool for diagnosing and characterizing these disorders.

2.3 Neurodevelopmental disorders

The DSM-V describes neurodevelopmental disorders as "[manifesting] early in development, often before the child enters grade school, and [being characterized]

by developmental deficits that produce impairments of personal, social, academic, or occupational functioning." (American Psychiatric Association, 2013, p 31). Many neurodevelopmental disorders involve impaired communication, although the nature of these impairments varies widely. To illustrate, pragmatic communication disorder involves difficulties in the social use of language (and nonverbal communication), such as following conventions in greeting or turn-taking (American Psychiatric Association, 2013, p 47), while childhood-onset fluency disorder (commonly referred to as "stuttering") involves impaired speech production, but not social difficulties. Neurodevelopmental disorders are frequently comorbid, meaning that a child may often have symptoms of more than one of them. For example, many individuals with an autism spectrum disorder also have impaired language. Finally, diagnosing neurodevelopmental disorders accurately and at an early stage is critical, as appropriate early intervention tends to be more effective than later intervention for various neurodevelopmental disorders, including both language impairments (Gillon, 2000, 2002) and ASD (e.g. Rogers, 1998; Lovaas and Smith, 2003; Eldevik et al., 2006). In this thesis, we are particularly interested in two neurodevelopmental disorders, namely specific language impairment and ASD, because they both impair language, and because we have a corpus of transcripts of spoken language collected from children with confirmed diagnoses of both of these disorders (the CSLU ADOS corpus, described in Section 3.2).

We note that children who do not have any sort of neurodevelopmental disorder are described as 'typically developing' (TD). The SALT and CSLU ADOS corpora have various criteria for determining which children are TD, as described in Chapter 3.

1. Persistent difficulties in the acquisition and use of language across modalities (i.e., spoken, written, sign language, or other) due to deficits in comprehension or production that include the following:
   (a) Reduced vocabulary (word knowledge and use).
   (b) Limited sentence structure (ability to put words and word endings together to form sentences based on the rules of grammar and morphology).
   (c) Impairments in discourse (ability to use vocabulary and connect sentences to explain or describe a topic or series of events or have a conversation).
2. Language abilities are substantially and quantifiably below those expected for age, resulting in functional limitations in effective communication, social participation, academic achievement, or occupational performance, individually or in any combination.
3. Onset of symptoms is in the early developmental period.
4. The difficulties are not attributable to hearing or other sensory impairment, motor dysfunction, or another medical or neurological condition and are not better explained by intellectual disability (intellectual developmental disorder) or global developmental delay.

Table 2.4: Diagnostic criteria for a language disorder taken from American Psychiatric Association (2013, p 42)

2.3.1 Language disorder and specific language impairment

The DSM-V (American Psychiatric Association, 2013) lists four major criteria involved in the diagnosis of a language disorder, and they are outlined in Table 2.4. We are particularly interested in *specific language impairment* (SLI), which can be considered a particular type of language disorder. SLI has received a great deal of attention in research, but it is not included in the DSM-V for a few reasons, including a lack of consensus on the reliability and robustness of the diagnosis itself. For example, nonverbal IQ is required to make a diagnosis of SLI, but even if formal testing is available, it is not necessarily appropriate: culturally appropriate tests are not available for speakers of nonstandard dialects or learners of English as a second language, thus excluding many children from a potential diagnosis of SLI (American Speech-Language-Hearing Association,

2012).

Although SLI is not included as such in the DSM-V, it is typically diagnosed in a similar manner to a language disorder by clinicians and researchers using the following criteria (see ex. Tomblin et al., 1997; Hill et al., 2015):

1. Nonverbal or performance IQ above a minimum level (typically 80 or 85)
2. Concurrent standard language test score at least a certain amount below the mean (typically 1.0 or 1.5 standard deviations)
3. No physical condition that would account for a communication deficit (ex. sensory or hearing loss, cleft palate, brain lesion)
4. No metabolic, genetic, or neurological condition that would account for a communication deficit (ex. an autism spectrum disorder)

Most research involving individuals with SLI tends to exclude speakers of non-standard dialects and second language learners, thereby addressing (and supporting) the ASHA's criticism of the SLI diagnosis. Even though these criteria are quite straightforward, it can be difficult to disentangle language impairments from comorbid disorders, for example deficits in attention, motor functioning, or learning ability. Goorhuis-Brouwer and Wijnberg-Williams (1996) found that these complicating factors led them to revise the diagnosis of 75% of children in a study involving 319 children diagnosed with SLI.

Individuals with SLI suffer from a wide variety of linguistic deficits. Some of the most salient of these deficits are morphological, for example difficulties with tense marking (Eadie et al., 2002; Leonard et al., 1997), subject-verb agreement (Leonard et al., 1997), and pronominal case (Charest and Leonard, 2004), or frequently omitting auxiliary verbs (Grela and Leonard, 2000). Clifford et al. (1995) found that children with SLI were able to form a coherent narrative, with events organized in the correct sequence, suggesting that SLI impacts production

of language, as opposed to other processes, for example social knowledge. For a thorough review of research characterizing the linguistic deficits observed with SLI, we refer the reader to Leonard (2014), in particular Part II, Chapter 3.

2.3.2 Autism spectrum disorders

Autism spectrum disorders (ASD) encompass a wide range of disorders that share a few common features, most notably impaired social communication and interaction, and restricted or repetitive interests, behaviors or activities (American Psychiatric Association, 2013). These symptoms vary widely in severity across individuals with autism. For example, social communication or interaction can be mildly impaired, as when an individual has difficulty initiating social interactions and carrying a conversation, but is able to speak in complete sentences. On the other hand, social interaction can be severely impaired, with the individual only responding to others with only a few intelligible words, and only when addressed directly. Similarly, the severity of restricted or repetitive interests can manifest itself mildly, as when an individual has difficulties switching between activities, or severely, as when repetitive behaviors such as finger flicking or hand flapping interfere with all activities. In this thesis we focus on the verbal communication deficits that are indicative of autism, and we refer the reader to the DSM-V (American Psychiatric Association, 2013) for a more thorough overview of autism spectrum disorders, including its diagnosis.

Autism can affect language in a variety of ways, although there is a great deal of variation between individuals. Two linguistic impairments characteristic of autism are *echolalia*, repeating words or phrases (Tager-Flusberg et al., 2005, 2009), and *neologisms*—novel words (ex. 'hana rra' to mean 'good') (Volden and Lord, 1991). Pragmatic deficits are also common in individuals with autism (Cromer, 1981; Tager-Flusberg, 1985). Individuals with autism may

also have difficulties with receptive language (Tager-Flusberg, 1981), possibly as a result of social deficits (Lord, 1985). Difficulties with prosody, both expressive (Shriberg et al., 2001) and receptive (Koning and Magill-Evans, 2001), are also present among individuals with autism. Finally, although there are individuals with autism who do not evidence grammatical impairments, the majority do (Kjelgaard and Tager-Flusberg, 2001). These are often similar in many ways to difficulties typical of specific language impairment, involving, for example, difficulty with complicated syntax, and morphology, in particular the past tense marker (-ed) (Roberts et al., 2004).

2.4 Conclusions

We have given a brief comparison of two ways of assessing language: structured instruments and LSA, along with an overview of the *de facto* standard tool for LSA, which is SALT. We have described the ways in which two neurodevelopmental disorders, namely SLI and ASD, can impair communication. Even though structured instruments are key to identifying SLI and language impairments in children with ASD, they are not effective for assessing all aspects of language. Furthermore, structured instruments elicit language in artificially simple settings, for example by asking children to list words, rather than in complex linguistic interactions such as a conversation. As a result, they provide a somewhat limited view of a child's language. On the other hand, even though LSA may be able to provide a richer view of a child's linguistic competence, it is more expensive to carry out than simply using structured instruments.

In Chapters 5 and 6, we will explore techniques to automate two of the key annotations in SALT, namely mazes and error codes. This work has the potential to expedite LSA. Less promisingly, our experiments in automating the application of these annotations also reveals weaknesses in the SALT guidelines,

and severe inconsistencies in the quality of annotations on the SALT corpora. Finally, in Chapter 7, we perform preliminary experiments to see just how useful SALT-annotated transcripts are for identifying language impairments and ASD, and whether they can be used to predict a child's score on structured instruments that assess linguistic ability.

Chapter 3

Data

This chapter provides an overview of the data we use in our experiments. We first present the SALT Corpora, which are included with the SALT software, and referred to as 'reference databases' in the SALT manual (Miller et al., 2011). We use the SALT corpora in our experiments on maze and SALT error code detection, presented in Chapters 5 and 6, respectively. Next we give an overview of the CSLU ADOS corpus, which we use in experiments predicting scores on structured instruments and discriminating between different diagnoses in Chapter 7. Finally, we discuss how these transcripts are processed for various experiments.

3.1 SALT Corpora

The SALT software includes nine reference databases, which we will refer to collectively as the *SALT corpora*. Table 3.1 shows where each was collected, the age ranges of the speakers, and the size of each corpus both in terms of transcripts and utterances. Note that only utterances spoken by the child are counted, as we throw out all others. One group of corpora comes from New

Corpus	Transcripts	Utterances	Age Range	Speaker Location
Conv	584	82,643	2;9 – 13;3	WI & CA
ENNI	377	56,108	4;0 – 9;11	Canada
Expository	242	4,918	10;7 – 15;9	WI
GillamNT	500	40,102	5;0 – 11;11	USA
NarSSS	330	16,091	5;2 – 13;3	WI & CA
NarSR	500	14,834	4;4 – 12;8	WI & CA
NZConv	248	25,503	4;5 – 7;7	NZ
NZPerNar	248	20,253	4;5 – 7;7	NZ
NZSR	264	2,574	4;0 – 7;7	NZ

Table 3.1: Description of SALT corpora

Zealand, while the majority come from North America. All of the corpora except for Expository include children at very different stages of language development.

Four research groups were responsible for the transcriptions and annotations of the corpora in Table 3.1. One group produced the Conv, Expository, NarSSS, and NarSR corpora. Another was responsible for all of the corpora from New Zealand. Finally, the ENNI and GillamNT corpora were transcribed and annotated by two different groups.

We provide details on each of the SALT corpora, including how they were collected and annotated, in turn. We also briefly describe any remarkable qualities of the SALT annotations that are apparent from either previously published descriptions of these corpora, or from a cursory inspection. Chapters 5 and 6 contain much deeper investigations into the SALT annotations in a few of the SALT corpora.

3.1.1 Conv Corpus

The Conv corpus is the largest of the four SALT corpora produced by the group at the University of Wisconsin, which also developed the SALT coding

system, and the SALT software. The other corpora produced by this group are: EXPOSITORY (Section 3.1.3), NARSSS (Section 3.1.5), and NARSR (Section 3.1.6). It contains 584 transcripts with a combined total of 82,643 utterances.

Participants

The CONV corpus contains transcripts of spoken language collected from children in both California and Wisconsin. The children in California were between the ages of 4 years 4 months and 9 years 11 months, while those in Wisconsin were between the ages of 2 years 9 months and 13 years 3 months. All of the participants were described as typically developing (TD), and this was determined using different criteria in California and Wisconsin. All of the children in California were of average ability as determined by performance on standardized tests, teacher reports, and the absence of special education services. These inclusion criteria were also used to determine that the participants in California were TD. In Wisconsin, participants came from a variety of ability levels, and these ability levels were determined by their teachers. These participants were determined to be TD based on their progress in school and the absence of special education services.

Elicitation

The documentation of the CONV corpus does not go into detail regarding the elicitation procedure. The description of the corpus simply states that the examiner is to have a conversation with the participant about one of the following topics, for which there are suggested prompts (Miller et al., 2011, p.190)

1. Classroom activities
    - "Tell me about some of the things you've been doing in school lately."
    - Ask about specific classroom units

2. Holidays
    - "Did you do anything special for Halloween (or appropriate holiday)?"
    - "Tell me about that."
    - "Are you going to do anything special for Christmas?"
3. Family activities, visits, locations, etc.
    - "Are you going to visit your grandma and grandpa?"
    - "Where do they live?"
    - "How do you get there"
    - "What do you do there?"
4. Family pets
    - "Do you have any pets at home?"
    - "Tell me about them."
    - "What do you have to do to take care of them?"

For young children, the examiners are simply instructed to elicit a language sample through play, for example by playing with playdough or small toys with the child and discussing the activity (SALT Software, 2014c).

Coding

Error	Count	% of utts.
[EO]	944	1.1%
[EU]	1,661	2.0%
[EW]	1,526	1.8%
[OM]	773	0.9%
[OW]	2,216	2.7%

Table 3.2: Counts of error codes in CONV corpus, which contains

The Conv corpus contains the five error codes shown in Table 3.2 along with the count of each code in the corpus. As with all other SALT corpora, omission errors are indicated with '*' prepended to the missing material (ex. *He like/*3s ice cream.*), but we convert all of these to either `[OM]` or `[OW]` for omitted morphemes or words, respectively. The set of error codes used in the Conv corpus is identical to the one used in the other three corpora compiled by the group at the University of Wisconsin (Expository, NarSSS, and NarSR).

3.1.2 ENNI Corpus

The ENNI corpus was produced by a research group at the University of Alberta, and it is the only SALT corpus produced by this group. It is the second largest SALT corpus, containing 377 transcripts with a total of 56,108 utterances.

Participants

Ages	Language group	N boys	N girls
4;0-4;11	TD	25	25
	SLI	9	3
5;0-5;11	TD	25	25
	SLI	8	6
6;0-6;11	TD	25	25
	SLI	6	5
7;0-7;11	TD	25	25
	SLI	10	3
8;0-8;11	TD	25	25
	SLI	10	7
9;0-9;11	TD	25	25
	SLI	5	5

Table 3.3: Summary of participants in ENNI database

The ENNI corpus was collected from 377 children between the ages of 4 years 0 months and 9 years 11 months old (Schneider et al., 2006). All of the children lived in or around Edmonton in Alberta, Canada. The authors of the ENNI

corpus put the children into two groups: typically developing (TD) and those with a specific language impairment (SLI). We note that these names appear to be conveniences, as diagnoses were not confirmed for either group. TD children were identified by asking public and separate school teachers in the Edmonton area for two children, one of each sex, from each of three achievement levels: low, intermediate, and high.

Children with SLI came from three places that serve children with language impairments: a public school that also serves children with learning disabilities, a rehabilitation hospital, and the Capital Health Authority. All of the children in the SLI group have language impairments that scored between 2 and 5 on the Capital Health's Severity Rating Scale, which goes from 1 (mild) to 5 (severe). Children were excluded from the SLI group if they had any of the following diagnoses: mental retardation, unmedicated ADD or ADHD, autism, hearing impairment, severe visual impairment that would result in inability to see pictures even with correction, or severe speech impairments. Children were not excluded from the SLI group if they had motor delays, medicated ADD or ADHD, a learning disorder, or a mild speech impairment.

Elicitation

The ENNI corpus contains transcripts of children telling a story to go along with a book of pictures, an activity commonly referred to as 'wordless picture book'. There were six different sets of pictures used throughout the corpus. The examiner prepared the child for the activity by having them tell a story for a practice set of pictures, and after that the child told stories for two different sets of pictures. For each set of pictures, the examiner began by showing the child each of the pictures. Next, the examiner held the pictures so that she could not see them, and more importantly, so that the child could not point to elements of them. Instead, the child had to describe the entire story verbally. The examiner

then asked the child to tell the story, turning the page when the child appeared to be ready. The transcripts are of the child telling the two non-practice stories.

Transcription

The samples of spoken language were originally transcribed and annotated following the CHAT transcription system used in the CHILDES database (MacWhinney and Snow, 1990; MacWhinney, 1992). The transcripts were converted from CHAT to SALT format, but the authors of the corpus did not specify whether this was done automatically or manually (Schneider et al., 2014).

Coding

Error	Count	% of utts.
[EU]	3,332	5.9%
[EW]	4,916	8.8%
[OM]	10	<0.1%
[OW]	766 1.4%	

Table 3.4: Counts of error codes in ENNI corpus

According to the SALT manual (Miller et al., 2011), there are only two error codes used consistently throughout the ENNI database: [EW:], indicating a word-level error (including the correction after the colon); and [EU], indicating an utterance-level error. The ENNI corpus also includes annotations for omitted words and a few annotations for omitted morphemes. Recall that we represent these errors with the codes [OM] and [OW], respectively, throughout this thesis, as discussed in Section 2.2.3. The error code counts in the entire ENNI corpus are shown in Table 3.4.

3.1.3 Expository Corpus

The Expository corpus is the smallest of the four SALT corpora produced by the group at the University of Wisconsin, which also developed the SALT coding system, and the SALT software. The other corpora produced by this group are: Conv (Section 3.1.1), NarSSS (Section 3.1.5), and NarSR (Section 3.1.5). It contains 242 transcripts with a combined total of 4,918 utterances.

Participants

The Expository corpus contains transcripts of spoken language collected from children in Wisconsin. The children were in middle and high school (grades 5-7, and grade 9), and were between the ages of 10 years 7 months and 15 years 9 months. There were 118 female participants, and 124 male participants. All of the participants were typically developing (TD). All of the participants were determined to be TD based on their progress in school and the absence of special education services. The participants came from a variety of ability levels, which were determined based on teacher reports and GPA.

Elicitation

The elicitation protocol for the Expository corpus is quite simple: the examiner reads a prompt asking the participant to explain whatever game or sport she chooses, the participant is given time to fill in a planning sheet, and then the participant responds to the prompt using the notes she wrote on the planning sheet. For the exact prompts, we refer the interested reader to the SALT manual (Miller et al., 2011, pp. 208-210).

Error	Count	% of utts.
[EO]	2	<0.01%
[EU]	109	0.3%
[EW]	459	1.1%
[OM]	30	<0.01%
[OW]	163	0.4%

Table 3.5: Counts of error codes in EXPOSITORY corpus

Coding

The EXPOSITORY corpus contains the five error codes shown in Table 3.5 along with the count of each code in the corpus. As with all other SALT corpora, omission errors are indicated with '*' prepended to the missing material (ex. *He like/*3s ice cream.*), but we convert all of these to either [OM] or [OW] for omitted morphemes or words, respectively. The set of error codes used in the EXPOSITORY corpus is identical to the one used in the other three corpora compiled by the group at the University of Wisconsin (CONV, NARSSS, and NARSR).

3.1.4 GILLAMNT Corpus

The Gillam Narrative Tasks (GILLAMNT) corpus was produced by a research group at the University of Texas at Austin, and it is the only SALT corpus produced by this group. It is the third largest SALT corpus, containing 500 transcripts with a total of 40,102 utterances.

Participants

The GILLAMNT corpus was collected from 500 children between the ages of 5 years 0 months and 11 years 11 months old (Miller et al., 2011). There were 50 participants in each of the following age groups: 5 year olds, 9 year olds,

10 year olds, and 11 year olds. There were also 100 participants in each of the following age groups: 6 year olds, 7 year olds, and 8 year olds. There were an equal number of male and female participants within each age group. The participants came from four regions around the US: Northeast, South, Midwest, and West, although the description of the GillamNT corpus does not define the boundaries of any of these regions. None of the participants in the GillamNT corpus had been identified as having any sort of disability, nor did any of them receive special education services.

Elicitation

The examiners administering the Gillam Narrative Tasks followed the scripts specified in the Gillam Test of Narrative Language guidelines (Gillam and Pearson, 2004). Each child completes three tasks. In the first, the examiner reads a story aloud to the child, asks the child comprehension questions, and finally asks the child to retell the story. Only the child's retelling is transcribed. The second task involves telling a story based on a sequence of five pictures: first the examiner tells a story based on a sequence of pictures, asks the child some questions about the story, and then asks the child to tell a different story based on a novel sequence of five pictures. Again, only the child's story is transcribed. The third task is very similar to the second, except that instead of a sequence of five pictures, the examiner and the child tell stories about a single picture.

Coding

Error	Count	% of utts.
[OW]	4	<0.1%

Table 3.6: Counts of error codes in GillamNT corpus

The GillamNT corpus does not contain error annotations: although cursory

manual inspection reveals there are grammatical errors in the corpus (ex. the overgeneralized past form 'comed' appears 12 times in the corpus), there are only four instances in which an omitted word (`[OW]`) error is annotated (with a prepended '*', as is customary).

3.1.5 NarSSS Corpus

Like the Conv (Section 3.1.1), Expository (Section 3.1.3), and NarSR (Section 3.1.6) corpora, the NarSSS corpus was produced by the group at the University of Wisconsin. It contains 330 transcripts with a combined total of 16,091 utterances.

Participants

The NarSSS corpus contains transcripts of narrative samples collected from children in Madison, Wisconsin between the ages of 5 years 2 months and 13 years 3 months. All of the participants were typically developing (TD), which the creators of the corpus determined with two criteria: lack of special education services, and normal progress in school. The children come from diverse economic backgrounds, which was determined by eligibility for the free lunch program at school. The children also have a variety of ability levels, which were determined by teacher ratings.

Elicitation

Examiners elicited a narrative from the child on one of the following four topics: 1) a movie the child had seen; 2) a book the child had read; 3) an episode of a TV show the child had seen; or 4) a familiar story such as Goldilocks or Red Riding Hood. Typically the fourth topic, retelling a familiar story, was reserved for younger kids. Examiners were also instructed to encourage the child to produce

more language or to encourage the child to move on in the story if she is having difficulties, using phrases such as 'tell me more', or 'what else?'. Examiners were expressly discouraged from helping the child tell the story, for example by providing missing vocabulary.

Coding

Error	Count	% of utts.
[EO]	488	3.0%
[EU]	540	3.4%
[EW]	1,151	7.2%
[OM]	177	1.1%
[OW]	431	2.7%

Table 3.7: Counts of error codes in NarSSS corpus

The NarSSS corpus contains the five error codes shown in Table 3.7 along with the count of each code in the corpus. As with all other SALT corpora, omission errors are indicated with '*' prepended to the missing material (ex. *He like/*3s ice cream.*), but we convert all of these to either [OM] or [OW] for omitted morphemes or words, respectively. The set of error codes used in the NarSSS corpus is identical to the one used in the other three corpora compiled by the group at the University of Wisconsin (Conv, Expository, and NarSR).

3.1.6 NarSR Corpus

The NarSR corpus, like the Conv, Expository, and NarSSS corpora, was produced by the group at the University of Wisconsin. It consists of 500 transcripts with a combined total of 14,834 utterances.

Participants

The NarSR corpus contains transcripts of spoken language collected from children in San Diego, California, as well as Madison and Milwaukee, Wisconsin.

All of the participants from Wisconsin are described as typically developing (TD), as determined by an absence of special education services, and normal progress in school. The participants from Wisconsin came from a variety of ability levels as determined by their teachers. The participants in California are described as being of average ability as determined by performance on standardized tests, teacher reports, and the absence of special education services. All of the participants came from a variety of economic backgrounds. In Wisconsin, this was determined by eligibility for the free lunch program at school, while in California, socioeconomic status was determined by the mother's highest level of education.

Elicitation

Two different protocols were used for elicitation, one for children in preschool, kindergarten, or first grade, and another for children in grades two through six. The protocol for the younger children begins with the examiner instructing the child that she will hear a story, and then she will have to retell it. A book is present for both tellings, although both the examiner and child are instructed not to read the story, but rather to loosely follow the pictures and the text. The examiner either reads the story, 'Frog, Where are You?', or plays a recording of someone reading it. Finally, the child retells the story. The protocol for older children is similar, although the story is different ('Pookins Gets her Way'), and the text is covered in the book when the child retells the story. During the first telling of the story, however, the child may either follow along with the book with text, or the one with out, as determined by the examiner. As in the elicitation of the NarSSS corpus, the examiner is instructed to provide generic encouragement to the child (ex. 'tell me more' or 'keep going'), but not to provide the student with cues or vocabulary.

Coding

Error	Count	% of utts.
[EO]	495	3.3%
[EU]	568	3.8%
[EW]	1,526	10.2%
[OM]	297	2.0%
[OW]	569	3.8%

Table 3.8: Counts of error codes in NarSR corpus

The NarSR corpus contains the five error codes shown in Table 3.8 along with the count of each code in the corpus. As with all other SALT corpora, we convert omission errors, indicated with '*' to either [OM] or [OW] for omitted morphemes or words, respectively. The set of error codes used in the NarSR corpus is identical to the one used in the other three corpora compiled by the group at the University of Wisconsin (Conv, Expository, and NarSSS).

3.1.7 NZConv Corpus

The New Zealand Conversation (NZConv) corpus was produced by a research group at the University of Canterbury in New Zealand. It contains 248 transcripts with a total of 25,503 utterances.

Participants

The NZConv corpus contains transcripts of spoken language collected from children ages 4;5-7;7 in three major urban areas in New Zealand: Hamilton, Christchurch, and Auckland. Participants were screened with the New Zealand speech and language screening test to asses language development. Transcripts from participants who performed very poorly on the receptive language portion of the screening test were excluded from the corpus, as were transcripts containing fewer than 45 utterances. The participants in the NZConv corpus come from a

wide variety of ethnic backgrounds (62% New Zealand European, 22% Maori, 5% Pasifika, 3% Asian, and 8% other). Finally, there was an even gender distribution among the participants.

Elicitation

Speech-language therapists elicited the language samples in the NZConv corpus following interview procedures described by Evans and Craig (1992). The examiners discussed three subjects with each child: first, an object of the child's choice that she was asked to bring from her classroom at school; the child's family; and after-school activities. Examiners attempted to elicit at least 50 complete, intelligible utterances from each child in 10 minutes of conversation.

Coding

Error	Count	% of utts.
[EO]	174	0.1%
[EU]	376	1.5%
[EW]	479	1.9%
[OM]	172	0.1%
[OW]	242	0.1%

Table 3.9: Counts of error codes in NZConv corpus

The NZConv corpus contains the five error codes shown in Table 3.9 along with the count of each code in the corpus. As with all other SALT corpora, we convert omission errors, indicated with '*' to either [OM] or [OW] for omitted morphemes or words, respectively. The set of error codes used in the NZConv corpus is identical to the one used in the other two corpora compiled by the group at the University of Canterbury (NZPerNar in Section 3.1.8, and NZSR in Section 3.1.9).

3.1.8 NZPerNar Corpus

The New Zealand Personal Narrative (NZPerNar) corpus was produced by a research group at the University of Canterbury in New Zealand. It contains 248 transcripts with a total of 20,253 utterances.

Participants

The NZPerNar corpus contains transcripts of spoken language collected from children ages 4;5-7;7 in three major urban areas in New Zealand: Hamilton, Christchurch, and Auckland. Participants were screened with the New Zealand speech and screening test to asses language development. Transcripts from participants who performed very poorly on the receptive language portion of the screening test were excluded from the corpus, as were transcripts containing fewer than 45 utterances. The participants in the NZPerNar corpus come from a wide variety of ethnic backgrounds (62% New Zealand European, 22% Maori, 5% Pasifika, 3% Asian, and 8% other). Finally, there was an even gender distribution among the participants.

Elicitation

Speech-language therapists elicited the language samples in the NZPerNar corpus following the Conversational Map technique (Peterson and McCabe, 1983). Each session began with an examiner telling the child a story based on a photo that they showed the child. After telling the story, the examiner asked the child if anything similar had happened to her. If the child said yes, the examiner asked her to tell a story about it. If she said no, the examiner repeated the story-telling procedure with a different photo. This process was repeated until the child produced at least three narratives and 50 complete and intelligible utterances. Examiners encouraged the children to speak more, for example by

asking "and then what happened?", but they did not assist or evaluate the child while she was telling a story.

Coding

Error	Count	% of utts.
[EO]	314	1.6%
[EU]	305	1.5%
[EW]	700	3.5%
[OM]	90	0.4%
[OW]	224	1.1%

Table 3.10: Counts of error codes in NZPERNAR corpus

The NZPERNAR corpus contains the five error codes shown in Table 3.10 along with the count of each code in the corpus. As with all other SALT corpora, we convert omission errors, indicated with '*' to either [OM] or [OW] for omitted morphemes or words, respectively. The set of error codes used in the NZPERNAR corpus is identical to the one used in the other two corpora compiled by the group at the University of Canterbury (NZCONV in Section 3.1.7, and NZSR in Section 3.1.9).

3.1.9 NZSR Corpus

The New Zealand Strory Retell (NZSR) corpus (also referred to as the New Zealand Expository Corpus in the SALT manual (Miller et al., 2011)) was produced by a research group at the University of Canterbury in New Zealand. It contains 264 transcripts with a total of 2,574 utterances.

Participants

The NZPERNAR corpus contains transcripts of spoken language collected from children ages 4;6-7;7 in three major urban areas in New Zealand: Hamilton, Christchurch, and Auckland. Participants were screened with the New Zealand

speech and screening test to asses language development. Transcripts from participants who were unable or unwilling to perform the task were excluded from the corpus.

Elicitation

The elicitation procedure began with the child listening to a recording of a story (Ana gets lost; Swan, 1992) twice while following along with a picture book. The story has only been published in Tokelauan, thus minimizing the chances that any of the children were familiar with it, and rendering it impossible for them to read the story while listening to it (the recording was of an English translation). After listening to the recording the first time, the examiner asked the child eight comprehension questions. Examiners provided the correct answers to questions that the child answered incorrectly to reduce the influence of comprehension on the child's retelling ability. After listening to the story a second time, the child was asked to retell it without the aid of the book. The examiner would encourage the child, but would not assist her in telling the story.

Coding

Error	Count	% of utts.
[EO]	55	2.1%
[EU]	55	2.1%
[EW]	98	3.8%
[OM]	13	0.5%
[OW]	32	1.2%

Table 3.11: Counts of error codes in NZSR corpus

The NZSR corpus contains the five error codes shown in Table 3.11 along with the count of each code in the corpus. As with all other SALT corpora, we convert omission errors, indicated with '*' to either [OM] or [OW] for omitted morphemes or words, respectively. The set of error codes used in the NZSR

corpus is identical to the one used in the other two corpora compiled by the group at the University of Canterbury (NZConv in Section 3.1.7, and NZPerNar in Section 3.1.8).

3.2 CLSU ADOS Corpus

The CSLU ADOS corpus was collected during a large scale study of autism and language impairments. In addition to the ADOS (discussed below), participants in the study (and their parents) also completed other tests, for example the WISC-4 test of intelligence (Wechsler, 2003), and the CCC-2 parent questionnaire to assess communication skills (Bishop and Volkmar, 2003). Most of these instruments are not relevant to this thesis, so we do not discuss them here. Two of these instruments are only relevant to Chapter 7, and are discussed there. Here we present the ADOS corpus, which is relevant to much of this thesis.

We process the CSLU ADOS corpus differently, depending upon the task we wish to perform (see Section 3.3), and in some cases this results in there being a different number of non-empty utterances. For example, if we excise mazes, then lines that only consist of a maze must be removed. Nevertheless, at most we make use of 145 transcripts containing a total of 61,949 utterances.

3.2.1 ADOS

The Autism Diagnostic Observation Schedule (ADOS) is a semistructured standardized assessment that is used as a part of the process to diagnose ASD (Lord et al., 2000). *Semistructured* instruments provide clinicians with standardized situations and activities to perform, but unlike structured instruments (see Sections 7.1.4 or 7.1.5 for detailed examples), semistructured instruments are designed to elicit spontaneous behavior, including language.

The ADOS itself consists of four modules, each of which takes 30 minutes to

Module Language level	1 Preverbal/single words/phrases	2 Flexible phrase speech	3 Fluent speech child / adolescent	4 Fluent speech adolescent / adult
Activities	Anticipation of a social routine	Construction task	Construction task	Construction task
		Make-believe play	Make-believe play	Current work, school, daily living
	Functional and symbolic imitation	Joint interactive play	Joint interactive play	SEQ: Plans and dreams
	Free play	Free play	Break	Break
	Snack	Snack		
	Response to name	Response to name	Cartoons	Cartoons
	Response to joint attention	Response to joint attention	SEQ: Emotions	SEQ: Emotions
	Birthday party	Birthday party	SEQ: Friends, loneliness, marriage	SEQ: Friends, loneliness, marriage
	Bubble play	Bubble play	SEQ: Social difficulties, annoyance	SEQ: Social difficulties, annoyance
	Anticipation of a routine with objects	Anticipation of a routine with objects	Creating a story	Creating a story
		Demonstration task	Demonstration task	Demonstration task
		Conversation	Conversation / reporting a nonroutine event	Conversation / reporting a nonroutine event
		Description of picture	Description of picture	Description of picture
		Looking at a book	Telling a story from a book	Telling a story from a book

Table 3.12: Overview of the four ADOS modules taken from (Lord et al., 2000, 3). Activities on the same line are similar in intent where possible.
SEQ=Socioemotional questions

administer. The choice of module is determined by the expressive language level of the individual being tested, and these modules are outlined in Table 3.12.

Coding

Error	Count	% of utts.
[EC]	43	<0.1%
[EO]	235	<0.1%
[EU]	477	0.1
[EW]	1,542	2.5%
[EX]	342	0.1%
[OM]	923	1.5%
[OW]	847	1.4%
[WO]	83	<0.1%

Table 3.13: Counts of error codes in CSLU ADOS corpus. [EC] = inappropriate response, [EX] = extraneous word, [WO] = word order error

ADOS sessions were recorded and child and examiner speech were later transcribed. Transcribers were unaware of the child's diagnostic status, cognitive level, and language ability.

Transcription followed SALT guidelines: utterances were separated into c-units, and the examiner and child's speech were separated; mazes were annotated; and grammatical errors were indicated as well. The set of error codes used in the CSLU ADOS corpus, along with their counts, is shown in Table 3.13.

3.2.2 Participants

The 111 participants in this study ranged from ages 4 through 8. All of the participants were native speakers of English who lived in the Portland, OR metro area. Participants were recruited by a variety of means: children with ASD were found through local healthcare specialists, education service districts, autism clinics, parent groups, and non-profit autism organizations, and those with language impairments were identified through local speech clinics, speech

language pathologists, and the Oregon Speech and Hearing Association.

There were five exclusionary criteria for this study: 1) identified metabolic, neurological, or genetic disorder; 2) gross sensory or motor impairment; 3) brain lesion; 4) orofacial abnormality, for example a cleft palate; and 5) intellectual disability. Furthermore, all participants included in the study produced utterances with a minimum mean length of 3.0 morphemes per utterance. All participants had a full-scale IQ of at least 70. IQ was determined using the Wechsler Preschool and Primary Scale of Intelligence (WPPSI-III; Wechsler, 2002) for children younger than 7, and with the Wechsler Intelligence Scale for Children (WISC-IV; Wechsler, 2003) for children ages 7 and older. Finally, during initial screening a speech language pathologist confirmed that the child did not have any impairments affecting speech intelligibility.

Diagnosis

Language	Autism		No autism	
Impaired	**ALI**	25	**SLI**	19
Not impaired	**ALN**	25	**TD**	42

Table 3.14: Diagnostic status of participants in CSLU ADOS corpus including group labels. For example, ALI = autism and language impairment.

Most of the participants in the CSLU ADOS corpus fall into one of four diagnostic categories, as shown in Table 3.14. We also note that the majority of the participants in the study were male. In each group, there were the following number of males: 23 in ALI (ASD + language impairment), 22 in ALN (ASD + no language impairment), 12 in SLI (language impairment, no ASD), and 30 in TD (neither ASD nor language impairment). Each child's diagnosis was confirmed in house following a standard set of procedures for identifying ASD and language impairments (Prud'hommeaux et al., 2011).

ASD was diagnosed by a team of two clinical psychologists, a speech language

pathologist, and an occupational therapist with experience working with autism spectrum disorders following the DSM-IV-TR criteria (American Psychiatric Association, 2000). A child was assigned a diagnosis of ASD only when this group reached a consensus. At present, best estimate clinical (BEC) judgement by experienced clinicians is considered to be the most effective way to diagnose ASD (Klin et al., 2000; Spitzer and Siegel, 1990). In addition to the group's BEC consensus, all children diagnosed with ASD had scores above the threshold indicative of ASD on both the ADOS-G (Gotham et al., 2007) and Social Communication Questionnaire (commonly referred to as the SCQ; Rutter et al., 2003). Specifically, all children diagnosed with ASD scored at least 12 on the SCQ, which is the recommended cutoff for research purposes (Lee et al., 2007).

Language impairments were identified using the Core Language Score on the Clinical Evaluation of Language Fundamentals (CELF) (Semel et al., 2003, 2004), which captures both receptive and expressive language capabilities. In particular, any child scoring at least 1 standard deviation below the mean on this test (Core Language Score < 85) was identified as having a language impairment. Two different variants of the CELF were used, dependent upon the age of the child: the CELF Preschool-2 (Semel et al., 2004) was administered to children younger than six years old, and the CELF-4 (Semel et al., 2003) to all children six years or older.

There are two groups of participants that meet the criterion for a language impairment, one with ASD (ALI), and one without (SLI). Inclusion in the SLI group had two additional criteria beyond the score on the CELF: 1) a documented history of language impairment or delay, and 2) BEC consensus that the child has a language impairment and does not have ASD taking into account a wide variety of evidence (including medical and family history, in-house and locally based assessments, and information from school). We note that several children

included in the SLI group scored above the threshold for ASD on either the SCQ (n=8) or the ADOS-G (n=4), but not both. Previous research suggests that it is common for children with a BEC diagnosis of SLI to score above thresholds indicative of ASD on diagnostic assessments (Bishop and Norbury, 2002; Leyfer et al., 2008). Children diagnosed with ASD following the procedures above were included in the ALI group if they also met the criterion for impaired language, and otherwise they were put into the ALN group.

Children in the TD group did not meet the criteria for either ASD or a language impairment. There were several additional criteria for exclusion from the TD group: a family history of autism or language impairments; a diagnosed psychiatric disorder (for example ADHD); and scoring above the threshold for ASD on either the SCQ or ADOS-G.

3.3 Preprocessing of SALT Annotated Corpora

We do not perform our experiments on unmodified SALT transcripts. Rather, we preprocess the transcripts by *desaltifying* and *normalizing* the transcripts, which we discuss in turn below. This preprocessing makes the experiments more straightforward to conduct, and renders the results more meaningful.

3.3.1 Desaltification

Before performing most of our experiments we remove some of the SALT annotations. We refer to this process as *desaltification*[1], and we perform desaltification using a script written at CSLU by Emily Tucker-Prud'hommeaux and Géza Kiss[2]. Desaltification can remove various items, for example sound effects (words preceded by '%'), incomplete words, error codes, or maze annotations. Desaltifi-

[1] Some prefer the term *desalination*.

[2] Specifically, we use the most recent version of `desaltify2.pl` as of July 2, 2014. This script is available upon request from the author.

Option	Example	Flag(s)	Maze	Error	Feature
Keep lines not associated with a valid activity label		`b`	✓	✓	✓
Preserve idiosyncratic words without corrections	hana_rra[=good]→ hanna_rra	`Dd`	✓	✓	✓
Include examiner speech		`e`			✓
Remove incomplete words	we*→ ∅	`i`	✓	✓	
Remove sound effects	%pfft→ ∅	`k`	✓	✓	✓
Preserve morphology segmentation	boy/s/z	`l`			✓
Preserve marking of phrases	ice_cream	`P`	✓	✓	✓
Preserve marking of mazes	(um I)	`n`	✓	✓	✓
Preserve marking of overlapping speech	<you are>	`o`			✓
Include omitted morphemes/words with annotation	*she go/*3s	`q`		✓	✓
Preserve error codes	goed [EW]	`r`		✓	✓
Preserve SALT contractions	gonna	`s`	✓	✓	✓
Segment at sentence boundaries even when time stamps are included		`T`	✓	✓	✓
Tokenize punctuation even if not tokenizing words	go. → go .	`Z`	✓	✓	✓
Do not tokenize	can/'t → can't	`z`	✓	✓	✓

Table 3.15: Desaltify script options used with different experiments: *maze* detection, *error* code detection, and *feature* extraction for predicting scores on structured instruments and diagnostic discrimination tasks. *Flag* column indicates flags used at the command line with `desaltify2.pl`.

SALT:	They (um) %pfft goed[EO] *to buy cookie/S.
MAZE:	they um goed buy cookies .
ERROR:	they um goed [EO] [OM] buy cookies .
FEATURE:	they (um) goed [EO] [OM] buy cookie/S .
SALT:	(Bu* bu*) but (I thought) < we ca/N'T> see it, can we? that.
MAZE:	but (I thought) we can't see it can we ?
ERROR:	but I thought we can't see it can we ?
FEATURE:	(Bu* bu*) but (I thought) <we ca/N'T> see it can we ?

Figure 3.1: Examples of utterances processed with different desaltification settings after normalization

cation can also involve removing the bound morpheme annotations (a form of normalization, discussed below in Section 3.3.2), for example converting 'go/3s' to 'goes' or 'run/ing' to 'running'. The desaltify script does so using a dictionary, although more recently Gorman et al. (2015) have found that a model based on linear classifiers and finite state transducers is highly effective for this task. We refer the interested reader to the desaltify script for implementation details. Here we only note the options used, not their implementations.

We produce three versions of each transcript using the desaltify script: one for maze detection (Chapter 5), one for error code detection (Chapter 6), and one from which we extract features when predicting scores on structured instruments and performing a diagnostic discrimination task (Chapter 7). The options used for each of these are shown and described in Table 3.15. For the maze and error code detection tasks we use desaltification (and normalization, discussed below) to create a very simple transcript that has only words and the annotations of interest, whether they be mazes or error codes. In the case of feature extraction, we use desaltification to simplify feature extraction. Figure 3.1 contains example utterances in their original SALT-annotated form (labeled 'SALT') along with their desaltified and normalized (see below) versions used

in our different investigations: MAZE for experiments around maze detection (Chaper 5), ERROR for those around error code detection (Chapter 6), and FEATURE for those around the clinical utility of SALT annotations (Chapter 7).

For clarity, Table 3.15 does not contain options that leave the text unmodified; in all experiments, we used two such options: `-F` 'keep file names unchanged', and `-g` 'keep empty lines'. These options assist with matching transcripts that have been desaltified differently, and the lines within them, respectively.

3.3.2 Normalization

In addition to desaltification, we also normalize the text before running experiments. *Text normalization* is the process of converting text to a standard form (Sproat et al., 2001). This may be as simple as lowercasing words for finding a case-insensitive exact match, or could involve spelling out numbers for pronunciation in a text to speech system ('41' → 'forty one'), or stemming for looking up words in a dictionary (ex. 'stemming' and 'stemmed' become 'stem'). More sophisticated forms of text normalization can involve abbreviation expansion, for example expanding '5th Ave' into 'fifth avenue'. Such techniques have a wide variety of applications, including in text-to-speech systems (Roark and Sproat, 2014) or expanding text messages (Beaufort et al., 2010; Liu et al., 2011) or social media data (Liu et al., 2012a,b) for further analysis.

First, we exclude any lines with unintelligible words from almost all of our experiments. We retain these lines in transcripts desaltified with the *feature* setting in Table 3.15 since we use the count of unintelligible words as a feature in these experiments; unintelligible words are thrown out in all other cases (for example before identifying mazes or error codes, and before computing the length of an utterance). Second, we remove empty mazes, which can arise when a maze

contains only sound effects and incomplete words, both of which we remove. We convert the original annotations for 'omitted word' and 'morpheme' to error codes:

(3.1) She *is go/*ing home. →
She *is go*ing home . →
She [OW] go [OM] home .

Third, we remove all non-final punctuation from every utterance because punctuation use appears to be inconsistent across corpora. Fourth, for our experiments in maze detection, we remove annotations for asides, which are delimited with double parenthesis. Not all corpora annotate asides, and automatically detecting asides is beyond the scope of this thesis. Finally, we perform sets of experiments in error code detection in which manually annotated mazes are excised, and others in which they are not, but in which the maze annotations are deleted. We do not use features related to maze annotations in any of the error code detector experiments. The examples in Figure 3.1 have been normalized following these standards, in addition to having been desaltified following the settings described above.

3.3.3 Partitioning into sets

We partition all of the SALT corpora, but not the CSLU ADOS corpus, into training, development, and test sets. We use these sets for our experiments in automatically detecting mazes (Chapter 5) and SALT error codes (Chapter 6). To do this, we: 1) desaltify and normalize all of the transcripts, using the appropriate settings in Table 3.15 for the relevant experiment; 2) merge all of the transcripts into a single file; 3) shuffle the lines randomly; and 4) partition the lines such that 80% are in the training set, and 10% in both the development and test sets.

3.4 Conclusions

We have presented an overview of the SALT-annotated corpora that sit at the heart of this dissertation. We have seen that they contain transcripts of a variety of tasks, and were collected from children with different ages and linguistic abilities. The variety of linguistic ability is further compounded by the presence of transcripts collected from children with autism, language impairments, or both, in addition to neurotypical children. Looking at basic summary statistics of these corpora, we also see hints of the diversity, or alternatively inconsistency, of annotation styles. For example, the GillamNT corpus contains only four error codes, and the ENNI corpus does not appear to use the overgeneralization code [EO] which appears in most of the other corpora. We consider the inconsistency of grammatical error code annotations in detail in Chapter 6, and similarly, we compare maze annotations across the SALT-annotated corpora in Chapter 5. Finally, in Chapter 7 we use the CSLU ADOS corpus to explore how useful SALT-annotated transcripts are for identifying and characterizing developmental disorders.

Chapter 4

Technical Background

This chapter contains an overviews of techniques that are used in this thesis, particularly in Chapters 5 and 6. We begin with an overview of the perceptron algorithm in Section 4.1. As we discuss, the perceptron algorithm is a simple but powerful algorithm for labeling a single item at a time. We then provide a background on graph algorithms in Section 4.2, which underpin the graph-based algorithms for structured prediction discussed in Section 4.3. Structured prediction is a particularly relevant to NLP because language contains so much structure, and we can begin to leverage some of it by using techniques for structured prediction.

Next we discuss a variety of tasks in NLP that are particularly relevant to this thesis. First, we present two approaches to inferring sentence structure, namely constituency parsing (Section 4.4.1) and dependency parsing (Section 4.4.2). These basic techniques are applied in both disfluency detection (Section 4.5), which is closely related to maze detection (Chapter 5), and our experiments in grammatical error detection in Chapter 6. We describe previous work in grammatical error detection below in Section 4.6, and finally conclude with

descriptions of the evaluation procedures used throughout this thesis.

4.1 Perceptron algorithm

for all labels y **do**
  $\boldsymbol{w}_y^{(0)} = \mathbf{0}$
end for
$i \leftarrow 0$
for $t : 1..T$ **do**
  for $n : 1...N$ **do**
    Let $y' = \text{argmax}_y \, \boldsymbol{w}_y^{(i)} \cdot \boldsymbol{x}_n$
    if $y' \neq y_n$ **then**
      $\boldsymbol{w}_{y'}^{(i+1)} = \boldsymbol{w}_{y'}^{(i)} - \boldsymbol{x}_n$
      $\boldsymbol{w}_y^{(i+1)} = \boldsymbol{w}_y^{(i)} + \boldsymbol{x}_n$
      $i \leftarrow i + 1$
    end if
  end for
end for
return $\boldsymbol{w}^i$

Figure 4.1: Pseudocode for the multilabel perceptron algorithm. T is the number of training iterations. $\boldsymbol{w}_i$ is the weight vector at the i^{th} iteration, while $\boldsymbol{x}_n$ and y_n are the n^{th} feature vector and labels, respectively. $\boldsymbol{w}_y$ is the weight vector for the label y.

The multilabel perceptron algorithm (Rosenblatt, 1958) is an *online*, *error-driven* algorithm for *supervised classification*, and the pseudocode for this algorithm is shown in Figure 4.1. Put formally, the perceptron's task is to learn a function $h : \mathcal{X} \rightarrow \mathcal{Y}$ that predicts a single label $y \in \mathcal{Y}$ from an input feature vector $\boldsymbol{x} \in \mathcal{X}$. In this thesis, we use the *averaged multiclass* perceptron, which means that there can be an arbitrary but finite number N of different labels. Since the perceptron algorithm is supervised, we need a training set of labeled examples $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$. During learning, the perceptron learns a set of N weight vectors $\mathcal{W}$, where $\boldsymbol{w}_y$ is the weight vector for the label y. We now describe prediction, and then we will return to learning with the perceptron algorithm.

Predicting a label y' for a feature vector $\boldsymbol{x}$ is quite simple once we have

learned the set of input weights $\mathcal{W}$:

$$y' = \operatorname*{argmax}_y \boldsymbol{w}_y \cdot \boldsymbol{x} \tag{4.1}$$

Prediction is also done during training (line 7 in Figure 4.1): every time there is an erroneous prediction, the weight vectors $\boldsymbol{w} \in \mathcal{W}$ get updated (lines 8-11 of the pseudocode). For a given training example $(\boldsymbol{x}, y)$, we predict its label y'. If the prediction is correct, we do nothing. If, however, we do not predict the true label y, then we update both $\boldsymbol{w}_y$ and $\boldsymbol{w}_{y'}$ (lines 9-10). In other words, we simply add $\boldsymbol{x}$ to the weight vector for the true label $\boldsymbol{w}_y$, and subtract $\boldsymbol{x}$ from the weight vector for the predicted label $\boldsymbol{w}_{y'}$ when we predict the wrong label. Because the weights are only updated when there is an erroneous prediction, we say that this classifier is *error driven*. Furthermore, the perceptron is an *online* training algorithm because the weights are updated after every training example. In other words, we do not need to retrain the entire model if we find some additional training data; we can simply update it.

One issue with the perceptron as we have described it thus far is that weight vectors are modified whenever they result in an incorrect prediction, even if they have been used in many correct predictions. The *averaged* perceptron prevents a single erroneous prediction from radically changing such a weight vector: instead of using the raw weights for prediction at test time, it uses the average weight vector, taken across all updates (Freund and Schapire, 1999). Therefore, if a particular weight vector stays unchanged across most training examples (because it does not result in any mistakes with them), the average weight vector will be quite close to it even if it yields a few mistakes later.

Current word	Score	Prediction	Truth	Update
you	0	-	-	None
go	0	-	-	None
home	0	-	-	None
she	0	-	-	None
go	0	-	+	C[go]+1 $\rightarrow$ 1
home	0	-	-	None
Iteration 2				
go	1	+	-	C[go]−1 $\rightarrow$ 0
go	0	-	+	C[go]+1 $\rightarrow$ 1

Table 4.1: Two iterations of perceptron training a to identify grammatical errors. '+': error, '-': no error.
The score is simply the weight of the entry $C[w]$ where C is the feature weight vector, and w is the current word. We only show steps with updates in Iteration 2.

4.1.1 Example

To make this description more concrete, let us look at a simple example of grammatical error detection, a task we approach throughout this thesis. We have two utterances: 'you go home', and 'she go home', the latter of which contains a single error associated with the word 'go' (assuming we are not discussing a dialect in which 'she go' is grammatical). Table 4.1 shows two rounds of training a simple perceptron to predict whether a word is associated with a grammatical error. We use a single feature for prediction: the word itself. As can be seen in 4.1, the feature weight for C[go] simply oscillates between 0 and 1 because the word 'go' is grammatical in one utterance, but ungrammatical in the other. Nevertheless, the perceptron algorithm correctly predicts that there is no grammatical error associated with any of the other words.

4.2 Graphs

Although the perceptron algorithm described above is suitable for a wide range of problems, it is not so effective for addressing those where *structure* is critical, as is the case for many problems in NLP, including our example problem of grammatical error detection. This is because the perceptron algorithm considers items in isolation, but language has structure. *Graphs* are a particularly powerful tool for capturing structure in language, among their many other applications in a diverse array of fields including sociology (ex. Breiger et al., 1975; Brandes, 2001), biology (ex. Enright et al., 2002), and logistics (ex. Zhan and Noon, 1998). Here we begin with a brief overview of graphs, and later in Sections 4.4.1 and 4.4.2 we provide more background on two types of graphs that are particularly relevant to language and NLP, namely *constituency* and *dependency parse* trees. Graphs are also relevant to methods for structured prediction, which we discuss in Section 4.3. For an excellent background on graphs and graph algorithms, we direct the interested reader to Even (2011).

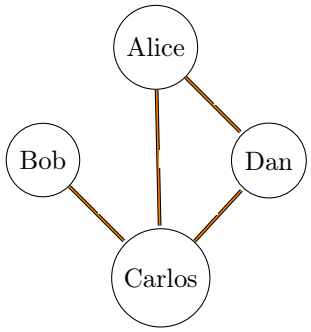
Formally, graphs consist of a set of vertices V, and a set of edges E, where each edge e connects two vertices (v_i, v_j). Two vertices v_i and v_j are *neighbors* if $(v_i, v_j) \in E$. Graphs can be *directed* in which case the edges (v_i, v_j) and (v_j, v_i) are distinct, or undirected, in which case they are not. Although this description may seem abstract, graphs are a natural way to encode a wide variety of information, including transit links between cities, social networks, and syntactic structure. Social networks, for example, can be represented with an undirected graph: each vertex represents a person, and each edge represents some sort of connection between two individuals, say friendship. Because friendship is mutual (at least in our graph), we don't need to distinguish between Alice being friends with Bob as opposed to Bob being friends with Alice. In some cases, however, we may want to model asymmetrical relationships, which we can

do with a *directed* graph.

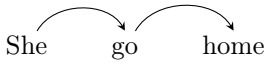
Graphs, whether directed or undirected, can be either *unweighted* or *weighted*. In an unweighted graph, edges either exist or they do not; to continue with the website example, the New York Times either links to a particular webpage (in which case there would be an edge), or it does not (no edge). Although this is a simple idea, it is quite powerful: it forms the core of the PageRank algorithm (Brin and Page, 1998; Page et al., 1999), which was Google's original algorithm for ranking webpages in its search.

For binary distinctions, such as when webpages are either linked or not, unweighted graphs are adequate. In many cases, however, some connections may be stronger than others. For example, we could represent the road system of, say, the United States with a graph: cities and towns are represented by vertices, and the roads by edges. Clearly highways allow more traffic between two cities than does a dirt road. To reflect this in our graph we assign weights to different edges, for example a weight of 100 to an edge connecting two cities connected by an interstate, and a weight of 1 to an edge connecting two cities (or more likely towns) connected by a dirt road. Figure 4.2c shows a weighted graph.

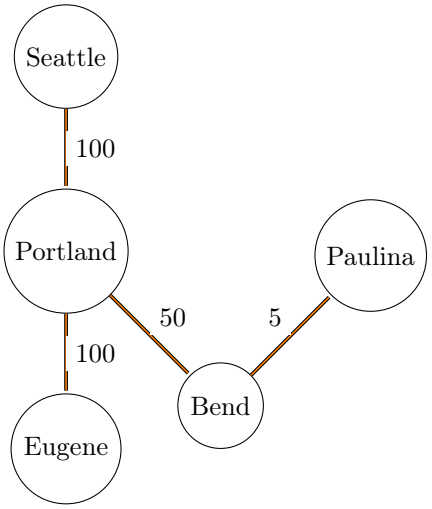
Returning to the two example utterances introduced above ('you go home' and 'she go home'), it is clear that we need context to know whether the word 'go' is grammatical or not. Based on these examples alone, we could conclude that 'go' is grammatical if it follows the word 'you', but not following 'she'. A more sophisticated analysis would posit that 'go' is ungrammatical if 'she' is the subject, thus enabling us to identify 'she probably go there often' as ungrammatical. Going further, we could add the exception that 'go' is grammatical with 'she' as the subject when it is in a subordinate clause, as in 'it is necessary that she go'. Each of these analyses suggests a different graphical structure: a sequence (Figure 4.2b), a dependency parse (Section 4.4.2) and a constituency parse tree



(a) Undirected, unweighted graph: social network



(b) Directed, unweighted graph: utterance as a sequence of words



(c) Undirected, weighted graph: roads

Figure 4.2: Examples of different types of graphs

(Section 4.4.1), respectively.

Random walks

Various properties of graphs can be explored using a *random walk*, which involves starting at a vertex v_i, and then taking some number of steps, each time going to a randomly selected neighbor of the current node. For example, if we keep traveling from port to port on a random container ship—ports are nodes, and trade links are edges—we would likely be in Shanghai far more often than Miami because more boats go to Shanghai. More generally, if we take enough boats, we can learn various properties of the world's trade networks: which ports are busiest, which are the most integrated, and so on.

In a random walk, neighbors can each be selected with uniform probability, as would typically be the case in an unweighted graph, or with different probabilities, as would be the case in a weighted graph (and the container ship example). Finally, in the case of directed graphs, a random walk can only go from vertex v_i to v_j if $(v_i, v_j) \in E$; having $(v_j, v_i) \in E$ is not sufficient to permit the random walk to go from v_i to v_j. Continuing with the container ship example, there may be a boat that goes from Portland to Seattle, but none from Seattle to Portland.

Although the random walk technique is quite simple, it has many applications in NLP and related fields, including the previously mentioned PageRank algorithm for information retrieval (Page et al., 1999), which identifies documents of interest based on how many times they are reached during a random walk on a graph in which documents are represented as nodes, and hyperlinks are directed edges. Random walks are also used in the closely related LexRank algorithm for text summarization (Erkan and Radev, 2004), as well as ones for word-sense disambiguation (Mihalcea, 2005) and providing recommendations for movies (Fouss et al., 2007).

4.3 Supervised structured prediction: tagging

Supervised structured prediction is the task of learning to predict *structured output* from labeled *structured input*. By structured, we mean that the input and output have a graph structure, as opposed to being items considered in isolation, as is the case with the basic perceptron (Section 4.1). Continuing with our error detection example, if the input is a sequence of words, then structured output could be a sequence of binary tags representing whether each word is associated with an error or not, for example:

(4.2) you go home → - - -

(4.3) she go home → - + -

Approaching grammatical error detection at the word level as a as a sequence *tagging* task should be far more fruitful than classifying a single word at a time because we are able to leverage each word's context to identify errors. For example, 'go' is grammatical in the first utterance where it follows 'you', but not in the second where it follows 'she'. In this thesis, we treat maze detection (Chapter 5) and error code detection (Section 6.5) as sequence prediction tasks, which have the graphical structure illustrated in Figure 4.2b. Our experiments treating SALT error code detection as a sequence prediction task is particularly closely related to our running example of identifying grammatical errors at the word level. We now introduce three techniques for structured prediction, which we use in this thesis to predict sequences: the structured perceptron (Collins, 2002), linear chain conditional random fields (CRF) (McCallum and Li, 2003), and max-margin Markov networks (M^3N) (Taskar et al., 2003).

More formally, in supervised structured prediction the task is to learn a function $h : \mathcal{X} \rightarrow \mathcal{Y}$ that predicts a structured output $\boldsymbol{y} \in \mathcal{Y}$ from structured input features $\boldsymbol{x} \in \mathcal{X}$. To be clear then, $h(\boldsymbol{x})$ is the set of labels predicted from

the input $\boldsymbol{x}$. In supervised structured prediction we assume a training set of N labeled data for training $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$. We also assume that there is a finite number of distinct labels in $\mathcal{Y}$.

Both CRFs and M^{3}N are types of Markov networks, which are undirected graph $\mathcal{G} = \{V, E\}$ in which each edge $(i, j) \in E$ is associated with a feature function $\boldsymbol{f}(\boldsymbol{y}_i, \boldsymbol{y}_j, \boldsymbol{x})$. We are particularly interested in predicting sequences, in which case the edges are all of the form $(t, t-1)$. Both CRFs and M^{3}Ns use a linear feature function of the input $\boldsymbol{x}$, output $\boldsymbol{y}$ and feature weights $\boldsymbol{w}$: $\boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{y}_t, \boldsymbol{y}_{t-1}; \boldsymbol{w}) = \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{y}_t, \boldsymbol{y}_{t-1})$ to assign a score to the output label sequence $\boldsymbol{y}$ given the input $\boldsymbol{x}$. CRFs and M^{3}Ns, however, compute weight vector $\boldsymbol{w}$ quite differently.

4.3.1 Structured perceptron

for all labels y **do**
  $\boldsymbol{w}_y^{(0)} = \mathbf{0}$
end for
$i \leftarrow 0$
for $n : 1...N$ **do**
  for $t : 1..T$ **do**
    $\boldsymbol{y'} \leftarrow \text{argmax}_{\boldsymbol{y'}}\, \boldsymbol{w}_{\boldsymbol{y}}^{(i)} \cdot f(\boldsymbol{x}, \boldsymbol{y'})$
    if $y' \neq y_t$ **then**
      $\boldsymbol{w}_{\boldsymbol{y'}}^{(i)} = \boldsymbol{w}_{\boldsymbol{y'}}^{(i)} - f(\boldsymbol{x}, \boldsymbol{y'})$
      $\boldsymbol{w}_{y}^{(i+1)} = \boldsymbol{w}_{\boldsymbol{y}}^{(i)} + f(\boldsymbol{x}, \boldsymbol{y})$
      $i \leftarrow i + 1$
    end if
  end for
end for
return $\boldsymbol{w}^i$

Figure 4.3: Pseudocode for the structured perceptron algorithm. T is the number of training iterations. $\boldsymbol{x}_t$ is the feature vector at time t, y_t is the corresponding label. $\boldsymbol{w}_y$ is the weight vector for the label y. $f(\boldsymbol{x}, \boldsymbol{y}$ is the score of the sequence of tags $\boldsymbol{y}$ given the sequence of feature vectors $\boldsymbol{x}$. Notation based on McDonald et al. (2010)

The structured perceptron (Collins, 2002) is a version of the perceptron

algorithm discussed in Section 4.1 that has been modified to perform structured prediction instead of classification. Although it is not used in this thesis, it is perhaps the most straightforward tool for structured classification. We give a brief overview of it here as a stepping stone for the reader unfamiliar with structured prediction before describing more complicated techniques. The pseudocode of the structured perceptron algorithm is shown in Figure 4.3.

As with the basic perceptron, the structured perceptron is an online, error driven algorithm. During training, we make a prediction (line 7) , and we update the model only when the prediction is wrong (lines 8-11). Instead of predicting a single tag, however, we predict the label sequence with the highest score using the Viterbi algorithm. Additionally, structured models use features that are dependent upon labels predicted earlier in the sequence. Continuing with our error detection example, let us use two features: the current word, and the current word along with the previous word. In the first iteration, we would predict the tag sequence for the grammatical utterance correctly, assuming 'no error' is the default tag. At the second utterance, we would predict the following, with the features and weights in the format (current word [weight], previous word + current word word [weight])

(4.4) (she [0], go [0]) (go [0], she go [0]) (home [0], go home[0]) → - - -

The second prediction is incorrect, and therefore we would need to increment the weights of the the relevant features: (go, she go). Returning to the first utterance, we would predict the following sequence:

(4.5) (you [0], go [0]) (go [1], you go [0]) (home [0], go home[0]) → - + -

Since the second prediction in that sequence is incorrect, we would decrease the feature weights for (go, you go). The model predicts the sequence of tags for the second utterance correctly:

(4.6) (she [0], go [0]) (go [0], she go [1]) (home [0], go home[0]) → - + -

At this point the structured perceptron has converged as it will always predict the correct tag sequence for these two training utterances, and therefore it will never be updated again.

4.3.2 Linear chain conditional random fields

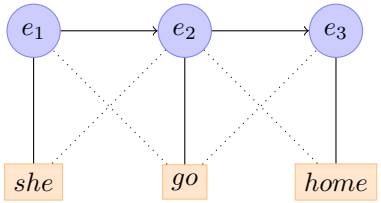


Figure 4.4: Illustration of the graphical structure of a conditional random field used for grammatical error detection at the word level: observations capture words (yellow squares), hidden states (blue circles) represent the predicted error tag at each step. While predicting the label for each state, we may leverage features from non-local observations (dashed edges).

The linear chain conditional random field (CRF) is a *discriminative graphical model* that has many applications in natural language processing, including named entity recognition (McCallum and Li, 2003), part-of-speech tagging (Toutanova et al., 2003), disfluency detection (Qian and Liu, 2013) and chunking (Sha and Pereira, 2003). We use the linear chain CRF to identify SALT error codes in Section 6.5. Although there are other varieties of CRF, we do not discuss them here because they are not used in this thesis, and we will only use the term 'CRF' to refer to linear chain CRFs throughout this thesis. We present a brief overview of how CRF training and testing work, and we refer the interested reader to Sutton and McCallum (2006) for a more thorough overview.

The CRF defines the conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$ as follows:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \prod_{t=1}^{T} \exp\left(\boldsymbol{w}_k \cdot \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{y}_t, \boldsymbol{y}_{t-1})\right) \tag{4.7}$$

where $\boldsymbol{x}_t$ is the t^{th} section of the structured input, and $\boldsymbol{y}_t$ is the t^{th} section of the structured output. For example, in our error tagging example, these would be the t^{th} word and the predicted tag, respectively, which we illustrate in Figure 4.4. The term $Z(\boldsymbol{x})$ in Equation 4.7 is a normalization function, typically referred to as the *partition function*. It is defined over a sequence of input feature vectors $\boldsymbol{x}$:

$$Z(\boldsymbol{x}) = \sum_{\boldsymbol{y}} \prod_{t=1}^{T} \exp\left(\boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{y}_t, \boldsymbol{y}_{t-1})\right) \tag{4.8}$$

Training

To train the CRF we simply need to estimate the weight vector $\boldsymbol{w}$. This is done with *maximum likelihood estimation*. Specifically, we maximize the log-likelihood of a labeled training data $\mathcal{D}$. Since the CRF models a conditional distribution, namely $p(\boldsymbol{y}|\boldsymbol{x})$, we use the *conditional log likelihood* function:

$$\ell(\boldsymbol{w}) = \sum_{\boldsymbol{x},\boldsymbol{y} \in \mathcal{D}} \log p(\boldsymbol{y}|\boldsymbol{x}) \tag{4.9}$$

We substitute the CRF model in (4.7) into the conditional log likelihood equation:

$$\ell(\boldsymbol{w}) = \sum_{\boldsymbol{x},\boldsymbol{y} \in \mathcal{D}} \log \left(\frac{1}{Z(\boldsymbol{x})} \prod_{t=1}^{T} \exp\left(\boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{y}_{t-1}, \boldsymbol{y}_t)\right) \right) \tag{4.10}$$

which can be simplified to:

$$\ell(\boldsymbol{w}) = \sum_{\boldsymbol{x},\boldsymbol{y} \in \mathcal{D}} \sum_{t=1}^{T} \boldsymbol{w} \cdot \boldsymbol{f}\left(\boldsymbol{x}_t, \boldsymbol{y}_t, \boldsymbol{y}_{t-1}\right) - \sum_{\boldsymbol{x} \in \mathcal{D}} \log Z(\boldsymbol{x}) \tag{4.11}$$

There is typically no way to maximize $\ell(\boldsymbol{w})$ analytically, and therefore numerical methods are used. Although both gradient ascent and Newton's method are applicable to this problem, they are typically too slow to be used in practice. The CRF implementation used in this thesis, CRF++ (Kudo, 2005), estimates $\boldsymbol{w}$ using the limited-memory Broyden-Fletcher-Goldfarb-Shanno

algorithm, which approximates Newton's method very efficiently (Nocedal, 1980), and is a common way of maximizing $\ell(\boldsymbol{w})$. We use L2 regularization in all of our experiments.

Prediction

Prediction with a CRF simply involves estimating the most probable sequence of labels:

$$h(\boldsymbol{x}) = \operatorname*{argmax}_{\boldsymbol{y}} p_{\boldsymbol{w}}(\boldsymbol{y}|\boldsymbol{x}) \tag{4.12}$$

, which can be done with the Viterbi algorithm.

4.3.3 Max margin Markov networks

M^{3}Ns, which we use for maze detection in Chapter 5, model the conditional probability of the label sequence $\boldsymbol{y}$ given the observed features $\boldsymbol{x}$ as follows:

$$p(\boldsymbol{y}|\boldsymbol{x}) \propto \prod_{t=1}^{T} \exp\left(\boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}_t, \boldsymbol{y}_{t-1})\right) \tag{4.13}$$

We note that Equation 4.13 is almost identical to the one defining the CRF (Equation 4.7), aside from the normalizing function $Z(\boldsymbol{x})$.

Training

	She	go	home	.
Truth	-	+	-	-
Pred.	+	-	-	-
Loss	1	0	1	0

Figure 4.5: Illustration of Hamming loss with identifying errors at the word level: + means error, - means no error. Total loss is 2.

As previously mentioned, the biggest difference between M^{3}Ns and CRFs

is in how they are trained: CRFs are trained by maximizing the conditional likelihood of the weight vector $\boldsymbol{w}$, while M^{3}Ns involve learning a weight vector $\boldsymbol{w}$ that maximizes the margin between the true label sequence $\boldsymbol{y}$ and the predicted label sequence $\boldsymbol{t}(\boldsymbol{x})$ by solving the following constrained optimization problem:

$$\min_{\boldsymbol{w},\boldsymbol{\xi}} \frac{1}{2} \|\boldsymbol{w}\|_2^2 + C \sum_{i=1}^{N} \xi_i \tag{4.14}$$

subject to the following two constraints:

$$\begin{aligned} \forall(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{D} : \boldsymbol{w} \cdot (\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}) - \boldsymbol{f}(\boldsymbol{x},h(\boldsymbol{x})) \geq l(\boldsymbol{y},h(\boldsymbol{x})) \\ \xi_i \geq 0 \end{aligned} \tag{4.15}$$

One approach to solving this constrained optimization problem is coordinate descent (Taskar et al., 2003).

In the above equations, ξ_i is a slack variable to absorb the errors in the i^{th} training example, and C is a tuning parameter. The difference in the scores of the predicted and true sequences $(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}) - \boldsymbol{f}(\boldsymbol{x},h(\boldsymbol{x}))$ is referred to as the *margin*. The M^{3}N is trying to maximize the margin between scores assigned to the correct label sequence and those assigned to incorrect sequences. Finally, $l(\boldsymbol{y},h(\boldsymbol{x}))$ is a loss function that compares the true label sequence $\boldsymbol{y}$ to the predicted one, $h(\boldsymbol{x})$. One of the most common loss functions is Hamming loss, which simply counts the number of divergent labels in $h(\boldsymbol{x})$, and which is illustrated in Figure 4.5.

Prediction

As with a CRF, prediction using a M^{3}N simply involves estimating the most probable sequence of labels:

$$h(\boldsymbol{x}) = \operatorname*{argmax}_{\boldsymbol{y}} p_{\boldsymbol{w}}(\boldsymbol{y}|\boldsymbol{x}) \tag{4.16}$$

, which can be done with the Viterbi algorithm.

4.4 Parsing

Sentences in human languages are not just bags of words with no internal structure: words can be grouped into phrases, and pairs of words can be related, for example a verb and its subject. This structure is referred to as *syntactic* structure, and it is typically represented with a graph in which words are nodes, and edges represent syntactic relationships. Although such representations are more complicated than a simple sequence of words, they are also far more powerful. Returning to our word-level grammatical error detection example, we saw in Section 4.1 that making predictions while looking at a single word is less effective than considering words as a sequence, because context is critical for grammatical error detection: the word 'go' is grammatical in 'you go home', but not in 'she go home'.

In this section we give an overview of two representations of syntactic structure, namely constituency parses and dependency parses. We also discuss approaches to predicting these structures, given an utterance (or sentence) as input. While we do not use constituency parse trees anywhere in this thesis, they are closely related to dependency parses, which are a key component of our investigation into automated SALT error code detection in Chapter 6. For more information on constituency parses, we refer the interested reader to Roark and Sproat (2007).

4.4.1 Constituency parsing

Constituency parse trees, also commonly referred to as syntax trees, are a graphical representation of sentence structure. Two example parse trees are shown in Figure 4.6. We see that they encodes many levels of structure. For

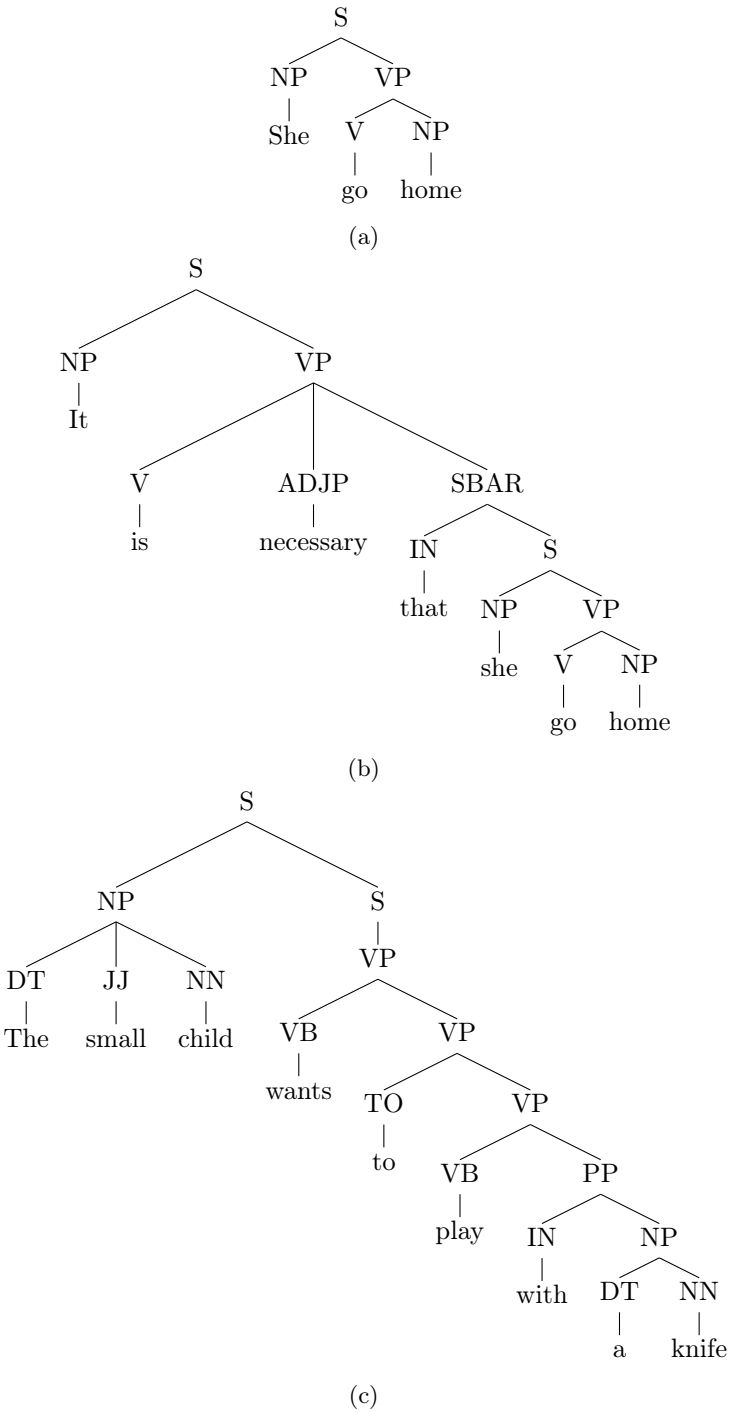


Figure 4.6: Constituency parse trees

example, both utterances are subsumed under an S node, and 'go home' is subsumed under a VP (verb phrase) node. These parse trees also encode grammatical relations: the left children of S nodes are the subjects of the right children, which are verb phrases, and 'she go home' in the second example is clearly a subordinate clause, as it is under an SBAR span.

As suggested by the name, constituency parses are particularly useful for identifying *constituents*, which are contiguous spans of words that form a single syntactic unit. Although there are a variety of constituency tests, one of the most straightforward is *substitution*, in which a constituent is replaced without really changing the meaning. For example, we can replace several constituents in Figure 4.6c: '*she* wants to play with *it*', '*she* wants to *do so*', or '*she does*' (ex. as an answer to the question 'Does she want to play with a knife?'). For more information on constituency tests in English, we refer the interested reader to Burton-Roberts (2013).

There are at least two major difficulties involved in using constituency parses for grammatical error detection in transcripts of spoken language. First, constituency parsing is computationally expensive: the worst-case complexity of constituency parsing is proportional to the cube of the number of words being parsed (ex. Earley, 1970; Younger, 1967). For comparison, dependency parsing, which we discuss below in Section 4.4.2, can be performed in linear time (Nivre, 2009). Similarly, taggers such as a CRF or M^3N can be far more efficient than methods based on constituency parsing.

The second issue in using constituency parses for error detection is that constituency parses encode more structure than we may need, while the structure we most obviously need is not necessarily encoded clearly. Many of the errors that we will be trying to detect involve either a single word (ex. overgeneralization errors such as 'goed') or pairs of words (ex. prepositional errors such as 'played

of', and subject/verb agreement errors such as 'she go'). They do not typically involve entire constituents. For example, identifying a prepositional error, which is indeed a common error in children's speech, does not necessarily require access to the verb phrase with a prepositional phrase as its left child (as in 'play with a knife' in Figure 4.6c). In general, the verb and its accompanying preposition will suffice, for example 'play with' vs 'play of'. Similarly, constituency parses do not encode some of the most basic grammatical relationships, for example direct objects, in an explicit way. Identifying these relationships can be very useful for word-level error detection: if we know that the word 'she' is a direct object of a verb, the utterance is likely ungrammatical (ex. 'I paid she'). As we shall see below, dependency parses do encode such relationships, and they do so explicitly.

4.4.2 Dependencies and dependency parsing

Dependency parsing is a widely-used formalism for representing the syntactic structure of a sentence as a graph, and it is the only such formalism used in this thesis. Dependency parsing has its roots in the work of Tesnière (1959), in which he claimed that "the structural relationships establish *dependency* relationships between the words. Each connection in principle unites a *superior* term to an *inferior* term." We refer to the superior term as the 'head', and the inferior term as the 'dependent' throughout this thesis. In this section, we give relevant background on dependency graphs and relations, and several techniques to learn a dependency grammar from manually labeled parses. There has been a great deal of research in NLP into dependency parsing, and here we give an overview of the main techniques used for this task. This thesis does not propose novel methods for dependency parsing, but dependency parsing is a critical component of our investigation into techniques for automatically detecting SALT error codes

(Chapter 6).

Dependency relations

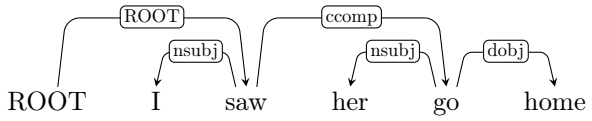


Figure 4.7: A dependency parse of 'I saw her go home'

The asymmetric relationship between the head and its dependent is referred to as a 'dependency'. We can represent dependency parses as graphs, as shown in Figure 4.7, and the dependency relationships themselves are encoded in such graphs as directed edges, or *arcs*. Finally, not all dependencies capture the same functional category: one may be from a verb and its subject, while another may be from a noun to its determiner. Different functional categories are encoded in the *arc labels*. In Figure 4.7, the *nsubj* relationship indicates that the nominal dependent is the subject of its verbal head, the *dobj* relationship indicates that the nominal dependent is the direct object of its verbal head, the *ccomp* relationship indicates that the clause headed by the dependent verb 'go' is the complement of the head verb 'saw', and finally the ROOT arc has an imagined head, and its dependent is typically the main verb of the sentence. For more detail on how dependency relationships and functional categories are determined, we refer the reader to Zwicky (1985), Chapter 6 of Hudson (1990), and De Marneffe and Manning (2008).

Unlike constituent parses (discussed above in Section 4.4.1), which identify phrases of words that are syntactically related, dependency parses identify pairs of related words, and these are likely to be highly informative for identifying grammatical errors. For example, by having access to subject/verb, noun/determiner, and verb/object relationships, we can easily identify several types errors that

occur throughout the transcripts that we investigate here: verb agreement errors ('he go'), determiner errors ('a dogs'), and some pronominal case errors ('see she'). These relationships are all explicitly encoded in dependency parses.

Dependency graphs

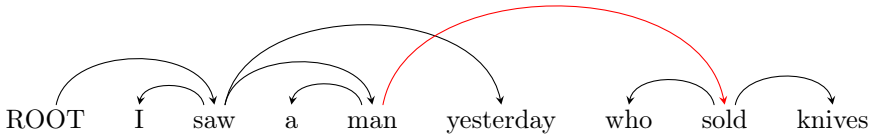


Figure 4.8: Example of a non-projective dependency parse. The non-projective arc is in red.

Dependency parses are represented as directed graphs $\mathcal{G} = \{V, E\}$, where each vertex $v \in V$ represents a word, and each edge $(i, j) \in E$ represents a dependency relationship in which the word at index i is the head of the word at index j. There are several restrictions on G. First, G must be *weakly connected*, meaning that for every node $v \in V$ there must be an edge incident to v. G must also be acyclic, which means that for every pair of nodes $(v_i, v_j) \in V$ either $(i, j) \notin E$ or $(j, i) \notin E$; G is not allowed to contain both of these edges. Each word must also have a *single head*, which means that if $(i, j) \in E$, then $(k, j) \notin E$, assuming $i \neq k$. Finally, many algorithms for dependency parsing require all of the dependencies to be *projective*. This means that for for every arc $(i, j) \in E$ there is a directed path from i to every word k where $i < k < j$. In more visual terms, this requirement prohibits arcs that cross each other, as illustrated in Figure 4.8. There, we see that there is an arc from 'man' to 'sold', but there is no directed path from 'man' to 'yesterday'. We only use projective dependency parses in this thesis, and therefore we focus our discussion on techniques for recovering such dependency parses. For a discussion of techniques for non-projective dependency

parsing, for example maximum spanning tree-based algorithms, which are less efficient than the transition-based dependency parsers discussed below, please see McDonald et al. (2005).

Parsing algorithms

Dependency parsing is a fundamental task in NLP. Its most widespread form involves learning a dependency grammar from a set of manually parsed sentences. To evaluate the parser's performance, one parses a set of test sentences, and compares the predicted dependencies to the manually labeled ones. Although some experiments in this thesis involve learning dependency grammars, we do not evaluate any of them in the standard way (labeled or unlabeled attachment score, which are simply the percentage of correct labeled or unlabeled dependency relations) because appropriate data, with both manual dependency and SALT annotations, is simply not available. We now present some basic algorithms for dependency parsing.

One of the most widespread types of dependency parsers, and the one of most relevance to this thesis, is the *transition-based* dependency parser. Transition-based parsers all share the same basic components, and operate in quite similar ways (Nivre, 2008). First, each sentence x is treated as a sequence of words $w_1, \ldots, w_n$. We prepend the artificial ROOT symbol to the beginning of x, as w_0 (Nivre, 2006).[1] A transition-based dependency parser starts with the ROOT symbol on the *stack* σ, and the rest of x in the *buffer* β. It also includes an empty set of predicted arcs A. Throughout parsing, we represent the state of the dependency parser as the triple $c = (\sigma, \beta, A)$. Starting from the initial state, the parser follows different *transitions*, which involve moving words between the stack σ and the buffer β, and adding dependency arcs to A. The parse is

[1] For an interesting discussion of the effects of ROOT placement, we refer the interested reader to Ballesteros and Nivre (2013)

complete once the buffer β is empty.

Learning a transition-based dependency grammar can be quite straightforward: an *oracle* produces the set of transitions to produce the manual parse for sentence x (Kay, 2000). These oracle transitions provide the gold labels for the learning task, and features are extracted from the parser state c. We note that features are not limited to words and their location in the stack or buffer; training data typically includes more information, for example part-of-speech tags or lemmatized forms (ex. 'play' for 'playing', 'plays', or 'played').

Although transition-based dependency parsers all share the same essential architecture, there are also ways in which they vary. One such way is in the set of transitions a parser uses. Two widespread transition systems, which we discuss below, are called *arc-standard* and *arc-eager*. Two other variable components of these parsers are the features used (ex. Zhang and Nivre, 2011), and the oracle (ex. Nivre et al., 2009; Goldberg and Nivre, 2013), both of which are also discussed below. Finally, we note that while we use dependency parsers in this thesis (Chapter 6), we do so not to produce dependency parses, but rather to identify SALT error codes. We do not modify the inner workings of the parsers, and we only compare the effectiveness of complete parsers for this task, as opposed to, say, investigating what kind of effect using a dynamic oracle has on SALT error code identification.

Step	Stack (σ)	Buffer (β)	Action
1	ROOT	John hit the ball	Shift
2	ROOT John	hit the ball	Shift
3	ROOT John hit	the ball	Left-arc ($hit \rightarrow John$)
4	ROOT hit	the ball	Shift
5	ROOT hit the	ball	Shift
6	ROOT hit the ball		Left-arc ($ball \rightarrow the$)
7	ROOT hit ball		Right-arc ($hit \rightarrow ball$)
8	ROOT hit		Right-arc ($ROOT \rightarrow hit$)

Figure 4.9: Steps to produce arc-standard parse of 'John hit the ball.'

Arc-standard parsing Arc-standard parsing (Nivre, 2008) has three transitions. In all of these, word i is the word at the top of the stack σ and word j is the second word on the stack σ:

1. LEFT-ARC: Add an arc (j, i) to the set of arcs A. Pop word j from the stack.

2. RIGHT-ARC: Add an arc (i, j) to the set of arcs A. Pop word i from the stack.

3. SHIFT: Move word j from the front of the buffer β to the top of the stack σ.

Figure 4.9 contains an example of arc-standard parsing.

Step	Stack (σ)	Buffer (β)	Action
1	ROOT	John hit the ball	Shift
2	ROOT John	hit the ball	Left-arc ($hit \rightarrow John$)
3	ROOT	hit the ball	Right-arc (ROOT $\rightarrow$ *hit*)
4	ROOT hit	the ball	Shift
5	ROOT hit the	ball	Left-arc ($ball \rightarrow the$)
6	ROOT hit	ball	Right-arc ($hit \rightarrow ball$)
7	ROOT hit		Reduce

Figure 4.10: Steps to produce arc-eager parse of 'John hit the ball.'

Arc-eager parsing Arc-eager parsing has four transitions (Nivre, 2008). Although some of these share the same names with transitions in arc-standard parsing, the definition of the transitions may be slightly different. Again, word i is the word at the top of the stack σ and word j is the word at the front of the buffer β:

1. LEFT-ARC: Add an arc (j, i) to the set of arcs A. Pop word i from the stack. This is allowed so long as i is not ROOT and i does not have a head.

2. REDUCE: Pop the stack σ. This is allowed so long as i has a head.

3. RIGHT-ARC: Add an arc (i, j) to the set of arcs A. Move j from the front of the buffer β and push it onto the stack σ. This is allowed so long as j does not have a head.

4. SHIFT: Move word j from the front of the buffer β to the top of the stack σ.

Figure 4.10 contains an example of arc-eager parsing.

Features and oracles

Transition-based dependency parsers extract features from the current parser state Σ at each step. During training, an *oracle* produces the gold label. During testing, the parser simply predicts a transition using the extracted features. At present, the most widely used feature set in dependency parsing is the one proposed by Zhang and Nivre (2011). This feature set includes simple features, for example words and part of speech tags for words in particular positions, for example at the front of the buffer or on the top of the stack. It also includes features capturing the distance between heads and dependents, and higher-order features, for example the head of the rightmost dependent of the word at the top of the stack. For a full exposition of these features, we refer the interested reader to the original paper.

The oracle determines the next transition the parser should choose. Until recently all oracles were *static*. Static oracles take a dependency parse, and following a set of rules, produce a sequence of transitions that produce that parse. In many cases dependency parses exhibit *spurious ambiguity*, which means that there may be more than one sequence of transitions that produce the exact same parse. In practice, however, the oracle's rules cause it to produce a single sequence of transitions for any given parse. More recently, Goldberg and Nivre (2013) proposed using a *dynamic oracle*. Static oracles assume that all previous

transitions up to the current point in a parse are correct, while dynamic oracles do not. Instead, dynamic oracles yield the optimal transition at the current point. Dynamic oracles improve parser performance, and one of the parsers we use in our experiments, Redshift (Honnibal and Johnson, 2014), uses a dynamic oracle.

4.4.3 Parsers

Parser / version	Transitions	Features	Oracle
MaltParser 1.8.1	Arc-eager	Zhang and Nivre (2011)	Static
Redshift - experimental	Arc-eager	Zhang and Nivre (2011), Honnibal et al. (2013)	Dynamic
Stanford Lexicalized Parser	Not transition-based		N/A
Zpar 0.7	Arc-eager	Zhang and Nivre (2011)	Static

Table 4.2: Parsers used in this thesis

Table 4.2 contains an overview of the parsers used in this thesis, particularly in Chapter 6. We now turn our attention to two higher-level tasks that incorporate techniques for structured inference as well as dependency parsing, and which are focus of this thesis: disfluency detection and grammar checking.

4.5 Disfluencies and disfluency detection

A key feature of spoken language is the extensive presence of *disfluencies*, which Bortfeld et al. (2001) found to occur at a rate of 5.97 per 100 words in spontaneous speech. As we discuss in more detail below, several phenomena fall under the umbrella of disfluencies, ranging from filled pauses ('um', 'uh') to complex revisions:

(4.17) I want to go to Dallas by uh I mean fly to Denver first.

Systems that interact with spoken language need to be able to automatically identify disfluencies: unless disfluent elements are deleted, it can be quite difficult to interpret the utterance. If the disfluency were retained in the above example, the user's interaction could be long and frustrating, with the system possibly trying to arrange a ticket from Dallas to Denver, or one just to Dallas, instead of offering a ticket straight to Denver, which is what the user actually wants.

Disfluencies are not just a source of inconvenience for automated systems—they also shed light on various phenomena in the biomedical domain. For example, disfluencies appear to be more common in populations with deficits in inhibitory control, such as the elderly (Kemper et al., 1992; Mortensen et al., 2006) and people with ADHD (Engelhardt et al., 2011). Engelhardt et al. (2013) found that lower inhibitory control, and to a lesser extent lower IQ, both increase the rate of disfluency production in typically developing individuals. *Mazes*, which are the equivalent of disfluencies in SALT (see Section 2.2.3), have been analyzed in research into develompental disorders, as noted in Section 2.3. Automating disfluency or maze detection has the potential to expedite such research by reducing, or even eliminating, the time needed to annotate transcripts. Furthermore, automated annotations could be more consistent than manual annotations, particularly ones applied by different groups. This is in fact an issue in the SALT corpora, as discussed in Section 5.6.

4.5.1 Switchboard disfluency annotations

Category	Example
Filled pause	I uh want a sandwich.
Editing term	He I mean she left already.
Discourse marker	She is, you know, a lawyer.

Table 4.3: Types of disfluencies annotated in Switchboard

The Switchboard annotations (Meteer et al., 1995), have become the *de facto*

standard for disfluency annotations in the field of natural language processing. The Switchboard annotations split disfluencies into two major types: non-sentence elements, and restarts. These two categories of disfluencies vary in how difficult they are to detect automatically.

The most straightforward type of disfluency to identify automatically is the *filled pause*. Filled pauses are a closed class that minimally contains 'uh' and 'um'. Uniquely among the disfluency categories, filled pauses are always disfluent.

There are three other closed-class[2] categories of disfluencies: *editing terms* ('I mean', 'excuse me'), *discourse markers* ('well', 'you know'), and *coordinating conjunctions* ('but', 'but anyway', 'and'). Unlike filled pauses, these terms are not always disfluent. For example, 'you know' is a discourse marker in the first of these utterances, but not the second:

(4.18) It's {F you know } that kind of place.

(4.19) You know that we can't afford that.

Filled pauses, editing terms, coordinating conjunctions and *asides* are all grouped together in a category labeled *non-sentence elements*. As shown in the example above, non-sentence elements are delimited with braces and coded with a the first letter of the relevant category (F, E, C or A). *Asides* are defined as "when the corresponding sequence of words interrupts the fluent flow of the sentence AND the sentence later picks up from where it left off" (Meteer et al., 1995, p. 15). For example:

(4.20) I think those satellites {D you know} {A or not satellites, but the spaceflights } could really spy.

This definition of aside is quite vague, and even the original annotation guidelines contain examples where the authors are not confident whether certain disfluencies

[2] *Closed-class* means that all possible words or phrases in each category could be enumerated, at least in theory.

should be annotated as asides. The authors present three examples of utterances that may contain asides, even though they admit that "it is not clear that this is the right analysis" (Meteer et al., 1995). One of these examples is:

(4.21) I don't know if it's even true that it's always unanimous because I thought there were cases [where + {F uh} {A I don't know if it's the difference between felonies and misdemeanors, but } where] it was okay for a state to have it {D like } eleven out of twelve.

Restarts are disfluencies that serve to correct whatever the speaker has just said. In the Switchboard annotations, restarts are broken into four parts, each of which can be seen in the following utterance:

(4.22)

$$\text{I want to go } [\underbrace{\text{to Dallas}}_{\text{RM}} \overbrace{+}^{\text{IP}} \underbrace{\{\text{F uh }\}\ \{\text{E I mean }\}}_{\text{IM}} \underbrace{\text{to Denver}}_{\text{RR}}] \text{ on Monday.}$$

First, there is the *reparandum* (RM), which contains the words the speaker wants to correct. The *interruption point* (IP) marks the beginning of the correction. The *interregnum* (IM) contains filled pauses and editing terms, and precedes the repair (RR), which contains the correction. The IP is the only component of a restart that is a single point in time; all other components are spans of time. Note that while the reparandum cannot be empty, the interregnum can be. It can be difficult to distinguish deletions (with an empty interregnum) and substitutions:

(4.23) They only give you [a +] so many cents for each bottle.

(4.24) They only give you [a + so] many cents for each bottle.

Switchboard also permits *nested* disfluencies:

(4.25) It must be depressing [to + to] [walk + walk] the halls [and see + [and + and] see] all these people.

As detailed in Section 2.2.3, SALT maze annotations are binary: words are either in a maze or they are not, and therefore there is no internal structure to mazes, and no need to distinguish between substitutions and deletions:

(4.26) They only give you (a) so many cents for each bottle.

(4.27) It must be depressing (to) to (walk) walk the halls (and see and) and see all these people.

In sum, the Switchboard disfluency annotations are far richer than the SALT maze annotations. Maze annotations are binary, thus collapsing all of the distinctions made by the Switchboard annotations between non-sentence elements, restarts, and the components of both of these categories. Maze annotations also cannot be nested, and therefore have minimal structure compared to Switchboard disfluencies.

4.5.2 Automated disfluency detection

Automatic disfluency detection is a well-established task in NLP, and is certainly the most relevant field of research to automating maze detection, which we address in Chapter 5. There are, however, four ways in which maze detection differs from disfluency detection. First, many disfluency detection systems leverage features from the speech signal, but as discussed below, using the speech signal is often not practical for maze detection. Second, nearly all previous research in the field of disfluency detection has ignored data collected from children, let alone ones who may have impaired language or communication. Third, there are limited resources for SALT-annotated data: we are not aware of SALT-annotated corpora that also have manual parse and part of speech

annotations, as is the case with the Switchboard corpus. Finally, SALT maze annotations are far simpler than other disfluency annotations, most notably the ones used in Switchboard, and as a result, some methods for disfluency detection are not directly applicable to maze detection. We address each of these distinctions in turn.

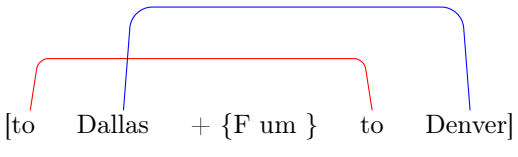
Techniques for disfluency detection can be divided into speech-first and text-first approaches (Nakatani and Hirschberg, 1993): in the former, one extracts features primarily from the speech signal (for example prosodic cues such as pitch), while in the latter, one ignores the speech signal, attempting to identify disfluencies only from text. Many approaches fall in-between, for example leveraging both textual and prosodic features (ex. Ferguson et al. 2015). Approaches leveraging the speech signal cannot be leveraged in several of the experiments performed here due to lack of audio. Even if the audio is available, as with the CSLU ADOS corpus (see Section 3.2), there is often all sorts of noise that make it difficult, if not impossible to extract reliable cues and time alignments from the speech signal, including clapping, overlapping speech, and banging toys. As a result, we only consider previous work in text-first disfluency detection that does not leverage features from the speech signal.

Previous research on automatically identifying disfluencies has almost exclusively focused on conversational language from adults who lack any acknowledged language impairment or neurological disorder (i.e. who are presumably TD). Such spoken language which may differ markedly from the spoken language we would typically annotate with SALT, which tends to come from children, and often ones suspected of having impaired language or communication. This focus is due at least in part to the paucity of data, particularly the lack of corpora comparable in size and annotation quality to the Switchboard corpus. Several corpora, all of which contain spoken language collected from presumably

TD adults, have fallen out of favor compared with Switchboard, both because they are smaller (in the case of the CallHome corpus (Canavan et al., 1997)), and because the language in many of them is more templatic, coming from narrow domains such as planning air travel with either an automated dialog system (ex. the ATIS corpus (Hemphill et al., 1990)) or another human (the AMEX (Kowtko and Price, 1989) and Verbmobil corpora (Wahlster, 2000)), or railroad freight transportation, as in the TRAINS corpus (Heeman and Allen, 1995). Although some transcripts in the CHILDES database (MacWhinney, 1992), which contains transcripts of spoken language collected from children, do contain limited disfluency annotations, we are not aware of any attempts to automatically identify these disfluencies.

One of the first investigations taking a text-first approach to disfluency detection was conducted by Charniak and Johnson (2001). There, they used boosted linear classifiers to identify edited words. Later, Johnson and Charniak (2004) improved upon the linear classifiers' performance with a tree adjoining grammar-based noisy channel model. Briefly, tree adjoining grammars are a formalism powerful enough to describe the cross-serial dependencies words commonly observed in disfluencies. More simply, tree adjoining grammars are powerful enough to describe the following structure, which cannot be described with a simple constituency parse because lines in them cannot cross:

(4.28)



Zwarts and Johnson (2011) improved the noisy channel model by adding in a re-ranker that leveraged features produced by a large language model. Georgila (2009) investigated a wide variety of techniques to identify candidate disfluencies, including conditional random fields, hidden-event language models, and maximum

entropy models, and then using integer linear programming to decide which candidate disfluencies to accept.

Recent research has focused on jointly performing dependency parsing (discussed in Section 4.4.2) and disfluency detection. This technique has its roots in the combination of constituency parsing and disfluency detection, which was first suggested by Charniak and Johnson (2001), although only later did this line of research become more popular (Johnson and Charniak, 2004; Lease et al., 2006; Miller, 2009; Miller et al., 2009). Approaching disfluency detection as an extension of constituency parsing is quite natural, as both involve identifying spans of words. Nevetheless, in this thesis we do not attempt to adapt constituency-parsing based techniques for disfluency detection to the task of maze detection because these techniques are no longer the best performing, nor are they as efficient as the best-performing techniques.

At present, the best-performing text-first disfluency detector is the one proposed by Honnibal and Johnson (2014), which does indeed perform dependency parsing and disfluency detection jointly. Differences in training and decoding make the performance of this system difficult to compare to the one proposed by Rasooli and Tetreault (2013), but Honnibal and Johnson's system narrowly outperforms another joint dependency parser/disfluency detector proposed by Rasooli and Tetreault (2014).

Although performing disfluency detection jointly with dependency parsing is clearly effective, this technique is not easily adaptable to the sort of data that one would typically annotate with SALT because there is simply no in-domain training data containing both manual maze and dependency parse annotations. Producing manual dependency parses requires trained annotators and is time-consuming. Using domain adaptation techniques with an automatic dependency parser is also unlikely to be effective: as we show in Chapter 5, the SALT corpora

do not constitute a homogeneous domain. Given that there are such discrepancies between corpora of spoken language collected from children ostensibly annotated following the same standards, it seems unlikely that a corpus of spoken language annotated collected from adults, annotated with entirely different standards, namely Switchboard, would provide appropriate training data for a joint maze detector/dependency parser.

Qian and Liu (2013) proposed the highest-performing text-first disfluency detector that does not leverage any sort of syntactic or dependency annotations. Their detector operates in three steps, and heavily leverages the Switchboard annotations. Unlike most of the aforementioned research, Qian and Liu do not approach disfluency detection as a *bracketing* task in which spans of disfluent words are identified. Instead, they approach disfluency detection as a *tagging* task in which sequences of tags are identified. Specifically, each word can have one of five tags: not disfluent (S-O), single word disfluency (S-D), begin multiword disfluency(B-D), inside multiword disfluency (I-D), and end multiword disfluency (E-D). Mapping between disfluency brackets and tags is trivial:

(4.29) (I) I (uh I think) I think he left

→

I/S-D I/S-O uh/B-D I/I-D think/E-D I/S-O think/S-O he/S-O left/S-O

The first pass of Qian and Liu's disfluency detector identifies non-sentence elements (filled pauses, editing terms, etc.), while the the second and third steps detect reparranda. The non-sentence element detector is a conditional random field tagger, and it uses features constructed from words, part-of-speech (POS) tags, and word/POS tuples (these features are presented in detail in Section 5.3). Specifically, it uses n-grams orders 1-3 (sequences of 1-3 words, POS tags, or word/POS tag tuples). It also uses binary features that capture whether one or two words or part of speech tags are repeated exactly (they refer to these as

'logic uni/bigrams'. For example, such a feature comparing the current word ('for', underlined) to the previous word (him) in:

(4.30) I bought it for him for her.

would be negative, but comparing it to two words back ('for') would be positive.

The second and third steps of Qian and Liu's disfluency detector identify reparranda using essentially the same features as the first step. One minor difference between the non-sentence element detector and the reparrandum detector is that the latter uses a M^3N tagger (Section 4.3.3) instead of a CRF tagger (Section 4.3.2). A more substantial difference is that each step after the first uses features extracted not only from the unmodified utterance, but also from the utterance after disfluent elements identified in the previous steps have been excised.

In some ways, Qian and Liu's disfluency detector is easily adaptable to maze annotation because it only uses features extracted from words and POS tags, and these can be automatically extracted without any difficulty, even from the sort of data that would typically be annotated with SALT. On the other hand, Qian and Liu's detector leverages distinctions made by Switchboard annotations that are absent in SALT maze annotations. Specifically, it identifies non-sentence elements in one step, and reparranda in another, while SALT does not distinguish between these two categories. Furthermore, Qian and Liu demonstrate that each additional pass improves disfluency detection. Nevertheless, we believe that Qian and Liu's detector is particularly appropriate to adapt to the task of maze detection due to the fact that it only uses very simple features, and because even a single pass of their detector performs well on Switchboard.

We now turn our attention to previous research in grammar checking, which unlike all of the work in disfluency detection, has focused primarily on written language.

4.6 Grammar checking

Previous research in NLP has addressed several tasks related to handling ungrammatical sentences: grammatical error *detection*, which only involves identifying whether a sentence contains a grammatical error; grammatical error *classification*, which requires identifying the type of any errors in a sentence (ex. subject-verb agreement error in 'She leave.'); and finally grammatical error *correction*, which requires producing the grammatical version of an ungrammatical sentence. We refer to these tasks collectively as *grammar checking*. Most previous research in grammar checking has focused on developing two types of tools: ones to assist native speakers while proofreading, which have been the focus of commercial research, and tools to assist second language learners, which have been a more popular interest in academic research.

The task of automatically identifying SALT error codes has three key characteristics that distinguish it from more traditional grammar checking, and these three characteristics lead us to identify three requirements for a useful SALT error code detector. First, SALT error codes are applied to transcripts of *spoken language* collected from children, including ones suspected of having a language impairment, and therefore a SALT error code detector must be *robust to phenomena in spoken language*. Second, SALT error codes are not typically applied in a consistent manner across corpora and research groups. For example, comparing Tables 3.4 and 3.7, we see that the ENNI corpus does not include the [EO] overgeneralization code, but the NarSSS corpus does; overgeneralization errors, along with a variety of others, are annotated with [EW] codes (word-level errors) in the ENNI corpus. Finally, there are various ways a SALT error detector could be used, for example: sampling a few utterances likely to contain an error, annotating utterances before manually correcting them, or removing utterances without errors before manual annotation of the rest. Each of these uses suggests

a different operating point in terms of precision and recall, and therefore a SALT error code detector should be *tunable*—*i.e.* the user should be able to adjust the trade-off between precision and recall. As we discuss below (Section 4.6.3), this ability is notably absent from many grammar checking systems.

Having enumerated the key differences between a SALT error code detector and a traditional grammar checker, we now turn to previous research in grammar checking, particularly as it relates to the task of automating SALT error code application and its three key requirements.

4.6.1 Spoken language

Most of the research in grammatical error checking has focused on developing tools to proofread formal, written English, which differs in several critical ways from the transcripts of spoken language that we are interested in analyzing. First, spoken language contains disfluencies, which are absent from written language, and incomplete utterances, which are acceptable in spoken language, even though incomplete sentences are typically considered ungrammatical in written language. We are aware of no previous work in grammar checking that has investigated the robustness of a grammar checker to either disfluencies or incomplete utterances. There has, however, been a small amount of research on grammar checking in transcripts of spoken language. Bowden and Fox (2002) propose a rule-based system to diagnose errors in spoken language from non-native speakers of English. Their system, however, is not trainable, and furthermore, it is not able able to evaluate sentences with more than a single clause. Lee and Seneff (2006) proposed a system to identify a variety of errors in transcripts of conversations in which users interact with a spoken dialog system to book a flight. This system uses an n-gram language model to produce candidate corrections for an utterance with simulated errors, and then a stochastic context free grammar to rerank

these candidates. Their system performs quite well, but it is only evaluated in a narrow domain. Although their system has two trainable components, it is unclear clear how one could use it to classify grammatical errors instead of correcting them. Lee and Seneff (2008) also proposed a system that uses template-matching on parse trees to correct verb errors in transcripts of spoken language from Japanese learners of English. This system is evaluated on open-domain dialogs, but since it is template-based, it is not trainable. Crucially, none of these investigations address disfluencies or incomplete utterances: the only mention of these phenomena is in Lee and Seneff (2008), which does not attempt to correct incomplete sentences, and which notes that "colloquial phenomena", for example using *like* as a filler, are difficult to process. In a sense then, these works have addressed grammar checking of language that happens to be spoken, as opposed to archetypal spoken language.

More recent investigations have dealt with spoken language that is quite distinct from written language. Caines and Buttery (2010) found a logistic regression model was able to identify the zero-auxiliary construction (e.g. 'You going home?'), which is comparatively rare in written language, with over 96% accuracy (Caines and Buttery, 2010). Even though the zero-auxilliary construction is not necessarily ungrammatical, identifying such constructions may be useful as a preprocessing step to a grammaticality classifier. They also demonstrated that their detector can be integrated into a statistical parser yielding improved performance, although they are vague about the nature of the parse improvement (Caines and Buttery, 2010, p. 6). However, when Hassanali and Liu (2011) reimplemented this zero-auxilliary detector and adapted it to transcripts of spoken language collected from children, they found that its performance was extremely poor.

Hassanali and Liu (2011), which is the most relevant investigation to this

thesis, evaluated several methods for grammatical error detection in transcripts of spoken language from children, some of whom had impaired language. They identified 11 types of errors in these transcripts, 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. The authors explicitly note that "as is prevalent in spoken language corpora, these transcripts had disfluencies, false restarts and incomplete utterances which sometimes pose problems to the parser" (Hassanali and Liu, 2011, p 4), but it is not clear from the paper how these were handled[3]

4.6.2 Trainability

Grammar checkers designed for written language are typically designed to identify deviations from standard formal, written English. Although there is no single definition of what constitutes 'correct' written English, some types of errors are universally acknowledged, including: subject-verb agreement errors, prepositional errors, and omitted articles. Other constructions may be grammatical, but unacceptable to certain writers, for example split infinitives or prepositions at the end of a sentence. And beginning a sentence with 'and'. In other words, most users of such grammar checkers are trying to produce language that conforms to similar, if not identical, standards. This situation means that grammar checkers do not have to be trainable for them to be effective.

The grammar checker in Microsoft Word (MS Word) is likely the world's most widely used. Like many previous systems, most notably IBM's Epistle (Heidorn et al., 1982; Jensen et al., 1983), the MS Word grammar checker consists of a hand-built grammar that is used to parse sentences. This grammar is quite

[3]The authors did not respond to an email asking about this either.

flexible, which allows even ungrammatical sentences to be parsed. For example, intransitive verbs are allowed to take direct objects, even though they will be penalized for doing so (Leacock et al., 2014). We note that this flexibility is not universal among hand-built parsers: link grammar parsers fail to parse ungrammatical utterances by design (Sleator and Temperley, 1991). In addition to these flexible rules, the MS word grammar checker also has the ability to fit parses of substrings into a complete parse. Typically a given sentence will yield multiple parses, which are then scored heuristically, and this score is used to determine whether or not the sentence contains a grammatical error. Due to the hand-crafted rules and heuristics, the MS Word grammar checker is not trainable. It is, however, somewhat customizable: users are allowed to select which errors they want to be alerted to (see page 173 for a description of these options). For more details on the grammar checker in MS Word, including a history of the system's development, we refer the interested reader to Leacock et al. (2014).

Although not available for consumer use, ETS E-Rater (Attali and Burstein, 2004), which is a system for automatically grading essays, contains a well-developed grammar checking component (Shermis and Burstein, 2013). ETS E-Rater does not only flag utterances that contain errors; it also identifies the types of errors in them. While the details of ETS e-Rater are proprietary, it consists of a variety of detectors of specific types of grammatical errors, some of which are statistical, and others of which are rule-based. Like the grammar checker in MS Word, these detectors are set to a high precision operating point so as to avoid false positives. Like MS Word, e-Rater has many manually-tuned components, and therefore is not easily adaptable to novel standards or different error sets. For a more detailed overview of ETS e-Rater, we refer the interested reader to Gamon et al. (2013, pp 262-263).

More recent research has focused on data-driven techniques for grammar checking, and most systems are based on techniques from machine learning, machine translation, language modeling, or a combination of these. Many also include hand-crafted rules because certain types of errors, for example verb overgeneralization (ex. 'eated' instead of 'ate'), are easier to identify and correct with rules than with data-driven approaches (Gamon et al., 2013). Nevertheless, each of these types of systems is trainable, at least to an extent, and we give a brief overview of each in turn.

The use of machine-learning based techniques, and in particular classifiers, for grammatical error detection has its roots in research on identifying spelling errors in English where either homophones ('your' and 'you're') or similar sounding words ('effect' for 'affect') are confused (Golding, 1995). These approaches looked at the context of the confused words to create separate models for each set of them. This approach is viable for such spelling errors because there are only so many of them, but extending it to arbitrary grammatical errors is not practical. More recently, several systems proposed for the CoNLL shared tasks on grammatical error correction (Ng et al., 2013, 2014) have used error-specific classifiers for grammar checking (Rozovskaya et al., 2013; van den Bosch and Berck, 2013; Rozovskaya et al., 2014). Approaches based on error-specific classifiers, however, are impractical for SALT error code detection, where the set of errors is not consistent across corpora. A different classifier based approach at the CoNLL 2013 shared task used a single classifier to propose corrections and a language model to filter these corrections (Jia et al., 2013). This system, however, refines all of the manual error labels using a set of rules, which limits its ability to be adapted rapidly to novel error sets.

Both the language-modeling and machine translation-based approaches in the CoNLL 2013 and 2014 shared tasks leveraged the corrections included in

the training data. The language modeling approaches typically compared the probability of an observed n-gram with the probability of n-grams of potential corrections, while the machine translation approaches treat grammar correction as a translation task: the input is a sentence that potentially contains an error, and the output is the corrected sentence (Ng et al., 2013, 2014).

Although the approaches described above can be successful for formal, written English, and could easily be extended to other domains where corrected text is available (ex. instant messaging), they are not necessarily appropriate for identifying SALT error codes. First, these approaches are far more useful for correcting a wide variety of errors than they are for classifying them. Identifying and classifying errors, however, are the primary goal of automatically identifying SALT error codes, as most error codes do not include corrections. Finally, given the inconsistencies we identify in the SALT maze and error code annotations (Chapters 5 and 6, respectively), it seems highly likely that the corrections would be inconsistent as well, thus making it quite difficult to learn an effective language model or model for machine translation, leaving aside issues of data sparsity.

4.6.3 Tunability

Commercial grammar checkers typically have very specific demands regarding their operating point, which is the balance between flagging true as well as false errors. For example, the grammar checker in MS Word underlines sentences with errors to draw a user's attention to them. In this scenario, the designers found it important to reduce the number of falsely flagged errors to below one per page on average, because users find a high number of falsely flagged errors to be irritating (Helfrich and Music, 2000). They also found that preferences about errors vary among users: German writers tend to care about case and

capitalization errors, while French and Spanish writers are more interested in agreement errors. Helfrich and Music describe how user feedback is used to tune the MS Word grammar checker very finely, but this process is highly involved. Such a laborious tuning procedure is reasonable for a commercial tool that needs to be tuned once, as is the case with MS Word's grammar check, but it is far too expensive to be applicable to a SALT error code detector. Some other grammar checkers, particularly the machine learning and language-modeling based tools proposed for the CoNLL shared tasks (Ng et al., 2013, 2014), are easily tuned, even if they are not adaptable to SALT error code detection for reasons discussed above in Section 4.6.2.

4.7 Evaluation

Throughout this thesis we evaluate systems in terms of *precision* (P), *recall* (R) and *F1 score* (F1). These are defined as follows:

$$\mathrm{P} = \frac{tp}{tp + fp} \tag{4.31}$$

$$\mathrm{R} = \frac{tp}{tp + fn} \tag{4.32}$$

$$\mathrm{F1} = \frac{2\mathrm{PR}}{\mathrm{P} + \mathrm{R}} \tag{4.33}$$

where tp and fp are the counts of true and false positives, respectively, and fn is the count of false negatives. All of these range between 0 and 1, with 1 indicating perfect performance. To illustrate, suppose we have a system that identifies ungrammatical utterances: if it has high precision, then most of the utterances it identifies as ungrammatical will in fact be ungrammatical; if it has high recall, then it will have identified most of the ungrammatical utterances in the data set.

In practice, there is typically a trade-off between precision and recall. In many cases, including the maze and SALT error code detectors described in Chapters 5 and 6, users of automated systems can set the *operating point* of the system, or in other words, change the balance between precision and recall. Finally, the F1 score combines precision and recall into a single metric. In general, we use the F1 score instead of, say, the mean of precision and recall, because the F1 score is not particularly sensitive to improvement in either one of precision or recall, but rather improvement in both of them.

4.7.1 Randomized paired-sample test

Many of the experiments in this thesis involve comparing the output of two systems A and B. In order to determine whether the distribution of the outputs yielded by two systems are significantly different, we adopt the *randomized paired-sample test* proposed by Noreen (1989) (see also Yeh (2000)).

Let A be the system with lower performance, and B be the system with higher performance. We first compute the difference d in performance between the two systems. Then, we repeatedly construct a random set of predictions for each input item by choosing between the outputs of system A and B with equal probability. We evaluate the performance of these randomly varied predictions n times, and if it exceeds the performance of the baseline system by at least d, we count the iteration as a success. We can then compute an upper bound on the significance level p:

$$p \leq \frac{s}{n+1} \tag{4.34}$$

where s is the number of successful iterations (Noreen, 1989).

This procedure can be used to compare the output of any two systems, and in this thesis we use it to compare systems that predict maze annotations (Chapter

5), scores on structured instruments (Chapter 7), and the presence of autism or a language impairment (also in Chapter 7). This procedure is also agnostic to the performance metric used, whether it be F1-score, a correlation metric such as Kendall's tau, or the area under the receiver operating curve.

Although the randomized paired-sample test does not tell us whether the system of interest B performs significantly better than the baseline system A, it does tell us whether the outputs of A and B are drawn from significantly different distributions. If the output of the two systems are drawn from significantly different distributions, and the output of B is better than the output of A, then this suggests, but does not show conclusively, that B outperforms A.

4.8 Conclusions

In this chapter we have provided the necessary background for the rest of this thesis. We began with an overview of classification, graphs, and parsing, in particular dependency parsing. These topics underpin the areas of NLP that are most relevant to this thesis, namely disfluency detection and grammar checking. Disfluency detection is very closely related to maze detection, which we explore in Chapter 5. There we see that minor modifications to Qian and Liu's disfluency detector enable it to perform maze detection with comparable performance to disfluency detection. Using this detector, we also identify severe inconsistencies in the maze annotations in the SALT reference corpora.

At first glance, grammar checking appears to be similar to SALT error code detection (Chapter 6); after all, both tasks consist of identifying and labeling ungrammatical language. Nevertheless, we have argued that a useful tool to identify SALT error codes has several differences from most previously proposed grammar checkers. First, a SALT error code detector needs to be *trainable* so that a given research group can have it produce annotations that are in line with

their own, rather than ones that conform to a universal standard. Second, it needs to be *robust to phenomena in spoken language*. Third and finally, the user must be able to *tune the trade-off* between precision and recall so that it can be used for a wide variety of tasks. Having provided the necessary background, we now continue to our own work, beginning with automating maze detection.

Chapter 5

Maze Detection

In this chapter we consider several issues pertaining to automating SALT *maze* annotations, which indicate disfluent speech. After a brief review of the maze guidelines and their utility in both research and language sample analysis, we present a simple tool that is able to identify mazes quite effectively.[1] We evaluate the maze detector both in terms of how well it replicates manual maze annotations, and in terms of how much certain statistics derived from transcripts with automatic maze annotations diverge from ones computed from transcripts with manual maze annotations. These statistics are all ones that are used for research into language development and impairment, as discussed in Chapter 2. We also explore whether it is possible to use the same maze detector for multiple corpora, or whether the annotations and language vary too much between corpora for this to be effective. Finally, we examine the consistency of maze annotations in the SALT corpora, as well as potential differences between the SALT corpora, for example differences in utterance segmentation or richness of vocabulary. We identify severe inconsistencies with standard maze annotation practices in the

[1] Much of the work in this chapter is based on our paper *Challenges in Automating Maze Detection* (Morley et al., 2014).

GillamNT corpus, which we discuss in Section 5.6.3. These inconsistencies affect the reference statistics, for example mean length of utterance, computed from the GillamNT corpus, making it of questionable value as a reference database.

5.1 Background

5.1.1 Annotation guidelines

(You have you have um there/'s only) there/'s ten people

(a) SALT maze annotation

[[You have + you have] + {F um}] [there/'s only + there/'s] ten people

(b) Switchboard disfluency annotation

Figure 5.1: Maze and disfluency annotations

As discussed in more detail in Section 2.2.3, the SALT manual gives very brief guidelines for annotating mazes, defining them simply as "filled pauses, false starts, and repetitions and revisions of words, morphemes and phrases" (Miller et al., 2011, p. 48). Although this definition implicitly acknowledges differences between filled pauses, false starts, and repetitions, SALT maze annotations do not: contiguous spans of words in mazes are simply delimited with a single set of parentheses. This means that mazes cannot be nested (either properly or improperly), maze spans cannot overlap, and mazes have no internal structure. SALT maze annotations are perhaps best viewed as binary word-level annotations: each word is either in a maze, or not in a maze. Figure 5.1 illustrates the flat nature of maze annotations, and for contrast, it also shows the same utterance annotated following the Switchboard guidelines (Shriberg, 1994), discussed in Section 4.5.1, which are widely used by speech and NLP researchers.

The self-paced SALT online training materials (SALT Software, 2014b) con-

tain a few more details regarding maze annotation. They indicate that in cases where there is a choice of words to mark, annotators are to consider the final production successful, and the previous productions as being in a maze. For example, in the utterance:

(5.1) (I) I saw it.

it would be incorrect to mark the second occurrence of the word 'I' as being in a maze. Additionally, both partial words, and stuttering in the middle of a word are always included in mazes:

(5.2) He ate (an ap_ p* p* _ple) a green apple.

(5.3) I saw (hi*) them go.

Recall that partial words are indicated by the spoken material followed by '*', and that stutters are delimited by '_' with any repeated sounds followed by '*', as described in Section 2.2.2.

5.1.2 Utility of maze annotations

Mazes have sparked interest in the literature about child language disorders, where they are typically analyzed from a language processing perspective in which disruptions to fluency are viewed as a consequence of monitoring, detecting and repairing language, potentially including speech errors (Levelt, 1993; Postma and Kolk, 1993; Rispoli et al., 2008). Several studies have found that as grammatical complexity and utterance length increase, the number of mazes increases in typically developing children and children with language impairments (MacLachlan and Chapman, 1988; Nippold et al., 2008; Reuterskiöld Wagner et al., 2000; Wetherell et al., 2007). Mazes in narrative contexts have been shown to differ between typical children and children with specific language impairment (MacLachlan and Chapman, 1988; Thordardottir and Weismer, 2001),

though others have not found reliable group differences (Guo et al., 2008; Scott and Windsor, 2000).

In addition to the potential usefulness of analyzing language in mazes, mazes must be identified and excised in order to calculate many statistics used in language sample analysis, even ones as basic as mean length of utterance, and type or token counts. In addition to these statistics, the SALT software also computes the mean number of mazes per utterance, and the mean number of words per maze, both of which obviously require maze annotations to compute. Finally, maze annotations are critical for analyzing errors: some mazes are in fact self-corrections of language or speech errors, and therefore errors within mazes should generally be ignored. In practice, error codes are not typically annotated within mazes.

5.1.3 Inter-annotator agreement

Heilmann et al. (2008) investigated inter-annotator agreement on several types of SALT annotations, including maze annotations. They investigated agreement on maze annotations under three conditions. In all three, they measured agreement between annotators in terms of the number of words and bound morphemes in mazes, as well as *maze placement*, which is essentially maze spans. For example, the following pair of annotations contain four disagreements at the word/morpheme level (you, have, there, /'s), and a single disagreement at the maze placement level:

(5.4) (You have you have um there/'s only) there/'s ten people

(5.5) You have (you have um there/'s only there/'s) ten people

If, however, excising different mazes results in the same utterance, then maze disagreements are ignored. For example, the following pair of annotations would

be counted as having zero disagreements because excising either maze produces the utterance 'I went to the store':

(5.6) I went (I went) to the store

(5.7) (I went) I went to the store

Heilmann et al. (2008) first measured inter-annotator agreement in terms of *transcription consensus*. First an annotator transcribed and annotated a sample of spoken language. A second annotator then reviewed the first transcriber's work, and the two resolved any discrepancies through discussion. The authors found both maze placement and maze word/morpheme agreement to be 99% when measured under this condition. For the second condition, *protocol coding accuracy*, they had annotators with ten hours of training and under a year of experience transcribe a recording of spoken language along with SALT annotations. They then had expert annotators review these transcripts and record the number of disagreements. Heilmann et al. (2008) reported 98% agreement for maze placement and 100% agreement for maze word/morpheme agreement under this condition. For the final condition, *independent transcription accuracy*, two teams of annotators with unspecified amounts of experience transcribed the same sample of spoken language and applied SALT annotations. Crucially, the annotators were each blind to the other's transcriptions and annotations. Heilmann et al. (2008) reported 98% agreement for maze placement and 100% maze word/morpheme agreement under this condition.

5.2 Automating maze detection

Here we describe how maze detection can be approached as a task in NLP. Specifically, we describe the input and output of the task, and discuss evaluation.

5.2.1 Input and output

We assume that the input to the maze detector will be a single *c-unit*, which is defined as "an independent clause with its modifiers" (Miller et al., 2011). We also assume that there are no SALT annotations, for example morphological annotations or error codes, within the c-unit, and that all partial words, non-speech sounds, and punctuation have been removed, as described in Section 3.3. Again, both training and test data are preprocessed in the same way, and we remove all utterances with unintelligible words from both training and test data. One would expect that including punctuation would facilitate maze detection, and preliminary experiments suggest this is indeed the case. Nevertheless, we remove punctuation for two reasons. The first reason is that doing so is standard practice in the disfluency detection literature, largely because the output of a speech recognition system would not necessarily include punctuation. Second, we do not want inconsistencies in punctuation between corpora to affect maze detection performance.

The maze detector outputs a single tag for each word in the input (see example 4.29), and this gets converted to a binary tag that indicates whether that word is or is not in a maze.

5.2.2 Evaluation

We evaluate the maze detector both *intrinsically* and *extrinsically*. Intrinsic evaluation, which is quite common in NLP tasks, captures how well a system replicates manual annotations, while *extrinsic* evaluation captures how useful the system output is for a downstream task. Extrinsic evaluation is critical to our investigation because maze detection is not simply a task in NLP, but rather a necessary step of language sample analysis using SALT. Here we discuss both intrinsic and extrinsic evaluation as they apply to an automatic maze detector.

Intrinsic evaluation

Gold: (and then it oh) and then it (um) put his wings out .
Prediction: (and then it) oh and then it (um) put his wings out .

Word	Gold	Pred.	Outcome
and	✗	✗	TP
then	✗	✗	TP
it	✗	✗	TP
oh	✗		FN
and, then, it			TNx3
um	✗	✗	TP
put, his, wings, out			TNx4

(a) Tagging evaluation

Span	Gold	Pred.	Outcome
(and then it oh)	✗		FN
(and then it)		✗	FP
(um)	✗	✗	TP

(b) Bracketing evaluation

Figure 5.2: Tagging and bracketing evaluation for maze detection; ✗indicates word or span is in a maze
TP = True Positive, FP = False Positive, TN = True Negative, FN = False Negative

We evaluate our maze detectors intrinsically in two different ways: *tagging* performance and *bracketing* performance, both of which are standard forms of evaluation for various tasks in the NLP literature. *Tagging* performance captures how effectively maze detection is done on a word-by-word basis, while *bracketing* performance describes how well each maze is identified in its entirety. For both tagging and bracketing performance, we count the number of true and false positives and negatives, as illustrated in Figure 5.2. In tagging performance, each word gets counted once, while in bracketing performance we compare the predicted and observed maze spans. We use these counts to compute performance in terms of precision, recall, and F1-score. See Section 4.7 for details on these scores.

Note that partial words and punctuation are both excluded from test data,

and hence from evaluation. We eliminate punctuation because it does not need to be included in mazes: punctuation is not counted in summary statistics (e.g. MLU, word count, etc.), and punctuation errors are not captured by the SALT error codes. Although we excise partial words for our experiments here, others may choose to include them. If this is done, partial words should be excluded from evaluation because they are always in mazes, and therefore can be detected trivially with a simple rule. We exclude partial words from detection for this reason, and so that performance metrics are comparable across corpora, even if they vary widely in the frequency of partial words. Finally, future work in maze detection may want to exclude both punctuation and partial words from evaluation so that our results can be used as a baseline.

We use the randomized paired-sample procedure described in Section 4.7.1 to compare pairs of models for maze detection. In particular, we use this test to see whether the tagging or bracketing F1 scores of the two models are significantly different.

Extrinsic evaluation

While precision and recall allow us to compare two different maze detectors, they do not give us a complete picture of how useful the maze detectors are in automating a step of SALT: in SALT, maze annotations permit the computation of certain summary statistics, and ideally automatically applied maze annotations will yield statistics similar to ones derived from manual annotations. Specifically, maze annotations are directly used to compute: MLU in words (and morphemes), token and type counts, the number of utterances with mazes, the total number of mazes, the number of words in mazes, and the percentage of words in mazes. After comparing various system configurations in terms of intrinsic performance (in Section 5.4), i.e. how well they are able to reproduce manual annotations, we will look at how summary statistics derived from the output of the best-

performing system compare to ones derived from manually annotated transcripts. This investigation is in Section 5.5.

5.3 Maze Detector

Category	Features
Unigrams	$t_{i-2},\ t_{i-1},\ t_i,\ t_{i+1},\ t_{i+2}$
Bigrams	$t_{i-1}t_i,\ t_it_{i+1}$
Trigrams	$t_{i-2}t_{i-1}t_i,\ t_{i-1}t_it_{i+1},\ t_it_{i+1}t_{i+2}$
Logic Unigrams	$I(t_i,\ t_{i-j})$ $-4 \leq j \leq 4;\ j \neq 0$
Logic Bigrams	$I(t_{i-1}t_i,\ t_{j-2}t_{j-1})$ $I(t_it_{i+1},\ t_jt_{j+1})$; $-4 \leq j \leq 4;\ j \neq 0$
Predicted tag	y_{i-1}

Table 5.1: Feature templates for maze word detection, following Qian and Liu (2013). We extract all of the above features from both words and part-of-speech (POS) tags, albeit separately. t_i indicates the current word or POS tag, while t_{i-1} is the previous one and t_{i+1} is the following. The function $I(a, b)$ is 1 if a and b are identical, and otherwise 0. y_{i-1} is the tag predicted for the previous word.

We carry out our experiments in automatic maze detection using a statistical maze detector we first presented at the CLPsych workshop[2] (Morley et al., 2014). This detector learns to identify mazes from manually labeled data using the features extracted from words and automatically predicted part of speech tags. For classification, we use the Max Margin Markov Network 'M^{3}N' classifier in the pocketcrf toolkit (available at `http://code.google.com/p/pocketcrf/`), which is the same classifier that Qian and Liu (2013) use in the second and third steps of their disfluency detector (discussed in Section 4.5). The M^{3}N classifier, described in Section 4.3, is particularly appropriate for maze and disfluency detection: it is a kernel-based classifier that is also able to leverage the sequential

[2]The results in this thesis differ from those in the CLPsych paper due to differences in normalization. For example, utterances with unintelligible words were included in the CLPsych experiments, but we exclude them here.

Index	Unigram		Bigram		Trigram		Logic unigram		Logic bigram		Label
	Word	POS	Word	POS	Word	POS	Word	POS	Word	POS	
0	I	PRP	$ I	$ PRP	$ $ I	$ $ PRP	0	0	0	0	S-O
1	saw	VBD	I saw	PRP VBD	$ I saw	$ PRP VBD	0	0	0	0	B-M
2	the	DT	saw the	VBD DT	I saw the	PRP VBD DT	0	0	0	0	E-M
3	saw	VBD	the saw	DT VBD	saw the saw	VBD DT VBD	1	1	0	0	S-O
4	the	DT	saw the	VBD DT	the saw the	DT VBD DT	1	1	1	1	S-O
5	dog	NN	the dog	DT NN	saw the dog	VBD DT NN	0	0	0	0	S-O

Figure 5.3: Features extracted for each word in the sentence 'I/PRP saw/VBD the/DT saw/VBD the/DT dog/NN' where w/X indicates word w has the POS tag X. $ is a symbol to pad the beginning and end of sentences. For space and clarity, we only include bigrams and trigrams to the left of the current position, and logic unigrams and bigrams with $j = -2$.

nature of the data in this problem (Taskar et al., 2003).

The feature set used by our maze detector is summarized in Table 5.1, and we note that it is identical to the feature set used by the 'filler word' detector in the disfluency detector proposed by Qian and Liu (2013). Figure 5.3 contains a concrete example of these features. We now present these features in turn.

Our maze detector extracts features from both words and of part-of-speech (POS) tags that are up to three items long. Such sequences are referred to as unigrams, bigrams, and trigrams, respectively, and sequences of unspecified length are called *n-grams*. One type of feature that we use is the word (or POS tag) n-gram itself. For example, features such as the unigram 'um', or the bigram 'the the', or its corresponding POS bigram 'DET DET', are likely to be strong indicators of mazes. POS n-grams may intuitively seem to be weaker indicators of mazes than word n-grams, but word n-grams are naturally sparser: there are far more sequences of three words that will never occur in the data as compared to sequences of three POS tags that we will not observe.

The other type of feature we extract is called a *logic n-gram*. Logic n-grams are a binary function of two n-grams. If the two n-grams are the same, it returns 1, and if they are different, it returns 0. We extract logic n-gram features by comparing word (or POS) n-grams starting at the current word with n-grams starting at nearby words. Logic n-gram features capture the intuition that repeated sequences of words or n-grams are strong cues of mazes.

We use the following label set: S-O (not in maze); S-M (single word maze); B-M (beginning of multi-word maze); I-M (in multi-word maze); and E-M (end of multi-word maze). The M^3N classifier allows us to set a unique penalty for each pair of confused labels, for example penalizing an erroneous prediction of S-O (failing to identify maze words) more heavily than spurious predictions of maze words (all -M labels). This ability is particularly useful for maze detection

because maze words are so infrequent compared to words that are not in mazes.

We note that in some bracketing tasks, for example phrase chunking and named entity recognition, converting between bracketing and tagging notation may not be trivial, for example in the case where there are adjacent chunks or entities. This is not the case for maze detection, however, because adjacent mazes are not permitted by the SALT guidelines. Therefore when interpreting the predictions of the M^3N classifier, we simply count any word with a tag ending in -M as being in a maze, and any word tagged S-O as not being in a maze. For tagging evaluation this is sufficient, and for bracketing evaluation, we convert these binary annotations to brackets, thus removing the possibility of adjacent mazes or illegal sequences (ex. S-O E-M).

5.4 Corpus-specific and generic models

In this section we train and test our maze detector on different sets of data in pursuit of two goals. First, we want to identify the maze detector with the highest performance in terms of intrinsic evaluation. In Section 5.5 we evaluate this detector further, by evaluating it extrinsically. Our second goal in this section is to get a preliminary idea of the inconsistencies in the data: does a maze detector trained on all of the SALT corpora outperform ones with smaller training sets because it is able to leverage more data, or are these corpora too heterogeneous for this to be the case? Our investigation into the consistency of maze annotations in the various SALT corpora (Section 5.6) will be guided by what we find here.

5.4.1 Baseline performance

As a baseline, we train corpus-specific maze-detection models on the training set of each corpus. We report maze detection performance on the development set

Corpus	Ages		MLU	Size Rank	Tag P	Tag R	Tag F1	Bracket P	Bracket R	Bracket F1
Conv	2;9	13;3	6.6	1	0.775	0.798	0.786	0.683	0.679	0.681
ENNI	4;0	9;11	8.9	2	0.898	0.832	0.864	0.791	0.791	0.791
GillamNT	5;0	11;11	9.8	3	0.892	0.881	0.886	0.795	0.803	0.799
NZConv	4;5	7;7	6.4	4	0.831	0.833	0.832	0.719	0.757	0.737
NZPerNar	4;5	7;7	7.0	5	0.882	0.840	0.861	0.778	0.759	0.768
NarSSS	5;2	13;3	9.1	6	0.796	0.795	0.795	0.648	0.651	0.649
NarSR	4;4	12;8	9.7	7	0.795	0.797	0.796	0.631	0.662	0.646
Expository	10;7	15;9	13.9	8	0.670	0.672	0.671	0.597	0.647	0.621
NZSR	4;0	7;7	8.2	9	0.836	0.875	0.855	0.661	0.703	0.681

Table 5.2: Baseline maze detection performance on development sections of SALT corpora: corpus-specific models; MLU=mean length of utterance including mazes

of each SALT corpus in Table 5.2, which also includes the rank of the size of each corpus (1 = biggest, 9 = smallest). For more information on how the SALT corpora are divided into training, development, and test sets see Section 3.3.3.

Table 5.2 shows that our maze detector performs far better on some corpora than on others, both in terms of tagging and bracketing performance. We note that maze detection performance is not solely determined by corpus size: tagging performance is substantially worse on the largest corpus (Conv) than the smallest (NZSR). Furthermore, utterance length does not appear to determine performance on maze detection: performance is relatively low on both the Expository and Conv corpora, which have the longest and second shortest mean length of utterance (MLU), respectivel. Meanwhile, performance is relatively high on the GillamNT and NZConv corpora, which have the second longest, and shortest MLUs, respectively. Finally, we see that maze detection performance appears to be independent of the age of the children whose speech is transcribed, as most corpora contain samples from children with a wide variety of ages, and maze detection performance can differ greatly between corpora collected from children with similar age ranges (ex. ENNI and NarSR).

We note that in general, the tagging performance on maze detection we observe here is similar to state-of-the-art disfluency detection performance on Switchboard (Rasooli and Tetreault (2014) report F1=0.841). This is not to say that maze detection is a solved task, but rather that techniques for disfluency detection are likely applicable to maze detection, and therefore advances in disfluency detection may be adapted to maze detection without much difficulty. This is not surprising given how similar the two tasks are. Nevertheless, some techniques for disfluency detection may require resources that are not available for maze detection, for example data that has both manual dependency parse

and maze annotations (Rasooli and Tetreault, 2014).

Cross-corpus performance

Corpus	Tag			Bracket		
	P	R	F1	P	R	F1
Conv	0.290	**0.815**	0.427	0.238	0.486	0.319
ENNI	0.653	0.784	0.712	0.608	0.665	0.635
Expository	0.363	0.762	0.492	0.370	0.496	0.424
GillamNT	0.892	0.881	0.886	0.795	0.803	0.799
NarSSS	0.601	**0.798**	0.686	0.503	0.595	0.545
NarSR	0.578	0.783	0.665	0.522	0.589	0.554
NZConv	0.393	**0.874**	0.542	0.343	0.610	0.439
NZPerNar	0.428	**0.891**	0.579	0.362	0.613	0.455
NZSR	0.730	0.814	0.770	0.570	0.658	0.611

Table 5.3: Model trained on GillamNT, tested on SALT development sets; see Appendix A.1 for complete results of cross-corpus experiments. Results in bold are better than baseline, all others are equal or worse.

We evaluate each of the corpus-specific models on the development set of each of the other SALT corpora. For the sake of brevity, we do not include all of the results here, although they are given in Appendix A.1.

In general, we find that the performance of corpus-specific models tend to degrade when they are tested on the development set of corpora other than the one on which they are trained. The GillamNT exhibits more severely degraded performance than any of the others, as can be seen in Table 5.3: tagging F1 on five of the nine SALT corpora is lower than the worst-performing baseline model (Expository, F1=0.671). We see in Table 5.3 that the GillamNT model tends to perform with low precision and high recall on many SALT corpora. This is likely because the GillamNT corpus has a higher proportion of words in mazes than the other corpora, as discussed in Section 5.6.2, and because parenthetical comments are annotated as mazes in this corpus, as discussed in Section 5.6.3.

We find that for most corpora, the highest performing model is the baseline model (i.e. the one trained on the training fold of the same corpus). This is

unequivocally the case for four of the eight corpora: Conv, ENNI, GillamNT and NarSR. The results are less clear for the NZSR corpus, where both the Conv and ENNI models perform as well as the baseline. The Conv model also performs as well as the baseline on the NZPerNar corpus. Somewhat surprisingly, the Conv model performs significantly better than the baseline models on both both the NarSSS and NZConv corpora. Specifically, on NarSSS tagging F1 improves from 0.795 to 0.803 ($p = 0.065$), and bracketing F1 improves from 0.649 to 0.677 ($p = 0.022$). On NZConv tagging F1 improves from 0.832 to 0.852 ($p \leq 0.001$), and bracketing F1 improves from 0.737 to 0.783 ($p \leq 0.001$). Overall then, it appears that the baseline models yield the highest performance on six of the eight SALT corpora (although in two of these cases there were other models as good as the baseline), and in two cases there were models that outperformed the baselines. We now explore whether we can outperform the baseline by training models on data from more than one corpus.

5.4.2 More general models

We now evaluate the performance of more general models for maze detection, namely ones that are trained on multiple corpora. We are interested in this for two reasons: first, a model trained on more data should outperform a model trained on less, provided the corpora, including both the language and annotations, are sufficiently similar; and second, models that can be effectively used with multiple corpora are more useful than corpus-specific models, particularly for labeling new data. Our procedure for training and testing such a model is as follows: first, we select a set of corpora S from the SALT corpora. We then combine all of the training sets of the corpora in S, and train a model. We note that the training and development proportions of each corpus in S are proportional: if 20% of the training utterances are from corpus c, then 20% of the development

utterances will be as well. Next, we combine the development sets of S, and use this combined development set to tune the loss matrix to balance precision and recall. Finally, we test the model on each of the development sets in S individually.

Throughout this section, we use $\alpha = 0.1$ to determine whether one system performs significantly higher than another, and we compute the p value using the randomized paired-sample procedure described in Section 4.7.1. When any of the more general models outperforms the corpus-specific baseline in terms of either tagging or bracketing performance, we will report: its F1 score; the difference in F1 score, relative to the baseline (Δ); and the significance level ($p \leq$). We use ✗ to indicate when a more general model yields a lower F1 score than the baseline model.

All Model

We construct the All model by letting S be the set of all SALT corpora. Table 5.4 shows the performance of the All model on the development folds of the SALT corpora, and indicates where it improves performance significantly over the corpus-specific baseline. We find that the All model does not yield significant improvement in terms of either tagging and bracketing performance on six of the nine corpora. There is improvement on the following three corpora in terms of both tagging and bracketing: Conv (tagging F1=0.791, $\Delta = 0.005$, $p \leq 0.010$; bracketing F1=0.696, $\Delta = 0.015$, $p \leq 0.003$); NarSSS (tagging F1=0.807 Δ=0.012, $p \leq 0.013$; bracketing F1=0.673, Δ=0.023, $p \leq 0.039$); and NZConv (tagging F1=0.848, Δ=0.0.016, $p \leq 0.001$; bracketing F1=0.774, Δ=0.037, $p \leq 0.002$).

Corpus	Sig. improved		Not improved		F1 (baseline → ALL)	
	Tag	Bracket	Tag	Bracket	Bracket	Tag
	$p \leq$	$p \leq$	$p \leq$	$p \leq$		
CONV	0.010	0.003			0.786 → 0.791	0.681→ 0.696
ENNI			✗	✗	0.864 → 0.833	0.791 → 0.760
EXPOSI-TORY			✗	✗	0.670 → 0.655	0.621 → 0.593
GILLAMNT			✗	✗	0.886 → 0.774	0797→ 0.686
NARSSS	0.013	0.039			0.795 → 0.807	0.649 → 0.673
NARSR			0.167	0.185	0.796 → 0.801	0.646 → 0.658
NZCONV	0.001	0.002			0.832 → 0.848	0.737→ 0.774
NZPERNAR			✗	✗	0.861 → 0.849	0.768 → 0.735
NZSR			0.480	✗	0.855 → 0.853	0.681 → 0.682

Table 5.4: Performance of the ALL maze detection model including comparison to the baseline on development sets of SALT corpora

Corpus	Sig. improved		Not improved		F1 (baseline → ALL)	
	Tag	Bracket	Tag	Bracket	Bracket	Tag
	$p \leq$	$p \leq$	$p \leq$	$p \leq$		
ENNI			✗	✗	0.864 → 0.829	0.791 → 0.745
GILLAMNT			✗	✗	0.886 → 0.787	0797→ 0.667
NARSSS			✗	✗	0.795 → 0.762	0.649 → 0.624
NARSR			✗	✗	0.796 → 0.781	0.646 → 0.638

Table 5.5: Performance of the AGE maze detection model including comparison to the baseline on development sets of SALT corpora

Age-specific model

We train a single AGE-specific model on the corpora listed in Table 5.5. These corpora contain transcripts collected from children roughly aged 4-12 (see Table 3.1). As can be seen in in Table 5.5, the age-based model performs worse than the baseline on all four of these corpora.

Task-Specific Models

Corpus	Sig. improved		Not improved		F1 (baseline → ALL)	
	Tag	Bracket	Tag	Bracket	Bracket	Tag
	$p \leq$	$p \leq$	$p \leq$	$p \leq$		
CONV	0.009	0.004			0.786 → 0.791	0.681→ 0.696
NZ-CONV	0.001	0.002			0.832 → 0.848	0.737→ 0.774

(a) CONV Model

Corpus	Sig. improved		Not improved		F1 (baseline → ALL)	
	Tag	Bracket	Tag	Bracket	Bracket	Tag
	$p \leq$	$p \leq$	$p \leq$	$p \leq$		
ENNI			✗	✗	0.864 → 0.820	0.791 → 0.736
GILLAMNT			✗	✗	0.886 → 0.756	0797→ 0.659
NARSSS			✗	0.226	0.795 → 0.787	0.649 → 0.658
NARSR			✗	✗	0.796 → 0.789	0.646 → 0.645
NZPERNAR			✗	✗	0.861 → 0.833	0.768 → 0.723
NZSR			✗	0.306	0.855 → 0.862	0.681 → 0.679

(b) NAR Model

Table 5.6: Performance of task-specific maze detection models including comparison to the baseline on development sets of SALT corpora

We construct two task-specific models for maze detection: one for conversations, and the other for narrative tasks. As shown in Table 5.6a, the CONV model

trained on the CONV and NZCONV significantly improves performance on the both corpora relative to the baselines, both in terms of tagging and bracketing performance. The improvements on the CONV corpus are statistically significant, but not particularly large for either tagging (F1=0.791, Δ=0.005, $p \leq 0.009$) or bracketing (F1=0.681, Δ=0.015, $p \leq 0.004$) performance. On the NZCONV corpus, tagging performance improves a small amount (F1=0.848, Δ=0.016, $p \leq 0.001$), while bracketing performance improves substantially (F1=0.774, Δ=0.037, $p \leq 0.002$). A model for narrative tasks (NAR) trained on the corpora shown in Table 5.6b does not improve performance on any of the constituent corpora, relative to the baseline.

Research Group-Specific Models

Corpus	Sig. improved		Not improved		F1 (baseline → ALL)	
	Tag	Bracket	Tag	Bracket	Bracket	Tag
	$p \leq$	$p \leq$	$p \leq$	$p \leq$		
NZ-CONV			✗	✗	0.832 → 0.820	0.737→ 0.733
NZPERNAR			✗	✗	0.861 → 0.845	0.768 → 0.736
NZSR			✗	✗	0.855 → 0.846	0.681 → 0.670

(a) NZ Model

	Tag			Bracket		
Corpus	F1	Δ	$p \leq$	F1	Δ	$p \leq$
CONV	0.790	0.004	0.088	0.700	0.019	0.001
EXPOSITORY	0.690	0.019	0.059	0.680	0.059	0.009
NARSSS	0.829	0.033	0.001	0.723	0.074	0.001
NARSR	0.820	0.024	<0.001	0.686	0.040	0.002

(b) Comparison of WI model performance to baseline

Table 5.7: Performance of the research group-specific maze detection models relative to the baseline on development sets of SALT corpora

There are two groups of researchers that have annotated multiple corpora:

Domain	Sentence	Universal features	Domain 1 features	Domain 2 features
1	She saw him	[She, saw, him]		[She, saw, him]
2	The sheep ate	[The, sheep, ate]	[The, sheep, ate]	

Figure 5.4: Illustration of features extracted with FEDA. Note that no 'Domain 1' features are extracted from utterances in 'Domain 2', and *vice versa*.

a group in New Zealand (NZ), which annotated the NZConv, NZPerNar, and NZSR corpora; and another group in Wisconsin (WI), which annotated the Conversation, Expository, NarSSS, and NarSR corpora. We train a research group-specific model for each of these two groups.

The WI model outperforms the baseline models in all cases, in some cases quite substantially (on Expository and to a lesser extent on NarSSS). This is shown in Table 5.7b. On the other hand, the NZ model does not improve performance on any of the corpora annotated by that group relative to the corpus specific baseline models, as shown in Table 5.7a.

Domain adaptation

We now apply the so-called Frustratingly Easy Domain Adaptation (FEDA) algorithm proposed by Daumé III (2007). This simple algorithm is essentially a preprocessing step that can be applied to any domain adaptation problem: the original feature set, which is common to all domains, is augmented to include a copy of features that are specific to each domain. As a result, there are no $\mathcal{D}_{src}$-specific features for data from domain $\mathcal{D}_{trg}$. For a concrete example, see Figure 5.4, which illustrates FEDA, using only word unigrams as features: there are no 'Domain 1' features extracted from the utterance from Domain 2; only 'Domain 2' and 'Universal' features are extracted from this utterance. We note that FEDA is not specific to maze detection, and after modifying the training

	Corpus	Sig. improved		Not improved		F1 (baseline → All)	
		Tag	Bracket	Tag	Bracket	Bracket	Tag
		$p \leq$	$p \leq$	$p \leq$	$p \leq$		
All	Conv	0.001			✗	0.786 → 0.806	0.681→ 0.729
	ENNI	0.001			✗	0.864 → 0.931	0.791 → 0.887
	Exposi-tory			✗	✗	0.670 → 0.383	0.621 → 0.276
	GillamNT	0.001			✗	0.886 → 0.928	0797→ 0.867
	NarSSS	0.001			✗	0.795 → 0.821	0.649 → 0.734
	NarSR	0.001			✗	0.796 → 0.823	0.646 → 0.708
	NZConv	0.001			✗	0.832 → 0.861	0.737→ 0.786
	NZPerNar	0.086			✗	0.861 → 0.867	0.768 → 0.770
	NZSR	0.028			✗	0.855 → 0.877	0.681 → 0.709
Age	ENNI	0.001			✗	0.864 → 0.927	0.791 → 0.875
	GillamNT	0.001			✗	0.886 → 0.932	0797→ 0.872
	NarSSS	0.001			✗	0.795 → 0.818	0.649 → 0.730
	NarSR	0.001			✗	0.796 → 0.820	0.646 → 0.705
Conv	Conv			✗	✗	0.786 → 0.785	0.681→ 0.686
	NZConv	0.001			✗	0.832 → 0.854	0.737→ 0.762
Nar	ENNI	0.001			✗	0.864 → 0.931	0.791 → 0.883
	GillamNT	0.001			✗	0.886 → 0.932	0797→ 0.875
	NarSSS	0.001			✗	0.795 → 0.818	0.649 → 0.726
	NarSR	0.001			✗	0.796 → 0.816	0.646 → 0.686
	NZPerNar			✗	✗	0.861 → 0.852	0.768 → 0.752
	NZSR	0.028			✗	0.855 → 0.878	0.681 → 0.705
NZ	NZConv	0.001			✗	0.832 → 0.858	0.737→ 0.777
	NZPerNar			0.178	✗	0.861 → 0.864	0.768 → 0.765
	NZSR	0.077			✗	0.855 → 0.870	0.681 → 0.705
WI	Conv	0.001			✗	0.786 → 0.800	0.681→ 0.729
	Exposi-tory			✗	✗	0.670 → 0.366	0.621 → 0.261
	NarSSS	0.001			✗	0.795 → 0.816	0.649 → 0.740
	NarSR	0.015			✗	0.796 → 0.808	0.646 → 0.694

Table 5.8: Performance of FEDA-trained models including performance to baseline corpus-specific models

and test features, training and testing proceeds as usual.

We re-create each of the more generic models explored above in Section 5.4.2 using FEDA. As Table 5.8 shows, generic models trained with FEDA tend to outperform the baseline models in terms of tagging performance, but not bracketing performance. The lack of improvement in bracketing performance is unsurprising: the M^3N classifier optimizes for tagging, not bracketing performance. Furthermore, each utterance yields fewer bracketing predictions compared to tagging predictions, thus a mistake in bracketing is more costly than a mistake in tagging.

5.4.3 Discussion

We have seen that simply combining different training sets is typically not an effective strategy for building models for maze detection, although there are notable exceptions (the WI and CONV models). In the majority of cases, multi-corpus models perform worse than corpus-specific models, even though they are trained on more data. By adopting a simple technique for domain adaptation, namely FEDA, we are able to construct models from multiple corpora that reliably improve maze detection performance (in terms of tagging) on nearly all of their constituent corpora, relative to the corpus-specific models.

These results suggest that there are inconsistencies between the different SALT corpora, but they do not tell us what these inconsistencies are. Some possible inconsistencies between these corpora could involve differences in language, utterance segmentation, or in maze annotation conventions varying across corpora. We examine the differences between these corpora in Section 5.6.

5.5 Extrinsic evaluation

Having compared various models for maze detection in terms of how well they replicate manual annotations, we now consider the utility of the annotations they produce. Specifically, maze annotations are critical for computing summary statistics from transcripts, most notably: utterance length, the number of types in an utterance, the number of mazes in an utterance, and maze lengths. These statistics are all computed by the SALT software from manually annotated transcripts, but ideally one could compute nearly identical statistics from unannotated transcripts.

We use two setups to compare summary statistics computed from automatically annotated transcripts to those computed from manually annotated ones. The first setup is *leave-transcript-out* (LTO) evaluation. In this scenario, we train a model on all of the transcripts save one, use the model to annotate the held-out transcript, then compute the summary statistics from the automatically annotated transcript. We repeat this procedure for every transcript. For simplicity, we use the penalty matrix tuned on each corpus's development set, which yields approximately balanced precision and recall.

This 'leave-one-out' procedure is ideal for extrinsic evaluation, but it is extremely expensive as a new model must be trained for each transcript. We propose the following procedure to generate *simulated transcripts*, which is much faster than LTO evaluation:

1. Create a simulated transcript by sampling utterances randomly from the development (or test) fold along with true and predicted maze annotations
2. Record the statistic (ex. MLU) computed from the manually annotated simulated transcript
3. Record the same statistic from the automatically annotated simulated

transcript

In practice, we create 1,000 simulated transcripts from which we compute the relevant statistics. These transcripts are each 200 utterances long, following the observation made by Gavin and Giles (1996) that MLU test/retest reliability was not sufficiently high for transcripts under 175 utterances long. We compare the difference in means in the two statistics over all of the trials, and we also use the Wilcoxon signed-rank test to see whether the differences are statistically significant. The statistics we compute are the means of: tokens per utterance (mean length of utterance or MLU), types per utterance, mazes per utterance, and maze length. We report the mean difference between the reference statistics derived from manually and automatically annotated transcripts, along with the p value from the Wilcoxon signed-rank test, and the effect size r. As a rule of thumb, an effect size r of 0.1 is interpreted as small, 0.3 as medium, and 0.5 as large. We also report the correlation between summary statistics derived from manually and automatically annotated transcripts both in terms of Pearson's ρ and Kendall's τ.

5.5.1 Corpus-specific models

Leave transcript out

We compare the four summary statistics (tokens per utterance (ie MLU), types per utterance, mazes per utterance, and maze length) derived from manual annotations to those derived from the baseline models' predictions. As shown in Table 5.9, nearly all of the summary statistics derived from automatic annotations are significantly different from those derived from manual annotations, yet the two values are typically very highly correlated. Appendix A.2 contains plots illustrating the relationship between the summary statistics derived from manually and automatically annotated transcripts, along with both linear and

Corpus	μ	σ	$\lvert\Delta\mu\rvert$	$p \leq$	r	ρ	τ
Conv	6.20	1.30	0.27	**0.001**	**0.842**	0.985	0.906
ENNI	8.09	1.37	0.10	**0.001**	0.254	0.995	0.942
Expository	13.04	1.93	0.31	**0.001**	**0.704**	0.989	0.904
GillamNT	8.57	1.96	0.16	**0.001**	0.266	0.990	0.924
NarSSS	8.05	1.47	0.29	**0.001**	**0.710**	0.979	0.878
NarSR	8.74	1.42	0.15	**0.001**	**0.367**	0.989	0.915
NZConv	5.81	0.95	0.06	0.894	0.008	0.996	0.951
NZPerNar	6.24	0.96	0.08	**0.001**	**0.663**	0.994	0.944
NZSR	7.41	1.06	0.15	**0.001**	**0.349**	0.977	0.890

(a) Token count

Corpus	μ	σ	$\lvert\Delta\mu\rvert$	$p \leq$	r	ρ	τ
Conv	5.99	1.18	0.21	**0.001**	**0.840**	0.988	0.915
ENNI	7.72	1.22	0.05	**0.001**	0.165	0.998	0.962
Expository	11.79	1.54	0.20	**0.001**	**0.751**	0.995	0.918
GillamNT	8.14	1.76	0.11	**0.001**	0.195	0.993	0.941
NarSSS	7.75	1.31	0.20	**0.001**	**0.658**	0.986	0.897
NarSR	8.34	1.27	0.10	**0.001**	0.293	0.994	0.939
NZConv	5.67	0.89	0.04	**0.001**	**0.335**	0.998	0.966
NZPerNar	6.09	0.90	0.06	**0.001**	**0.761**	0.998	0.957
NZSR	7.20	0.96	0.10	**0.001**	**0.391**	0.989	0.917

(b) Type count

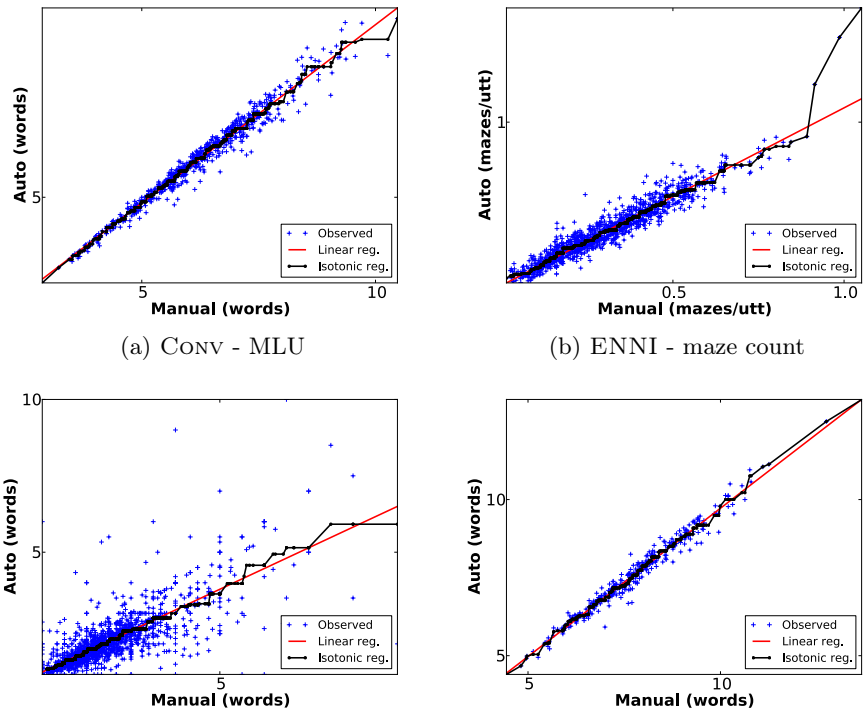
Corpus	μ	σ	$\lvert\Delta\mu\rvert$	$p \leq$	r	ρ	τ
Conv	0.28	0.14	0.03	**0.001**	**0.346**	0.940	0.816
ENNI	0.30	0.16	0.04	**0.001**	**0.574**	0.952	0.823
Expository	0.49	0.21	0.08	0.389	0.092	0.930	0.725
GillamNT	0.45	0.24	0.06	**0.001**	0.129	0.925	0.799
NarSSS	0.41	0.19	0.07	**0.001**	**0.536**	0.909	0.745
NarSR	0.37	0.19	0.06	**0.001**	**0.497**	0.944	0.790
NZConv	0.26	0.13	0.03	**0.001**	**0.398**	0.970	0.844
NZPerNar	0.29	0.14	0.03	0.661	0.029	0.956	0.853
NZSR	0.38	0.23	0.07	0.719	0.027	0.899	0.752

(c) Maze count

Corpus	μ	σ	$\lvert\Delta\mu\rvert$	$p \leq$	r	ρ	τ
Conv	1.78	0.41	0.43	**0.001**	**0.566**	0.523	0.396
ENNI	1.98	0.58	0.36	**0.001**	**0.682**	0.707	0.591
Expository	1.96	0.57	0.64	**0.001**	**0.845**	0.604	0.464
GillamNT	2.29	1.03	0.44	**0.001**	**0.639**	0.796	0.700
NarSSS	1.96	0.55	0.36	0.823	0.012	0.511	0.423
NarSR	2.02	0.58	0.38	**0.001**	**0.612**	0.628	0.533
NZConv	1.75	0.45	0.31	**0.001**	**0.767**	0.771	0.602
NZPerNar	1.91	0.50	0.42	**0.001**	**0.841**	0.747	0.618
NZSR	1.99	0.92	0.44	**0.001**	**0.603**	0.628	0.602

(d) Maze length

Table 5.9: Reference statistic prediction accuracy from LTO procedure with corpus-specific maze detection models; all values of ρ and τ significant at $p \leq 0.001$ level. All values of μ and σ refer to counts from manual counts. Correlations and differences in mean are computed over transcripts.



(a) Conv - MLU

(b) ENNI - maze count

(c) GillamNT - maze length

(d) NarSSS - type count

Figure 5.5: Comparison of observed summary statistics with predictions from LTO procedure on various SALT corpora

isotonic regression lines. These plots convey similar information to Table 5.10, in particular the tight correlation between the summary statistics.

Even where the summary statistics derived from automatic annotations are significantly different, they are generally not substantially different (the maze length statistics on certain corpora may be exceptions). This is illustrated in the plots in Figure 5.5 as well. In the case of MLU, a difference of 0.103 words corresponds to roughly one month of development (between the ages of 2 and 5) (Scarborough et al., 1986). Put into temporal terms, the mean error in MLU produced by the baseline systems is roughly between two and three weeks.

Simulated Transcripts

Corpus	μ	σ	$\lvert\Delta\mu\rvert$	$p \leq$	r	ρ	τ
Conv	6.08	0.28	0.04	**0.001**	0.211	0.982	0.884
ENNI	8.13	0.24	0.06	**0.001**	**0.743**	0.974	0.863
Expository	12.03	0.39	0.07	**0.018**	0.075	0.974	0.860
GillamNT	8.79	0.31	0.05	**0.001**	0.250	0.983	0.886
NarSSS	8.08	0.28	0.06	0.103	0.052	0.970	0.848
NarSR	8.77	0.29	0.05	**0.095**	0.053	0.976	0.868
NZConv	5.82	0.22	0.04	**0.014**	0.078	0.979	0.875
NZPerNar	6.38	0.23	0.04	**0.001**	**0.590**	0.981	0.882
NZSR	7.49	0.11	0.05	**0.001**	**0.837**	0.969	0.853

(a) Token count

Corpus	μ	σ	$\lvert\Delta\mu\rvert$	$p \leq$	r	ρ	τ
Conv	5.89	0.26	0.03	0.233	0.038	0.991	0.918
ENNI	7.76	0.21	0.04	**0.001**	**0.777**	0.990	0.916
Expository	10.94	0.30	0.06	**0.001**	**0.544**	0.978	0.871
GillamNT	8.33	0.27	0.04	0.514	0.021	0.987	0.902
NarSSS	7.78	0.25	0.04	**0.001**	**0.591**	0.985	0.895
NarSR	8.38	0.25	0.04	**0.001**	**0.438**	0.986	0.900
NZConv	5.69	0.21	0.03	**0.001**	**0.481**	0.990	0.915
NZPerNar	6.22	0.22	0.04	**0.001**	**0.800**	0.991	0.919
NZSR	7.30	0.10	0.05	**0.001**	**0.864**	0.980	0.887

(b) Type count

Corpus	μ	σ	$\lvert\Delta\mu\rvert$	$p \leq$	r	ρ	τ
Conv	0.26	0.04	0.02	**0.001**	**0.374**	0.841	0.653
ENNI	0.31	0.03	0.02	**0.001**	**0.377**	0.863	0.689
Expository	0.49	0.03	0.09	**0.001**	**0.866**	0.736	0.537
GillamNT	0.42	0.04	0.02	**0.001**	**0.698**	0.866	0.688
NarSSS	0.44	0.04	0.03	**0.001**	**0.546**	0.794	0.598
NarSR	0.37	0.04	0.04	**0.001**	**0.818**	0.791	0.609
NZConv	0.25	0.03	0.03	**0.001**	**0.797**	0.844	0.670
NZPerNar	0.30	0.03	0.01	**0.002**	0.097	0.861	0.689
NZSR	0.35	0.02	0.03	**0.001**	**0.861**	0.741	0.572

(c) Maze count

Corpus	μ	σ	$\lvert\Delta\mu\rvert$	$p \leq$	r	ρ	τ
CONV	1.80	0.19	0.14	**0.001**	0.267	0.595	0.422
ENNI	2.08	0.20	0.18	**0.001**	**0.796**	0.686	0.490
EXPOSITORY	2.05	0.16	0.37	**0.001**	**0.865**	0.169	0.100
GILLAMNT	2.27	0.21	0.10	**0.001**	**0.464**	0.855	0.661
NARSSS	2.02	0.16	0.11	**0.001**	0.237	0.589	0.409
NARSR	2.11	0.17	0.14	**0.001**	**0.680**	0.649	0.439
NZCONV	1.80	0.22	0.15	**0.001**	**0.534**	0.688	0.492
NZPERNAR	1.98	0.21	0.12	**0.001**	**0.429**	0.800	0.581
NZSR	2.37	0.12	0.13	**0.001**	**0.819**	0.696	0.490

(d) Maze length

Table 5.10: Reference statistic prediction accuracy using baseline maze detection models; all values of ρ and τ significant at $p \leq 0.001$ level. All values of μ and σ refer to counts from manual annotations on simulated transcripts. Correlations and differences in mean are computed over simulated transcripts.

We now use the simulated transcripts procedure (described at the beginning of this Section) to compare the four summary statistics (tokens per utterance (ie MLU), types per utterance, mazes per utterance, and maze length) derived from manual annotations to those derived from the baseline models' predictions. We remind the reader that the simulated transcript procedure is far less computationally expensive than LTO extrinsic evaluation, which it is trying to approximate. We see in Table 5.10 that the results of the simulated transcript procedure are broadly in line with those from LTO extrinsic evaluation: all of the summary statistics derived from the automatically annotated simulated transcripts are highly correlated with the true values, and while they are typically significantly different from one another, they are not substantially different. This suggests that the simulated transcript procedure is an effective method to perform extrinsic evaluation of maze detection models. This finding is particularly relevant to our experiments below evaluating the generic models extrinsically: performing LTO evaluation under these circumstances is prohibitively expensive since some of these models may be trained on several thousand transcripts, and training a single model may take half an hour.

Appendix A.2 contains plots illustrating the relationship between the summary statistics derived from manually and automatically annotated transcripts, along with both linear and isotonic regression lines. These plots convey similar information to Table 5.10, in particular the tight correlation between the summary statistics.

5.5.2 Generic models

We now compare the four summary statistics (tokens per utterance (ie MLU), types per utterance, mazes per utterance, and maze length) derived from manual annotations to those derived from predictions made by models trained on multiple corpora, i.e. *generic models*. We perform these experiments primarily using the *simulated transcript* procedure described on page 138 because performing these experiments in a leave-transcript out manner is so computationally expensive—there can be thousands of transcripts incorporated into a single model. Nevertheless, we do present partial results from leave-transcript-out experiments below in Section 5.5.2.

As shown in Table 5.9, nearly all of the summary statistics derived from automatic annotations are even closer to the true values than the ones extracted from automatic annotations produced with the baseline corpus-specific models. Although the summary statistics from the annotations produced by the generic model are significantly different from ones derived from manual annotations, the observed and predicted values are typically very highly correlated. Appendix A.2 contains plots illustrating the relationship between the summary statistics derived from manually and automatically annotated transcripts, along with both linear and isotonic regression lines. These plots show similar information to Table 5.11, in particular the tight correlation between the summary statistics.

Simulated Transcripts

Corpus	μ	σ	$\lvert\Delta\mu\rvert$	$p \leq$	r	ρ	τ
Conv	6.10	0.29	0.04	**0.001**	0.291	0.984	0.891
ENNI	8.13	0.23	0.06	**0.001**	**0.751**	0.975	0.865
Expository	12.04	0.40	0.07	0.586	0.017	0.974	0.855
GillamNT	8.78	0.30	0.05	**0.001**	0.198	0.981	0.880
NarSSS	8.09	0.28	0.06	0.421	0.025	0.968	0.842
NarSR	8.79	0.29	0.05	0.674	0.013	0.975	0.863
NZConv	5.81	0.23	0.04	**0.006**	0.087	0.980	0.882
NZPerNar	6.39	0.23	0.04	**0.001**	**0.573**	0.981	0.882
NZSR	7.49	0.11	0.05	**0.001**	**0.825**	0.964	0.838

(a) Token count

Corpus	μ	σ	$\lvert\Delta\mu\rvert$	$p \leq$	r	ρ	τ
Conv	0.26	0.04	0.02	**0.001**	**0.368**	0.826	0.638
ENNI	0.31	0.03	0.02	**0.001**	**0.358**	0.858	0.679
Expository	0.48	0.03	0.09	**0.001**	**0.866**	0.740	0.549
GillamNT	0.41	0.04	0.03	**0.001**	**0.713**	0.857	0.678
NarSSS	0.43	0.04	0.03	**0.001**	**0.584**	0.813	0.626
NarSR	0.37	0.04	0.04	**0.001**	**0.829**	0.778	0.586
NZConv	0.25	0.03	0.02	**0.001**	**0.784**	0.846	0.664
NZPerNar	0.30	0.03	0.02	**0.001**	0.128	0.854	0.675
NZSR	0.35	0.02	0.03	**0.001**	**0.856**	0.766	0.595

(b) Type count

Corpus	μ	σ	$\lvert\Delta\mu\rvert$	$p \leq$	r	ρ	τ
Conv	0.26	0.04	0.02	**0.001**	**0.368**	0.826	0.638
ENNI	0.31	0.03	0.02	**0.001**	**0.358**	0.858	0.679
Expository	0.48	0.03	0.09	**0.001**	**0.866**	0.740	0.549
GillamNT	0.41	0.04	0.03	**0.001**	**0.713**	0.857	0.678
NarSSS	0.43	0.04	0.03	**0.001**	**0.584**	0.813	0.626
NarSR	0.37	0.04	0.04	**0.001**	**0.829**	0.778	0.586
NZConv	0.25	0.03	0.02	**0.001**	**0.784**	0.846	0.664
NZPerNar	0.30	0.03	0.02	**0.001**	0.128	0.854	0.675
NZSR	0.35	0.02	0.03	**0.001**	**0.856**	0.766	0.595

(c) Maze count

Corpus	μ	σ	$\lvert\Delta\mu\rvert$	$p \leq$	r	ρ	τ
Conv	1.77	0.19	0.14	**0.001**	**0.354**	0.601	0.419
ENNI	2.09	0.21	0.18	**0.001**	**0.784**	0.712	0.503
Expository	2.06	0.17	0.38	**0.001**	**0.865**	0.223	0.136
GillamNT	2.26	0.21	0.10	**0.001**	**0.392**	0.838	0.642
NarSSS	2.03	0.17	0.11	**0.001**	0.210	0.602	0.415
NarSR	2.13	0.16	0.15	**0.001**	**0.732**	0.629	0.431
NZConv	1.80	0.21	0.14	**0.001**	**0.492**	0.656	0.457
NZPerNar	1.98	0.21	0.11	**0.001**	**0.375**	0.793	0.587
NZSR	2.37	0.13	0.13	**0.001**	**0.804**	0.691	0.484

(d) Maze length

Table 5.11: Reference statistic prediction accuracy using All maze detection model; all values of ρ and τ significant at $p \leq 0.001$ level. All values of μ and σ refer to counts from manual annotations on simulated transcripts. Correlations and differences in mean are computed over simulated transcripts.

As we saw in Section 5.4.2, models trained on multiple corpora do not routinely outperform the corpus-specific baselines unless they are trained with FEDA, a simple technique for domain adaptation. In nearly all cases, the FEDA-trained models outperform the corpus-specific baseline models. Here, we evaluate the FEDA-trained All model extrinsically, because it is appropriate for all of the SALT corpora, while Appendix A.2 contains similar results for all of the other FEDA-trained models, including plots illustrating the relationship between the summary statistics derived from manually and automatically annotated transcripts, along with both linear and isotonic regression lines. We note, however, that some manually annotated data is required to apply a FEDA-trained model to a new data set because FEDA involves both corpus-independent and corpus-specific features.

As shown in Table 5.11, the All model allows us to compute token and type statistics that are in most cases significantly, but not substantially, different from the ones computed from manually annotated transcripts. The major exception to this is on the Expository corpus, where we observe severely degraded performance compared to the baseline model. This degradation is likely due to

the language in the EXPOSITORY corpus being different from the language in the other corpora since is the only one with transcripts of an expository task. For all other corpora, however, the correlations between the summary statistics computed from manually and automatically annotated transcripts are extremely high. Therefore, unless there is a severe domain mismatch, as is the case with the EXPOSITORY corpus, using a generic model to apply maze annotations from which one will compute summary statistics appears to be an effective strategy, even though baseline models tend to produce annotations of higher intrinsic quality.

Leave transcript out

The leave-transcript-out procedure for predicting maze annotations is computationally expensive to perform, particularly with the generic models, which are built from thousands of transcripts. Furthermore, the exact results produced by the leave-transcript-out procedure do not appear to be substantially different from those produced by the simulated transcript procedure described above, even though the simulated transcript procedure is far cheaper to execute. For example, in Table 5.12 we see that the summary statistics predicted from transcripts automatically annotated using the simulated transcript procedure are comparable to those predicted from simulated transcripts. Furthermore, there do not appear to be systematic differences between the summary statistics from transcripts automatically annotated using these two procedures.

5.5.3 Discussion

In this section we have seen that the most effective way of training a maze detector on the SALT corpora is typically to train a different model for each corpus. This is somewhat surprising, as one would expect more training data to

Corpus		μ	σ	$\lvert\Delta\mu\rvert$	$p \leq$	r	ρ	τ
Conv	LTO	6.20	1.30	0.22	**0.001**	**0.818**	0.991	0.924
	SIM	6.08	0.29	0.05	**0.001**	0.266	0.982	0.882
Expository	LTO	13.04	1.93	0.50	**0.001**	**0.846**	0.986	0.891
	SIM	12.04	0.41	0.08	0.152	0.045	0.974	0.858
NarSSS	LTO	8.06	1.47	0.16	**0.001**	**0.485**	0.990	0.921
	SIM	8.07	0.29	0.06	0.122	0.049	0.971	0.849
NarSR	LTO	8.74	1.42	0.11	**0.001**	0.226	0.994	0.935
	SIM	8.78	0.29	0.05	**0.011**	0.080	0.974	0.860

(a) Token count

Corpus		μ	σ	$\lvert\Delta\mu\rvert$	$p \leq$	r	ρ	τ
Conv	LTO	5.89	0.26	0.03	0.792	0.008	0.990	0.914
	SIM	5.99	1.18	0.18	**0.001**	**0.829**	0.993	0.932
Expository	LTO	10.95	0.32	0.07	**0.001**	**0.586**	0.978	0.868
	SIM	11.79	1.54	0.27	**0.001**	**0.839**	0.993	0.912
NarSSS	LTO	7.78	0.26	0.04	**0.001**	**0.568**	0.986	0.897
	SIM	7.75	1.31	0.10	**0.001**	**0.498**	0.994	0.940
NarSR	LTO	8.38	0.26	0.04	**0.001**	**0.391**	0.984	0.892
	SIM	8.34	1.27	0.07	**0.002**	0.163	0.997	0.954

(b) Type count

Corpus		μ	σ	$\lvert\Delta\mu\rvert$	$p \leq$	r	ρ	τ
Conv	LTO	0.28	0.14	0.03	**0.001**	0.254	0.949	0.823
	SIM	0.26	0.03	0.02	**0.001**	**0.400**	0.829	0.642
Expository	LTO	0.49	0.21	0.10	**0.001**	**0.744**	0.941	0.761
	SIM	0.49	0.04	0.09	**0.001**	**0.866**	0.770	0.578
NarSSS	LTO	0.41	0.19	0.07	**0.001**	**0.607**	0.921	0.772
	SIM	0.43	0.04	0.02	**0.001**	**0.585**	0.822	0.627
NarSR	LTO	0.37	0.19	0.05	**0.060**	0.101	0.950	0.808
	SIM	0.37	0.04	0.04	**0.001**	**0.821**	0.780	0.596

(c) Maze count

Corpus		μ	σ	$\lvert\Delta\mu\rvert$	$p \leq$	r	ρ	τ
Conv	LTO	1.78	0.41	0.33	**0.001**	**0.407**	0.574	0.424
	SIM	1.79	0.21	0.15	**0.001**	**0.354**	0.596	0.417
Expository	LTO	1.96	0.57	0.76	**0.001**	**0.846**	0.483	0.353
	SIM	2.06	0.16	0.39	**0.001**	**0.866**	0.171	0.113
NarSSS	LTO	1.96	0.55	0.37	**0.001**	**0.692**	0.681	0.554
	SIM	2.02	0.17	0.11	**0.001**	0.235	0.611	0.415
NarSR	LTO	2.02	0.58	0.40	**0.001**	**0.741**	0.720	0.628
	SIM	2.13	0.17	0.15	**0.001**	**0.674**	0.599	0.412

(d) Maze length

Table 5.12: Comparison of summary statistics derived from maze annotations produced by WI model: leave-transcript-out (LTO) and simulated transcript (SIM) procedures

yield a better model, and therefore one would also expect that using all available SALT-annotated data would produce the best model. As we have seen in this section, and above in Section 5.4, however, models trained on multiple SALT corpora do not appear to produce better maze annotations than corpus-specific models, regardless of whether we evaluate them intrinsically (as in Section 5.4) or extrinsically, as we have done in this section. This suggests that there are systematic differences between the SALT corpora. In the next section, we show that one major difference between these corpora lies in the quality of the maze annotations: the annotations in at least one of the corpora (GillamNT) do not follow the SALT guidelines.

5.6 Comparison of SALT corpora

Although there are instances where a maze-detection model trained on multiple corpora outperforms corpus-specific models on each of its constituent corpora, this is generally not the case. We now turn our attention to the SALT corpora themselves to see why this might be the case; after all, one would expect a model trained on more data, particularly data that ostensibly comes from the same domain, to outperform a model trained on less data. In particular, we will look at high-level characteristics of the SALT corpora, such as token and type counts. We will also investigate whether there are differences in what words frequently appear in mazes across tasks, and whether there are similar differences between research groups. In particular, we are interested in whether there are differences between the SALT corpora that are not accounted for by the nature of the tasks, or the ages of the participants. We find that the maze annotations in one of the SALT corpora, GillamNT, deviates from the SALT guidelines, annotating parenthetical comments as mazes alongside 'true' mazes. This issue with annotations is discussed further in Section 5.6.3.

Note that throughout this section we do not excise mazes before computing any statistics unless explicitly noted. Thus, for example, when we discuss 'utterance length', this means the number of words in an utterance, regardless of whether these words are in mazes. This is in contrast to standard practice in language sample analysis, in which mazes are excised before computing, say, mean length of utterance. However, given that we are trying to pinpoint differences in the SALT corpora, we do not want to risk introducing additional variability from potentially inconsistent maze annotations.

5.6.1 Tokens and types

In each utterance of the SALT corpora, we count: the number of tokens, the number of types, and the token to type ratio. Table 5.13 contains statistics summarizing the distribution of these counts in each of the SALT corpora, sorted by the mean.

In Table 5.13a, we see that the Expository corpus has far more words per utterance than any of the other corpora. From Table 5.13b, we see that these utterances do not simply consist of the same word repeated over and over; the mean type count per utterance in Expository is substantially higher than for any of the other SALT corpora. This is not surprising because unlike the other corpora, none of the participants in the Expository corpus is younger than 10. Among the other corpora, we see that the conversational ones (Conv and NZConv) have fewer words per utterance than the others.

One could quite reasonably divide the SALT corpora into three groups, based on the mean length of utterance in terms of either tokens or types: 1) Expository; 2) Conv, NZConv (and potentially NZPerNar); and 3) the four other corpora together in a group. In other words, the mean length of utterances in each of the corpora seems to correspond nicely to the task it contains. To

Corpus	Mean	Std	Skew	Max
NZConv	6.4	3.9	1.5	47
Conv	6.6	4.7	1.8	67
NZPerNar	7.0	4.1	1.5	56
NZSR	8.2	3.8	1.1	30
ENNI	8.9	4.0	1.3	46
NarSSS	9.1	5.3	1.6	60
NarSR	9.7	5.1	1.7	67
GillamNT	9.8	5.1	1.4	49
Expository	13.9	8.4	1.7	76

(a) Tokens per utterance

Corpus	Mean	Std	Skew	Max
NZConv	6.0	3.3	1.0	32
Conv	6.1	4.0	1.3	41
NZPerNar	6.5	3.4	0.8	33
NZSR	7.5	3.0	0.5	21
ENNI	8.1	3.1	0.9	31
NarSSS	8.3	4.2	1.1	37
NarSR	8.8	3.9	1.3	42
GillamNT	8.9	4.0	1.0	33
Expository	12.2	6.2	1.1	52

(b) Types per utterance

Corpus	Mean	Std	Skew	Max
NZConv	1.043	0.127	9.465	7.500
Conv	1.045	0.115	4.038	3.00
NZPerNar	1.054	0.135	4.734	4.000
NZSR	1.074	0.144	2.828	2.286
ENNI	1.083	0.147	2.872	3.000
NarSSS	1.077	0.146	3.165	3.000
NarSR	1.087	0.151	3.098	3.750
GillamNT	1.087	0.153	3.430	4.000
Expository	1.099	0.135	1.886	2.042

(c) Token/type ratio per utterance

Table 5.13: Summary statistics from the SALT corpora, including all words in mazes

verify this quantitatively, we pool the corpora of conversational tasks into one group, and the corpora of narrative tasks into another. The EXPOSITORY corpus stands alone as its own group. We then compare the utterance lengths of each pair of groups using the Mann-Whitney U test with Bonferroni correction[3] to see whether the populations are the same (the null hypothesis), or whether the utterance lengths from one group tend to be larger than the other. In all cases the Mann-Whitney U statistic is extremely large, indicating that the utterance lengths in the pairs of groups follow different distributions: Conversation vs. Expository ($p < 0.001$; r=0.228), Conversation vs. Narrative ($p < 0.001$; r=0.293) and Expository vs. Narrative ($p < 0.001$; r=0.130). We interpret this as meaning the differences within the transcripts in these three groups, at least in terms of utterance length, are dwarfed by the between-group differences, and as a result, we can consider them as three distinct groups.

As Table 5.13a shows, the SALT corpora contain some very long utterances. Here, for example, is the longest utterance from NARSR, which also includes a very long and complicated maze.

(5.8) (if she didn't get her way she would if if she didn't get her way if she didn't get her way she wouldn't in the in if if she if she didn't get her if she and if she didn't get her way really if she and) if she didn't get her way she would make bad faces (and) and throw apples and yell very loudly .

As illustrated by the mean utterance lengths in that table, these utterances appear to be exceptional, and therefore they should not be taken as conclusive evidence of poor utterance segmentation. There are, however, no major surprises in the statistics in Table 5.13 that would suggest radical differences between the corpora beyond what one would expect due to the different tasks they contain.

[3]Bonferroni correction is typically considered the most conservative form of correction, and therefore results identified as statistically significant after such correction should remain statistically significant under other forms of correction as well

5.6.2 Maze counts and counts of words in mazes

Corpus	Mean	Std	Skew
NZConv	5.5%	12.4	3.1
Conv	5.1%	11.9	3.3
ENNI	5.8%	11.2	2.2
NZPerNar	6.2%	12.5	2.6
Expository	6.7%	11.9	3.0
NarSR	7.0%	12.4	2.3
NZSR	7.4%	13.6	2.3
NarSSS	8.0%	13.8	2.4
GillamNT	10.3%	18.9	2.5

(a) Percent of words in mazes per utterance

Corpus	Mean	Std	Skew	Max
NZConv	0.3	0.5	2.2	6
Conv	0.3	0.6	2.4	7
NZPerNar	0.3	0.6	2.1	5
ENNI	0.3	0.6	1.9	5
NZSR	0.4	0.7	1.9	5
NarSR	0.4	0.7	1.8	6
NarSSS	0.5	0.7	1.8	5
Expository	0.5	0.8	1.7	6
GillamNT	0.5	0.7	1.7	6

(b) Mazes per utterance

Corpus	Mean	Std	Skew	Max
NZConv	1.8	1.6	3.6	22
Conv	1.8	1.5	3.2	21
NZPerNar	1.9	1.7	3.0	16
ENNI	2.0	1.6	2.8	20
NZSR	2.0	1.7	3.0	16
NarSSS	2.0	1.8	3.5	23
Expository	2.0	1.9	3.5	23
NarSR	2.1	1.9	4.9	45
GillamNT	2.3	2.2	3.0	29

(c) Maze lengths

Table 5.14: Summary statistics from the mazes in the SALT corpora

We count the following for each utterance in each of the SALT corpora: the percentage of tokens in mazes, the number of mazes, and the length of each maze. Table 5.14 contains statistics summarizing the distribution of these counts in each

of the SALT corpora, sorted by the mean. Due to the discrepancies in utterance length across the SALT corpora, we felt that presenting the counts of tokens in mazes would be uninformative.[4] We also omit the maximum percentage of words in mazes in each corpus because there are always utterances that consist entirely of words in mazes, for example 'um'.

Looking at Table 5.14a, we see that the maze rate is lower in conversational tasks than in narrative tasks. This is perhaps unsurprising, as one would expect the conversational tasks to have a lower cognitive load than the narrative tasks, and high cognitive load has been found to increase the rate with which speakers produce mazes, including both disfluencies and filled pauses (Müller et al., 2001). That being said, the three tables in 5.14 illustrate that the GillamNT corpus has an unremarkable number of mazes per utterance, but these mazes are relatively longer than the ones in other corpora. In terms of the statistics in Table 5.14, the other SALT corpora appear quite similar, and none seems particularly remarkable.

Comparing Tables 5.13a (or 5.13b) and 5.14a, we see that unsurprisingly, the longer the utterances in a corpus (including words in mazes), the higher the percentage of words in mazes it contains. One exception to this is Expository: it has by far the longest mean length of utterance, but the percentage of words in mazes in the corpus is near the middle. Quantitatively, the Pearson correlation between mean tokens per utterance and the mean percentage of words in mazes per utterance is only 0.378 ($p = 0.315$) when we include all corpora. However, when we exclude the Expository corpus, we observe a significant correlation between these two summary statistics ($r = 0.739$, $p = 0.036$). A likely explanation for why the Expository corpus bucks this trend is simply that the transcripts in it are collected from much older children than any of the other corpora. Even

[4]This differs from maze length: in an utterance of 10 words, if every other word were in a maze, we would have an average maze length of one word, but if the first five words were, then the average maze length is five words. In both cases, however, there are five words in mazes.

the youngest participants in the Expository corpus (10 years 7 months) are older than the oldest participants in many of the other corpora.

5.6.3 Frequency of N-Grams in Mazes

Rank	Conv	ENNI	NZPerNar	Expository
1	and and then	and then and	and um and	they can also
2	we go we	then the then	and and we	um there's no
3	it was it	the the rabbit	then then I	mainly used for
4	and I and	the then the	and and my	team serves and
5	um I have	the the giraffe	I went I	then the red

(a) Trigrams most likely to appear in mazes

Rank	Conv	ENNI	NZPerNar	Expository
1	I like to	and then he	I went to	you have to
2	you have to	and then the	and um and	and then you
3	and then we	and then they	and then we	you can get
4	and then I	and then and	when I was	um and then
5	I don't know	and the elephant	I had to	if the ball

(b) Most frequent trigrams in mazes

Rank	Conv	ENNI	NZPerNar	Expository
1	they they um	and she and	um there was	um and then
2	um the um	castle castle castle	and and and	when you are
3	went to um	is he is	and the and	you just like
4	go to um	um the um	um well um	if it goes
5	it's it's a	and he and	I um I	if the other

(c) Trigrams in mazes with highest pointwise mutual information; minimum trigram count of 5 in each corpus

Table 5.15: Common trigrams in mazes in three different corpora

We expect that certain n-grams should appear more frequently in a maze than others. To capture this intuition, we inspect the most common n-grams in mazes in the SALT corpora, and the n-grams that appear particularly frequently in mazes. We also inspect the n-grams which share the highest pointwise mutual information with being in a maze while occurring at least five times in each of the SALT corpora. For example, in the Conv corpus, the most common bigram in mazes is 'and then', the bigram 'um I' is in a maze more frequently than

any other bigram (98.7% of its occurrences), and the bigram 'um so' has the highest pointwise mutual information with being in a maze (12.72). While none of these observations is surprising, inspecting n-grams from mazes organized by frequency does provide insights into ways maze annotations in particular SALT corpora diverge from the standard.

In all but one corpus (GillamNT, discussed in Section 5.6.3) we find nothing particularly surprising in either the n-grams that tend most strongly to appear in mazes, or in the n-grams that appear the most in mazes. To illustrate, Table 5.15a shows the trigrams that are most likely to appear in mazes when they occur in four of the SALT corpora. The trigrams from three of the four corpora (not Expository) are unremarkable: all of them contain either repeated words or filled pauses. Although the trigrams from the Expository corpus look interesting, most of them tend to appear quite rarely. For example, 'they can also' appears three times, and two of those times it is in a maze, and 'then the red' only occurs twice, once in a maze.

Table 5.15b shows the trigrams in these same corpora with the highest number of occurrences in mazes. Again, most of these contain repeated words. The trigrams that do not occur frequently appear outside of mazes as well, which suggests that the entire trigram is repeated, for example:

(5.9) (And the elephant) and the elephant is sitting there.

Finally, Table 5.15c shows the trigrams in these corpora with the highest pointwise mutual information with being in a maze. The trigrams in Table 5.15c all occurred at least five times, but we explored other thresholds as well. Lower thresholds, however, tend to yield noisier n-grams, namely ones that appear only a time or two in the entire corpus, possibly only in mazes.

n-gram	maze freq.	fluent freq.
I don't remember	62	4
I forgot the	56	4
that's all I can	42	2
I don't know the	18	2
I can think of	18	2

Table 5.16: Selected n-gram frequencies in mazes and fluent segments in the GillamNT corpus

GillamNT

Table 5.16 shows some n-grams from the GillamNT corpus that appear much more frequently in mazes than outside of them. The fact that these n-grams appear so rarely outside of mazes rules out the possibility that they are included in mazes because they are repetitions. Examples of utterances with these phrases include:

(5.10) And (**I don't remember** his name).

(5.11) (**I don't remember** the rest of it).

(5.12) (And then **I don't remember** the rest).

(5.13) (Th* **that/'s all I can** think of).

(5.14) (No **that/'s all I can** know).

(5.15) (**I forgot the** boys name).

(5.16) (And then um **I forgot the** next part).

In addition to some missing morphological annotations (*don't*, which occurs 632 times in the corpus, should be *do/n't*, which appears only four times), it appears that utterances containing these n-grams are parenthetical remarks that

have been annotated as mazes. According to SALT conventions, parenthetical remarks are to be annotated with double parentheses (SALT Software, 2014a):

(5.17) The boy ((I can/'t remember his name)) left the house.

but in the GillamNT corpus, there is not a single utterance that contains this annotation. It is therefore not the case that the examples above are simply one-off annotation errors; rather, parenthetical remarks are systematically annotated as mazes in the GillamNT corpus. This explains why the GillamNT corpus has a relatively high percentage of its words in mazes, as shown in Table 5.14a. In light of this finding, it is not surprising that including data from the GillamNT corpus in the training data for a maze detector degrades system performance on other corpora as seen in Table 5.3.

Collapsing the distinction between mazes and parenthetical remarks affects the statistics computed from the GillamNT corpus: mazes are always excluded from MLU, while parenthetical remarks may optionally be included; the number of mazes in a transcript, and the average number of words per maze are both statistics that SALT software reports. Estimates of MLU, maze counts, and the number of words per maze derived from transcripts in the GillamNT corpus will be systematically inflated because parenthetical remarks are annotated as mazes in this corpus.

5.7 Conclusions

We have seen that automated approaches to disfluency detection can be applied in a straightforward manner to the related task of maze detection. Specifically, we adapt the disfluency detector proposed by Qian and Liu (2013) to this task, and we see that in general this maze detector performs comparably to the disfluency detector in terms of F1 score. We evaluate corpus-specific baseline models, and

find that in most cases, we are not able to train more effective models by simply training on the merged training sets of multiple corpora. However, by using a simple technique for domain adaptation (FEDA, see Section 5.4.2), we are able to leverage training data from multiple corpora effectively, and train models that outperform the corpus-specific baselines. We also find that summary statistics extracted from automatically annotated transcripts tend to be significantly, but not substantially, different from those computed from manually annotated ones.

We have also investigated the SALT corpora to see whether there are systematic differences in language or maze annotations. Many of our findings are unsurprising, for example that mean utterance length is shortest in corpora of conversations, longer in ones of narrative tasks, and longest in the corpus of an expository task (EXPOSITORY). We find that examining the most frequent n-grams in mazes, specifically those that occur frequently in mazes but rarely outside of them, to be a particularly effective way of inspecting maze annotations. Doing so has alerted us to a severe issue with the GILLAMNT corpus: both mazes and parenthetical comments are annotated the same way there (with single parentheses), while SALT conventions specify that mazes should be annotated with single parentheses and parenthetical comments with double parentheses. Deviant annotation standards are not only an issue for training automatic maze detectors, but also for interpreting reference statistics computed from the GILLAMNT corpus.

Based on our findings, we believe that the SALT annotation system should include more rigorous guidelines for maze annotations, and that the mazes reference databases should be re-annotated following the same standards. At present, the maze annotations in different corpora tend to vary slightly, as illustrated by the fact that the corpus-specific models typically outperform models trained on more data. We believe the most likely culprit for this are the

vague maze guidelines in the SALT manual. Adding detail to these guidelines will help clinicians and researchers annotating their own data. More importantly, it will allow valid comparisons between summary statistics derived from these transcripts and the reference transcripts in the SALT corpora. At present, however, this is not possible because all of these corpora are annotated differently, following standards that are not sufficiently detailed to allow them to be applied elsewhere.

Chapter 6

SALT Error Code Detection

6.1 Introduction

SALT error codes are used to capture a wide variety of grammatical errors. As discussed in more detail in Section 2.2.3, these errors include, but are not limited to, overregularization errors (`[EO]`), omitted words (`[OW]`) and morphemes (`[OM]`), and other word-level (`[EW]`) and utterance-level (`[EU]`) errors. Nevertheless, particular error codes do not necessarily capture the same set of errors in each SALT corpus, and different research groups tend to use particular error codes differently. This can be seen with a quick glance at the counts of error codes in the various SALT corpora (for more detail see Section 3.1): the ENNI corpus has no `[EO]` codes, and hardly any `[OM]` codes, while the Conv corpus has a fair number of both. It is not that the two corpora contain radically different types of errors, it is simply that the errors annotated with [EO] and [OM] codes in the Conv corpus are annotated with [EW] in the ENNI corpus.

There are three major impediments to using existing techniques for grammat-

ical error detection and classification, discussed in Section 4.6, to detect SALT error codes. First, many existing error detectors tend to be designed to identify a specific set of errors, and they cannot be re-trained easily, if at all, to identify different ones. The grammar checker in Microsoft Word and ETS e-rater, both of which we use as baselines, are examples of such systems. Using such systems to detect SALT error codes requires the user to map the system's output to error codes, which is not a trivial task, as discussed in Section 6.3.1. The second impediment is that the vast majority of error detectors are designed for formal, written language. Although there are many errors that affect both spoken and written language, for example verb agreement errors, there are many differences between spoken and written language that are relevant to error detection. For example, punctuation and spelling errors are completely irrelevant to transcribed speech, yet they are critical to identify in a written setting. Similarly, disfluencies do not affect written language, but they are common in spoken language. As discussed in Section 4.6, some current techniques in grammatical error correction may be applicable to SALT error code detection, but there are likely to be major obstacles in adapting them to this task. For example, some of the cutting-edge research in grammatical error detection leverages parallel corrected data, which is simply unavailable for SALT-annotated transcripts.

We propose several systems for SALT error code detection, each of which model different amounts of linguistic structure. These range from a minimal amount in the utterance-level classification approach (Section 6.4), through neighboring words in a tagging approach (Section 6.5), to dependency structure (Section 6.6 and 6.8). All of these approaches are entirely data-driven, and are not designed with any corpus-specific features, and thus they should be applicable to any SALT-annotated corpora

After exploring various systems with different configurations, we look at the

output of one of them to see what sorts of errors it identifies, and which ones it fails on (Section 6.9). Finally, in Section 6.9.1 we examine the consistency of the manual labels in different corpora. We argue that labeling consistency is the most plausible explanation for why the detectors' performances vary so widely across corpora. Such inconsistency should come as no surprise given one of our findings in Chapter 5, namely that the quality of maze annotations are highly variable between the SALT corpora.

6.1.1 Scope of error detection

Minimal context to identify	Error type	Example	Notes
Word	Overgeneralization	I goed [EO : went] to the store .	
Sentence	Subject agreement	She are [EW : is] over there .	
	Subject case	Him [EW : he] left .	
	Omitted word	What [OW] you selling ?	Standard North American English
Conversation	Gender errors	She [EW : he] left.	Antecedent in conversation
	Factual errors	Cathy [EW : Joan] is tall .	If already discussed in conversation
	Pragmatic errors	Mom [EU] .	Response to 'Did you stop?'
Real world	Factual errors	The house is blue [EW : red] .	If not discussed in conversation
	Pragmatic errors	Good morning [EU] .	Said at night

Table 6.1: Overview of errors and the context required to identify them. Errors above the double line are within the scope of this thesis.

The amount of language and linguistic structure needed to identify a grammatical error varies depending upon the type of error. At the minimum, a single word is required: overgeneralized forms such as *goed* and *robuster* are ungrammatical regardless of context. Next, we have errors that can be identified

by looking at a single utterance, even if they only involve a few words within it, for example case errors ('Him left.') and using the wrong preposition ('played of the ball'). Other types of errors, for example factual errors, gender errors, and pragmatic errors, may require either multiple utterances from a single transcript, or real world knowledge to identify. In this thesis, we are only attempting to identify errors that can be identified with a single utterance, and this set of errors naturally includes those which can be identified with a single word. We indicate these errors, along with examples of errors that require more context, and are therefore out of scope, in Table 6.1. This table is by no means exhaustive with regards to enumerating all of the classes errors identifiable in each context.

To be clear, we do not alter the SALT corpora based upon Table 6.1. As a result, there will be training and test utterances with errors that we are not interested in detecting. Furthermore, some of these errors will be ones that we cannot reasonably expect any existing system to detect, for example pragmatic errors or factual errors. In the error analysis in Section 6.9, however, we do consider whether the errors missed by our system could reasonably be caught, or whether they are errors that require more context than a single utterance to identify.

6.2 Evaluation

Evaluating system performance in tagging tasks on manually annotated data is typically straightforward: 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 and thus best mimics the behavior of one or more annotators. This is not necessarily the case with the task of automatically applying SALT error codes, however, and different users may want to use systems with different operating points. For example, some users may want to inspect

a small sample of utterances that likely contain errors, while others may wish to exclude utterances that are unlikely to contain errors from manual labeling. Here we explain the metrics we use to summarize system performance, and how we are able to tune these using confidence scores.

6.2.1 Metrics

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

A tool that identifies utterances with any error that would be marked by a SALT error code can be applied in a variety of ways. Most obviously, it can be used to expedite an annotator's work by removing utterances without any error codes from consideration. Such a tool could also be used to get a quick sample of utterances with errors, which may be of interest to clinicians. As discussed in Chapter 7, 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. These applications suggest that one key way of evaluating a system to automatically apply SALT error codes is as a binary classifier: 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 UTTERANCE, and it is illustrated in Figure 6.1.

A tool that identifies specific SALT error codes also has several applications. Most obviously, it could speed up manual annotation. As explored in Chapter 7, automatically detected error codes can also be used to predict scores on several

structured instruments. To this end, we 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, and it is illustrated in Figure 6.1. Our graph-based system, presented in Section 6.8, is unable to predict repeated error codes, and therefore we evaluate how well it can identify specific error codes in a slightly different way, described in Section 6.8.1.

We are not aware of any analyses performed with SALT error codes that takes into account their location. As a result, we see no reason to evaluate any of the detectors in terms of how well they predict the locations of specific error codes.

6.2.2 Confidence Scores

Each of the predictions output by systems we propose is accompanied by a confidence score, although this is not the case for the output of MS Word or ETS e-rater (in Section 6.3). We are able to count each prediction as positive or negative at a particular threshold θ quite easily: if the confidence score of that prediction is at least θ, we count it as positive, and if not, we count it as negative. This applies to both individual error code predictions and to predictions of whether an utterance contains an error. So, for example, if we predict that the utterance 'him can go home' contains an `[EW]` code with a confidence of 70, then for $\theta \geq 70$ we retain the prediction, and for $\theta < 70$, we ignore it.

We present our results in the form of plots illustrating the trade-off between precision and recall at a wide range of confidence levels, as can be seen in Figure 6.9. We also include the area under the curve (AUC) for each curve in the legend of the plot. AUC summarizes the overall range operating points accessible with a particular system, and ranges from 0 (worst) to 1 (best).

6.2.3 Interpretation of evaluation

Precision, recall and F1 score are common ways of evaluating system performance in the field of natural language processing. In general, systems with higher F1 score are preferred. External considerations, however, can inform the choice of how to balance the trade-off between precision and recall. For example, users tend to prefer grammar checkers (such as the one in Microsoft Word) to be biased towards precision, or in other words ones that do not flag grammatical sentences unnecessarily, even if they fail to flag ungrammatical ones (Helfrich and Music, 2000). It is for this reason that the 2014 CoNLL shared task on grammatical error correction chose to use $F_{0.5}$, which weights precision more heavily than recall, instead of the more widely used F1 score, which weights precision and recall equally (Ng et al., 2014).

In the case of automatically applying SALT error codes, there does not seem to be an ideal operating point because there are many ways one could use the output of such a system. For example, a research group could use a SALT error code detector to identify utterances without any error codes so that time is not wasted manually coding them. In such a scenario, one would want to use a system with high recall so that utterances with errors do not get removed from the coding process. On the other hand, if a researcher simply wants to see a small sample of ungrammatical utterances in a transcript, a system with high precision would be appropriate so that they are not given a sample containing grammatical utterances. One can imagine other scenarios with different ideal operating points, for example using automatically predicted error codes to predict scores on structured instruments (see Chapter 7). The bottom line is that it is important for a system that detects SALT error codes to be able to trade off precision and recall, and that these systems should be evaluated at a variety of operating points.

6.2.4 Corpora

The SALT corpora are not all the same size, and they do not contain the same number of error codes. Some, in fact, contain very few error codes. We exclude GillamNT from this chapter because it only has four [OW] codes in the entire corpus. Other corpora, specifically Expository and NZSR have so few that their development folds (10% of the entire corpus) have fewer than 100 error codes each. As a result, a single prediction can have a dramatic impact upon precision and recall, thus making it very difficult to compare system performance. Throughout this chapter, rather than simply presenting a high-level overview of performance on all of the SALT corpora, we focus our experiments on the three largest corpora (Conv, ENNI, and NarSR). We do so because SALT error code is a novel task, and we believe that presenting a smaller variety of results in depth will yield a clearer, more informative investigation than simply carrying out these experiments on as many corpora as possible. Nevertheless, we do include baseline results for all of the SALT corpora that include error codes, which is all of them except for GillamNT.

6.2.5 Setting operating points by manipulating the proportion of errors in training data

Corpus	Utterances	Errors	% Error
Conv	64,034	5,176	8.1%
ENNI	46,556	5,821	12.5%
NarSR	11,502	1,660	14.4%
Expository	3,916	427	10.9%
NarSSS	12,645	1,369	10.8%
NZConv	19,663	1,020	5.2%
NZPerNar	15,728	1,169	7.4%
NZSR	2,148	229	10.7%

Table 6.2: Errors in unmodified training folds of SALT corpora. Corpora used in this chapter are in the top portion of the table.

In all of our experiments we explore the effect of varying the proportion of utterances in the training data that contain an error. We set the percentage of utterances with an error in the training data by removing utterances randomly: if we want a higher percentage of errors than the original corpus contains, we remove utterances without any errors, and similarly, we remove utterances with errors to produce a training corpus with a lower percentage of errors. We evaluate training folds with 10%, 20%, 30%, ..., 100% of the utterances contain an error, in addition to the unmodified training corpora with the error percentages shown in Table 6.2. We do not alter the development or test folds under any circumstances.

Varying the percentage of utterances with an error in the training data affects system performance. In the case of confidence-based systems, different percentages permit different ranges of operating points. For these systems, we present results from the training fold yielding the best range of operating points, which we defined as the one with the highest area under the precision/recall curve (see Section 6.2.3). For the dependency-based error detection system, which does not produce confidence scores, adjusting the percentage of utterances with an error in the training data allows us to control the operating point: by increasing the percentage of utterances with an error, we can improve recall at the expense of precision. We combine the resulting models to produce confidence scores, as discussed in Section 6.6.1.

6.3 Baseline systems

As discussed in Section 4.6, the vast majority of grammatical error detectors are designed for formal English written by typically developed adults, as opposed to transcripts of spoken language collected from children who may have developmental disorders. Here, we investigate how well the grammar check function in

Microsoft Word, and ETS e-rater, two of the best-developed grammatical error detectors, perform at the task of identifying SALT error codes. We describe each of the systems in turn, and report their performance detecting utterances with errors in the SALT corpora. Since neither of these systems is able to detect specific SALT error codes, we do not report ERROR performance.

6.3.1 Microsoft Word

Microsoft (MS) Word includes a function to identify grammatical errors in the user's writing. This grammar checker was designed with the errors produced in formal, written English by native speakers in mind, as opposed to informal, spoken English (Shermis and Burstein, 2013).[1] For more on the differences between As a result, it can identify many types of errors that are irrelevant to our investigation, for example punctuation errors or misused words (ex. substituting 'effect' for 'affect'), as shown in Table 6.3. Nevertheless, MS Word's grammar check can identify certain errors one would observe in SALT-annotated data, for example subject-verb agreement errors. These categories of errors are also shown in Table 6.3.

We test the performance of MS Word's grammar check on the development folds of each SALT corpus, as shown in Table 6.4. Note that we are only able to evaluate it in terms of UTTERANCE performance, as Microsoft Word only identifies errors; it does not categorize them. Furthermore, even if we were to identify which errors in the data corresponded to each option in Table 6.3, we would still have to map them to SALT error codes. These mappings, however, are not one-to-one. For example, 'subject-verb agreement' errors include errors with both `[EW]` and `[OM]` codes. As a result, using MS word to identify SALT error codes is likely to be quite difficult, and since error code conventions can

[1] See Section 4.6 for more on the differences on performing grammar checking with spoken as opposed to written language.

Error	Description	Used here?
Capitalization	Capitalization problems, such as proper nouns ("Mr. jones" should be "Mr. Jones") or titles that precede proper nouns ("aunt Helen" should be "Aunt Helen"). Also detects overuse of capitalization.	✗
Fragments and run-ons	Sentence fragments and run-on sentences.	✗
Misused words	Incorrect use of adjectives and adverbs, comparatives and superlatives, "like" as a conjunction, "nor" versus "or," "what" versus "which," "who" versus "whom," units of measurement, conjunctions, prepositions, and pronouns.	✗
Negation	Use of multiple negatives.	✓
Noun phrases	Incorrect noun phrases; a/an misuse; number agreement problems in noun phrases ("five machine" instead of "five machines").	✓
Possessives and plurals	Use of a possessive in place of a plural, and vice versa. Also detects omitted apostrophes in possessives.	✗
Punctuation	Incorrect punctuation, including commas, colons, end-of-sentence punctuation, punctuation in quotations, multiple spaces between words, or a semicolon used in place of a comma or colon.	✗
Questions	Nonstandard questions such as, "He asked if there was any coffee left?", "Which makes an offer a good solution?", and "She asked did you go after all?".	✗
Relative clauses	Incorrect use of relative pronouns and punctuation, including "who" used in place of "which" to refer to things, "which" used in place of "who" to refer to people, unnecessary use of "that" with "whatever" and "whichever," or "that's" used in place of "whose."	✗
Subject-verb agreement	Disagreement between the subject and its verb, including subject-verb agreement with pronouns and quantifiers (for example, "All of the students has left" instead of "All of the students have left").	✓
Verb phrases	Incorrect verb phrases; incorrect verb tenses; transitive verbs used as intransitive verbs.	✓

Table 6.3: Options available and used here in Microsoft Word 2010's grammar check (Microsoft, 2015)

Corpus	P	R	F1
CONV	0.464	0.139	0.214
ENNI	0.530	0.276	0.363
EXPOSITORY	0.364	0.170	0.232
NARSSS	0.367	0.131	0.193
NARSR	0.425	0.145	0.216
NZCONV	0.333	0.174	0.228
NZPERNAR	0.385	0.162	0.228
NZSR	0.200	0.077	0.111

Table 6.4: Performance of MS Word Grammar Check on development folds of SALT corpora; UTTERANCE evaluation; mazes not removed

vary across corpora, the mapping would need to be revised repeatedly.

As can be seen in Table 6.4, Microsoft Word's grammar check does not perform very well. In general, it yields predictions with higher precision and lower recall. We note that the relatively high-precision operating point is to be expected, as one of the design goals of Microsoft Word's grammar check is to avoid presenting users with false positives (Helfrich and Music, 2000).

6.3.2 ETS e-rater

ETS e-rater is an automatic system to assess writing proficiency (). As with MS Word, ETS e-rater was designed to identify deviations from formal, written English, as opposed to informal, spoken English. Like MS Word, e-rater can identify many types of errors that are irrelevant to our investigation, for example punctuation and capitalization errors, as shown in Table 6.5. E-rater, however, identifies grammatical errors more finely than MS Word. For example, grammar check can identify certain errors one would observe in SALT-annotated data, for example subject-verb agreement errors, and these categories of errors are also shown in Table 6.3.

We test the performance of the ETS e-rater on the development folds of

Error type	Used here?	Error type	Used here?
Fragments	✓	Run-on sentences	✗
Garbled sentences	✓	Subject-verb agreement	✓
Ill-formed verbs	✓	Pronoun errors	✓
Possessive errors		Wrong or missing word	✓
Proofread this!	✓	Determiner noun agreement	✓
Missing or extra article	✓	Confused words	✗
Wrong form of word	✓	Faulty comparisons	✗
Preposition error	✓	Nonstandard word form	✓
Negation error	✓	Wrong part of speech	✓
Wrong article	✓	Spelling	✗
Capitalize proper nouns	✗	Missing initial capital letter in a sentence	✗
Missing question mark	✗	Missing final punctuation	✗
Missing apostrophe	✗	Missing comma	✗
Hyphen error	✗	Fused words	✗
Compound words	✗	Duplicates	✗
Extra comma	✓		

Table 6.5: Error types available and counted here in ETS e-rater ()

Corpus	P	R	F1
Conv	0.081	0.419	0.136
ENNI	0.151	0.244	0.187
Expository	0.073	0.191	0.106
NarSSS	0.102	0.283	0.149
NarSR	0.190	0.235	0.210
NZConv	0.055	0.459	0.098
NZPerNar	0.066	0.296	0.108
NZSR	0.064	0.125	0.085

Table 6.6: Performance of ETS e-rater on development folds of SALT corpora; Utterance evaluation; mazes not removed

each SALT corpus, as shown in Table 6.6. We only evaluate it in terms of Utterance performance, since mapping the errors identified by e-rater do not correspond perfectly to SALT error codes. For example, 'subject-verb agreement' errors include errors with both `[EW]` and `[OM]` codes. As a result, using ETS e-raterd to identify SALT error codes is likely to be quite difficult, and since error code conventions can vary across corpora, the mapping would need to be

revised repeatedly.

As can be seen in Table 6.6, the ETS e-rater performs worse than the MS Word grammar checker (see Table 6.4). As a result, we will only use the MS Word performance as a baseline in the remaining experiments in this chapter. Unlike MS Word, ETS e-rater yields predictions with higher recall and lower precision. This is understandable, given that the ETS e-rater is designed to assess writing skills, and is not subject to the same sort of user interface considerations as MS Word.

6.4 Classifier-based error detection

The classifier we discuss here is based on one proposed by Hassanali and Liu (2011), and we originally presented it in the context of using automatically produced SALT error codes to predict the presence of Autism spectrum disorder and a language impairment (Morley et al., 2013). This classifier is only able to make predictions at an utterance level, and it ignores any deeper structure present in an utterance because the only features it uses are pairs of words that appear close to one another.

6.4.1 Methods

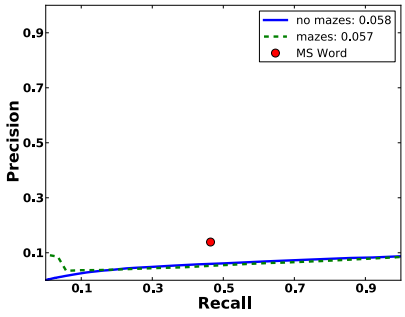
Hassanali and Liu (2011) investigated rule- and classifier-based approaches to identify six specific errors in the Paradise corpus (Paradise et al., 2005), which contains transcripts of spoken language collected from children, some of whom have *otitis media*, a hearing disorder. We re-implemented their classifiers,and found that only one of them ('misuse of -ing participle') was able to detect SALT error codes. It is this classifier that we discuss here. We refer readers curious about the other classifiers to their original paper (Hassanali and Liu, 2011) as well as our follow-up work (Morley et al., 2013).

The classifier uses two types of features: word bigrams, which are all pairs of adjacent words in the original utterance (ex. 'the classifier' and 'classifier uses' at the beginning of this sentence); and word skip-1 bigrams, which are all pairs of words that have a single word between them in the original utterance (ex. 'this uses' and 'classifier two' at the beginning of this sentence). These features are extracted from the entire utterance, as this classifier works at the utterance level.

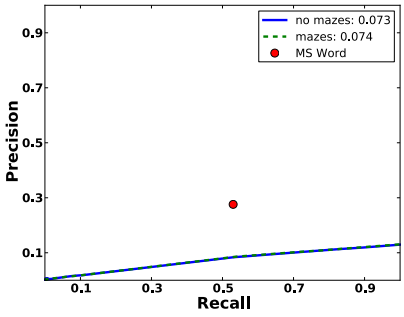
Hassanali and Liu (2011) used a Naive Bayes classifier, but here we use the C-support vector classifier in scikit (Pedregosa et al., 2012) instead. We decided to do so because this classifer's probability estimates are more reliable than those from the Naive Bayes classifier as it does not make the same independence assumptions. Classification is performed on one utterance at a time. The support vector classifier allows us to predict which utterances contain any SALT errors without any difficulty. Using it to predict specific error codes, however, is complicated because a single utterance can have more than one error code, but expanding the number of labels to be predicted (ex. labels representing combinations of error codes observed together in an utterance) risks introducing issues with data sparsity. Furthermore, if we were to use multiple classifiers (ex. one for each error code) or a multilabel classifier, we would still not be able to predict that an utterance contains more than one instance of a particular error code. We therefore only evaluate this classifier in terms of UTTERANCE performance, i.e. how well it can identify utterances with an error.

We train and test the utterance-level classifier, varying the presence of mazes, using the manual maze annotations. We also vary the proportion of utterances with an error in the training data to be 10%, 20%, ..., 100% by randomly removing utterances, but we only report the results with the highest area under the P/R curve, as described in Section 6.2.2.

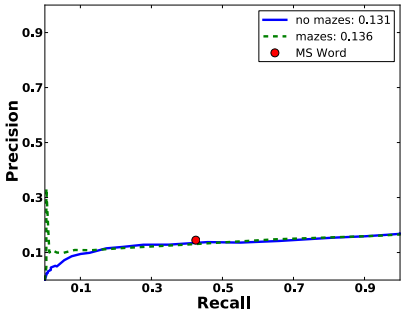
6.4.2 Results and conclusions



(a) Conv



(b) ENNI



(c) NarrativeSR

Figure 6.2: Classifier performance on the development folds of several SALT corpora varying the presence of manually annotated mazes
MS word performance on data with mazes

Overall, we find that the utterance classifier yields very poor performance, even worse than MS Word, as can be seen in Figure 6.2. On most corpora, there is a minimal range of operating points available, even though the classifier's predictions come with confidence scores. The utterance-level classifier is, however, robust to the presence of mazes across most operating points.

6.5 Tagging-based error detection

The utterance-level classifier described in the previous section models essentially models utterances as a bag of words and n-grams. Its poor performance suggests that perhaps more linguistic structure needs to be modeled in order to identify SALT error codes effectively. Therefore in this section, we approach SALT error code detection as a *tagging* task, meaning we model the utterance as a sequence of words, and that we try to find the most likely sequence of tags for each utterance. To see how this contrasts with the classification approach, let us consider the common task of part-of-speech (POS) tagging. In a classification approach, we could simply label each word with its most common tag, so perhaps 'bear' is labeled as a noun (NN) wherever it occurs. This is fine in phrases like 'the bear', but not for ones like 'we bear', in which 'bear' is actually a verb. Tagging approaches, however, take into account the frequency with which particular sequences of tags are observed. Continuing with the example of 'we bear', 'PRP VB' is very common, but 'PRP NN' is rare, and therefore the sequence 'PRP VB' with 'bear' as a verb is more likely than the sequence 'PRP NN' with 'bear' as a noun.

To approach SALT error code detection as a tagging task, we assign a single tag to each word: if there is an error code to the right of the word, then that word is tagged with the error code, and if there is no error code there, then it is labeled as 'no error'. So, for example, the utterance 'him [EW] can go home' would be labeled: 'EW NE NE NE', where NE is the label for 'no error'.

6.5.1 Methods

We perform classification using the CRF++ toolkit with the default settings (Kudo, 2005). CRF++ is a linear-chain conditional random field (CRF) tagger, and for more details on CRFs, we refer the reader to Section 4.3.2. We use the features

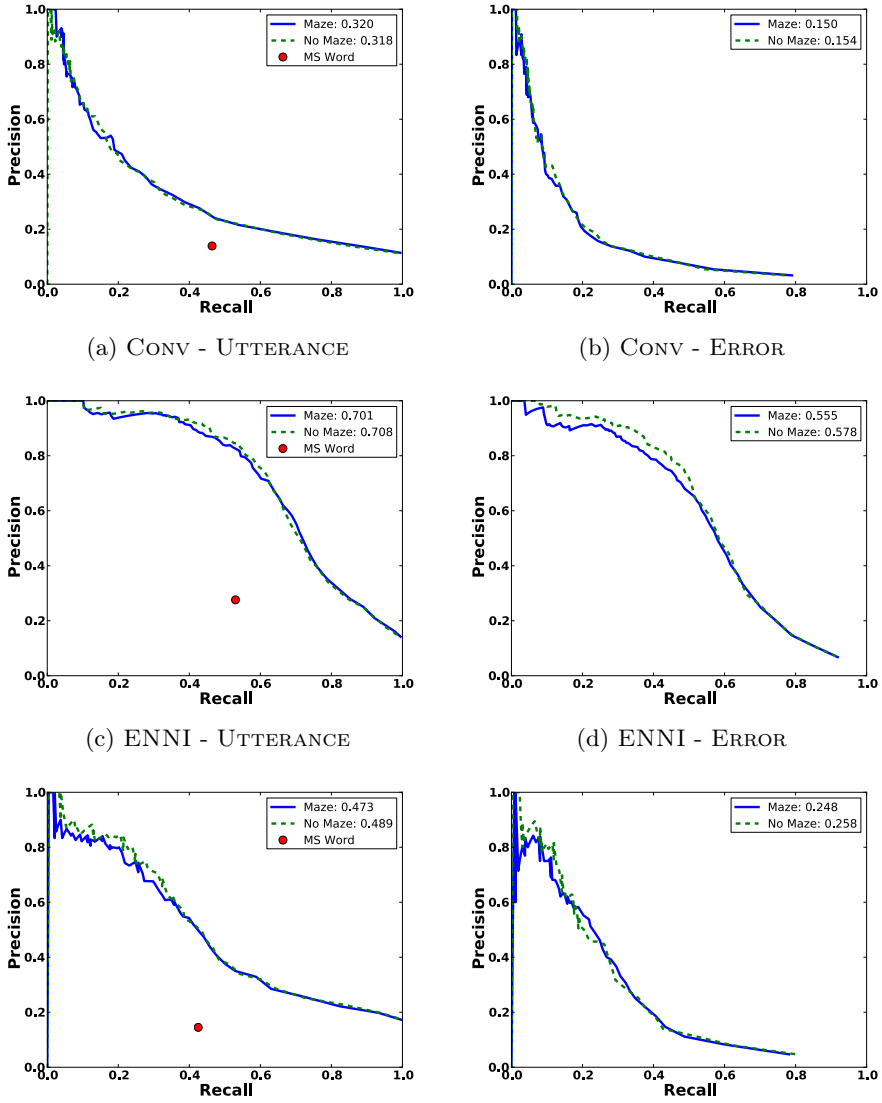
Description	Feature	Location	Example ($i = 0$ at 'goed')
Word unigrams	w_i	$-1 \leq i \leq 1$	goed
Word bigrams	$w_i w_{i+1}$	$-1 \leq i \leq 0$	goed home
POS unigrams	p_i	$-1 \leq i \leq 1$	VBD
POS bigrams	$p_i p_{i+1}$	$-1 \leq i \leq 0$	VBD ADV
POS trigrams	$p_i p_{i+1}$	$-2 \leq i \leq 0$	VBD ADV </S>
Word+POS	$w_i p_i$	$-1 \leq i \leq 1$	goed+VBD
Word+POS / Word	$w_i p_i w_{i+1}$	$-1 \leq i \leq 0$	goed+VBD home
Word / Word+POS	$w_i w_{i+1} p_{i+1}$	$-1 \leq i \leq 0$	goed home+ADV
Word+POS / POS	$w_i p_i p_{i+1}$	$-1 \leq i \leq 0$	goed+VBD ADV
POS / Word+POS	$p_i w_{i+1} p_{i+1}$	$-1 \leq i \leq 0$	goed+VBD ADV

Table 6.7: Features used in tagger, index of current word is 0, word to left is -1, and word to right is 1.
Examples in table have $i = 0$ and are taken from sentence: 'we goed home' with POS tag sequence 'PRP VBD ADV' when current word is 'goed'. </S> is a padding symbol.

illustrated in Table 6.7, all of which are extracted from words and POS tags. We train and test the CRF tagger on data that contains mazes, and on data that has had the manually annotated mazes excised.

6.5.2 Results and Conclusions

In Figure 6.3 we see that the CRF tagger comfortably outperforms the MS Word baseline in terms of Utterance performance and that it is largely robust to mazes. The CRF tagger also outperforms the utterance-level classifier by a wide margin (see Figure 6.2). The plots in Figure 6.3 show that performance varies greatly across different corpora, both in terms of Utterance and Error evaluation. In particular, performance on ENNI is substantially better than the other two corpora. Finally, we see that Utterance performance is in all cases somewhat higher than Error performance. This is unsurprising, as the correct Error level predictions are a proper subset of correct Utterance level



(a) Conv - Utterance (b) Conv - Error

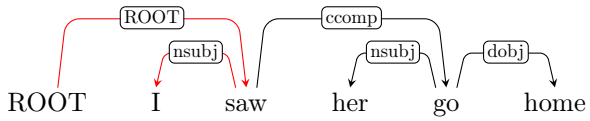
(c) ENNI - Utterance (d) ENNI - Error

(e) NarSR - Utterance (f) NarSR - Error

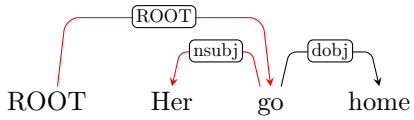
Figure 6.3: CRF tagger performance on the development folds of several SALT corpora varying the presence of manually annotated mazes
MS word performance on data with mazes

predictions.

6.6 Dependency-based error detection



(a) Dependency parse of 'I saw her go home'



(b) Dependency parse of *'Her go home'

Figure 6.4: Dependency structure indicative of a grammatical error

Certain grammatical errors can be identified by looking at a single word, or even a sequence of words, but many cannot. For example, the overgeneralized form 'goed' is always ungrammatical, but whether the word 'her' is grammatical depends upon its context. Consider the utterances shown in Figure 6.4. The utterance in 6.4b is contained fully within utterance 6.4a, yet it is ungrammatical. 'Her go home' is ungrammatical because the subject of the main verb is in the accusative case. This is not obvious if we only consider these utterances as strings of words. In terms of dependencies, however, we see that in the grammatical utterance, the dependent in the `nsubj` dependency headed by the main verb (ie the verb headed by `ROOT`) is in the nominative case ('I'), while in the ungrammatical utterance, there is an accusative pronoun ('Her') in this same structural position. This can be seen in the parses in Figure 6.4 by following the red arcs. We now explore ways of using dependency parses to identify SALT error codes, motivated by the observation that dependency structure can be helpful for identifying grammatical errors.

We propose a simple method to train a dependency grammar that can be used to identify specific SALT error codes. Our procedure has several parameters that can easily be set by the user, and we explore the effect of these parameters on system performance. Recall from the discussion in Section 6.2.3 that unlike many tasks in NLP, the SALT error code detection system with the highest F1 score is not necessarily the best one for all purposes; depending upon the intended use, one may prefer a system with higher precision at the expense of recall, or *vice versa*.

6.6.1 Methods

Setting	Description
Parser	Which parser we use
Arcs labeled	Which arc labels are augmented with error codes
Label encoding	How error codes are encoded in arc labels
Recovery	How to decide which error codes are in the output

Table 6.8: Template for settings of dependency parse-based SALT error detection system

Our method for training and using a dependency grammar to identify specific SALT error codes is as follows (and a concrete example is given below):

1. Produce dependency parses of utterances that have been annotated with SALT error codes
2. Augment the parses to include representation of any SALT error codes
3. Optionally modify the proportion of utterances in the training set with any errors
4. Train a new dependency grammar on these augmented parses
5. Parse new data using this grammar

6. Recover the error codes from these parses

Steps 1, 3, 4, and 5 are simple enough: dependency parsers can parse unseen data (steps 1 and 5); they can be trained (step 4); and using random sampling, we can vary the proportion of utterances that contain an error (Step 3). The most effective method for Step 2, incorporating the SALT error codes into the dependency parse, is not obvious. In this thesis, we investigate ways of encoding SALT error codes in dependency arc labels. This raises two key challenges: 1) determining which arcs should be labeled, and 2) determining how these arcs should be labeled. How we encode SALT error codes in dependency arc labels determines the range of ways in which the error codes can be recovered. Therefore, each time we propose a different way of encoding the SALT error codes in the dependency parses, we will also discuss how to decide which SALT error codes are predicted in the parser's output (Step 6).

We now present several methods to encode SALT error codes in dependency arc labels. We use the template in Table 6.8 throughout this section and the presentation of our experimental results (Section 6.6.2) to summarize the experimental settings.

Basic algorithm

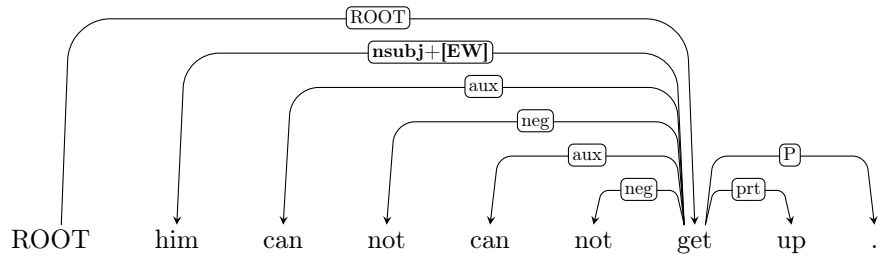


Figure 6.5: Basic error-code augmented parse of 'Him [EW] (can not) can not get up .'

Setting	Description
Parser	Any
Arcs labeled	Arc going into word to left of error code
Label encoding	As itself (ex. `nsubj` becomes `nsubj+EW`)
Recovery	Each augmented arc label corresponds to a single error code

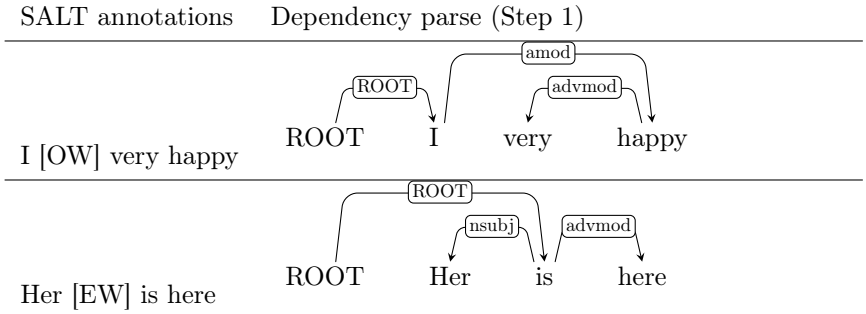
Table 6.9: Basic settings of dependency parse-based SALT error code detection system

A very simple way of encoding SALT error codes in dependency arc labels is by augmenting a single arc label with the appropriate SALT error code. In Figure 6.5, we show such a parse. Specifically, we append the appropriate SALT error code to the label of the arc going into the word to the SALT error code's left. To recover the error codes, we simply look at the predicted arc labels for any that have an error code in them. Table 6.9 shows the settings for the basic algorithm.

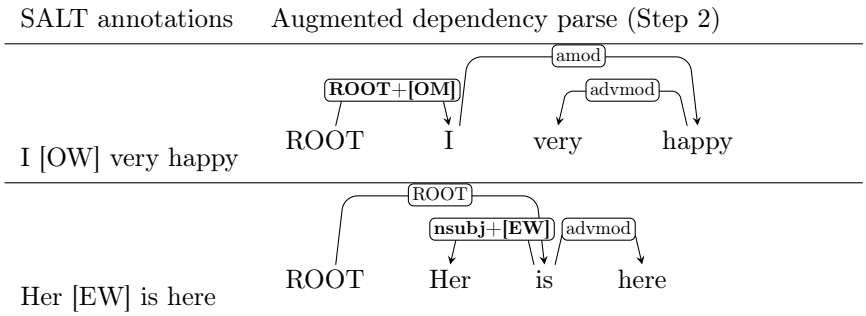
Illustration of basic algorithm

Now that we have presented a concrete method to encode SALT error codes in a dependency parse, let us walk through a toy example of all five steps of the algorithm. In Step 1, we take an existing dependency grammar, and use it to parse utterances that have been annotated with SALT error codes. In all of our experiments, we perform Step 1 with a dependency grammar trained on the Switchboard Treebank (Godfrey et al., 1992). The Switchboard Treebank contains manually produced constituency parses of transcribed conversations. We pre-process the Switchboard Treebank by removing all partial words as well as all words dominated by EDITED nodes. 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. Next, we parse the SALT-annotated utterances with the grammar trained on the Switchboard Treebank:

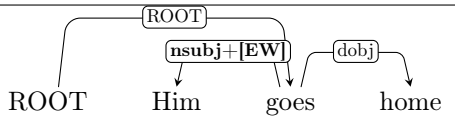
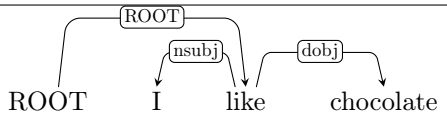


In Step 2, we augment the arc label going into the word to the left of each SALT error code:



In Step 3 we can alter the percentage of utterances that contain an error by randomly selecting utterances from the output of Step 2. In this example we will not remove any utterances, but in our experiments we vary the proportion of utterances with an error in the training data to be 10%, 20%, 30%, ..., 100%. We note that since the vast majority (>80%) of utterances are grammatical in all of the SALT corpora, most of these operating points involve removing grammatical utterances, not ungrammatical ones. In Step 4, we train a dependency grammar on the modified parses that remain after Step 3. In Step 5, we parse new

utterances using this grammar, and in Step 6, we recover the error codes from these parses:

Utterance	Predicted parse (Step 5)	Predicted Error Codes (Step 6)
Him go home		`[EW]`
I like chocolate		None

Getting confidence scores

Parse	Predicted error codes
Parse 1	None
Parse 2	`[EW]`
Parse 3	`[EW]`, `[EW]`
Evaluation	**Confidence Scores**
UTTERANCE	2
ERROR	`[EW-1]`: 2; `[EW-2]`: 1

Table 6.10: Illustration of confidence scores from three parses of utterance: 'Him can go home'

Our dependency-based system does not have an obvious way to produce confidence scores: even if a parser is able to produce them for individual parses, it is not able to do so for specific arc labels. To address this shortcoming, we train 11 models for each experiment, each trained on data with a different percentage of utterances containing an error (10%, 20%, 30%, ..., 100%, along with the original data). For a given utterance, we are then able to produce 11 predictions, one for each model.

Using these 11 models we are able to extract confidence scores for UTTERANCE evaluation quite easily: each prediction of an error contributes one point so that the confidence score for each utterance ranges from 0 (no model predicts an error) through 11 (all models predict an error). Producing confidence scores for ERROR evaluation is only slightly more complicated: for each utterance, we count the number of parses in which each error code is predicted. When a particular error code is predicted more than once in a parse, we assign indices to the error codes in each parse (ex. first `[EW]` code or third `[OM]` code) before counting confidence scores. This is illustrated in Table 6.10.

Error code representation: generic error code

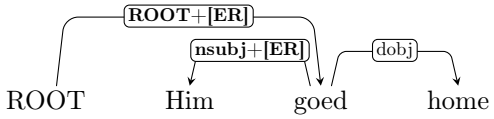


Figure 6.6: Arcs augmented with generic error label: 'Him [EW] goed [EO] home'

Setting	Description
Parser	Any
Arcs labeled	Arc going into word to left of error code
Label encoding	All errors labeled as `[ER]`
Recovery	An augmented arc label means the utterance contains a SALT error code

Table 6.11: Generic error code settings of dependency parse-based SALT error code detection system

In the basic algorithm, we augment a single arc with the error code itself. This approach has the advantage of encoding specific error codes, but if we are only trying to identify utterances that contain an error code, then we only need a generic error code. Specifically, in Step 2, we can augment arc labels with a generic error code (say '+ER'), as illustrated in Figure 6.6 rather than

with specific SALT error codes. This configuration is presented in Table 6.11. Using generic error codes in Step two has the advantage of alleviating data sparsity relative to using specific error codes because there are many more arc labels to learn than with a single generic error code, and therefore there will be more observations per arc label when we use a single generic error code. The disadvantage of this method is that it precludes identifying specific SALT error codes.

Error code representation: concatenated error codes

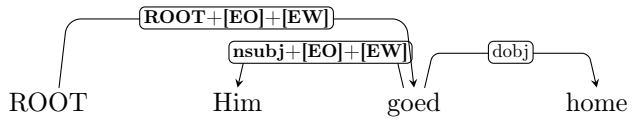


Figure 6.7: Arcs augmented with concatenated error label: 'Him [EW] goed [EO] home'

Setting	Description
Parser	Any
Arcs labeled	Arc going into word to left of error code
Label encoding	All errors codes in utterance are concatenated into a single label used throughout the utterance
Recovery	Each constituent label within an augmented arc label corresponds to a single error code

Table 6.12: Concatenated error code settings of dependency parse-based SALT error code detection system

It is possible that there are interactions between errors in utterances with multiple errors (ex. 'Him [EW] goed [EO] home'), and therefore augmenting arc labels with concatenated error codes may improve system performance relative to the basic arc labels, particularly in terms of Error level performance on utterances with multiple errors. We construct the concatenated error labels without repeated errors. We sort the error codes alphabetically to avoid introducing unnecessary data sparsity (we see no reason to learn the labels 'nsubj+[EO]+[EW]'

and 'nsubj+[EW]+[EO]' separately). Table 6.12 contains this configuration.

Augmenting multiple arc labels: NEIGHBOR-N

Setting	Description
Parser	Any
Arcs labeled	Arc going into word to left of error code and N neighboring or all words
Label encoding	Basic, generic, or concatenated
Recovery	Dependent upon arc label encodingl

Table 6.13: NEIGHBOR-N error code settings of dependency parse-based SALT error code detection system

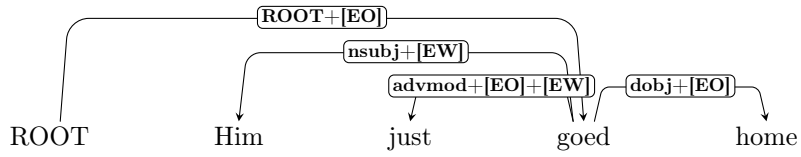


Figure 6.8: Arcs augmented with NEIGHBOR-1: 'Him [EW] just goed [EO] home'

By augmenting more than a single arc label with a SALT error code we increase the number of augmented arcs in the training data thereby reducing data sparsity. On the other hand, augmenting multiple arc labels with error codes will likely result in decreased precision as a result of overpredicting augmented arc labels at test time. We propose a simple procedure for augmenting multiple arc labels with a SALT error codes, which we will refer to as NEIGHBOR-N: augment the arcs going into the N words to the right and left of the SALT error code. We consider values of N from 1 through 5, and NEIGHBOR-ALL, in which we augment all arc labels in utterances with a SALT error code.

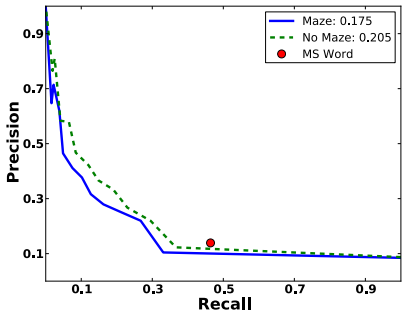
Following the NEIGBOR-N procedure with either generic or concatenated encoding schemas illustrated in Figures 6.6 and 6.7, respectively, is straightforward because the error codes are encoded in the same way in every arc label. If, however, we wish to extend the baseline encoding schema in which arc labels

are augmented at most with one error code, then we must decide what to do in utterances with multiple SALT error codes. As illustrated in Figure 6.8, we propose that if an arc is near enough to more than one error code, then it should be augmented with a concatenation of the relevant error codes, and that other arc labels should be augmented with a single error code. As with the concatenated encoding schema, if an arc label is to be augmented with multiple error codes, we remove duplicate error codes and sort them alphabetically before augmenting the arc label. The configuration for NEIGHBOR-N encoding is presented in Table 6.13.

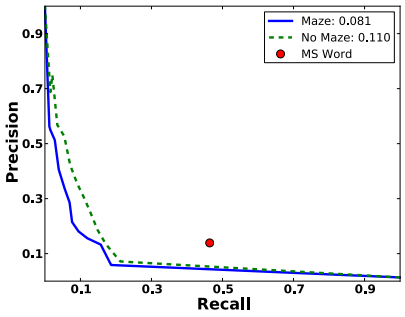
6.6.2 Results

We first use the basic algorithm to answer two key questions: 1) is this method for detecting SALT error codes robust to mazes?, and 2) are some parsers better than other for this task? As detailed below, we find that neither the presence of mazes nor the choice of parser has much of an impact on error code detection performance. The finding about mazes is particularly important: there is no need to perform maze detection as a preprocessing step to SALT error code detection.

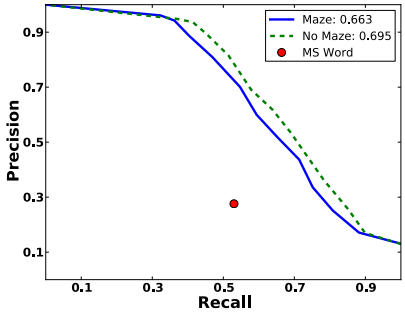
After the initial experiments, we investigate the utility of different ways of representing error codes in the dependency parses. We perform these experiments on data that includes mazes, and the parser we use is MaltParser (Nivre, 2003) with the default settings. We selected MaltParser for these experiments because it is far faster to train than Zpar, and RedShift was unable to train a model in certain cases (specifically on the NZSR corpus with 40% or more of the utterances containing an error).



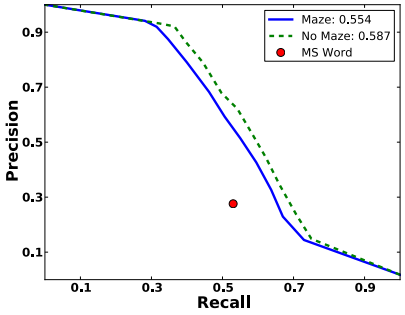
(a) Conv - Utterance performance



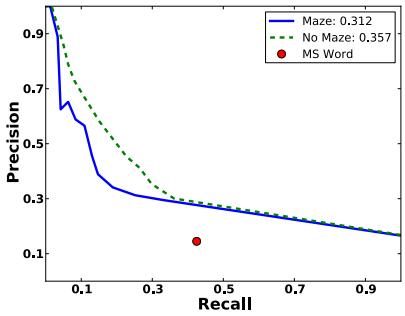
(b) Conv - Error performance



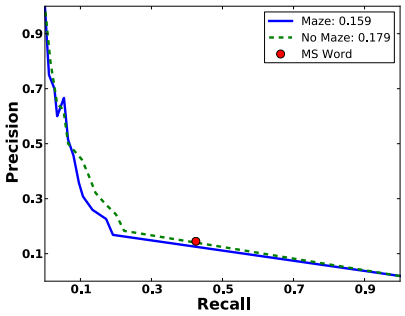
(c) ENNI - Utterance performance



(d) ENNI - Error performance



(e) NarSR - Utterance performance



(f) NarSR - Error performance

Figure 6.9: Dependency-based detector performance on SALT development folds: varying the presence of mazes

Setting	Description
Parser	MaltParser with default settings
Arcs labeled	Arc going into word to left of error code
Label encoding	As itself (ex. `nsubj` becomes `nsubj+EW`)
Recovery	Each augmented arc label corresponds to a single error code

Table 6.14: Settings of dependency parse-based SALT error code detection system for maze detection experiments

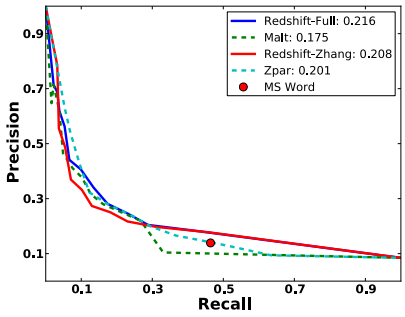
Basic algorithm: Do mazes matter?

We train the error detection system with the configuration described in Table 6.14 (the basic algorithm) on two versions of each SALT corpus: one with mazes present, and the other with all manually identified mazes excised. We do not vary the presence of mazes between the training and development folds; models trained on data with mazes present are only ever tested on development folds with mazes present, and models trained on data with mazes excised are only ever tested on development folds with mazes excised.

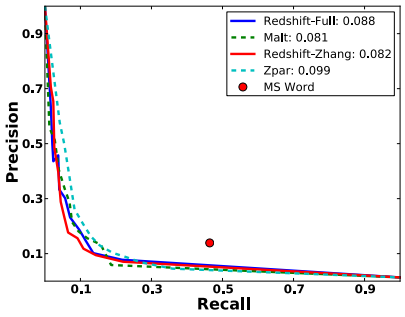
The results of these trials are shown in Figure 6.9. We see that the presence or absence of mazes does not substantially affect system performance for most corpora, either in terms of Utterance or Error evaluation. Given these results, we will not perform an in-depth investigation of how to combine error code and maze detection.

Basic algorithm: Does the choice of parser matter?

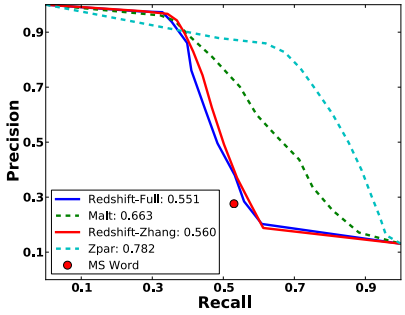
We evaluate the effectiveness of several parsers for SALT error code detection following the configuration in Table 6.15. Specifically, we use: Malt parser (Nivre, 2003), Redshift (Honnibal et al., 2013), and ZPar (Zhang and Clark, 2011). We only evaluate the default settings of Malt Parser and ZPar, but we try two



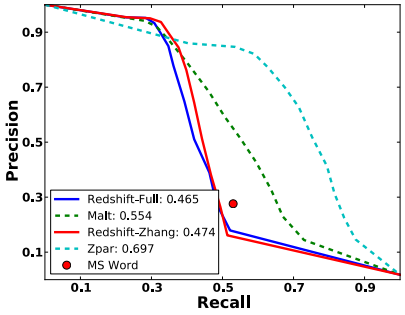
(a) Conv - Utterance performance



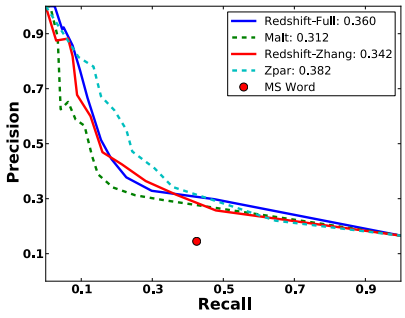
(b) Conv - Error performance



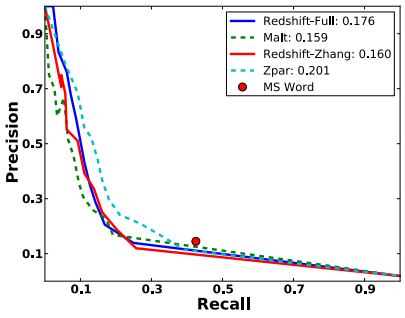
(c) ENNI - Utterance performance



(d) ENNI - Error performance



(e) NarSR - Utterance performance



(f) NarSR - Error performance

Figure 6.10: Dependency-based detector performance on SALT development folds: different parsers

Setting	Description
Parser	MaltParser and ZPar with default settings, Redshift with ZHANG and FULL feature sets
Arcs labeled	Arc going into word to left of error code
Label encoding	As itself (ex. `nsubj` becomes `nsubj+EW`)
Recovery	Each augmented arc label corresponds to a single error code

Table 6.15: Settings of dependency parse-based SALT error code detection system for comparing different parsers

settings for Redshift: the ZHANG feature set, and the FULL feature set, both of which are described in Section 4.4.

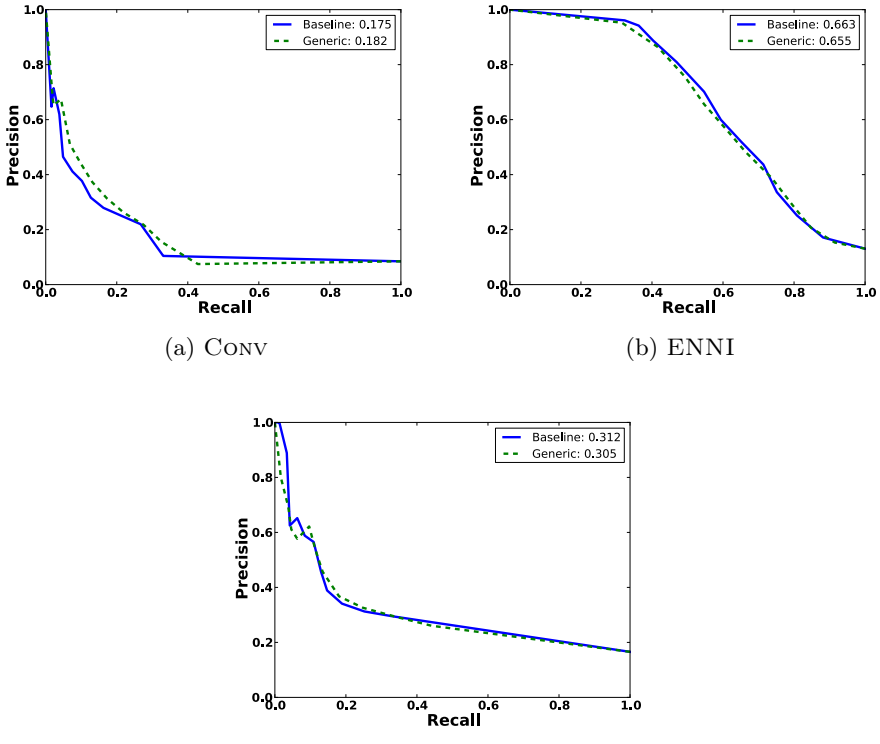
Figure 6.10 shows the ERROR and UTTERANCE performance of the error detectors built with each of these parsers on several SALT corpora. These plots illustrate that while the choice of parser does have an effect upon SALT error code detection performance, none of the parsers yields better performance than the others on all corpora. Looking at Figure 6.10, we see that Zpar is the best parser by a wide margin for ENNI and by a smaller one onNARSR. Zpar's performance on CONV is similar to the other parsers' even though it is the second lowest. We perform our remaining experiments with MaltParser because it is the fastest to train, and its performance is on the whole comparable to the other parsers'.

Error code representation: generic error codes

We now evaluate our error detection system with generic error codes, following the settings in Table 6.16. This system is unable to identify specific errors, and therefore it can only be evaluated with UTTERANCE evaluation, and not ERROR evaluation. In Figure 6.11, we see that using generic error codes does not have much of an effect on system performance.

Setting	Description
Parser	MaltParser with default settings
Arcs labeled	Arc going into word to left of error code
Label encoding	All errors as `[ER]`
Recovery	Each augmented arc label corresponds to a single error code

Table 6.16: Settings of dependency parse-based SALT error code detection system for generic error code experiments



(a) Conv (b) ENNI

(c) aNarSR

Figure 6.11: Dependency-based detector Utterance performance on SALT development folds: generic error codes

Error code representation: concatenated error codes

Setting	Description
Parser	Any
Arcs labeled	Arc going into word to left of error code
Label encoding	All errors codes in utterance are concatenated into a single label used throughout the utterance
Recovery	Each constituent label within an augmented arc label corresponds to a single error code

Table 6.17: Concatenated error code settings of dependency parse-based SALT error code detection system

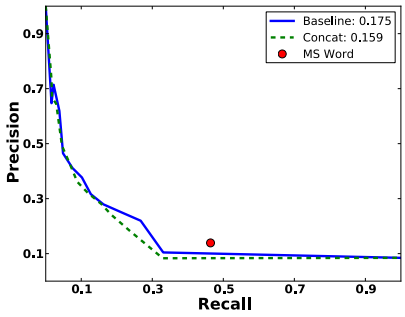
We now evaluate our error detection system with concatenated error codes, following the settings in Table 6.17. In Figure 6.12, we see that using concatenated error codes has hardly any effect on UTTERANCE performance. ERROR performance on ENNI drops relative to the dependency baseline (with the 'basic' encoding schema) when concatenated error codes are used, but it is effectively the same as the baseline for the other two corpora.

Augmenting multiple arc labels: NEIGHBOR-N

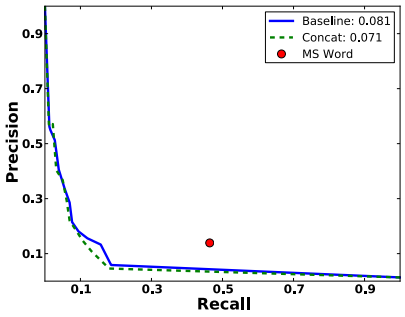
Setting	Description
Parser	Any
Arcs labeled	Arc going into word to left of error code and N neighboring or all words
Label encoding	Basic, generic, or concatenated
Recovery	Dependent upon arc label

Table 6.18: NEIGHBOR-N error code settings of dependency parse-based SALT error code detection system

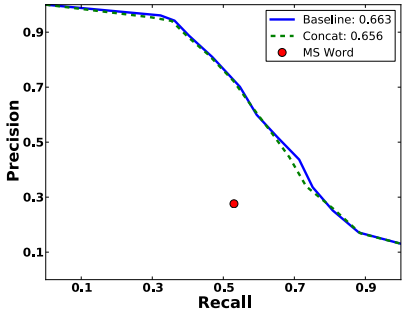
We run the experiments to evaluate the NEIGHBOR-N labeling schemas following the configuration in Table 6.18. Figure 6.13 shows the effect on performance of using the NEIGHBOR-N encoding schema with basic arc error labels (i.e. the arc labels are augmented with specific error codes) for three of the SALT corpora. The most notable effect of the NEIGHBOR-N labeling schema



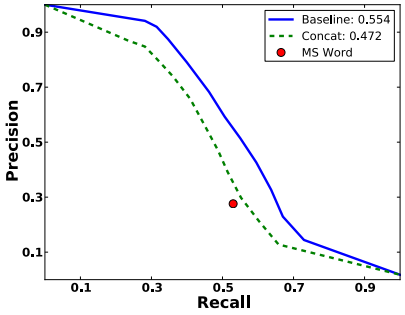
(a) Conv - Utterance performance



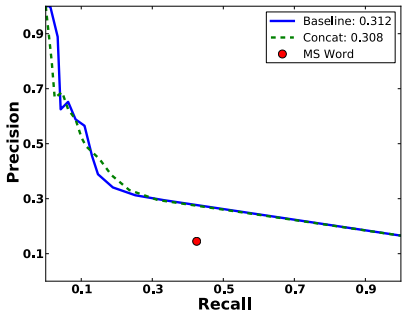
(b) Conv - Error performance



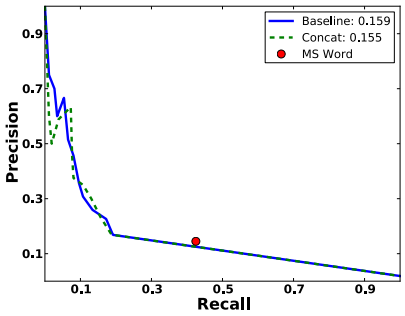
(c) ENNI - Utterance performance



(d) ENNI - Error performance

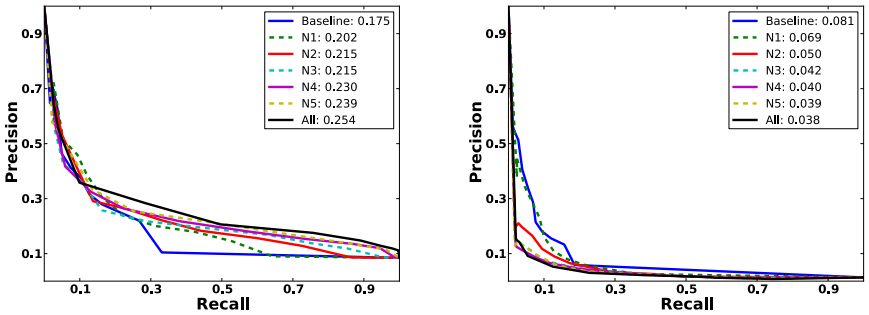


(e) NarSR - Utterance performance



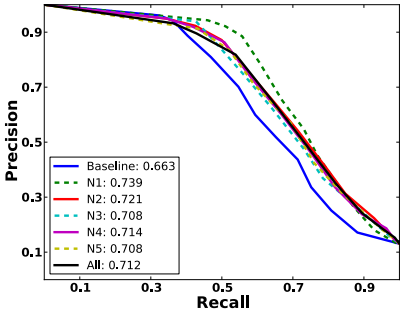
(f) NarSR - Error performance

Figure 6.12: Dependency-based detector performance on SALT development folds: concatenated error codes

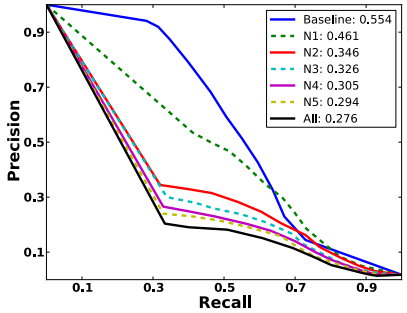


(a) Conv - Utterance

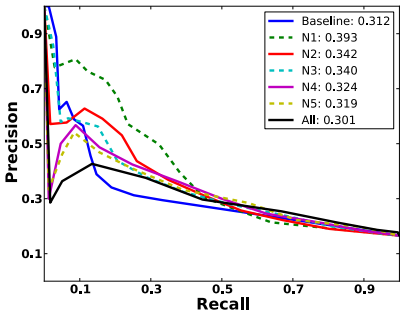
(b) Conv - Error



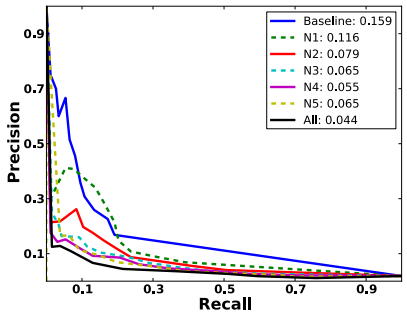
(c) ENNI - Utterance



(d) ENNI - Error



(e) NarSR - Utterance



(f) NarSR - Error

Figure 6.13: Dependency-based detector Utterance performance on SALT development folds: Neighbor-N labeling schema with basic error arc labels

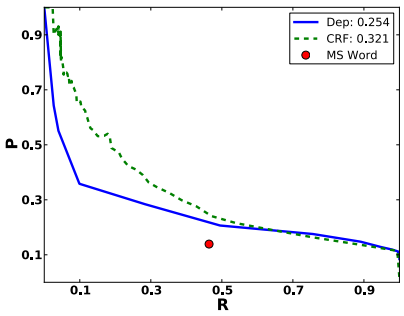
is to enable high-recall operating points to have precision above the minimum level (i.e. only predicting errors). In terms of UTTERANCE evaluation, this does not come at the expense of precision at any given values of recall. This is not the case for ERROR evaluation, however, where all of the NEIGHBOR-N systems have lower precision than the baseline at given values of recall.

Comparing the dependency-based system to the tagger

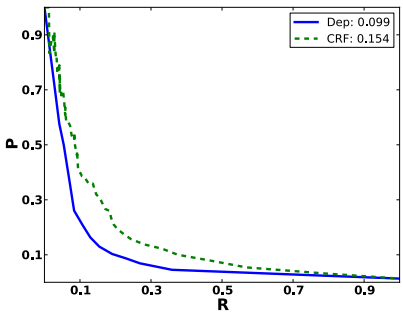
Figure 6.14 compares the performance of the dependency-based algorithm with the baseline systems, all detecting SALT error codes with mazes present. The results for the dependency-based algorithm in Figure 6.14 are from the highest performing systems for each corpus, noted alongside each plot. First, we see that on the ENNI corpus, the dependency-based detector clearly outperforms the tagger both in terms of UTTERANCE and ERROR performance. All of the systems perform substantially better on the ENNI corpus than on the CONV and NARRATIVESTORYRETELL corpora. For these two corpora, we find that the CRF-based tagger outperforms the dependency-based error detector both in terms of UTTERANCE and ERROR evaluation. Finally, we note that the Zpar baseline system is the best performing of all the dependency-based systems in terms of AUC on ERROR evaluation for all three corpora, even though it still performs worse than the CRF-based tagger in two of these three corpora.

The effect of labeling schemas on the number of arc labels

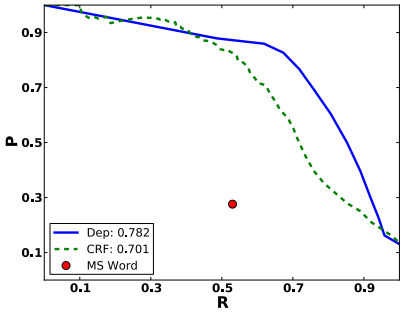
Table 6.19 shows the observed number of arc labels in each of the SALT corpora under each of the labeling schemes examined thus far along with the theoretical maximum number of arc labels, which is simply the number of error labels times the number of unaugmented arc labels. First, we see that in many corpora the specific error labels are somewhat restricted in their distribution, relative to arc labels: in many corpora the observed number of arc labels under the



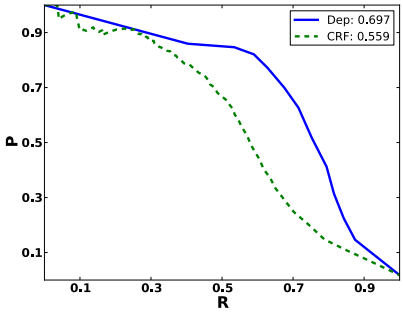
(a) Conv - Utterance performance, Neighbor-All



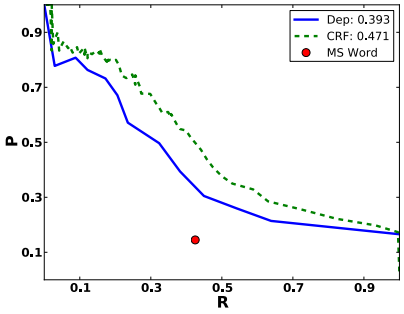
(b) Conv - Error performance, Zpar baseline



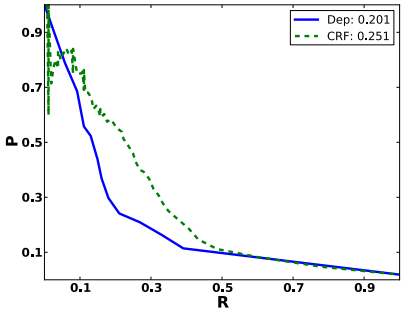
(c) ENNI - Utterance performance, Zpar baseline



(d) ENNI - Error performance, Zpar baseline



(e) NarSR - Utterance performance, Neighbor-1



(f) NarSR - Error performance, Zpar baseline

Figure 6.14: Comparison of dependency-based algorithm to CRF tagger and MS Word

	N	N	Basic		Generic		Concatenated	
Corpus	Codes	Labels	Max	Obs	Max	Obs	Max	Obs
CONV	6	45	270	229	90	87	2,880	409
ENNI	4	45	180	139	90	85	720	196
EXPOSITORY	5	45	225	118	90	78	2,880	141
NARSSS	6	45	270	184	90	85	2,880	274
NARSR	6	45	270	173	90	84	2,880	250
NZCONV	6	45	270	170	90	80	2,880	234
NZPERNAR	6	45	270	167	90	84	2,880	227
NZSR	6	41	246	109	82	68	2,624	130

Table 6.19: Arc label counts under different labeling schemas

'basic' labeling schema is around 60-70% of the theoretical maximum. Not surprisingly, the generic error labels are more widely distributed. We attribute the massive discrepancy between the theoretical maximum and observed counts of arc labels under the 'concatenated' schema to the fact that many of the possible combinations of error codes simply do not occur. For example, CONV has 28 concatenated error codes, which is the highest of any of the SALT corpora, but still far lower than the $2^6 - 1 = 63$ concatenated error codes possible in that corpus.

Finally, we note that it is somewhat surprising that all of the schemas for encoding error codes in dependency arc labels yield similar performance. Simply based on the number of arc labels to learn, one would expect the 'generic' labeling schema to yield the best UTTERANCE performance, and the 'basic' labeling schema to yield the best ERROR performance, but this is not always the case.

6.6.3 Conclusions

In this section we have seen that dependency grammars can be trained to identify SALT error codes. This approach tends to outperform tools developed

for formal, written language, namely Microsoft Word and the ETS e-rater. Like the CRF tagger, the dependency-based method is agnostic to the set of error labels used, and is robust to mazes. Although it has an operating point that can be manipulated by varying the proportion of training utterances that contain an error, it does not output a confidence score. We have proposed a way to combine models, which enables us to rank outputs, although control over the trade-off between precision and recall is less fine-grained than with systems that produce true confidence scores. In most cases the CRF tagger outperforms the dependency-based method, but there are some instances (ENNI - UTTERANCE) where the opposite is true. This suggests that there is some merit to the dependency approach, even if it is not appropriate for every corpus.

6.7 System combination: tagging- and dependency-based error detectors

The tagging- and dependency-based methods for SALT error code detection outperform all of the other methods we explore (including the random walk-based error detector explored below in Section 6.8). Neither the tagging- nor the dependency-based system clearly outperforms the other, which suggests model combination. We combine these two systems by incorporating the predictions of the dependency-based error detector into features used by the CRF tagger.

6.7.1 Methods

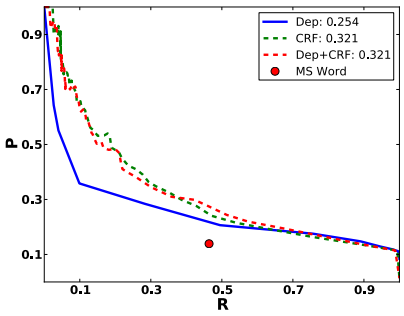
We add the possibly-augmented arc label output by the dependency-based error detector to the original feature set used by the CRF tagger. While we experimented with other features, for example arc label bigrams and POS tags combined with arc labels, we found using arc labels alone to be equally effective.

Furthermore, we found that using the augmented arc labels yielded slightly higher performance than simply using the predicted error code. For reference, the original feature set used by the CRF tagger is shown in Table 6.7.

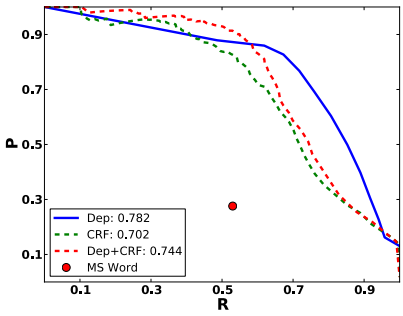
Using augmented arc labels as a feature complicates training: we cannot train a CRF tagger using the usual version of this feature in our training sets because they contain the true error code, which is the label we are trying to predict. Furthermore, when we extract features in testing, we will not find dependency arcs augmented with true error codes, but rather the augmented arc labels predicted by the dependency-based error detector. We address this issue by using arc labels predicted using cross-validation as features in training. Specifically, we partition the training set of a particular corpus into ten folds, and then train a dependency-based error detection model on nine of these folds. We use the labels predicted by this model as the features for the utterances in the held-out fold. We repeat this procedure for all ten folds.

6.7.2 Results and Conclusions

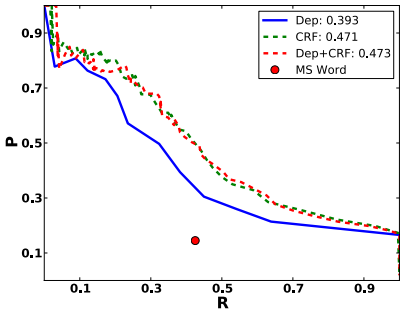
Table 6.15 shows the performance of the CRF tagger that incorporates the output of the dependency-based error detector. We see that combining these systems improves performance in some cases, but not across the board. For example, the joint system allows higher levels of precision on ENNI corpus at levels of recall below about 0.6 compared to either the CRF tagger or dependency-based error detector alone, but performance on the Conv and NarSR corpora is essentially the same as the original CRF tagger. Based on these preliminary results, it appears that combining these two systems can be fruitful, but not dramatically or reliably so.



(a) Conv - Utterance



(b) ENNI



(c) NarSr - Error

Figure 6.15: Performance of CRF tagger incorporating features from dependency-based error detector on the development folds of several SALT corpora. Mazes included, Utterance evaluation.

6.8 Random Walk-Based Error Detection

We now introduce a random walk-based algorithm to identify SALT error codes. This method, like our classifier, tagger, and dependency-based error detection methods, is data-driven, and is agnostic to the set of error codes. Like the dependency-based method, the random walk-based method allows us to capture more complicated relations between words in an utterance than does the tagger, which simply treats each utterance as a sequence of words, or the classifier, which essentially treats each utterance as a bag of words and bigrams. The random walk-based method also addresses the main weakness of the dependency-based method, namely its inability to produce confidence scores.

6.8.1 Methods

We begin by describing our basic random walk-based algorithm for SALT error code detection. We then describe some variants of this method, including ones that have information from dependency grammars trained to identify SALT error codes encoded in the graph structure. For simplicity, we begin with methods to detect utterances that contain any error codes. We then propose a method to identify specific error codes, each with accompanying confidence scores.

Basic Algorithm

Our basic algorithm for random walk-based SALT error code detection is as follows, and we illustrate the first two steps with the SALT-annotated utterance 'Him [EW] can go home':

1. Convert the utterance into a graph in which each node represents a single word. In the basic algorithm, the graph is unweighted, undirected, and fully connected, as shown in Figure 6.16.

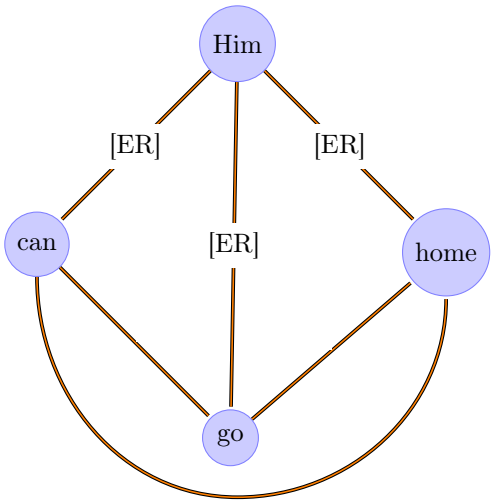


Figure 6.16: Basic graph representation of utterance 'Him [EW] can go home'

2. Label edges in the graph with any SALT error code in the utterance. In the basic algorithm, for each SALT error code e in the utterance, we label any edge incident to the node representing the word to the left of e (in the original SALT-annotated utterance) as an error edge. For example, in Figure 6.16, we label the edges incident to the node labeled 'Him' with `[ER]`.

3. Take a random walk of N steps, beginning at any node. At each step:

   (a) If training with an offline algorithm (ex. SVM), extract features, and store these along with the true arc label. Offline training is not explored in this thesis.

   (b) If training with an online algorithm (ex. perceptron), extract features, predict whether or not the edge just crossed has an error label, and update the classifier.

   (c) If testing, extract features and use a classifier to predict whether or

not the edge just crossed has an error label. Store this prediction. At the end of the random walk, we return the number of times an error was predicted. This number ranges from 0 (least likely to have an error) to N (most likely to have an error).

The basic algorithm described above can be elaborated upon in many ways. First, in Step 1, we could construct a directed or weighted graph, or one that is both directed and weighted. In Step 2, we can vary both which edges we label, and how we label them. In Step 3, we have a choice of what features to extract, and of which classifier to use. We explore variations of all of these, aside from the choice of classifier, and they are described below.

Throughout our experiments, we use an averaged perceptron (Collins, 2002) for classification. We selected this algorithm due to its ability to handle large numbers of features, as well as the fact that it is an online learning algorithm, which simplifies training in practice because we do not need to extract features from every step of every random walk before training. Averaged perceptrons also perform well on a wide variety of tasks in NLP, including part of speech tagging (Spoustová et al., 2009) and dependency parsing (Honnibal et al., 2013). For more information on perceptrons, please see Section 4.1.

As was the case with the CRF tagger, we can vary the proportion of utterances in the training data that contain an error. In all of our experiments, we train on data in which 10%, 20%, 30%, ..., 100% of the utterances have an error. For the sake of clarity, we only report results from the training data that yields the highest AUC for each corpus.

We evaluate the random walk-based detector at different confidence levels θ, which in our experiments range from 0 and 100, because we set the length of the random walk (N) to be 100 in all of our experiments. We then count utterances with a score of at least t as containing an error, and utterances with a score

below t as not containing an error. UTTERANCE evaluation then proceeds as described in Section 6.2.

Detecting Specific Error Codes

Most of our experiments in this section are only designed to identify utterances that contain an error. For many scenarios, for example reducing an annotator's workload by filtering out utterances that do not contain any errors, or getting a sample of the 'worst' utterances to see what kinds of error the child produces, this is sufficient. Nevertheless, it may be the case that one wants to predict specific error codes. In this case, we modify the basic algorithm to label 'error edges' (see Figure 6.16) with specific error codes rather than a generic error label. The averaged perceptron classifier can perform multiclass classification, and therefore we simply train it to predict the label of the edge just crossed.

For evaluation, we keep track of c confidence scores where c is the number of different error codes in the data. The extra confidence score is for 'no error code'. We also need to set c thresholds for evaluation. Then, whenever the confidence score for a particular error code meets or exceeds that threshold, we count the utterance as containing that particular error code.

Using the random-walk based algorithm to predict *counts* of individual error codes is not straightforward, and precludes us from evaluating the random walk-based system's ERROR performance. To illustrate, consider again the utterance 'him [EW] can go home', and its graph representation (see Figure 6.16 on page 207). If an `[EW]` code is predicted crossing the edge between 'him' and 'go', and then again between 'home' and 'him', should these contribute towards a single predicted `[EW]` code or multiple `[EW]` codes? Now suppose that a particular edge is traversed several times during the walk, and each time a different error is predicted. If we can uniquely map this edge to a word, should we predict a long sequence of error codes, which is potentially unrealistic (ex. 'Him [EW] [EU]

	Individual error codes		
Gold error codes:	[EW]	[EW]	
Gold set error codes:	[EW]		
Predicted error codes:	[EW]		[OW]
ERROR Evaluation:	TP	**FN**	FP
ERROR-SET Evaluation:	TP	**TN**	FP

Figure 6.17: Illustration of ERROR-SET evaluation
TP = true positive; FP = false positive; FN = false negative

[OM] can go home')? If we cannot map this edge to a particular word, should we allow more error codes to be predicted than there are words in the utterance?

Although these issues are beyond the scope of this thesis, we do still evaluate how effectively the random walk-based method can predict specific error codes. We do this by removing duplicate error codes from manual annotations before performing ERROR evaluation. We call this form of evaluation ERROR-SET, and it is contrasted with ERROR evaluation in Figure 6.17.

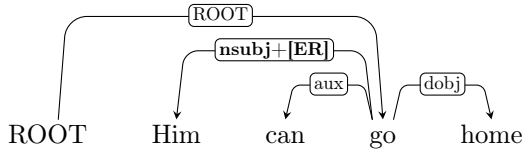
Features

During the random walk, we extract features from the current word and the previous h words in the history of the random walk. We experiment with two sets of features: basic features, and features that depend upon a dependency parse using a grammar trained to identify SALT error codes, as described in Section 6.6. These features are illustrated in Figure 6.18.

Incorporating dependency features requires a more complicated training procedure. To see why this is so, consider what happens with the 'error label' features if we train our detector on gold features and gold labels. Recall that the error label features are simply the SALT error codes that are appended to dependency arc labels. The gold error label feature is then a highly reliable indicator of whether the arc just crossed should be labeled True or False (as

Category	Feature	Example features
Basic	Word	him; can
	Part of speech	PRP; MD
Dependency	Head word	go; go
	Part of speech	MD; MD
	Arc label	nsubj; aux
	Error label	[ER]; none
Labels		
False; True		

(a) Features and labels in random walk-based error detection. Examples are from a random walk after two steps: 'can', 'him'.



(b) Dependency parse using a dependency grammar for SALT error code detection; generic error code

Figure 6.18: Description of features along with examples using the utterance: 'Him [EW] can go home'

an error arc, or not). When testing, however, the error labels are not the gold SALT error codes attached to dependency arcs. Instead, they are the error codes predicted by a grammar. They are therefore substantially less reliable than the gold error arc labels. Similar problems affect the other dependency features.

To address this issue, we use cross-validation to produce dependency parses with predicted error codes:

1. Split the training data into 10 folds of equal size. The training data contains both manual SALT annotations and dependency parses (from a grammar trained on, say, Switchboard)

2. For each fold f:

   (a) For all utterances *not* in f, augment the arc labels with the generic

error label `[ER]` based on the manual SALT annotations.

(b) Train a grammar on these augmented parses.

(c) Use the resulting grammar to parse the utterances in f.

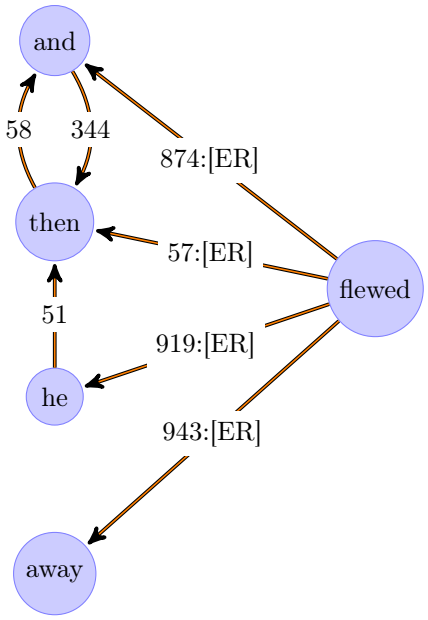
(d) Extract features from the parses of f, and labels from the manual SALT annotations.

Varying Arc Weights

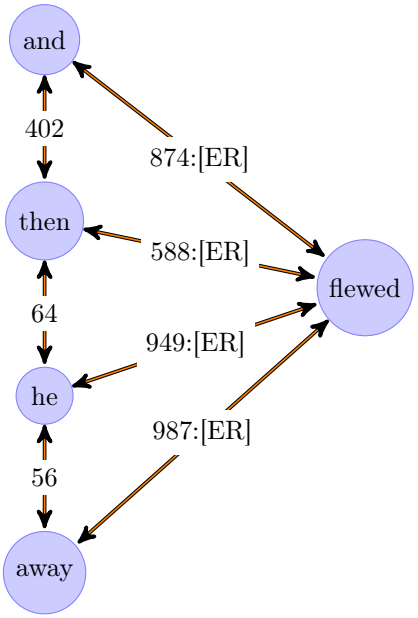
By using a fully connected, unweighted graph, we ensure that no pair of words is precluded from consideration during the random walk. On the other hand, not all pairs of words are equally tightly coupled. Adding weights to the graph encoding of an utterance captures the intuition that some pairs of words are more tightly coupled than others. For example, words that are nearer to each other in a dependency parse are typically more tightly coupled than words that are far apart in a dependency parse. We expect "closer" pairs of words to be more relevant for identifying grammatical errors than more distant ones. For example, in 'Him [EW] can go home', 'him' and 'go', which are adjacent in the correct dependency parse, and this pair of words is key to identifying the grammatical error. On the other hand, 'him' and 'home' are farther apart, and do not seem to be useful for identifying the grammatical error.

To assign weights to each arc in the graph representing utterance u, we first get the k-best dependency parses of u. Then, the raw weight of the arc (a, b) from word a to word b is simply the number of times that the arc (a, b) appears in the k-best dependency parses of u.

We explore two types of weighted graphs, namely directed and undirected weighted graphs. To construct the directed weighted graph, we simply use the raw arc weights. To construct the undirected weighted graph, we set the weight of the edge between word nodes a and b to be the raw weights of (a, b) and (b, a)



(a) Directed weighted graph



(b) Undirected weighted graph

Figure 6.19: Weighted graphs from 1000 best parses of 'And then he flewed [EW] away' (from ENNI). Edges with weight < 50 not shown.

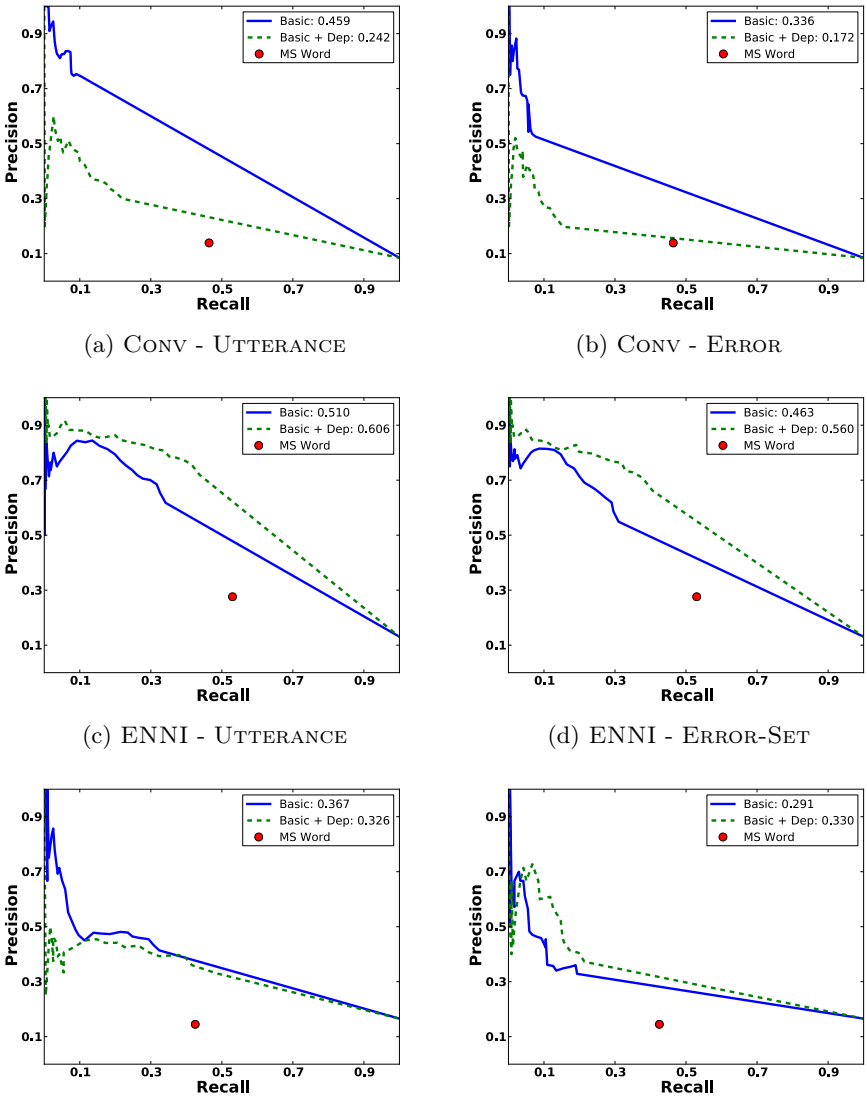
added together. To account for the fact that not all words will be connected in the resulting graph, including ones that possibly should be, we smooth the observed frequencies with Laplacian (add-one) smoothing. Thus, all of the words are connected in the resulting graph, which also prevents the random walk from getting stuck at any point. Figure 6.19 illustrates both the directed and undirected weighted graphs, albeit with low-weight edges hidden for the sake of clarity.

6.8.2 Results

We now present from our results with the random walk-based error detector. Recall that we train and test 11 models for each corpus as we vary the proportion of utterances in the training data that contain an error. For the sake of clarity, we do not present results from each model. Instead, for each configuration, we present a single curve for each corpus, specifically the one with the highest area under the precision/recall curve.

Basic algorithm: basic features

We first evaluate graph error detector using only the basic features, namely words and part of speech tags, with a history length of two (i.e. extracting features from only the current word and the previous word in the random walk). As in all of our experiments, the random walk in each utterance is 100 steps long, and therefore the confidence scores for each utterance are between 0 (least likely to contain an error) and 100 (most likely to contain an error). We see the performance of the random walk-based error detector change as we vary the threshold t at which we count an utterance as containing an error in Figure 6.20; the left end of each line is with $t = 100$, yielding the highest precision, and the right end is with $t = 0$, yielding the highest recall.



(a) Conv - Utterance

(b) Conv - Error

(c) ENNI - Utterance

(d) ENNI - Error-Set

(e) NarSR - Utterance

(f) NarSR - Error-Set

Figure 6.20: Performance on SALT development folds: graph error detector with basic features alone, and both basic and dependency features

Basic algorithm: adding dependency features

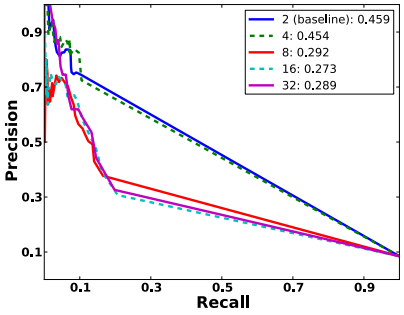
We now add in the dependency features illustrated in Figure 6.18, and again we use a history length of two (i.e. extracting features from only the current word and the previous word in the random walk). Plots comparing the performance enabled by the dependency features as compared to the basic features for several corpora are also shown in Figure 6.20. For some corpora, the addition of the dependency features clearly improves performance. This is the case with ENNI. The dependency features improve ERROR-SET performance on NARSR while degrading UTTERANCE performance. The dependency features degrade performance on the CONV corpus.

Basic algorithm: varying the history length

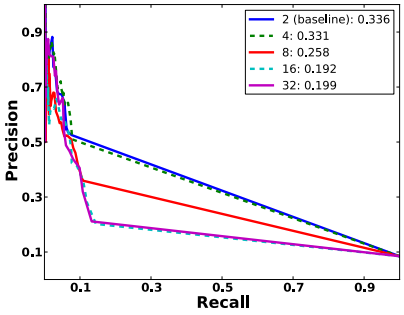
In previous experiments we have used a history length of two, meaning we extract features from the current word and the previous word in the random walk. Here, we evaluate our detector using other history lengths: 4, 8, 16, and 32. For each detector, we use the feature set that yields the better performance of the two we have tried: basic and basic + dependency. Figure 6.21 shows the results of these experiments, noting which feature set is used with each corpus. In most cases, using a longer history length degrades performance, although doing so does open up some higher-precision, lower-recall operating points on the NARSR corpus.

Varying Arc Weights

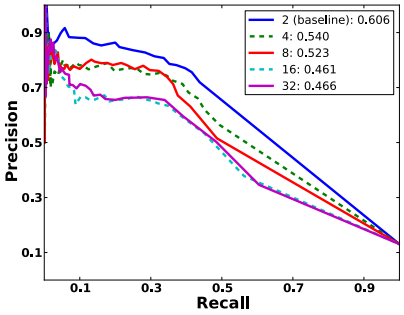
The plots in Figure 6.22 clearly illustrate that using k-best dependency parses to set arc weights degrades performance relative to the baseline in which the arcs are unweighted.



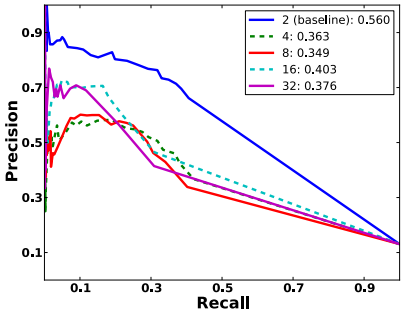
(a) Conv – Utterance – basic features



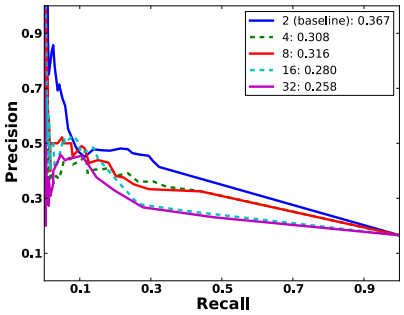
(b) Conv – Error-Set – basic features



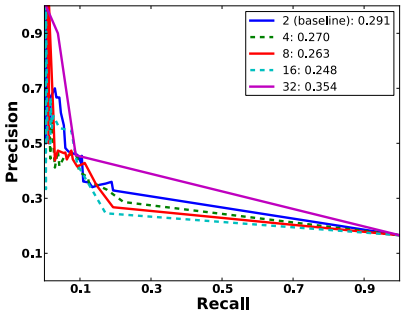
(c) ENNI – Utterance – basic + dependency features



(d) ENNI – Error-Set – basic + dependency features

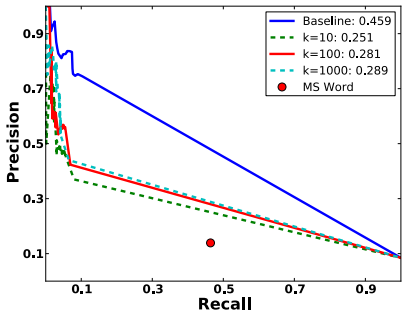


(e) NarSR – Utterance – basic features

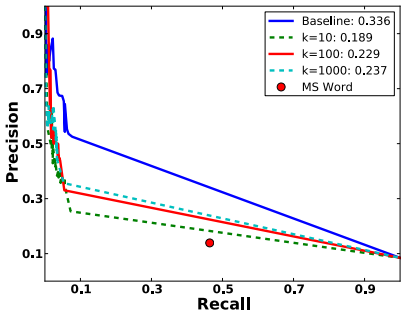


(f) NarSR – Error-Set – basic features

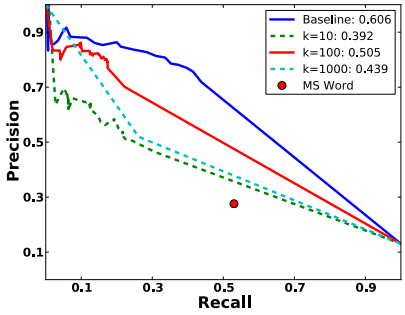
Figure 6.21: Performance on SALT development folds varying the history length: 2 (baseline), 4, 5, 10, 20



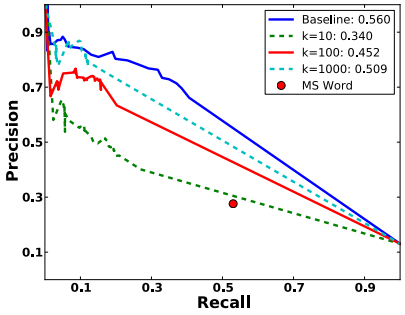
(a) Conv - Utterance



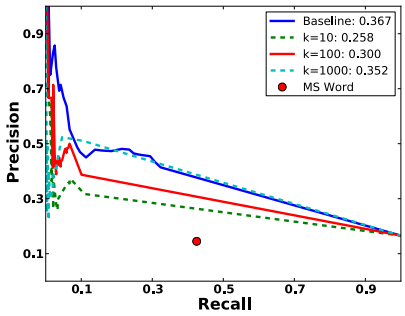
(b) Conv - Error-Set



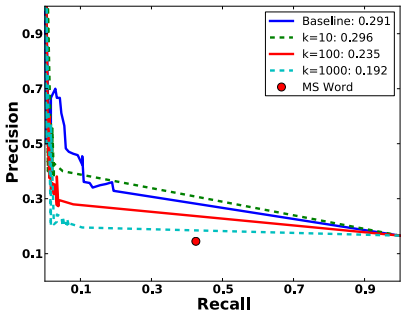
(c) ENNI - Utterance



(d) ENNI - Error-Set



(e) NarSR - Utterance



(f) NarSR - Error-Set

Figure 6.22: Performance on SALT development folds: using k =10, 100, 1000 best dependency parses to set arc weights

6.8.3 Conclusions

Like the CRF tagger (and the poorly-performing utterance classifier) the random-walk based error detector presented in this section is able to produce confidence scores for each utterance. It does so more naturally than the dependency-based method, which requires multiple models to assign a confidence score to an utterance (see Section 6.6.1). Unlike these two methods, however, it is unable to predict multiple error codes per utterance. Furthermore, its performance is lower than both the CRF tagger and the dependency-based method on all three corpora.

6.9 Error Analysis

In this chapter we have explored many variants of four systems for identifying utterances with SALT error codes (a classifier, a CRF tagger, the dependency-based system, and the random walk-based system) and we have evaluated it on the development folds of seven SALT corpora. Here we look at the errors produced by the baseline dependency-based system using Zpar on the ENNI corpus. We selected this system because its UTTERANCE performance is nearly the best, and its ERROR performance is quite good as well (see Figure 6.14). We set a minimum threshold of 9 for counting predicted errors, and we chose this threshold because it gives the highest UTTERANCE F1 score. Concretely, its UTTERANCE performance is P=0.83, R=0.67, and F1=0.74, and its ERROR performance is P=0.82, R=0.59, F1=0.69.

First, we re-annotated the first 3,000 utterances in the development portion of the ENNI corpus at the utterance level, indicating which ones contain an error, and which do not. All utterances that we marked as containing an error contain at least one *overt* error, namely one that can be identified by

looking at that utterance without any context (see Section 6.1.1). We did not re-annotate the individual error codes. Although the ENNI documentation contains descriptions of the error codes used in the corpus, we were not trained alongside the annotators. As a result, we do not feel qualified to redo the error codes; identifying ungrammatical utterances, however, is much more straightforward. After re-annotation, we find that UTTERANCE performance is P=0.71, R=0.69, F1=0.70. The small decrease in precision and increase in recall both have the same cause: utterances being re-classified as not having an error, likely due to the error being covert.

Category	Example	Count	TP	FP	FN
A/an substitution	a elephant	33	31	1	1
Negation	And the lifeguard can not do nothing	8	3	0	5
Omitted word	They are going eat outside.	32	22	0	10
Overgeneralization	sitted	46	32	0	14
Pronominal case	Then her put a bandage on.	20	19	0	1
Verb agreement or tense error	Then he give it to the girl.	124	85	3	36
Other/multiple errors	Then the horse went for swimming.	79	40	1	38

Table 6.20: Breakdown of errors in first 3,000 utterances of the re-annotated ENNI development fold. TP means the utterance was identified as having an error during re-annotation and the baseline dependency system using Zpar, and analagously for FN for FN.

Next, we assigned a category to each of the utterances we identified as containing an error. The categories, along with examples and counts in the ENNI development fold are shown in Table 6.20. Some of the errors in Table 6.20 are harder to detect than others: a/an substitution errors and overgeneralization errors can easily be identified by regular expressions and a dictionary, respectively. Our system is quite effective at identifying a/an substitution errors, but less so at identifying generalization errors. Our system appears to be quite effective at identifying pronominal case errors, only missing one of the 18 observed. It

is somewhat less effective at identifying omitted words and verb agreement or tense errors, although it still is able to identify the majority of these.

The major area where our system struggles is with 'other/multiple' errors, which is a true grab-bag. Many of the utterances with an error in this category that our system successfully identifies in fact have several errors that fall under other categories, for example a verb tense error along with an a/an substitution error. Some of these utterances that are missed by our system have an error that we would hope to catch, alongside other, more difficult-to-detect errors. An example of this is 'and the long time he catch it far away [EU] .'. This utterance contains a verb agreement error, but even after that is corrected the utterance does not make any sense and so should receive an `[EU]` code. Finally, some of the 'other' errors that our system misses are lexical errors for example '(uh) they told questions [EW] .'. We expect these errors to be particularly difficult to identify because they are likely to be sparse in the training data.

Of the 3,000 utterances we re-annotated, there were originally 409 that were marked as containing an error. During re-annotation, we identified six utterances that had an unmarked error. We also identified 75 utterances that were originally coded as containing an error, but which we do not believe to contain any errors. The discrepancies in our annotations relative to the original may be because our standards differ from those of the ENNI annotators, which is inevitable given that we were not trained together. For example, we see no reason for the the utterance 'she had to sit on the bench til her knee felt better .' to contain an error code, but the original ENNI annotations include an `[EW]` code after 'til'. Another potential source of disagreement is that we, unlike the original annotators, are looking at utterances out of context. To illustrate:

(6.1) E: Where are the balloons?

C: And the balloons [OW] up in the sky.

(6.2) E: What did you see?

C: A bird.

::pause::

C: And the balloons up in the sky.

Since our detectors only considers an utterance at a time, it is not reasonable to expect it to identify errors that are only clear in context. Although such errors are clearly present in the data, the vast majority of errors that have been manually annotated are overt errors that are identifiable without context.

6.9.1 Other corpora

Original → Revised	ENNI	Conv	NarSR
Error → no error	20	32	73
No error → error	0	44	36
Original error ct	140	93	163
Revised error ct	121	103	125
Overturn rate	2.0%	7.6%	10.4%

Table 6.21: Comparison of original and re-done utterance-level error annotations on first 1,000 utterances in development folds

Throughout this chapter we have observed that most of the systems we have tried perform far better on the ENNI corpus than on either Conv or NarSR. To investigate why this is the case, we re-examine the first 1,000 utterances in the development sections of these two corpora, marking which contain any overt errors. We refrain from re-annotating these utterances with SALT error codes because our standards of when to apply specific error codes are likely to be different due to the fact that we were not trained with the original annotators. Nevertheless, we should agree with the original annotators on utterance-level annotations.

After re-annotating which utterances have an error in the first 1,000 utterances of the development codes of these corpora, we count how many differences we

observed relative to the original annotations. Counts of the differences in annotation are shown in Table 6.21. Specifically, we count how many utterances the original annotators marked as containing an error that we identify as not containing an error (error → no error), and how many utterances have errors that were unmarked in the original annotations (no error → error).

The overturn rates in Table 6.21 are the percentage of utterances where we disagree with the original annotators regarding whether there are any errors in that utterance. We see that ENNI, which is the corpus most amenable to automatic error code detection, has a low overturn rate (2%), while the other two corpora have much higher rates of overturn. These utterances are ultimately a random sample from the three corpora, and therefore there is every reason to believe they are representative of the corpora as a whole. If, as is almost certainly the case, the labels in the training folds are as consistent as the ones we re-annotated, then it is not surprising that all of the systems perform far better on ENNI than on Conv and NarSR: the training fold of ENNI has reliable labels, while the labels used in training on the other corpora are very noisy.

Other plausible explanations for why performance on ENNI is so much higher than on Conv and NarSR fall apart upon closer examination. One potential explanation for the good performance on ENNI is that it is a narrative task and therefore has far more restricted language than conversational corpora. This, however, suggests that performance on NarSR should be comparable, but it is in fact quite low. Another possibility is that many of the errors marked in Conv and NarSR are covert, while those in ENNI are overt. Looking again at Table 6.21, we see that while there are potentially covert errors in all of the corpora (error → no error), there are still far more missed errors in Conv and NarSR than in ENNI.

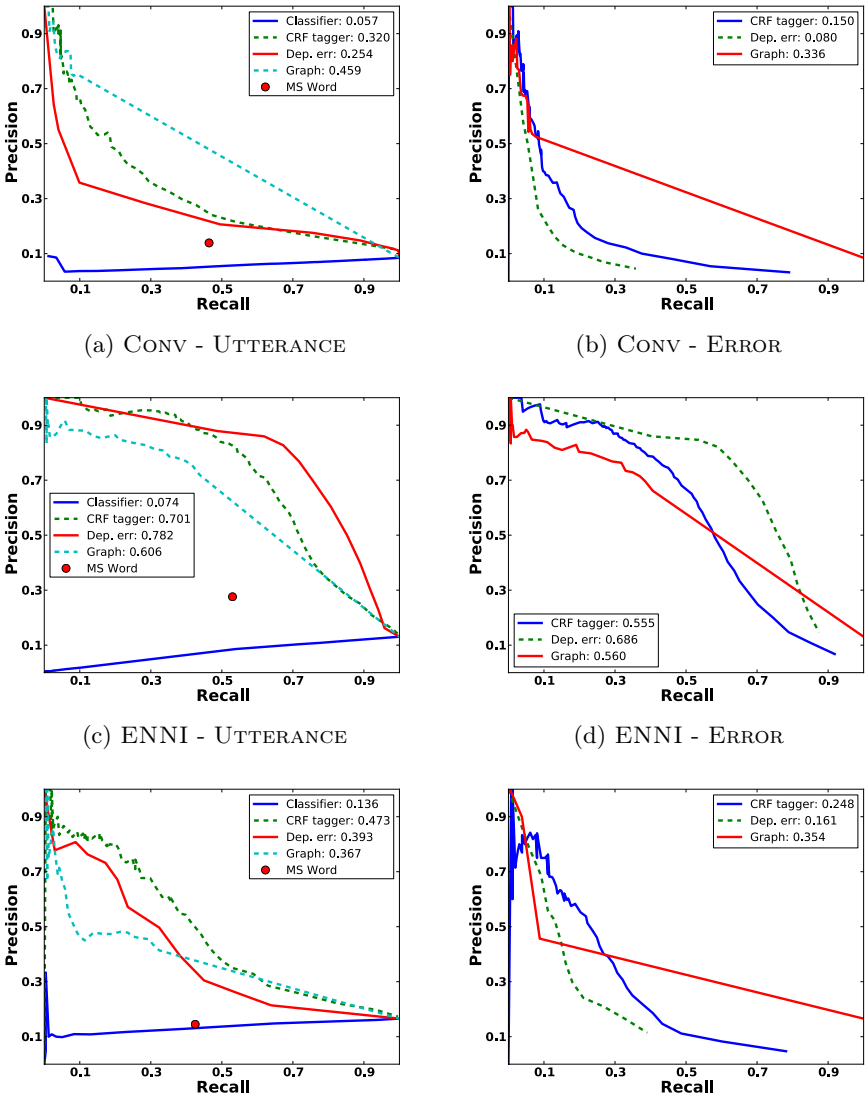
6.10 Conclusions

System	Trainable	Predict specific errors	Confidence scores
MS-Word	✗	✗	✗
ETS e-rater	✗	✗	✗
Classifier	✓	✗	✓
Tagger	✓	✓	✓
Dependency	✓	✓	Using multiple models
Random walk	✓	Yes, but not repeated ones	✓

Table 6.22: Summary of systems' features presented in this chapter

In this chapter we have presented several data-driven methods for SALT error code prediction: an utterance level classifier, a CRF tagger, a dependency-based method, and a random-walk based method, the features of which are summarized in Table 6.22. We show the performances of the best systems of each type in Figure 6.23. Of these systems, the CRF tagger and the dependency-based method typically yield the best performance, whether it is in terms of identifying utterances with any error codes (UTTERANCE evaluation) or identifying specific error codes (ERROR evaluation), although neither method clearly outperforms the other on different corpora. A simple approach to combining these two systems did not improve performance unambiguously, and perhaps a better way of combining them would yield better results. Importantly, we found that neither the CRF tagger nor the dependency-based method is negatively affected by the presence of mazes. This means that there is no need to either detect mazes and error codes jointly (although this approach may be a fruitful direction for future research), or to detect them in a pipelined approach.

We re-annotated the first 1,000 utterances in the development folds of all three corpora, and found that ENNI, on which we observed relatively high performance throughout our experiments, has far more consistent annotations



(a) Conv - Utterance

(b) Conv - Error

(c) ENNI - Utterance

(d) ENNI - Error

(e) NarSR - Utterance

(f) NarSR - Error

Figure 6.23: Comparison of best performing systems of each type, NB: all graph methods are evaluated with Error-Set instead of Error evaluation

Method	P	R	F1
MS Word	**0.464**	0.139	0.214
ETS e-rater	0.081	**0.419**	0.136
Classifier	0.086	0.034	0.049
Tagger	0.324	0.353	**0.338**
Dependency	0.218	0.236	0.226
Joint	0.310	0.397	0.348
Random walk	0.283	0.247	0.263

(a) CONV

Method	P	R	F1
MS Word	0.530	0.276	0.363
ETS e-rater	0.151	0.244	0.187
Classifier	0.013	0.049	0.021
Tagger	0.656	0.658	0.657
Dependency	0.566	0.604	0.585
Joint	**0.662**	**0.664**	**0.663**
Random walk	0.637	0.434	0.516

(b) ENNI

Method	P	R	F1
MS Word	0.425	0.145	0.216
ETS e-rater	0.190	0.235	0.210
Classifier	0.109	0.113	0.111
Tagger	**0.477**	**0.445**	**0.461**
Dependency	0.280	0.252	0.265
Joint	0.431	0.433	0.432
Random walk	0.388	0.391	0.389

(c) NARSR

Table 6.23: Comparison of different error detection systems. Results for tunable methods reported at highest F1 with approximately balanced P/R. Mazes removed, UTTERANCE evaluation. Maximum performances for each corpus are in **bold**.

than the Conv and NarSR corpora, on which we consistently observed low performance. We ascribe the poor performance on these two corpora to the inconsistent annotations that we believe are present throughout all folds. We found in informal experiments that the performance of some of our SALT error code detectors is quite low on the other SALT corpora, hence it is likely that these corpora may have low-quality error code annotations as well. The fact that the dependency-based system outperformed the CRF tagger on what may be the only consistently-annotated corpus further underscores the utility of this method in addition to the CRF tagger for SALT error code detection.

In our error analysis, we found that the of all error types, the ENNI-trained system had the most trouble identifying utterances with verb agreement and tense errors, as well as ones containing multiple or rarely-observed errors. Even so, our system was able to identify the majority of these errors, and it did so with quite high precision. Furthermore, the ability to adjust the operating point of the system allows users to adjust the output depending on their needs: if they are annotating data then a high-recall operating point is appropriate, and if they are trying to get an idea of what errors a child produces, then a high-precision operating point is better.

Chapter 7

The clinical utility of SALT annotations

In this chapter we investigate the clinical utility of features derived from SALT-annotated transcripts of spoken language. First, in Section 7.1.1, we look at whether such features can be used to predict scores on structured instruments that capture various aspects of linguistic competence, and in particular, whether these features can be used to predict scores on structured instruments more effectively than verbal IQ. Next, in Section 7.2, we investigate whether these features can be used to distinguish between different diagnostic pairs that vary in the presence of either autism or a language impairment.

7.1 Predicting scores on structured instruments

We begin by investigating whether we can use features derived from SALT-annotated transcripts to predict sub-scores on two widely-used structured instruments for assessing language: CCC-2 and CELF-4. We begin by describing

the procedures we follow for extracting features from these transcripts and then predicting scores, as well as for evaluation. We then present the two structured instruments in turn, along with the results from our experiments.

7.1.1 Prediction

We use a *leave-transcript-out* procedure to predict scores on structured instruments. Concretely, we are given a set of children C, and for each child $c_i \in C$, we have her SALT-annotated transcript t_i, age a_i, and a structured instrument score s_i. Using ages and various features extracted from the SALT-annotated transcripts collected from all children except for c_i, we train a model m_i to predict the score on the structured instrument of child c_i. In our experiments, we use ordinary least squares linear regression to predict the test score for each child. We use ordinary least squares because it provides a natural baseline for this relatively novel task. Furthermore, while we have performed informal experiments to evaluate other regression methods (elastic net and LASSO), none of them obviously outperforms ordinary least squares linear regression.

7.1.2 Features

We use seven different sets of features in all of our experiments, each reflecting a transcript with a different level of manual annotation. These features are shown in Table 7.1. The Baseline feature set captures very basic data about the child: age, verbal IQ (VIQ), utterance length (captured with TKCT), and type count. The Transcript feature set excludes VIQ, instead capturing phenomena that are easy to derive from a basic manual transcript of the child's spoken language. The SALT feature sets require progressively more complicated types of annotations, ranging from mazes (SALT-1) to marking utterances as containing errors (SALT-2) through full SALT annotation (SALT-5). As described in Table

Group	Feature	Description
Baseline	Age	Child's age in months
	TkCt	Token count
	TpCt	Type count
	VIQ	Concatenated WPPSI3 & WISC4 VIQ standard scores
Transcript	All baseline features excluding VIQ along with:	
	CEOlp	# of times examiner speaks while child is talking
	ECOlp	# of times child speaks while examiner is talking
	IncCt	Incomplete word count
	UmUhRat	Ratio of 'um' to 'uh'
	UnintCt	Unintelligible word count
SALT-1	All Transcript features along with:	
	MpCt	Morpheme count
	MazeCt	Maze count
	MazeTkCt	Token count within mazes
	MazeTpCt	Type count within mazes
	NoMazeTkCt	Token count outside of mazes
	NoMazeTpCt	Type count oustide of mazes
SALT-2	All SALT-1 features along with:	
	NErrUtt	Number of utterances with any SALT error codes
SALT-3	All SALT-1 features along with:	
	ErrCt	Count of SALT error codes
SALT-4	All SALT-1 features along with:	
	UtLErrCt	Count of utterance level errors (EC / EU)
	WdLErrCt	Count of word level errors (all other error codes)
SALT-5	All SALT-1 features along with:	
	XCt	Count of individual error codes (X=EC, EO, ...; see Table 2.3)

Table 7.1: Features used for test score prediction and diagnostic classification

7.1, we then derive counts from these annotations. For example, the SALT-2 feature NERRUTT is simply the count of the number of utterances with any error codes.

We compute most of the features in Table 7.1 for each utterance. We then use the following summary statistics as features for test score prediction: minimum, maximum, median, mean, standard deviation. The only exception to this is the ratio feature UMUHRAT, which we compute using the entire transcript, and of course the AGE and VIQ features, which are simply given for each child.

For each test, we first predict scores using features derived from the manually annotated ADOS transcript. In the event that we observe a significant correlation between the observed and predicted test scores at the $\alpha = 0.05$ level, we investigate which features are the most important for predicting the test score in question. We then use features extracted from transcripts with automatically produced maze and SALT error code annotations. This can be seen as another way of extrinsically evaluating the maze and error code detectors, and in particular whether automatically produced SALT annotations capture the same informative cues as the manual annotations.

As shown in Table 7.1, we use features extracted from manual maze and error code annotations. When these features are useful for predicting a particular metric, we also investigate whether the same features derived from automatic SALT annotations are as effective. To do so, we obviously need to apply automated maze and error code annotations.

We use the M^3N-based maze detector described in Chapter 5 to produce automatic maze annotations. We compare two models for maze detection. The first model is trained on the training set of the ENNI corpus, and it is tuned to have balanced precision and recall on the ENNI development set. The second model for maze detection is in fact a collection of models: we predict

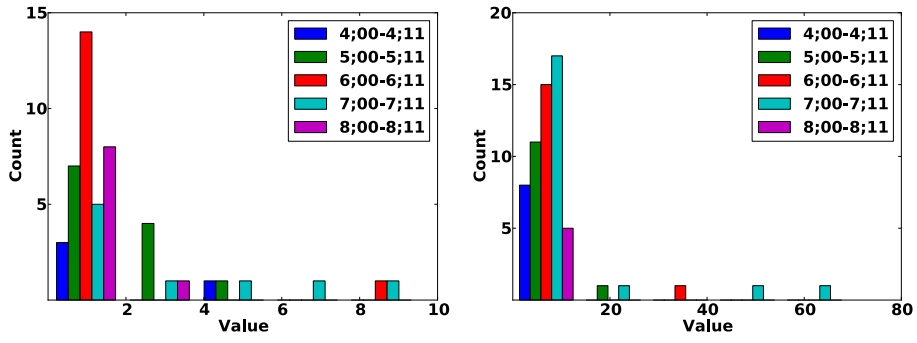
maze annotations on the CSLU transcripts in a leave-transcript-out (LTO) manner. We use the same penalty matrix (which determines the operating point) throughout the LTO procedure as is used to train the ENNI model. For clarity and conciseness, we will refer to features extracted from automatically produced annotations using these models as *ENNI features* and *LTO features*, respectively. Similarly, we will refer to features extracted from the manual annotations as *manual features*.

We use the CRF tagger described in Section 6.5 to produce SALT error code annotations. Again, we use two models: one trained on the training set of the ENNI corpus, and the other trained following the LTO procedure with the CSLU transcripts. We selected the CRF tagger for error detection because it performs well on a wide variety of corpora, even though there are some cases where the dependency-based system using Zpar as the parser outperforms the CRF tagger.

Distribution of features

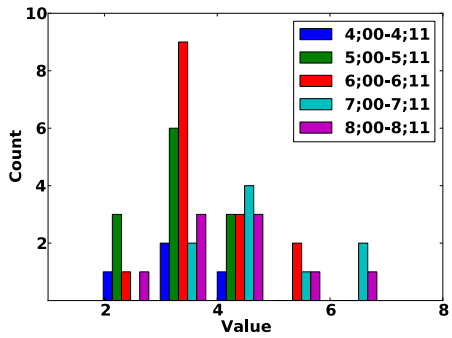
The features in Table 7.1 capture some aspects of each child's spoken language. Although we do not perform an in-depth analysis of how these features are distributed across children of different ages and with different diagnoses, we do wish to show the reader some of the variety we observe.

Figure 7.1 shows how a few of the features from the Baseline and SALT-1 sets are distributed over different diagnostic groups and children of different ages. In Figures 7.1a and 7.1b we see that the um-uh ratio in children with autism (ASD) is vastly lower than in children without autism (nASD). On the other hand, looking at the distribution of the token counts per utterance (TkCt) in Figures 7.1c, 7.1d, 7.1e, and 7.1f tells us that this feature appears to be more affected by the presence of a language impairment (i.e. between the language normal (LN) and language impaired (LI) groups) than by the presence of autism (i.e. between the autism (ASD) and no autism (nASD) groups). These distributions of features

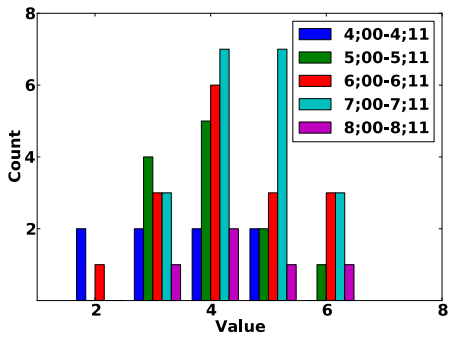


(a) UmUhRat - ASD

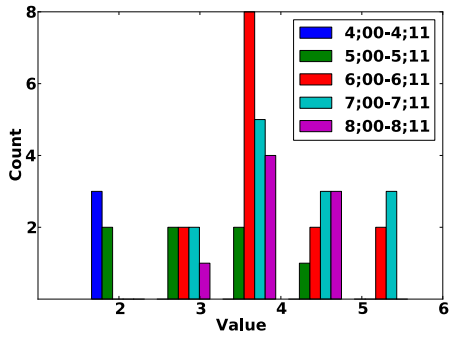
(b) UmUhRat - nASD



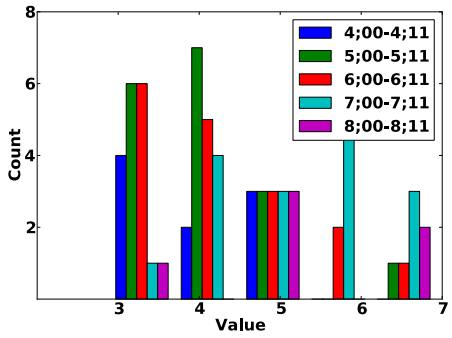
(c) TkCt - ASD



(d) TkCt - nASD



(e) TkCt - LI



(f) TkCt - LN

Figure 7.1: Distribution of features across diagnostic groups and ages

are entirely what one would expect based on the literature (see discussion in Section 2.3), but it is nevertheless useful to confirm that this is the case, rather than to assume it.

Appendix A contains similar plots for all of the Baseline and SALT features, for all of the diagnostic categories (ALI, ALN, SLI, TD), as well as the two composite categories: ASD (ALI + ALN), nASD (SLI + TD), LI (ALI + SLI), and LN (ALN + TD). The plots in Appendix A are grouped by feature rather than by diagnostic group to facilitate the comparison of the distributions of these features between diagnostic groups.

Accounting for age

A potential criticism of the feature set in Table 7.1 is that the child's age is simply thrown in as if it were an additive factor. Simply removing age will not do, however, as age has a great impact upon language development, and therefore upon many, if not all, of the features we use in this chapter. To address these issues, we evaluate a second set of features, which we will refer to as *age-group percentile features*. These features are derived from the ones in Table 7.1: instead of using raw feature values or ratios, we use the percentile of each feature's value, taken among children of the same age. For example, instead of using the raw TkCt feature for a child that is 8 years, 3 months old, we would report the percentile of the TkCt feature among all children in the study aged 8;0-8;11. We contrast the age-group percentile features with the *default features* described above.

7.1.3 Evaluation

We evaluate the predicted test scores produced by our leave-transcript-out regression procedure in two ways. First, we look at the correlation between

the predicted test score and the true test score in terms of Kendall's tau rank correlation coefficient (τ). We use the τ statistic because it is non-parametric. We compute τ as follows: let o_i and p_i be the observed and predicted test scores for child i. We then count the number of *concordant* and *discordant* pairs of observations (o_i, p_i), (o_j, p_j), iterating over all pairs of observations. A pair is concordant if both $o_i > o_j$ and $p_i > p_j$, or $o_i < o_j$ and $p_i < p_j$. Similarly, a pair is discordant if $o_i < o_j$ and $p_i > p_j$, or $o_i > o_j$ and $p_i < p_j$. If there are no ties, then we report τ_a:

$$\tau_a = \frac{c - d}{c + d}$$

We account for ties with the τ_b statistic, as follows:

$$\tau_b = \frac{c - d}{\sqrt{(c + d + t_o)(c + d + t_p)}}$$

where t_o is the number of ties where $o_i = o_j$,and similarly t_p is the count of ties where $p_i = p_j$. If there are no ties, then naturally $\tau_a = \tau_b$. We note that the τ statistic ranges between -1 (perfect disagreement in rankings) and 1 (perfect agreement in rankings).

We use the randomized paired-sample test described in Section 4.7.1 to see whether the predicted scores yielded by different feature sets come from significantly different distributions. In particular, we compare the performance yielded by different feature levels (ex. SALT-1 vs SALT-3), and features of the same level extracted from three different sources: manual, ENNI, and LTO features. When comparing features of the same level, at the onset of each of the iterations of the randomized paired-sample test we randomly select which variant (manual, ENNI or LTO) to use for each feature. We use the same version of each feature across all children, so we do not train or test on data that includes

both manual and automatic versions of the same feature. To compare different feature levels, we select one of the levels at random at each iteration of the LTO procedure, i.e. each time we train and test a model. Whether comparing automatic and manual features, or different feature levels, we perform 1,000 iterations of the randomized paired-sample test, each time producing an estimate of Kendall's tau that we then compare to the baseline in order to determine whether the trial is a success.

We will be performing multiple comparisons of correlation coefficients for each feature set: seven comparisons for the manual features, and five comparisons for the ENNI and LTO feature sets. We correct for this by using the Bonferroni correction: given a desired level of significance α, and n comparisons, we test for significance at the α/n level. In our experiments, we opt to use an α level of 0.05. After correction then, we test for statistically significance of the correlation coefficients using $\alpha = 0.007$ for experiments involving manual features, and $\alpha = 0.01$ for experiments involving ENNI and LTO features.

We evaluate the relative importance of different features by testing how well each feature is able to predict a particular metric using by averaging the relative importance of each feature (in terms of R^2) over all possible orderings (Kruskal, 1987), which is implemented in the R package `relaimpo` (Grömping et al., 2006). We use this method because it accounts for both direct and indirect effects, which occur when a feature of interest f is among the first and last regressors, respectively.

Stability of features

The features in Table 7.1 are all derived from transcripts of children's speech. One potential issue is that these features may be sensitive to which utterances are analyzed. This is a relevant issue in cases where there are not enough resources to annotate all of the available transcribed speech collected from a child: if these

features are sensitive to which utterances are selected for annotation, then extra care needs to go into selecting them. Furthermore, features that are sensitive to which utterances are analyzed should be less useful than ones that are not, since their apparent usefulness may only be due to chance.

We use the *split half* procedure to determine how sensitive the features in Table 7.1 are to the choice of which utterances are analyzed (Guttman, 1945). A single iteration of the split half procedure is as follows:

1. Randomly split each child i's transcript in two, t_{i1} and t_{i2}, by assigning the utterances to either with equal probability
2. Perform two rounds of leave-transcript-out prediction, one with the set of t_{i1}, and the other with the set of t_{i2}
3. Record the correlation of metrics predicted for each child from t_{i1} with the ones predicted from t_{i2}

We report the average correlation between the two predicted metrics from 100 iterations of the split-half procedure. For correlation, we use Pearson's r, which quantifies the strength of linear dependence between the two predicted metrics.

7.1.4 Predicting CCC-2 scores

The Children's Communicative Checklist-2 (Bishop and Volkmar, 2003) is a questionnaire used to assess a child's ability to communicate. The CCC-2 contains 70 items that capture different aspects of communication, and the entire questionnaire is to be filled by one person who knows the child well, for example a parent or teacher. Each item contains a single behavior, such as speaking repetitively about subjects that nobody is interested in, and the person filling out the questionnaire is asked to assign a numerical frequency to that behavior ranging from 0 (never) to 3 (several times a day or always). Each question falls

	Category	Example item
A	Speech	Pronounces words in a babyish way, such as 'chimbley' for 'chimney' or 'bokkie' for 'bottle'.
B	Syntax	Leaves out 'is', e.g. 'Daddy going to work' instead of 'daddy is going to work'.
C	Semantics	Is vague in choice of words, making it unclear what s/he is talking about, e.g. saying 'that thing' rather than 'kettle'.
D	Coherence	Is hard to make sense of what s/he is saying (even though words are clearly spoken).
E	Inappropriate initiation	It's difficult to stop him/her from talking.
F	Stereotyped language	Says things s/he does not seem to fully understand (may appear to be repeating something s/he heard an adult say). So, for instance, a 5-year old may be heard to say of a teacher "she's got a very good reputation."
G	Use of context	Asks a question even though s/he has been given the answer.
H	Non-verbal communication	Smiles appropriately when talking to people.
I	Social relations	Shows concern when other people are upset
J	Interests	Reacts positively when a new and unfamiliar activity is suggested.

(a) Categories and example items from CCC-2

Composite score	Items
General communication	Sum of items A through H
Social interaction deviance	(A + B + C + D) - (E + H + I + J)

(b) Items included in different composite scores following category labels in table above

Table 7.2: CCC-2 item categories and composite scores

Features	Manual	ENNI	LTO
Baseline	0.330		
Transcript	0.240		
SALT-1	0.231	0.248	0.258
SALT-2	0.306	0.248	0.244
SALT-3	0.290	0.241	0.246
SALT-4	0.308	0.241	0.246
SALT-5	0.225	0.224	0.213

(a) Coherence sub-score; all correlations significant at $\alpha = 0.05/n$ level (adjusting for Bonferroni correction)

Features	Manual	ENNI	LTO
Baseline	0.378		
Transcript	0.268		
SALT-1	0.301	0.336	0.315
SALT-2	0.391	0.317	0.297
SALT-3	0.383	0.324	0.302
SALT-4	0.378	0.324	0.302
SALT-5	0.313	0.300	0.239

(b) Semantics sub-score; all correlations significant at $\alpha = 0.05/n$ level (adjusting for Bonferroni correction)

Features	Manual	ENNI	LTO
Baseline	0.345*		
Transcript	0.093		
SALT-1	0.071	0.124*	0.131*
SALT-2	0.257*	0.120	0.190*
SALT-3	0.240*	0.118	0.193*
SALT-4	0.240*	0.118	0.193*
SALT-5	0.201*	0.102	0.132*

(c) Speech sub-score; $*$=correlation significant at $\alpha = 0.05/n$ level (adjusting for Bonferroni correction)

Features	Manual	ENNI	LTO
Baseline	0.478		
Transcript	0.247		
SALT-1	0.248	0.313	0.263
SALT-2	0.381	0.342	0.261
SALT-3	0.378	0.329	0.264
SALT-4	0.366	0.329	0.264
SALT-5	0.331	0.308	0.219

(d) Syntax sub-score; all correlations significant at $\alpha = 0.05/n$ level (adjusting for Bonferroni correction)

Table 7.3: Correlation (Kendall's τ) between observed and predicted CCC-2 sub-scores for all non-TD children in the CSLU data using manual, ENNI and LTO default features

Features	Manual	ENNI	LTO
Baseline	0.351		
Transcript	0.362		
SALT-1	0.351	0.326	0.334
SALT-2	0.362	0.311	0.322
SALT-3	0.349	0.300	0.318
SALT-4	0.340	0.291	0.313
SALT-5	0.266	0.270	0.269

(a) Coherence sub-score; all correlations significant at $\alpha = 0.05/n$ level (adjusting for Bonferroni correction)

Features	Manual	ENNI	LTO
Baseline	0.340		
Transcript	0.356		
SALT-1	0.313	0.304	0.285
SALT-2	0.320	0.276	0.275
SALT-3	0.336	0.254	0.287
SALT-4	0.283	0.245	0.281
SALT-5	0.250	0.218	0.219

(b) Semantics sub-score; all correlations significant at $\alpha = 0.05/n$ level (adjusting for Bonferroni correction)

Features	Manual	ENNI	LTO
Baseline	0.277*		
Transcript	0.208*		
SALT-1	0.145	0.173*	0.157
SALT-2	0.206*	0.155	0.219*
SALT-3	0.220*	0.137	0.201*
SALT-4	0.240*	0.127	0.190*
SALT-5	0.177*	0.108	0.128

(c) Speech sub-score; $*$=correlation significant at $\alpha 0.05/n$ level (adjusting for Bonferroni correction)

Features	Manual	ENNI	LTO
Baseline	0.474		
Transcript	0.436		
SALT-1	0.394	0.402	0.399
SALT-2	0.411	0.387	0.399
SALT-3	0.421	0.372	0.397
SALT-4	0.429	0.364	0.399
SALT-5	0.368	0.353	0.369

(d) Syntax sub-score; all correlations significant at $\alpha = 0.05/n$ level (adjusting for Bonferroni correction)

Table 7.4: Correlation (Kendall's τ) between observed and predicted CCC-2 sub-scores for all non-TD children in the CSLU data using manual, ENNI and LTO age-group percentile features

into one of the ten categories listed in Table 7.2a, and the administrator of the test computes sub-scores for each of these categories simply by summing the appropriate frequency ratings. Finally, various sets of sub-scores are combined to produce the two *composite* scores in Table 7.2b. We note that all composite scores in this study are US-scaled, which normalizes these sub-scores for different ages. Normalization of scores for other countries follows different standards (Bishop and Volkmar, 2003).

Some of the questions on the CCC-2 inquire about behaviors that should be observable in SALT-annotated transcripts. Such behaviors include producing mazes, making pronominal case errors (which would be marked with the `[EW]` error code), and using the past tense properly. Many of the items on the CCC-2, however, capture phenomena that cannot be derived from SALT-annotated transcripts in a straightforward manner, if at all. These phenomena fall into three broad categories. The first category is phenomena that SALT either does not label explicitly, or ones that are not typically labeled. For example, SALT annotators are instructed to transcribe all words with standard spelling, thus making it impossible to recover mispronunciations. Similarly, although particular research groups may annotate vague or stereotyped language, this is not standard SALT practice. Neither stereotyped nor vague language can be identified by looking at a single utterance without grounding knowledge, and as a result, identifying either of these phenomena with the automated SALT error code detection techniques described in Chapter 6 is beyond the scope of this thesis. The second category of items on the the CCC-2 that are not captured in SALT-annotated transcripts deal with behaviors that require interpreting spoken language and behavior that the transcribed activity would not necessarily elicit. For example, none of the tasks in the SALT corpora (narrative, conversational or expository) necessarily provide much of an opportunity for a child to show

concern when others are upset. The final category of phenomena included in the CCC-2 items, but excluded from SALT, are non-verbal aspects of communication, such as smiling appropriately. Non-verbal behaviors are typically excluded from transcripts entirely, whether SALT-annotated or not, and if such behaviors are included, then it is typically only certain ones (e.g. clapping, which makes a noise, but not blinking, which does not).

Predicting sub-scores from manual SALT annotations

We predict CCC-2 sub-scores for each child using the LTO procedure, in particular ones that capture some of the same phenomena as certain SALT annotations: coherence, semantics, speech, and syntax. We see in Tables 7.3 and 7.4 that the Baseline feature set is able to predict all four CCC-2 sub-scores with a statistically significant positive correlation between the true and predicted scores. The Baseline feature set is the only feature set that includes VIQ, and it yields the best predicted scores for the Coherence, Speech, and Syntax sub-scores.

Looking closer at the predictive performance on the various sub-scores in Table 7.3, we see that the Semantics sub-score because this is the only sub-score that the non-Baseline feature sets can predict competitively with the Baseline feature set. Specifically, we see in Table 7.3b that the SALT-2 and -3 feature sets both outperform the Baseline feature set for predicting the Semantics sub-score. Nevertheless, the randomized paired-sample test does not suggest that there is any difference in predictions yielded by either the SALT-2 or -3 features, and the Baseline features ($p \leq 0.097$ and $p \leq 0.578$, respectively). On the other hand, the results of the randomized paired-sample test do suggest that the SALT-2 and -3 feature sets may improve performance over the Transcript feature set ($p \leq 0.002$ and $p \leq 0.003$, respectively, albeit before Bonferroni correction).

We observe somewhat different results using the age-group percentile features, as shown in Table 7.4. Unlike the default features, some of the manual age-group

percentile features are able to predict CCC-2 sub-scores as well as the baseline features (ex. the Transcript through SALT-4 features predicting the Coherence sub-score). Still, all of the feature sets we consider, whether comprising default or age-group percentile features, are weak predictors of the CCC-2 sub scores, with none of the correlations rising above 0.478.

For all four of the CCC-2 sub-scores we consider, at least some of the SALT feature sets improve prediction relative to the Transcript feature set, and typically, the SALT-2, -3, and -4 feature sets tend to yield the best performance. The NErrUtt feature, which is a count of the number of utterances with any SALT error codes, and the only SALT-derived feature in the SALT-2 feature set, appears to be particularly effective. When we look at the relative contribution of predictors in the SALT-2 feature set, the NErrUtt feature is between the most useful features from the SALT-1 and Transcript feature sets: NoMazeTkCt ($R^2 = 0.046$), and IncCt ($R^2 = 0.024$). This demonstrates that the NerrUtt feature does have some predictive power. Thus, even if the SALT-derived features are not as effective as VIQ for predicting CCC-2 sub-scores, they do have some predictive power beyond features extracted from an unannotated transcript.

We note that none of the feature sets is a particularly good predictor of the CCC-2 Speech sub-score, as can be seen in Table 7.3c. As explained above, this is to be expected: the CCC-2 Speech sub-score captures phonetic and phonological issues, which are not necessarily captured in SALT-annotated transcripts, and which are not explicitly captured by any of our features.

One potential issue with SALT-derived features is their consistency: are the SALT feature sets highly sensitive to which utterances are analyzed? If this is the case, then great care would be needed to select utterances that are suitable for analysis, which somewhat defeats a purported advantage of language sample analysis over structured instruments, namely analyzing organic samples

of language as opposed to linguistic competence on artificial tasks. Using 100 iterations of the split-half procedure (described in Section 7.1.3), we see that the SALT features are quite consistent: the average value of r from 100 predictions of the CCC-2 semantics score is 0.80 for the the SALT-2, -3, -4 and -5 features. This indicates that the features derived from SALT annotations that we considered are not sensitive to which utterances are analyzed. As a result, we expect that two transcripts will yield comparable features so long as they are both sufficiently long and are of a similar activity.

Predicting sub-scores from automatic SALT annotations

Table 7.3b shows the correlation between the true CCC-2 Semantics sub-scores and ones predicted using the ENNI and LTO features. We see that the ENNI features are less useful than the manual ones for predicting this metric, and the LTO features are less useful still. It is possible that the ENNI features are more effective than the LTO ones because the maze and error annotation models are trained on more data. Alternatively, it may be that the ENNI corpus was annotated slightly more consistently than the CSLU corpus, thus resulting in better maze and error annotation models; given the wide variety in the quality of SALT annotations we have found in Chapters 5 and 6, this would not be surprising. Nevertheless, the differences in the correlations yielded by the manual and automatic SALT features produced by the ENNI models are not statistically significant in any case ($\alpha = 0.05/n$ level, including Bonferroni correction).

7.1.5 Predicting CELF-4 scores

The Clinical Evaluation of Language Fundamentals – fourth edition (CELF-4) is a structured instrument designed to evaluate several components of language in children ages 5 to 21 (Semel et al., 2003). In particular, the CELF-4 assesses

phonological awareness, morphology, syntax, semantics, and pragmatics. SALT annotations capture many of these aspects of language to some degree, in particular morphology and syntax, and to a lesser extent semantics, and even potentially pragmatics (with non-standard annotations).

The CELF-4 contains the 13 core and five supplementary items presented in Table 7.5. Each of these tests are scored individually. Additionally, subsets of the CELF-4 tests are combined into the composite scores described in Table 7.6. Compared to the CCC-2, the CELF-4 has two major advantages from our point of view: 1) the CELF-4 index scores are more similar to information potentially captured by SALT annotations than are many CCC-2 sub-scores; and 2) the relevant CELF-4 index scores may be more consistent across children than the relevant CCC-2 sub-scores because the CELF-4 index scores come from tests administered by a clinician, while the CCC-2 sub-scores are derived from questionnaires filled out by parents reporting on their children.

We predict the following CELF-4 index scores for each child using the LTO procedure: Language Structure Index, Core Language Score, Receptive Language Index and Expressive Language Index. Results are presented in Tables 7.7 and 7.8, which contains results from the default features and the age-group percentile features, respectively. We do not consider either the Language Memory Index or Working Memory Index scores since these appear unlikely to relate to phenomena captured by our SALT-derived features. Furthermore, some CELF-4 subtests were not administered, and as a result, we do not have Language Content Index scores for any of the children. Finally, we do not have CELF-4 scores for children in the TD group, so we are only able to train and test models on children in the ALI, ALN, and SLI groups.

Test	Description
Concepts and following directions	The child points to pictured objects in response to oral directions
Word structure	The child completes sentences using the targeted structure(s)
Recalling sentences	The child imitates sentences presented by the examiner
Formulated sentences	The child formulates a sentence about visual stimuli using a targeted word, or phrase
Word classes	The child chooses two related words and describes their relationship
Sentence structure	The child points to a picture that illustrates the given sentence
Expressive vocabulary	The child identifies a pictured object, person, or activity
Word definitions	The child defines a word that is presented and used in a sentence
Understanding spoken paragraphs	The child responds to questions about orally presented paragraphs; questions target main idea, details, sequence, inferential, and predictive information
Sentence assembly	The child produces two semantically and grammatically correct sentences from visually and orally presented words or groups of words
Semantic relationships	The child listens to a sentence and selects the two choices that answer a target question
Number repetition	The child repeats a series of numbers forward, then backwards
Familiar sequences	The child names days of the week, counts backward, orders other information while being timed
Rapid automatic naming	The child names colors, shapes, and color-shape combinations while being timed
Word Associations	The child names words in specific categories while being timed
Phonological awareness	The child rhymes, segments, blends, identifies sounds and syllables in words and sentences
Pragmatics profile	The examiner elicits information from a parent or teacher about the child's social language skills
Observational Rating Scales	Parent, teacher, and child each rate the child's classroom interaction and communication skills

Table 7.5: Description of CELF-4 tasks taken from Pearson Education, Inc. (2008)

Score	Summary
Core language score	Overall language performance. Used to identify the presence of a language impairment
Receptive language index	Measures listening and reading comprehension
Expressive language index	Measures child's ability to express themselves with language
Language content index	Captures semantic development, including vocabulary, relationships between words, interpretation of facts, and ability to make meaningful, grammatical sentences
Language structure index	Evaluates both receptive and expressive language, but only used with children ages 5-8
Language memory index	Measures ability to recall spoken directions, formulate sentences with given words, and and identify semantic relationships
Working memory index	Measures attention, concentration, and recall

Table 7.6: Description of CELF-4 index scores taken from Pearson Education, Inc. (2008)

Features	Manual	ENNI	LTO
Baseline	0.643*		
Transcript	0.126*		
SALT-1	0.156	0.422*	0.165
SALT-2	0.280*	0.453*	0.173
SALT-3	0.265*	0.458*	0.176
SALT-4	0.239	0.458*	0.176
SALT-5	0.306*	0.379*	0.163

(a) Core Language Score

Features	Manual	ENNI	LTO
Baseline	0.640*		
Transcript	0.162		
SALT-1	0.213*	0.436*	0.167
SALT-2	0.378*	0.476*	0.228*
SALT-3	0.381*	0.481*	0.228*
SALT-4	0.345*	0.481*	0.228*
SALT-5	0.363*	0.428*	0.169

(b) Expressive Language Index

Features	Manual	ENNI	LTO
Baseline	0.584*		
Transcript	0.182*		
SALT-1	0.140	0.442*	0.133
SALT-2	0.224*	0.458*	0.141
SALT-3	0.235*	0.465*	0.149
SALT-4	0.206*	0.465*	0.149
SALT-5	0.288*	0.429*	0.047

(c) Language Structure Index

Features	Manual	ENNI	LTO
Baseline	0.449*		
Transcript	0.025		
SALT-1	0.018	0.256*	0.046
SALT-2	-0.009	0.212*	0.060
SALT-3	-0.013	0.214*	0.060
SALT-4	0.021	0.214*	0.060
SALT-5	0.199*	0.151	0.087

(d) Receptive Language Index

Table 7.7: Correlation (Kendall's τ) between observed and predicted CELF-4 index scores for all non-TD children in the CSLU data using manual, ENNI and LTO features
$*$=correlation significant at $\alpha = 0.05/n$ level (adjusting for Bonferroni correction)

Features	Manual	ENNI	LTO
Baseline	0.619*		
Transcript	0.156		
SALT-1	0.128	0.220	0.074
SALT-2	0.340*	0.220	0.128
SALT-3	0.326*	0.222	0.090
SALT-4	0.258*	0.207	0.083
SALT-5	0.220	0.069	0.064

(a) Core Language Score

Features	Manual	ENNI	LTO
Baseline	0.624*		
Transcript	0.217		
SALT-1	0.227	0.330*	0.219
SALT-2	0.443*	0.341*	0.268*
SALT-3	0.417*	0.314*	0.232
SALT-4	0.383*	0.307*	0.220
SALT-5	0.296*	0.208	0.167

(b) Expressive Language Index

Features	Manual	ENNI	LTO
Baseline	0.618*		
Transcript	0.217		
SALT-1	0.159	0.348*	0.175
SALT-2	0.272*	0.311*	0.178
SALT-3	0.262*	0.310*	0.136
SALT-4	0.168	0.309*	0.122
SALT-5	0.149	0.198	0.160

(c) Language Structure Index

Features	Manual	ENNI	LTO
Baseline	0.425*		
Transcript	0.002		
SALT-1	-0.042	-0.002	-0.132
SALT-2	-0.029	-0.033	-0.097
SALT-3	-0.015	-0.059	-0.094
SALT-4	0.009	-0.070	0.100
SALT-5	0.053	-0.083	-0.012

(d) Receptive Language Index

Table 7.8: Correlation (Kendall's τ) between observed and predicted CELF-4 index scores for all non-TD children in the CSLU data using manual, ENNI and LTO age-group percentile features
$*$=correlation significant at $\alpha = 0.05/n$ level (adjusting for Bonferroni correction)

Predicting sub-scores from manual SALT annotations

Looking at Tables 7.7 and 7.8, we see that the Baseline feature set is able to predict the four CELF-4 index scores far more effectively than the Transcript feature set or any of the SALT feature sets. This is the case for both the default and age-group percentile features.

We see in Table 7.7 that for three of the four CELF-4 index (excepting the Receptive Language Index), the predictions yielded by the Transcript and SALT (default) feature sets correlate significantly with the true scores. These features are not effective at predicting Receptive Language Index scores: aside from the Baseline, the only feature set that is able to predict scores that correlate significantly with the true scores is SALT-5, which does so poorly compared to the Baseline feature set ($\tau = 0.199$, $p \leq 0.034$ as opposed to $\tau = 0.449$, $p \leq 0.001$). The randomized paired-sample test shows indicates that the distributions of CELF-4 Receptive Language Index score predictions yielded by the Baseline and SALT-5 feature sets are significantly different ($p \leq 0.001$).

Continuing further with the default feature set, the SALT-2 through -5 feature sets yield predictions that correlate significantly with the true scores for the Core Language Score, Expressive Language Index, and Language Structure Index. In all three cases, the SALT-5 feature set yields the best predictions of the SALT feature sets. Nevertheless, even predictions from the SALT-5 feature set appear to be worse than ones yielded by the Baseline features. For example, as can be seen in Table 7.7b, the SALT-5 feature set is able to predict the Expressive Language Index with $\tau = 0.363$. Using the randomized paired sample test, we see that these predictions are drawn from a significantly different distribution ($p \leq 0.001$) compared to the Baseline predictions ($\tau = 0.640$). This strongly suggests that the SALT-5 predictions are worse than the Baseline ones. Similarly, the SALT-5 feature set predicts the Core Language Score with $\tau = 0.306$ compared to the

Baseline τ of 0.663 (again, the randomized paired sample test yields $p \leq 0.001$).

Predicting sub-scores from automatic SALT annotations

We now predict the four CELF-4 index scores using SALT features extracted from automatic annotations. Comparing Tables 7.7 and 7.8, we find that the default features are more useful than the age-group percentile features. The latter of these are able to predict Expressive Language Index and the Language Structure Index scores that correlate significantly with the true scores, but this is not the case for either the Core Language Score or the Receptive Language Index score. The default LTO features are not effective predictors of these two index scores either, as can be seen in Table 7.7. The randomized paired-sample test suggests that the predictions of the Expressive Language and Language Structure Index scores using the ENNI and manual features are more faithful to the true scores than those yielded by the LTO features.

Overall, the default ENNI feature sets appear to be more effective at predicting CELF-4 index scores than either the manual or LTO feature sets, as can be seen throughout Table 7.7. In all but one case (SALT-5 for the Receptive Language Index), the ENNI features predict the CELF-4 index scores with higher correlations to the true index scores than the manual features. Nevertheless, the differences in performance yielded by the manual and ENNI features are not significant at the $\alpha = 0.05$ level for any feature set/index score combination. Finally, as was the case with the CCC-2 sub-scores, the SALT-2 through -4 default feature sets appear to be the most effective for predicting CELF-4 index scores, excluding the Baseline feature set, which includes VIQ.

Group 1		Group 2		
	N		N	Matched on
ALI	22	ALN	23	Age, SCQ total score, ADOS CSS
ALI	24	SLI	17	Age, VIQ, PIQ
ALN	21	TD	37	Age, VIQ, PIQ
SLI	14	TD	38	Age, SCQ total score, ADOS CSS

Table 7.9: Details of matched diagnostic groups
ADOS CSS=Autism Diagnostic Observation Schedule Calibrated Severity Score; PIQ=Performance IQ; SCQ=Social Communication Questionnaire; VIQ=Verbal IQ

7.2 Using SALT to discriminate between diagnostic pairs

We now turn to a slightly different question: how useful are features derived from SALT-annotated transcripts for predicting the presence of either autism or a language impairment? We address this question using a 'leave pair out' classification procedure, described in Section 7.2.1, which we first introduced for this task in our workshop publication 'The utility of manual and automatic linguistic error codes for identifying neurodevelopmental disorders' (Morley et al., 2013). We perform these experiments using the matched groups shown in Table 7.9. These matched pairs were generously provided by Kyle Gorman, and were created with a pre-release version of the `ldamatch` package for R, which 'performs group matching by backward elimination using linear discriminant analysis' (Gorman, 2015). We also compare the effectiveness of features extracted from manual SALT annotations to ones extracted from automatic SALT annotations.

7.2.1 Leave-pair-out prediction and evaluation

Due to the small number of children in all of the matched groups, we use a cross-validation procedure to evaluate the utility of various features. In particular, we use a leave-pair-out (LPO) classification schema (Cortes et al., 2007), which is quite similar to the LTO procedure we use to predict scores on structured instruments. Let C_1 be the set of all children with the first diagnosis d_1 in the matched group, and C_2 be the set of all children with the second diagnosis d_2 in the matched group. For LPO classification, we iterate over all pairs of children from different diagnostic groups: $c_1 \in C_1$ and $c_2 \in C_2$. We train a model to predict the diagnoses of c_1 and c_2 on all of the other children in C_1 and C_2. Then, we predict the probabilities of each child having either diagnosis. In practice, we use the logistic regression classifier in Scikit-learn (Pedregosa et al., 2012) to compute these probabilities.

We evaluate the LPO procedure by computing the *area under the receiver operating curve* (AUC). We count each iteration as a success if $p(d_1|c_1) > p(d_1|c_2)$ or in other words, if the classifier says that c_1 is more likely to have diagnosis d_1 than is c_2. We report the number of successes divided by the number of pairs, which provides an unbiased estimate of AUC (Airola et al., 2011). AUC is simply the probability that the classifier will assign a higher score to a randomly chosen "positive" example than to a randomly chosen "negative" example. AUC effectively ranges from 0.5—chance performance—to 1.0—a perfect classifier. If AUC is less than 0.5, then we simply count "positive" predictions as "negative", which redefines AUC as $1 - \text{AUC}$, thus guaranteeing that it is at least 0.5.

We use the randomized paired-sample test described in Section 4.7.1 to compare how effectively different feature sets allow us to perform this diagnostic discrimination task. In particular, we compare the performance yielded by different feature levels, and features of the same level extracted from three

different sources: manual SALT annotations ('manual features') and the two different automatic SALT annotations ('automatic features'), namely ones produced using a model trained on the ENNI data set, and another trained using a leave-transcript-out procedure.

When comparing manual and automatic features, at the onset of each of the iterations of the randomized paired-sample test we randomly select whether to use the manual or automatic variant of each feature. We use the same version of each feature across all children, so we do not train or test on data that includes both manual and automatic versions of the same feature. To compare different feature levels (ex. Baseline vs SALT-3), we select one of the levels at random at each iteration of the LPO procedure, i.e. each time we train and test a model. Whether comparing automatic and manual features, or different feature levels, we perform 1,000 iterations of the randomized paired-sample test, each time producing an estimate of AUC that we then compare to the baseline in order to determine whether the trial is a success.

7.2.2 Results

Table 7.10 contains the AUC on the diagnostic discrimination task for all feature levels extracted from both manual and automatic SALT annotations. First, we see that manual and automatic features are equally effective for this task. In only one case does the randomized paired-sample test indicate that there are any cases where results yielded by manual features differ significantly from those yielded by automatic features. The sole exception is the SALT-1 features derived from LTO annotations, which yield significantly different predictions from the manual features ($p < 0.001$) when discriminating between the TD and SLI groups. The LTO SALT-1 features do not, however, yield significantly different results than the ENNI SALT-1 features ($p < 0.931$). Moreover, the LTO SALT-1 features

Group 1	Group 2	Feature Set	Manual AUC	ENNI AUC	LTO AUC
ALI	ALN	Baseline	0.834		
		Transcript	0.646		
		SALT-1:5	0.650	0.666	0.658
		SALT-1:5 + VIQ	0.826†	0.824†	0.824†
ALI	SLI	Baseline	0.586		
		Transcript	0.588		
		SALT-1	0.581	0.569	0.578
		SALT-2	0.600	0.571	0.578
		SALT-3	0.605	0.571	0.578
		SALT-3+VIQ	0.583	0.569	0.561
		SALT-4	0.605	0.571	0.578
		SALT-4+VIQ	0.581	0.569	0.561
		SALT-5	0.588	0.571	0.578
ALN	TD	Baseline	0.544		
		Transcript	0.668		
		SALT-1	0.655	0.694	0.655
		SALT-2	0.655	0.694	0.655
		SALT-3	0.656	0.694	0.655
		SALT-3+VIQ	0.672	0.669	0.663†
		SALT-4	0.655	0.694	0.655
		SALT-5	0.655	0.694	0.655
SLI	TD	Baseline	0.966		
		Transcript	0.769		
		SALT-1	0.791	0.793	0.801*
		SALT-2	0.799	0.793	0.801
		SALT-3	0.806	0.793	0.801
		SALT-3+VIQ	0.806	0.793	0.801
		SALT-4	0.806	0.793	0.801
		SALT-4+VIQ	0.927†	0.923†	0.923†
		SALT-5	0.799	0.793	0.801

Table 7.10: AUC on diagnostic discrimination task using baseline features, as well as ones extracted from manual and automatic SALT annotations.
*: results are significantly different at the $\alpha = 0.05$ level from those yielded by the manual features at the same level; significance determined using randomized paired sample test
†: results are significantly different at the $\alpha = 0.05$ level from those yielded by the same feature set without VIQ; significance determined using randomized paired sample test

do not yield substantially better performance than the manual SALT-1 features discriminating between the TD and SLI groups (AUC=0.801 vs 0.791).

Pairs that are not matched on VIQ (ALI/ALN and SLI/TD) differ only in the presence of a language impairment. For these pairs, the Baseline feature set, which contains VIQ, far outperforms any of the other feature sets. This is not surprising, as low VIQ is a very strong cue of a language impairment. The SALT-derived feature sets significantly improve discrimination between the SLI and TD groups, relative to the 'Transcript' features (ex. $p \leq 0.001$ for manual SALT-3 features vs Transcript). For this pair, the SALT-3 and 4 feature sets, which perform identically, significantly outperform the SALT-1 ($p \leq 0.008$ for manual features), but not the SALT-2 and -5 ones ($p \leq 0.066$ for both with manual features). On the other hand, the SALT-derived features do not significantly improve discrimination between the ALI/ALN groups over the Transcript features, whether they are manual ($p \leq 0.412$), or automatic (ENNI $p \leq 0.066$, LTO $p \leq 0.196$). When we add the VIQ feature to the best-performing SALT feature set we see a dramatic improvement in discriminative performance relative to the original SALT feature set, but performance is still worse than using the Baseline features. Performance likely degrades with the additional features because the logistic regression classifier is not trained to maximize AUC over all iterations, but rather using maximum likelihood estimation at each iteration. SALT features are not ignored completely because they have some discriminatory value, even if it is substantially less than VIQ.

Pairs that are matched on VIQ (ALI/SLI and ALN/TD) differ only with respect to the presence of autism, but not the presence of a language impairment. As shown in Table 7.10, none of the feature sets we consider is particularly effective for discriminating between these pairs. Even in cases where the SALT-derived features outperform the Transcript features, the results are not significantly

different (ex. $p \leq 0.999$ comparing SALT-4 to Transcript for ALI/SLI). Similarly, the Transcript features do not yield significantly different results from the Baseline features for discriminating between the ALN/TD groups ($p \leq 0.997$). Unsurprisingly, adding VIQ to the best performing feature set for discriminating between the ALI and ALN groups—SALT-3—degrades performance relative to that feature set without VIQ. Finally, adding VIQ to the LTO SALT-3 feature set yields significantly different results compared to the original LTO SALT-3 feature set ($p \leq 0.015$), but the increase in AUC is not substantial (0.655 to 0.663).

7.3 Conclusions

Throughout these experiments we have seen that VIQ is a highly effective predictor of scores on other structured instruments that capture a child's ability to use language, specifically the CCC-2 and CELF-4. Features derived from transcripts with varying degrees of SALT annotations can can typically predict these scores to some degree, albeit not as effectively as when VIQ is used for prediction. Promisingly, we find that features extracted from automatic SALT annotations are as reliable as ones extracted from manual SALT annotations for predicting the CCC-2 sub-scores and CELF-4 index scores that we considered. In particular, we find that ENNI features—ones extracted from automatic annotations produced by models trained on an external data set—are able to predict CCC-2 sub-scores and CELF-4 index scores at least as well as LTO features, and in some cases even better. Taken together, these findings suggest that whatever utility manual SALT annotations may have is present in automatic annotations as well. The discrepancies in the quality of the predictions yielded by the ENNI and LTO findings underscore the importance of the quality and thoroughness of annotations in model training data. In Chapters 5 and 6 we

demonstrate that the maze and error-code annotations in the ENNI corpus are more consistent than the other SALT corpora. Although we have not evaluated the CSLU ADOS corpus in the same way, it is possible that it is less consistently annotated than the ENNI corpus, and that this inconsistency is in turn responsible for the ENNI-trained models yielding slightly better performance than the LTO models. We note that any differences in performance are unlikely to have been caused by corpus size: the ENNI corpus contains 56,108 utterances, while the CSLU ADOS corpus contains 61,949 utterances.

We have also investigated whether features derived from transcripts of spoken language can be used to discriminate between different diagnostic groups. In particular, we have explored whether increasingly detailed SALT-annotations improve performance on this task. We find that SALT-derived features can be used to discriminate between the SLI/TD pair, which is the only pair in which one group has typical language (TD), and the other has impaired language (SLI)[1]. As with predicting CCC-2 and CELF-4 scores, features derived from manual and automatic SALT annotations are as effective as one another for this task.

SALT-derived features are not useful for discriminating between groups that are matched on the presence of a language impairment (ALI/SLI and ALN/TD), or perhaps more narrowly, between groups that are matched on VIQ. Adding VIQ to the SALT feature sets does not improve discrimination performance for these two diagnostic pairs. This suggests that features derived from SALT annotated transcripts are not effective indicators of autism. Combined with our findings that SALT feature sets are not as effective as VIQ for predicting scores on structured instruments (Chapter 7), we conclude that at best, the SALT features we consider capture essentially the same information as VIQ, albeit

[1] As discussed in Sections 2.3.2, 7.1.4, and 7.1.5 the ALN group has deficits in communication compared to the TD group, even though none of the children in it has a language impairment.

noisily.

This brings us to a larger question, namely whether features derived from SALT annotated transcripts provide information that is complementary to VIQ, or whether at best they essentially capture VIQ with added noise. The experiments in this chapter suggest that the latter is more likely to be the case. First, the CCC-2 sub-scores and CELF-4 index scores that we considered capture many aspects of language, but we have not found any cases where features derived from SALT transcripts yielded substantially better predictions than the VIQ-including Baseline feature set. Furthermore, we have found that VIQ is a far more powerful feature for discriminating between groups that differ in the presence of a language impairment (ALI/ALN or SLI/TD) than features derived from a SALT-annotated transcript. In no case did adding VIQ to a SALT feature set improve performance on the diagnostic discrimination task over the Baseline feature set, which does include VIQ. This suggests that the SALT-derived features we evaluated capture largely the same information as VIQ, albeit less effectively. If the SALT features do in fact capture information complementary to VIQ, then this information appears not to be captured by any of the CCC-2 or CELF-4 sub-/index scores we evaluated, nor does it appear to be predictive of the presence of either a language impairment or autism.

Chapter 8

Conclusions

The primary focus of this thesis has been to automate two key annotations used in the Systematic Analysis of Language Transcripts (SALT) conventions for language sample analysis. In Chapter 5, we show that an existing detector for the closely related task of disfluency detection (Qian and Liu, 2013) can be adapted to maze detection. In Chapter 6 we explore several methods for automating SALT error code annotations, which indicate grammatical errors. Specifically, we evaluate methods to identify grammatical errors that could be identified by observing a single utterance in isolation, thus excluding, for example, pronominal gender errors and pragmatic errors. Our investigation includes a variety of techniques to identify grammatical errors, including utterance-level classifiers, taggers, a method based on dependency parsing, and a random-walk-based method. The methods based on tagging and dependency parsing perform better than the others, but no single method performs best across all corpora. Crucially, these methods, along with the others that we proposed, are both trainable and tunable. Since these methods are trainable, they can be used with novel error sets, as the standards used to apply error code annotations appear to be variable

across corpora. By tunable, we mean that the user can set the operating point of the system in terms of precision and recall. This permits the error detectors to be used for a wide variety of downstream objectives, including identifying a small sample of ungrammatical utterances (high precision), or annotating utterances that are then to be corrected by hand (high recall).

During our investigations into automating maze and error code detection we have found that annotation standards are highly variable across SALT-annotated corpora. Some of these differences are obvious, while others are relatively subtle. For example, the GillamNT corpus annotates asides, which are typically delimited with double parentheses, as mazes. On the other hand, we find that corpus-specific models for maze detection tend to outperform various models trained on multiple corpora, which suggests that there may be other, more subtle differences in maze annotation across corpora. In the case of error codes, we find that the different SALT corpora use different sets of error codes, and the same error code may be used differently in different corpora: the `[EO]` (overgeneralization) code is used throughout the ENNI corpus, but hardly at all in the Conv corpus, where overgeneralization errors are captured by `[EW]`. Furthermore, we find that annotation quality varies widely across SALT-annotated corpora by manually relabeling whether 1,000 utterances in certain corpora contained an error: we disagree with the annotators' decisions on 20 utterances in the ENNI corpus (all of which could be true errors when considered in context), but 104 in the NarSR corpus.

The variable quality and standards of annotations in the SALT corpora calls into question their value as "reference databases". As described throughout the SALT manual (Miller et al., 2011), these corpora are included with the SALT software as references against which one can compare statistics computed from a transcript of interest. If each of the SALT corpora is annotated following

idiosyncratic standards, as appears to be the case, and a transcript of interest is annotated following yet another set of standards, it is unclear what value is to be found in this comparison. To illustrate, asides are always excluded from utterance length statistics for the GillamNT corpus, because they are annotated as mazes. If the transcript of interest annotates asides in the standard manner and includes them computations of utterance length, any comparisons of utterance length with the GillamNT corpus will be spurious. Similarly, it is unclear what meaning one can attach to the frequency of grammatical errors found in some SALT corpora if one in ten of these annotations is wrong. We recommend that the SALT corpora either come with more thorough descriptions of the annotation standards used, and that some annotations (for example the error codes in NarSR) be redone completely. The automated techniques that we proposed for both maze detection and SALT error code detection could perhaps be used to expedite this process.

We have also explored the clinical and diagnostic utility of SALT annotations. In Chapter 7, we consider whether features derived from SALT-annotated transcripts can be used to predict scores on the CCC-2 and CELF-4—two structured instruments that assess language. We found that in most cases, features derived from SALT annotations provide predictive power beyond ones derived from an unannotated transcript, but they rarely are as useful as VIQ. We also evaluate whether these same features are useful for identifying the presence of either autism or a language impairment. We find that SALT-derived features cannot be used to identify autism reliably, but can be used to identify a language impairment. As with predicting scores on structured instruments, the SALT-derived features are useful, but not as effective as VIQ for this task. One likely possibility is that the set of SALT annotations used on the CSLU corpus, which we note is a slightly expanded set compared to the one described

in the SALT manual, do not capture any more information than verbal IQ. Interestingly and encouragingly, however, we found that features derived from automatic SALT annotations are essentially as effective as ones derived from manual SALT annotations.

We see several fruitful directions for future work, including building on the techniques in natural language processing that are proposed here, both for maze and error code detection. We believe, however, that carrying out such work successfully is contingent upon better quality data. The SALT corpora, with the possible exception of the ENNI corpus (which we believe to be well annotated), should be re-annotated following more rigorous standards. In doing so, it may be useful to have separate codes to indicate whether the error is always visible (ex. 'goed'), or whether it is only visible in context (ex. 'she left' instead of 'he left'). Without better quality data, it is difficult to be confident that particular methods perform well across corpora and domains such as conversation and narrative tasks. Better quality data will also allow us to test whether a single error detection or maze model can be used across corpora, or whether there is truly a justification for task- or corpus-specific models.

The work we presented in Chapter 7, in which we predict scores on structured instruments and the presence of autism or a language impairment, should be repeated, but using different data, specifically transcripts of narrative or expository tasks. The ADOS data that we have used consists of transcripts of conversations, but conversational data tends to elicit utterances that are short and relatively simple syntactically. On the other hand, expository and narrative tasks elicit longer, more complex utterances, and these are better able to identify the presence of a language impairment (Hadley, 1998). In fact, Nippold et al. (2008) found no group differences between children with and without a language impairment when performing a conversational task, but

clear ones during an expository task. We expect that simply repeating these experiments using narrative or expository data will not improve performance in identifying autism; for that, we expect that the SALT annotations would need to be expanded to include richer pragmatic codes than the rarely used `[EC]` 'inappropriate utterance'. Nevertheless, it is possible that features derived from SALT annotated transcripts of expository or narrative tasks would have more diagnostic utility than the ADOS transcripts we have explored here.

Appendix A

Maze detection experiments

A.1 Cross-corpus maze detection experiments

This section contains the complete results of the cross-corpus maze detection experiments described in Section 5.4.1. Briefly, in these experiments, we train a single model on the training set of one corpus. We tune the maze detector so that precision and recall are roughly balanced on the development set of the training corpus. We then evaluate the performance of this model on the development set of each of the other SALT corpora.

Corpus	Tag			Bracket		
	P	R	F1	P	R	F1
CONV	0.806	0.785	0.795	0.711	0.690	0.700
ENNI	0.826	0.755	0.789	0.699	0.688	0.694
EXPOSITORY	0.700	0.633	0.665	0.636	0.669	0.652
GILLAMNT	0.802	0.622	0.701	0.652	0.593	0.621
NARSSS	0.826	0.781	0.803	0.693	0.661	0.677
NARSR	0.796	0.765	0.780	0.638	0.642	0.640
NZCONV	0.846	0.859	0.852	0.767	0.800	0.783
NZPERNAR	0.863	0.867	0.865	0.753	0.779	0.766
NZSR	0.951	0.806	0.872	0.708	0.676	0.691

Table A.1: Model trained on CONV, tested on SALT development sets

Corpus	Tag			Bracket		
	P	R	F1	P	R	F1
CONV	0.680	0.753	0.715	0.576	0.638	0.605
ENNI	0.898	0.832	0.864	0.791	0.791	0.791
EXPOSITORY	0.601	0.614	0.608	0.552	0.635	0.591
GILLAMNT	0.752	0.639	0.691	0.614	0.605	0.610
NARSSS	0.776	0.772	0.774	0.635	0.664	0.649
NARSR	0.777	0.777	0.777	0.609	0.642	0.625
NZCONV	0.743	0.824	0.781	0.618	0.728	0.668
NZPERNAR	0.814	0.856	0.834	0.694	0.737	0.715
NZSR	0.927	0.821	0.871	0.704	0.685	0.694

Table A.2: Model trained on ENNI, tested on SALT development sets

Corpus	Tag			Bracket		
	P	R	F1	P	R	F1
CONV	0.611	0.750	0.674	0.502	0.602	0.547
ENNI	0.744	0.727	0.735	0.608	0.639	0.623
EXPOSITORY	0.670	0.671	0.670	0.597	0.647	0.621
GILLAMNT	0.702	0.613	0.654	0.537	0.548	0.542
NARSSS	0.696	0.747	0.720	0.549	0.567	0.558
NARSR	0.725	0.737	0.731	0.553	0.578	0.565
NZCONV	0.674	0.825	0.742	0.574	0.727	0.641
NZPERNAR	0.705	0.841	0.767	0.546	0.701	0.614
NZSR	0.832	0.772	0.801	0.575	0.586	0.580

Table A.3: Model trained on EXPOSITORY, tested on SALT development sets

Corpus	Tag			Bracket		
	P	R	F1	P	R	F1
Conv	0.290	0.815	0.427	0.238	0.486	0.319
ENNI	0.653	0.784	0.712	0.608	0.665	0.635
Expository	0.363	0.762	0.492	0.370	0.496	0.424
GillamNT	0.892	0.881	0.886	0.795	0.803	0.799
NarSSS	0.601	0.798	0.686	0.503	0.595	0.545
NarSR	0.578	0.783	0.665	0.522	0.589	0.554
NZConv	0.393	0.874	0.542	0.343	0.610	0.439
NZPerNar	0.428	0.891	0.579	0.362	0.613	0.455
NZSR	0.730	0.814	0.770	0.570	0.658	0.611

Table A.4: Model trained on GillamNT, tested on SALT development sets

Corpus	Tag			Bracket		
	P	R	F1	P	R	F1
Conv	0.714	0.767	0.740	0.623	0.655	0.638
ENNI	0.742	0.781	0.761	0.607	0.657	0.631
Expository	0.614	0.638	0.626	0.548	0.620	0.582
GillamNT	0.716	0.643	0.678	0.572	0.569	0.570
NarSSS	0.796	0.794	0.795	0.648	0.651	0.649
NarSR	0.759	0.788	0.773	0.590	0.631	0.610
NZConv	0.761	0.837	0.797	0.685	0.739	0.711
NZPerNar	0.798	0.858	0.827	0.698	0.749	0.723
NZSR	0.902	0.802	0.849	0.636	0.613	0.624

Table A.5: Model trained on NarSSS, tested on SALT development sets

Corpus	Tag			Bracket		
	P	R	F1	P	R	F1
Conv	0.635	0.785	0.702	0.557	0.627	0.590
ENNI	0.718	0.792	0.753	0.604	0.671	0.636
Expository	0.575	0.660	0.614	0.539	0.643	0.587
GillamNT	0.691	0.660	0.675	0.571	0.580	0.576
NarSSS	0.745	0.805	0.774	0.616	0.643	0.629
NarSR	0.795	0.797	0.796	0.631	0.662	0.646
NZConv	0.726	0.855	0.785	0.655	0.734	0.692
NZPerNar	0.774	0.867	0.818	0.679	0.748	0.712
NZSR	0.893	0.821	0.855	0.661	0.649	0.655

Table A.6: Model trained on NarSR, tested on SALT development sets

Corpus	Tag			Bracket		
	P	R	F1	P	R	F1
CONV	0.776	0.728	0.751	0.660	0.641	0.650
ENNI	0.793	0.711	0.750	0.640	0.608	0.624
EXPOSITORY	0.677	0.563	0.615	0.525	0.519	0.522
GILLAMNT	0.773	0.581	0.664	0.601	0.544	0.571
NARSSS	0.817	0.743	0.778	0.667	0.642	0.654
NARSR	0.802	0.742	0.771	0.619	0.626	0.622
NZCONV	0.831	0.833	0.832	0.719	0.757	0.737
NZPERNAR	0.870	0.838	0.854	0.737	0.755	0.746
NZSR	0.958	0.779	0.860	0.709	0.658	0.682

Table A.7: Model trained on NZCONV, tested on SALT development sets

Corpus	Tag			Bracket		
	P	R	F1	P	R	F1
CONV	0.811	0.701	0.752	0.713	0.622	0.665
ENNI	0.804	0.705	0.751	0.677	0.592	0.632
EXPOSITORY	0.671	0.528	0.591	0.600	0.508	0.550
GILLAMNT	0.782	0.574	0.662	0.648	0.532	0.584
NARSSS	0.818	0.726	0.769	0.687	0.609	0.646
NARSR	0.806	0.723	0.762	0.658	0.600	0.628
NZCONV	0.862	0.815	0.838	0.761	0.755	0.758
NZPERNAR	0.882	0.840	0.861	0.778	0.759	0.768
NZSR	0.944	0.776	0.852	0.710	0.640	0.673

Table A.8: Model trained on NZPERNAR, tested on SALT development sets

Corpus	Tag			Bracket		
	P	R	F1	P	R	F1
CONV	0.441	0.789	0.566	0.375	0.518	0.435
ENNI	0.468	0.809	0.593	0.361	0.530	0.430
EXPOSITORY	0.328	0.658	0.438	0.260	0.398	0.315
GILLAMNT	0.454	0.656	0.536	0.333	0.444	0.381
NARSSS	0.557	0.816	0.662	0.432	0.541	0.480
NARSR	0.508	0.820	0.628	0.365	0.491	0.419
NZCONV	0.564	0.854	0.680	0.496	0.663	0.567
NZPERNAR	0.611	0.879	0.721	0.505	0.665	0.574
NZSR	0.836	0.875	0.855	0.661	0.703	0.681

Table A.9: Model trained on NZSR, tested on SALT development sets

A.2 Extrinsic evaluation of maze detection

These tables capture the complete results of our experiments using the simulated transcripts procedure (described in Section 5.5) to compare the four summary statistics (tokens per utterance (ie MLU), types per utterance, mazes per utterance, and maze length) derived from manual annotations to those derived from various maze detection models' predictions. As in Section 5.4, we do not evaluate multi-corpus models on the development folds of corpora that they were not trained on. Statistical significance is determined using the Wilcoxon signed-rank test ($\alpha = 0.1$), as was the case in Section 5.5.

A.2.1 Baseline models

Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
NARSSS	0.04	**0.001**	0.211	0.982	0.884
ENNI	0.06	**0.001**	**0.743**	0.974	0.863
NARSSS	0.07	**0.018**	0.075	0.974	0.860
GILLAMNT	0.05	**0.001**	0.250	0.983	0.886
NARSSS	0.06	0.103	0.052	0.970	0.848
NARSR	0.05	**0.095**	0.053	0.976	0.868
NZCONV	0.04	**0.014**	0.078	0.979	0.875
NZPERNAR	0.04	**0.001**	**0.590**	0.981	0.882
NZSR	0.05	**0.001**	**0.837**	0.969	0.853

(a) Token count

Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
NARSSS	0.03	0.233	0.038	0.991	0.918
ENNI	0.04	**0.001**	**0.777**	0.990	0.916
NARSSS	0.06	**0.001**	**0.544**	0.978	0.871
GILLAMNT	0.04	0.514	0.021	0.987	0.902
NARSSS	0.04	**0.001**	**0.591**	0.985	0.895
NARSR	0.04	**0.001**	**0.438**	0.986	0.900
NZCONV	0.03	**0.001**	**0.481**	0.990	0.915
NZPERNAR	0.04	**0.001**	**0.800**	0.991	0.919
NZSR	0.05	**0.001**	**0.864**	0.980	0.887

(b) Type count

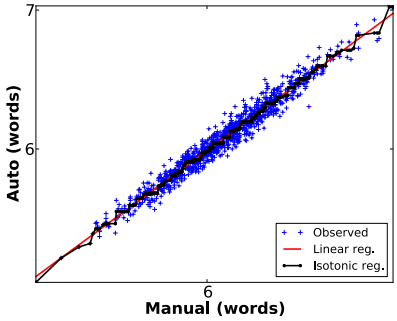
Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
NARSSS	0.02	**0.001**	**0.374**	0.841	0.653
ENNI	0.02	**0.001**	**0.377**	0.863	0.689
NARSSS	0.09	**0.001**	**0.866**	0.736	0.537
GILLAMNT	0.02	**0.001**	**0.698**	0.866	0.688
NARSSS	0.03	**0.001**	**0.546**	0.794	0.598
NARSR	0.04	**0.001**	**0.818**	0.791	0.609
NZCONV	0.03	**0.001**	**0.797**	0.844	0.670
NZPERNAR	0.01	**0.002**	0.097	0.861	0.689
NZSR	0.03	**0.001**	**0.861**	0.741	0.572

(c) Maze count

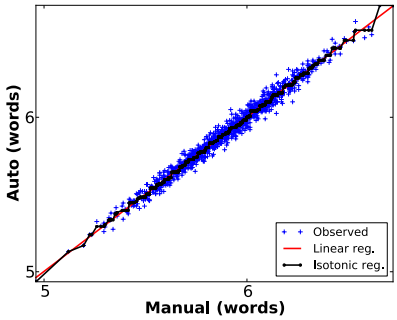
Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
NARSSS	0.14	**0.001**	0.267	0.595	0.422
ENNI	0.18	**0.001**	**0.796**	0.686	0.490
NARSSS	0.37	**0.001**	**0.865**	0.169	0.100
GILLAMNT	0.10	**0.001**	**0.464**	0.855	0.661
NARSSS	0.11	**0.001**	0.237	0.589	0.409
NARSR	0.14	**0.001**	**0.680**	0.649	0.439
NZCONV	0.15	**0.001**	**0.534**	0.688	0.492
NZPERNAR	0.12	**0.001**	**0.429**	0.800	0.581
NZSR	0.13	**0.001**	**0.819**	0.696	0.490

(d) Maze length

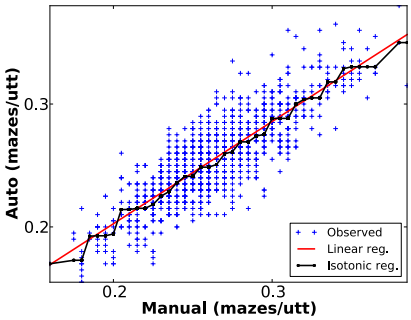
Table A.10: Reference statistic prediction accuracy using baseline maze detection models; all values of ρ and τ significant at $p \leq 0.001$ level



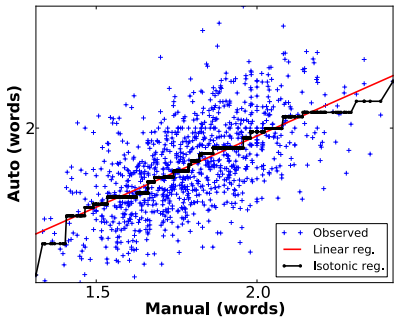
(a) Token count



(b) Type count

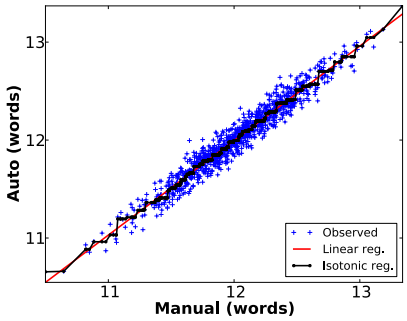


(c) Maze count

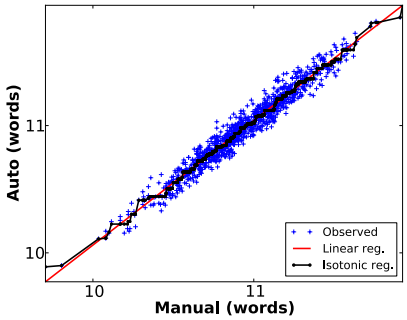


(d) Maze length

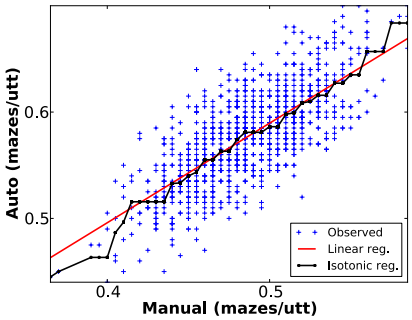
Figure A.1: Relationship between summary statistics computed from manual transcripts and baseline model; development fold of Conv corpus



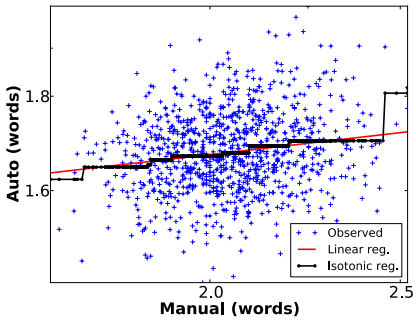
(a) Token count



(b) Type count

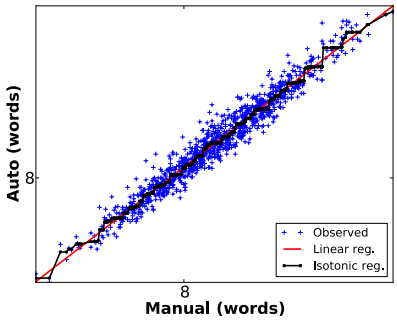


(c) Maze count

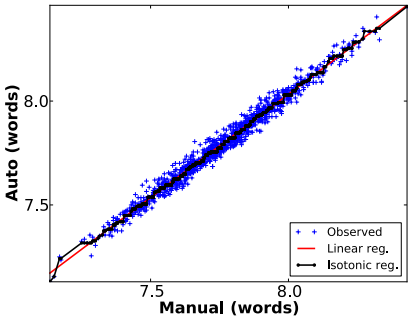


(d) Maze length

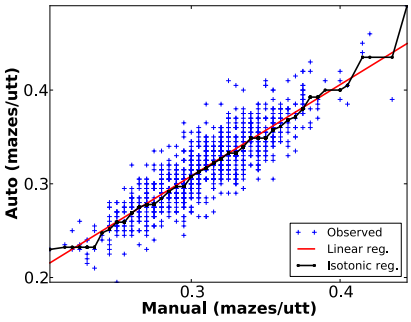
Figure A.2: Relationship between summary statistics computed from manual transcripts and baseline model; development fold of EXPOSITORY corpus



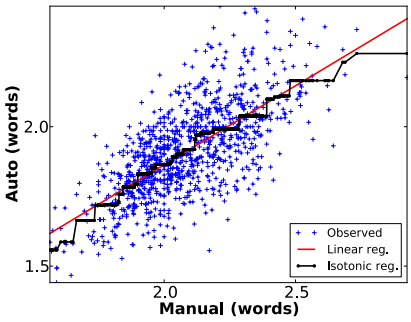
(a) Token count



(b) Type count

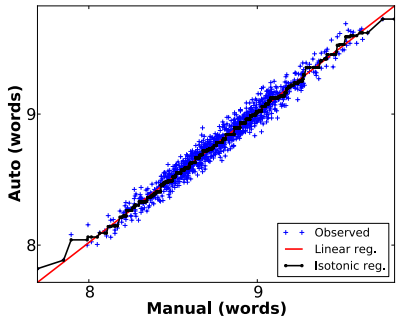


(c) Maze count

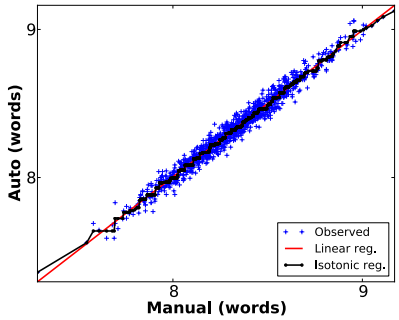


(d) Maze length

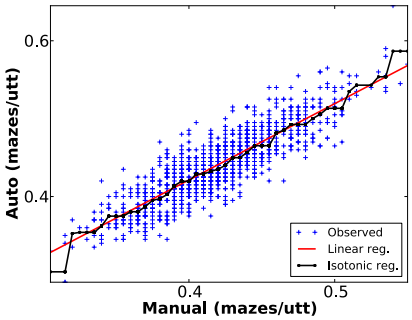
Figure A.3: Relationship between summary statistics computed from manual transcripts and baseline model; development fold of ENNI corpus



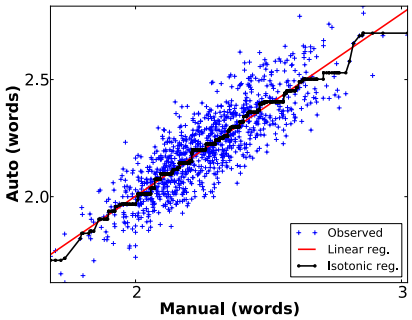
(a) Token count



(b) Type count

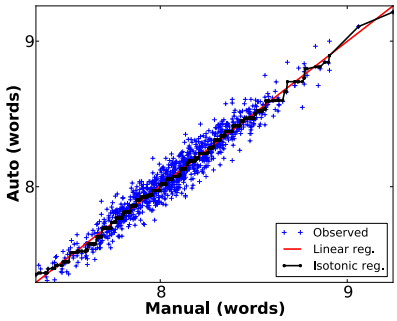


(c) Maze count

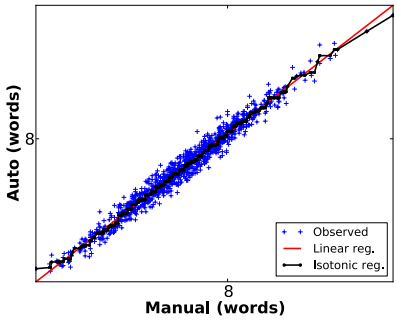


(d) Maze length

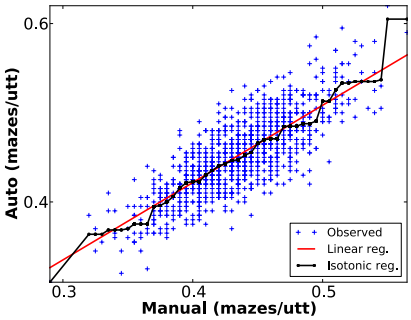
Figure A.4: Relationship between summary statistics computed from manual transcripts and baseline model; development fold of GillamNT corpus



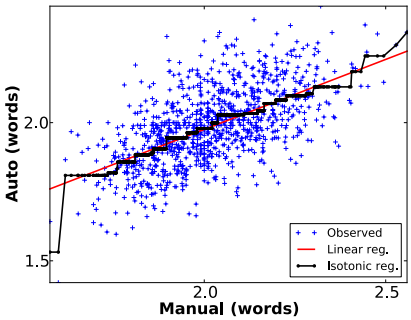
(a) Token count



(b) Type count

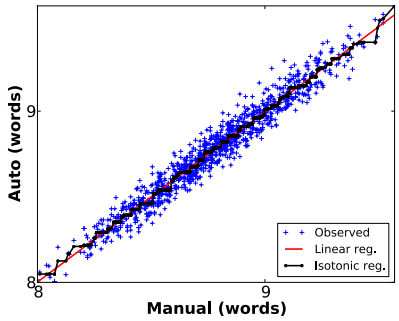


(c) Maze count

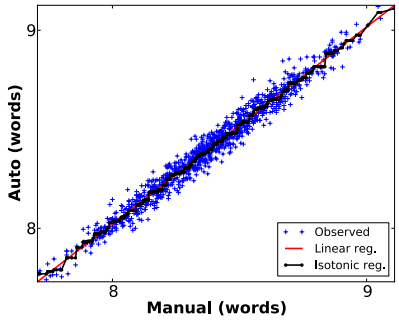


(d) Maze length

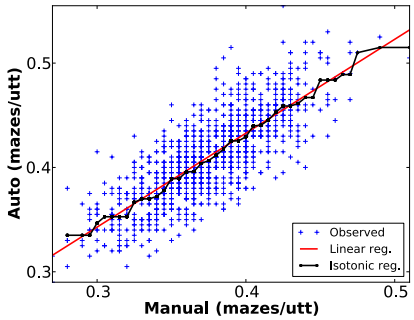
Figure A.5: Relationship between summary statistics computed from manual transcripts and baseline model; development fold of NarSSS corpus



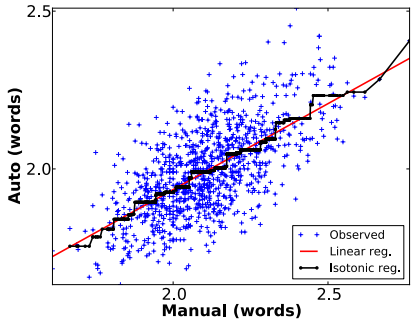
(a) Token count



(b) Type count

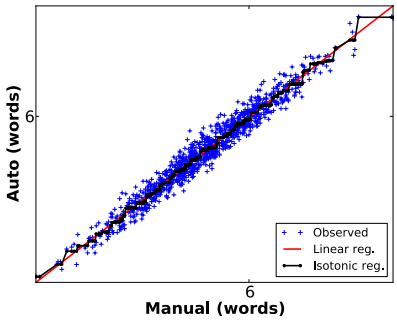


(c) Maze count

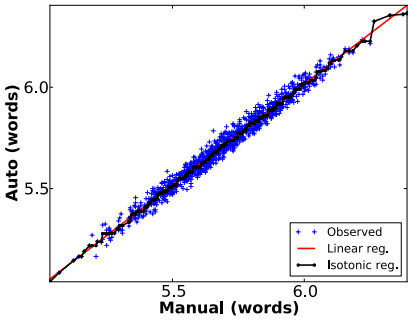


(d) Maze length

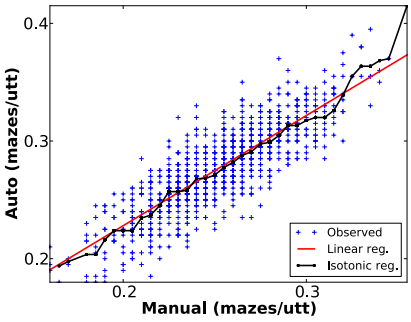
Figure A.6: Relationship between summary statistics computed from manual transcripts and baseline model; development fold of NarSR corpus



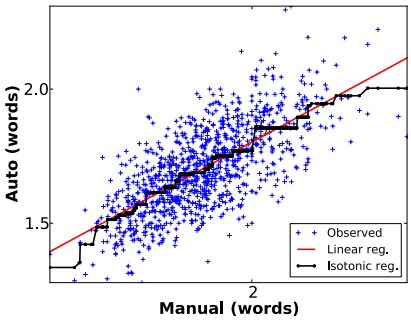
(a) Token count



(b) Type count

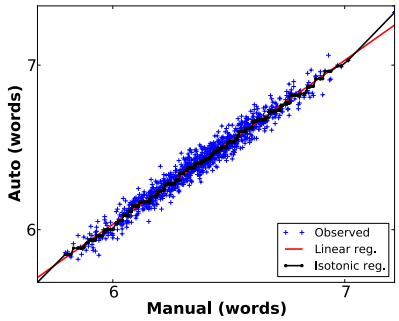


(c) Maze count

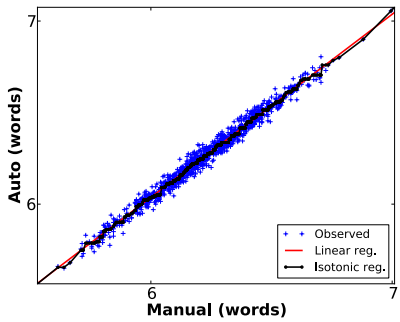


(d) Maze length

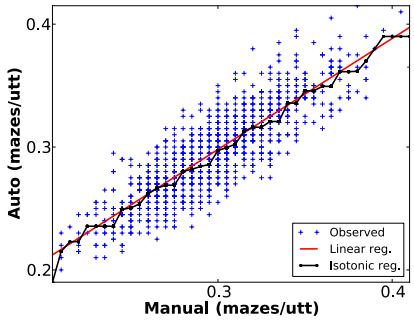
Figure A.7: Relationship between summary statistics computed from manual transcripts and baseline model; development fold of NZCONV corpus



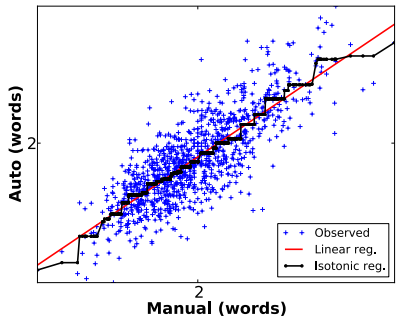
(a) Token count



(b) Type count

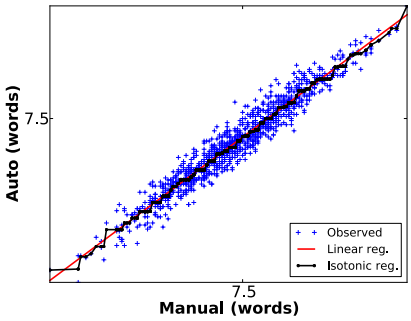


(c) Maze count

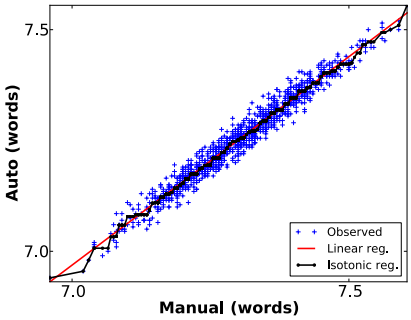


(d) Maze length

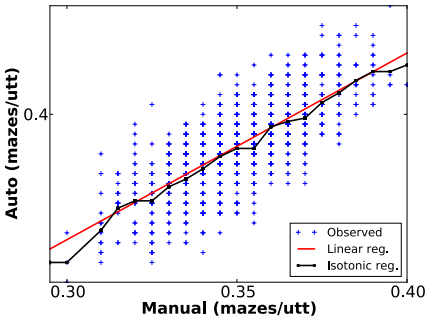
Figure A.8: Relationship between summary statistics computed from manual transcripts and baseline model; development fold of NZPERNAR corpus



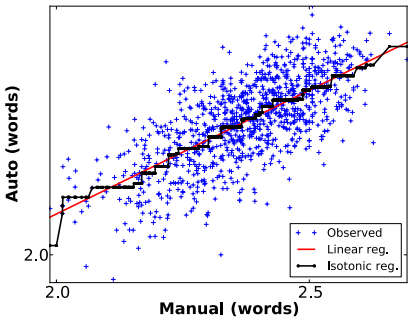
(a) Token count



(b) Type count



(c) Maze count



(d) Maze length

Figure A.9: Relationship between summary statistics computed from manual transcripts and baseline model; development fold of NZSR corpus

A.2.2 FEDA All model

Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
NarSSS	0.04	**0.001**	0.291	0.984	0.891
ENNI	0.06	**0.001**	**0.751**	0.975	0.865
NarSSS	0.07	0.586	0.017	0.974	0.855
GillamNT	0.05	**0.001**	0.198	0.981	0.880
NarSSS	0.06	0.421	0.025	0.968	0.842
NarSR	0.05	0.674	0.013	0.975	0.863
NZConv	0.04	**0.006**	0.087	0.980	0.882
NZPerNar	0.04	**0.001**	**0.573**	0.981	0.882
NZSR	0.05	**0.001**	**0.825**	0.964	0.838

(a) Token count

Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
NarSSS	0.03	0.457	0.023	0.991	0.922
ENNI	0.04	**0.001**	**0.788**	0.990	0.916
NarSSS	0.06	**0.001**	**0.562**	0.978	0.870
GillamNT	0.04	0.721	0.011	0.985	0.895
NarSSS	0.04	**0.001**	**0.573**	0.986	0.896
NarSR	0.04	**0.001**	**0.445**	0.985	0.891
NZConv	0.03	**0.001**	**0.471**	0.990	0.920
NZPerNar	0.04	**0.001**	**0.791**	0.991	0.919
NZSR	0.05	**0.001**	**0.858**	0.975	0.870

(b) Type count

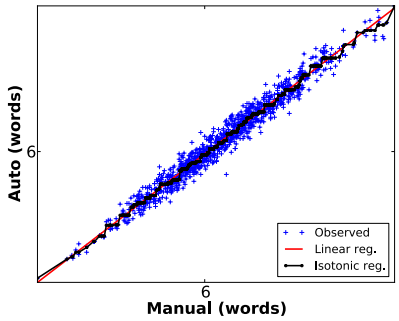
Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
NarSSS	0.02	**0.001**	**0.368**	0.826	0.638
ENNI	0.02	**0.001**	**0.358**	0.858	0.679
NarSSS	0.09	**0.001**	**0.866**	0.740	0.549
GillamNT	0.03	**0.001**	**0.713**	0.857	0.678
NarSSS	0.03	**0.001**	**0.584**	0.813	0.626
NarSR	0.04	**0.001**	**0.829**	0.778	0.586
NZConv	0.02	**0.001**	**0.784**	0.846	0.664
NZPerNar	0.02	**0.001**	0.128	0.854	0.675
NZSR	0.03	**0.001**	**0.856**	0.766	0.595

(c) Maze count

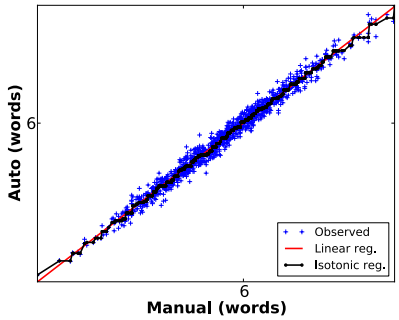
Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
NarSSS	0.14	**0.001**	**0.354**	0.601	0.419
ENNI	0.18	**0.001**	**0.784**	0.712	0.503
NarSSS	0.38	**0.001**	**0.865**	0.223	0.136
GillamNT	0.10	**0.001**	**0.392**	0.838	0.642
NarSSS	0.11	**0.001**	0.210	0.602	0.415
NarSR	0.15	**0.001**	**0.732**	0.629	0.431
NZConv	0.14	**0.001**	**0.492**	0.656	0.457
NZPerNar	0.11	**0.001**	**0.375**	0.793	0.587
NZSR	0.13	**0.001**	**0.804**	0.691	0.484

(d) Maze length

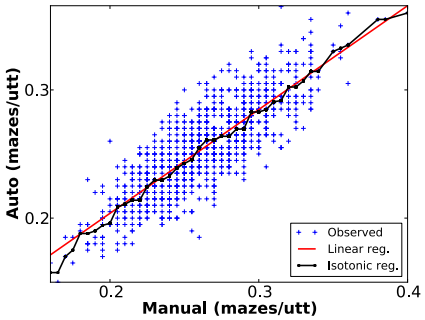
Table A.11: Reference statistic prediction accuracy using FEDA All maze detection model; all values of ρ and τ significant at $p \leq 0.001$ level



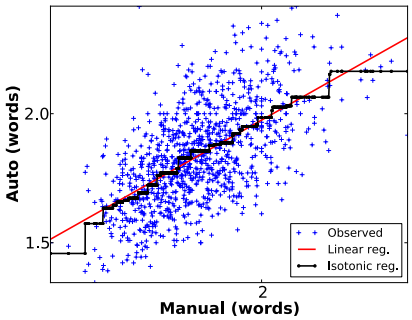
(a) Token count



(b) Type count

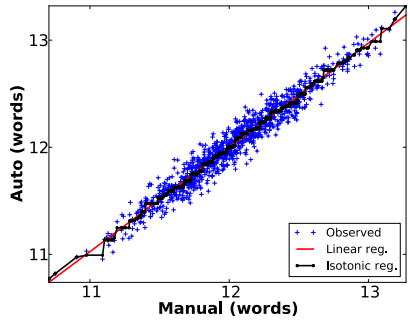


(c) Maze count

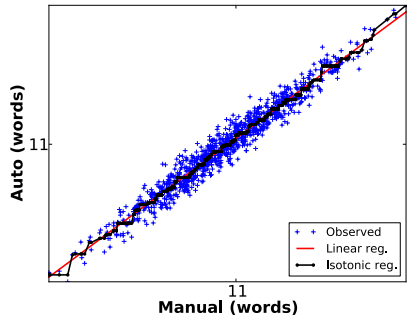


(d) Maze length

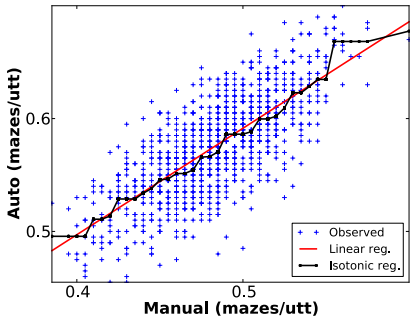
Figure A.10: Relationship between summary statistics computed from manual transcripts and FEDA ALL model; development fold of CONV corpus



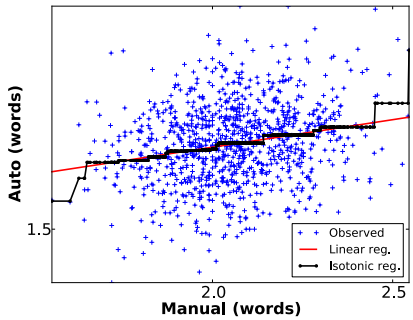
(a) Token count



(b) Type count

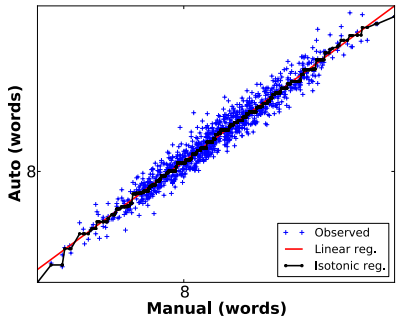


(c) Maze count

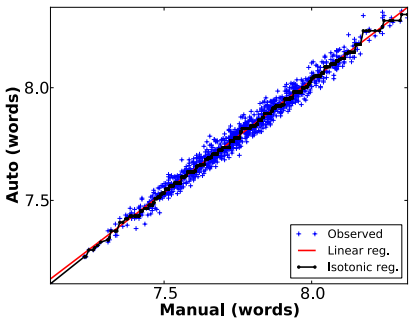


(d) Maze length

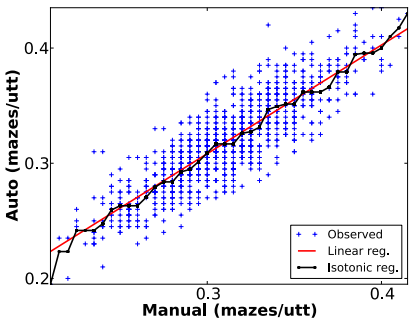
Figure A.11: Relationship between summary statistics computed from manual transcripts and FEDA ALL model; development fold of EXPOSITORY corpus



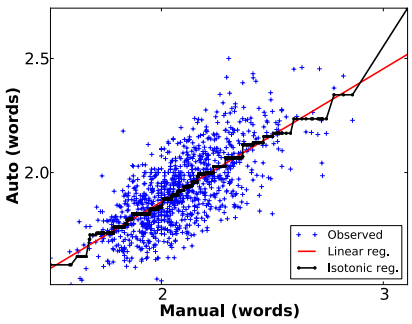
(a) Token count



(b) Type count

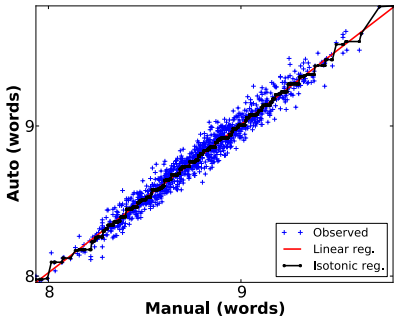


(c) Maze count

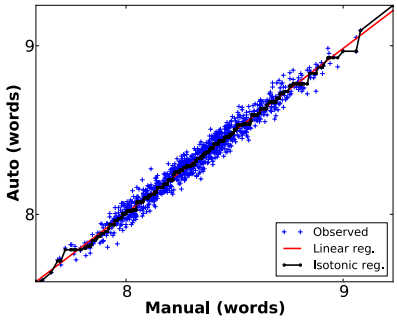


(d) Maze length

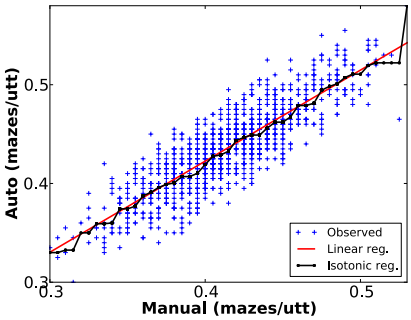
Figure A.12: Relationship between summary statistics computed from manual transcripts and FEDA ALL model; development fold of ENNI corpus



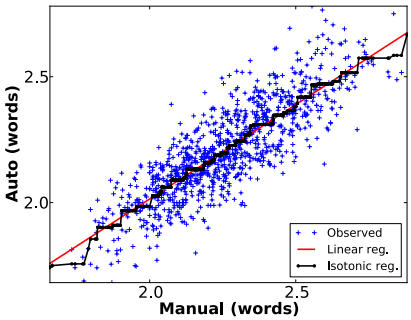
(a) Token count



(b) Type count

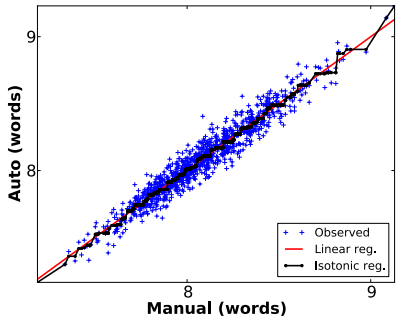


(c) Maze count

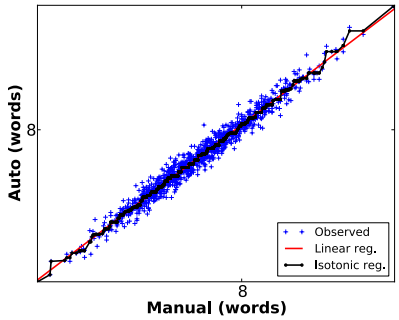


(d) Maze length

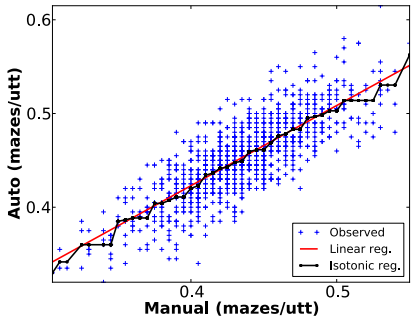
Figure A.13: Relationship between summary statistics computed from manual transcripts and FEDA ALL model; development fold of GILLAMNT corpus



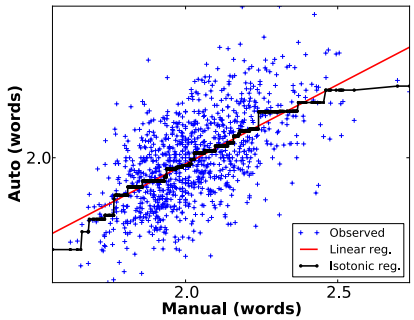
(a) Token count



(b) Type count

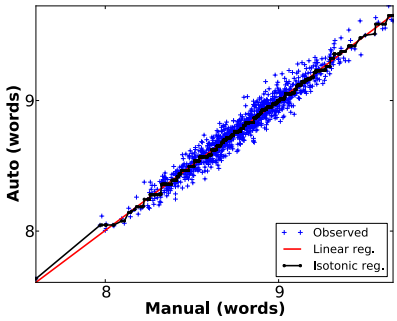


(c) Maze count

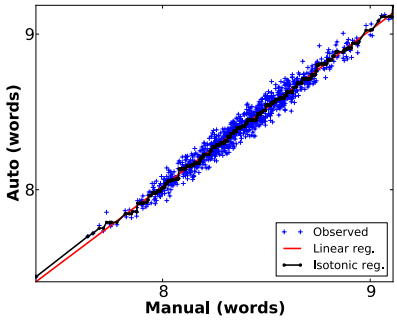


(d) Maze length

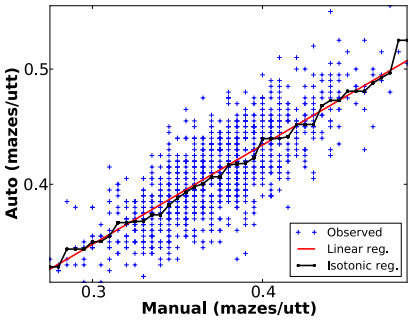
Figure A.14: Relationship between summary statistics computed from manual transcripts and FEDA ALL model; development fold of NARSSS corpus



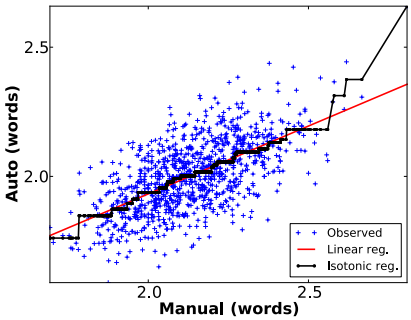
(a) Token count



(b) Type count

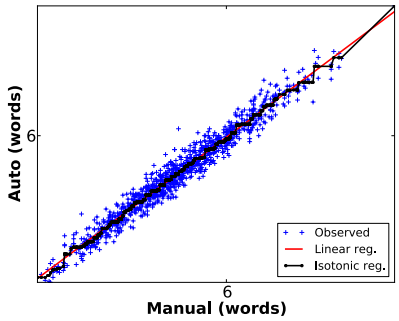


(c) Maze count

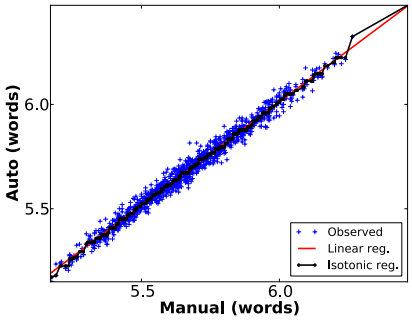


(d) Maze length

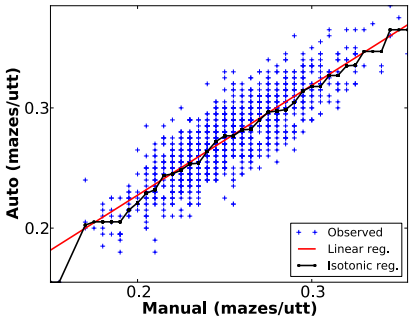
Figure A.15: Relationship between summary statistics computed from manual transcripts and FEDA ALL model; development fold of NARSR corpus



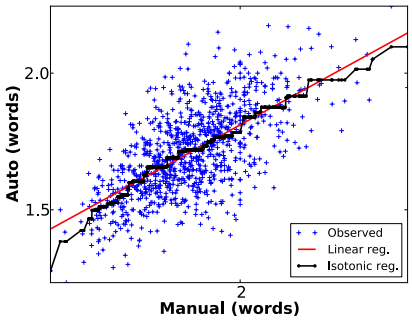
(a) Token count



(b) Type count

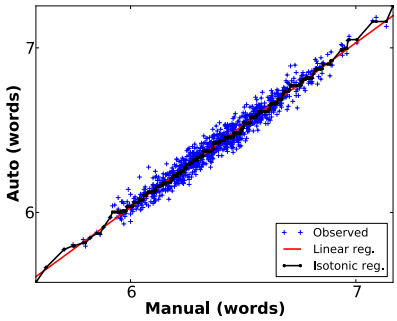


(c) Maze count

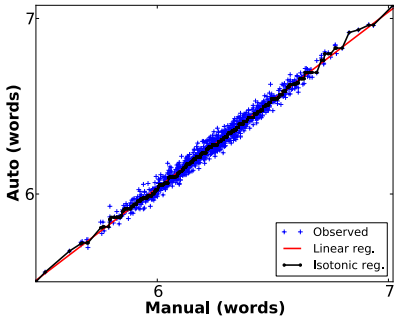


(d) Maze length

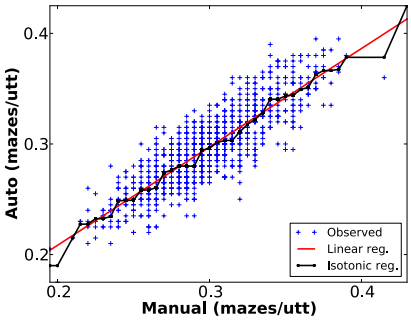
Figure A.16: Relationship between summary statistics computed from manual transcripts and FEDA ALL model; development fold of NZCONV corpus



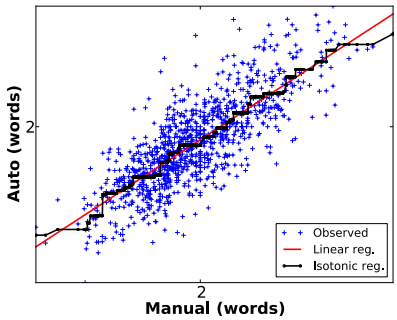
(a) Token count



(b) Type count

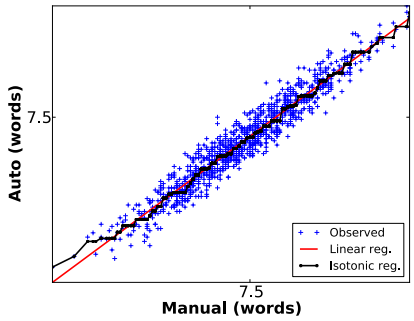


(c) Maze count

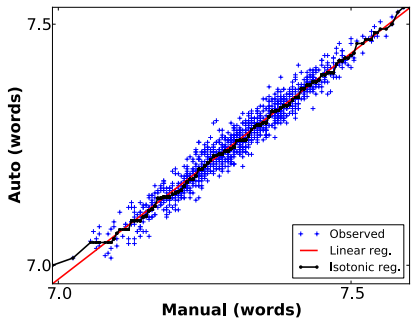


(d) Maze length

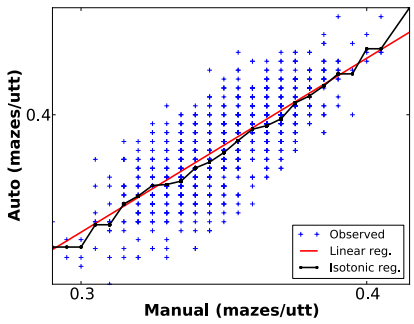
Figure A.17: Relationship between summary statistics computed from manual transcripts and FEDA ALL model; development fold of NZPERNAR corpus



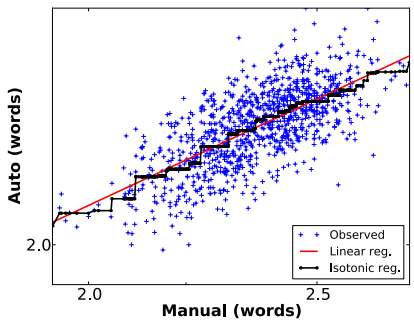
(a) Token count



(b) Type count



(c) Maze count



(d) Maze length

Figure A.18: Relationship between summary statistics computed from manual transcripts and FEDA ALL model; development fold of NZSR corpus

A.2.3 AGE model

Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
ENNI	0.06	**0.001**	**0.742**	0.975	0.865
GILLAMNT	0.05	**0.001**	0.213	0.983	0.883
NARSSS	0.05	0.534	0.020	0.970	0.845
NARSR	0.05	**0.008**	0.084	0.976	0.864

(a) Token count

Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
ENNI	0.04	**0.001**	**0.797**	0.990	0.917
GILLAMNT	0.04	0.801	0.008	0.986	0.898
NARSSS	0.04	**0.001**	**0.560**	0.986	0.895
NARSR	0.04	**0.001**	**0.381**	0.984	0.893

(b) Type count

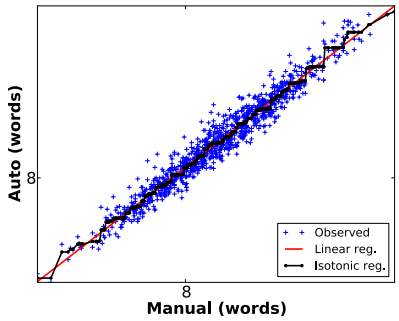
Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
ENNI	0.02	**0.001**	**0.375**	0.856	0.675
GILLAMNT	0.02	**0.001**	**0.707**	0.867	0.693
NARSSS	0.03	**0.001**	**0.555**	0.796	0.612
NARSR	0.04	**0.001**	**0.818**	0.780	0.591

(c) Maze count

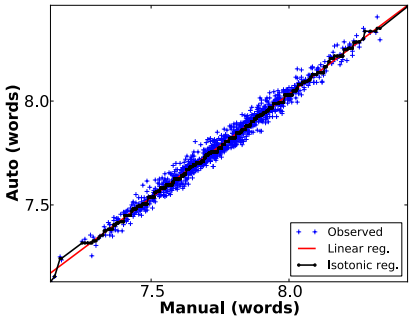
Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
ENNI	0.18	**0.001**	**0.799**	0.720	0.516
GILLAMNT	0.10	**0.001**	**0.409**	0.843	0.645
NARSSS	0.11	**0.001**	0.192	0.594	0.415
NARSR	0.14	**0.001**	**0.711**	0.642	0.451

(d) Maze length

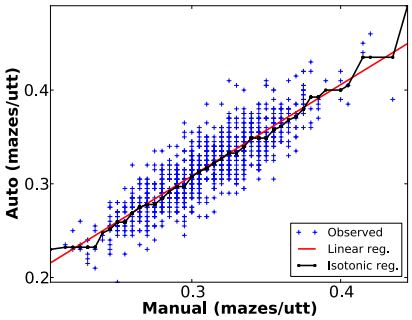
Table A.12: Reference statistic prediction accuracy using AGE maze detection model; all values of ρ and τ significant at $p \leq 0.001$ level



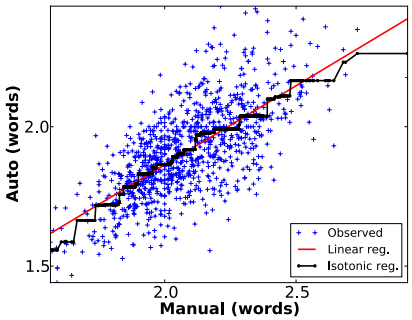
(a) Token count



(b) Type count

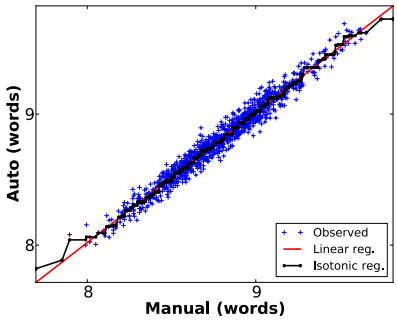


(c) Maze count

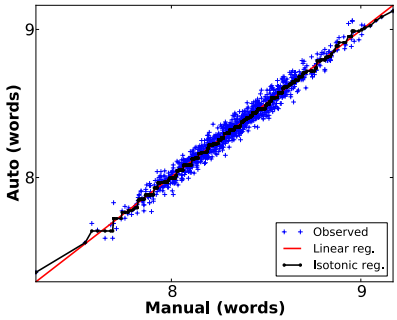


(d) Maze length

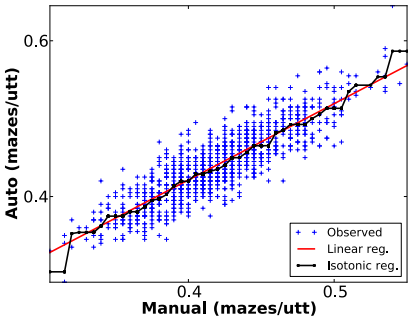
Figure A.19: Relationship between summary statistics computed from manual transcripts and FEDA AGE model; development fold of ENNI corpus



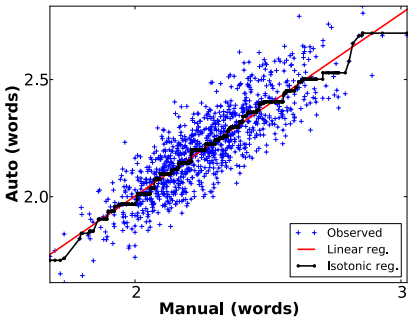
(a) Token count



(b) Type count

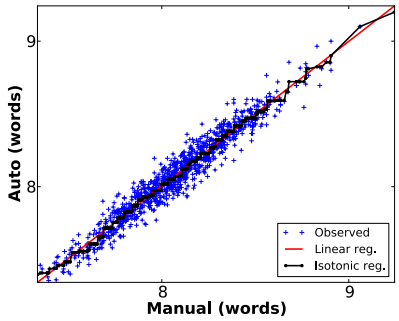


(c) Maze count

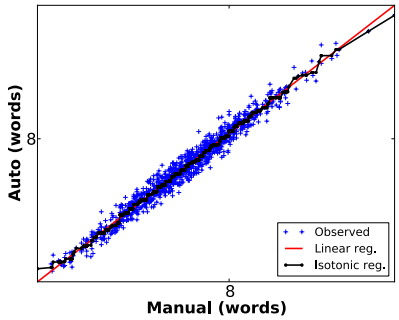


(d) Maze length

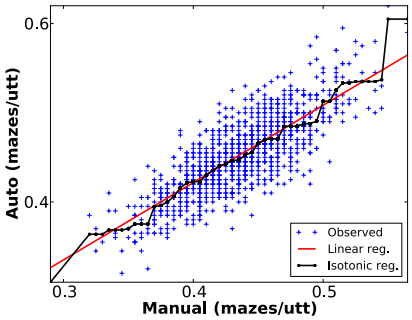
Figure A.20: Relationship between summary statistics computed from manual transcripts and FEDA AGE model; development fold of GILLAMNT corpus



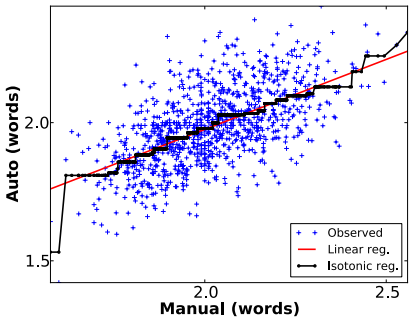
(a) Token count



(b) Type count

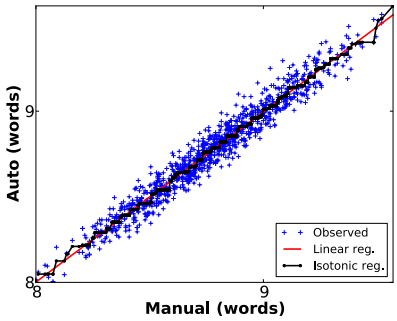


(c) Maze count

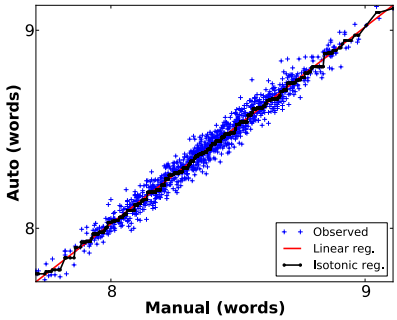


(d) Maze length

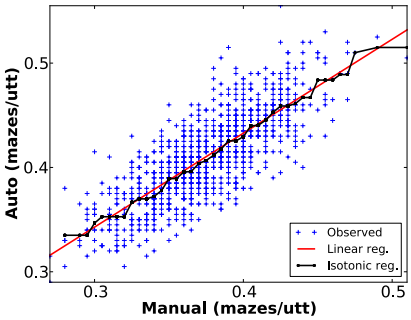
Figure A.21: Relationship between summary statistics computed from manual transcripts and FEDA AGE model; development fold of NARSSS corpus



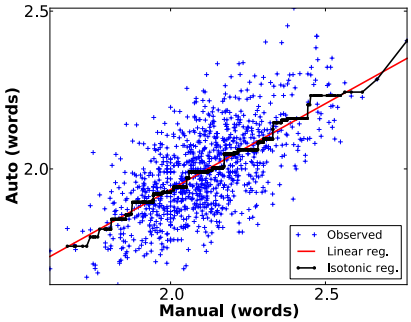
(a) Token count



(b) Type count



(c) Maze count



(d) Maze length

Figure A.22: Relationship between summary statistics computed from manual transcripts and FEDA AGE model; development fold of NARSR corpus

A.2.4 Conversational model

Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
Conv	0.04	**0.001**	0.276	0.983	0.888
NZConv	0.04	0.106	0.051	0.978	0.872

(a) Token count

Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
Conv	0.03	0.895	0.004	0.991	0.919
NZConv	0.03	**0.001**	**0.498**	0.989	0.913

(b) Type count

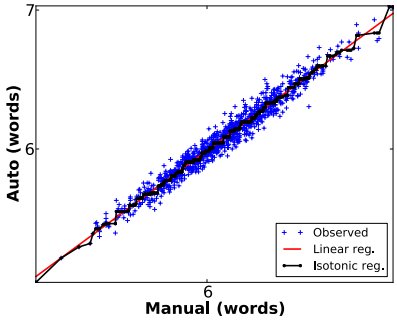
Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
Conv	0.02	**0.001**	**0.369**	0.845	0.662
NZConv	0.03	**0.001**	**0.783**	0.842	0.660

(c) Maze count

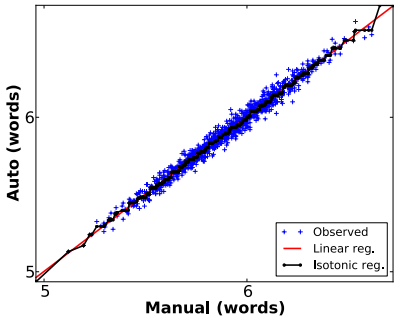
Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
Conv	0.15	**0.001**	**0.332**	0.593	0.415
NZConv	0.15	**0.001**	**0.525**	0.632	0.442

(d) Maze length

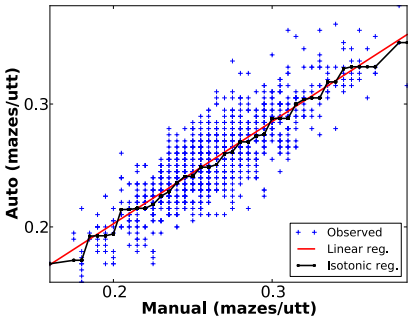
Table A.13: Reference statistic prediction accuracy using Conversational maze detection model; all values of ρ and τ significant at $p \leq 0.001$ level



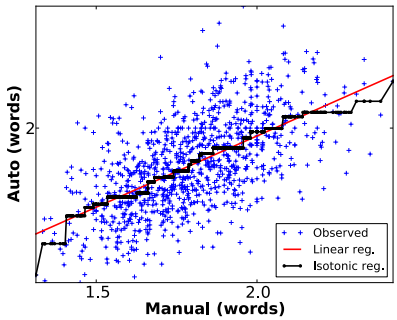
(a) Token count



(b) Type count

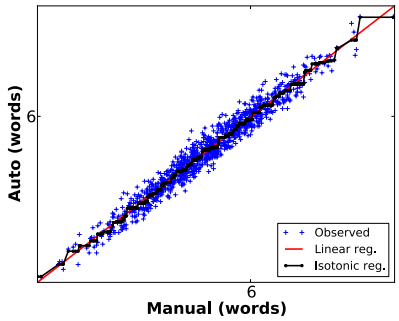


(c) Maze count

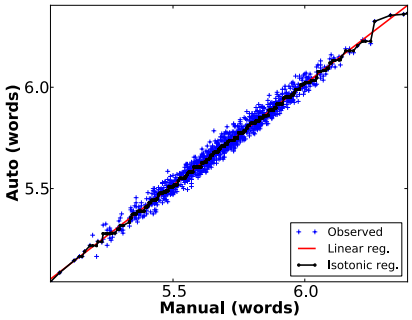


(d) Maze length

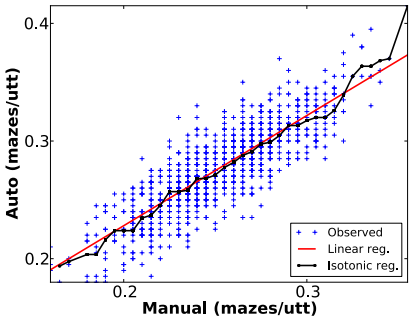
Figure A.23: Relationship between summary statistics computed from manual transcripts and FEDA Conversational model; development fold of Conv corpus



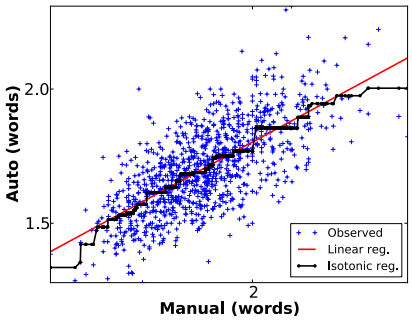
(a) Token count



(b) Type count



(c) Maze count



(d) Maze length

Figure A.24: Relationship between summary statistics computed from manual transcripts and FEDA CONVERSATIONAL model; development fold of NZCONV corpus

A.2.5 NARRATIVE model

Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
ENNI	0.06	**0.001**	**0.716**	0.973	0.860
GILLAMNT	0.05	**0.001**	0.238	0.981	0.880
NARSSS	0.06	**0.050**	0.062	0.970	0.845
NARSR	0.05	**0.071**	0.057	0.973	0.860
NZPERNAR	0.04	**0.001**	**0.563**	0.978	0.874
NZSR	0.05	**0.001**	**0.841**	0.969	0.853

(a) Token count

Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
ENNI	0.04	**0.001**	**0.765**	0.990	0.916
GILLAMNT	0.04	0.366	0.029	0.986	0.897
NARSSS	0.04	**0.001**	**0.598**	0.986	0.895
NARSR	0.04	**0.001**	**0.397**	0.982	0.887
NZPERNAR	0.04	**0.001**	**0.800**	0.989	0.915
NZSR	0.05	**0.001**	**0.863**	0.978	0.879

(b) Type count

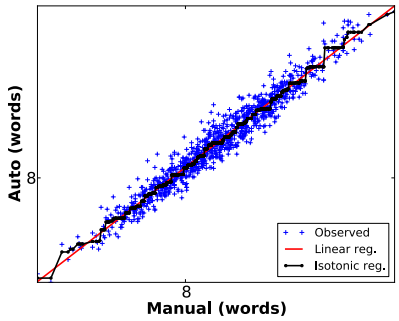
Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
ENNI	0.02	**0.001**	**0.423**	0.872	0.695
GILLAMNT	0.03	**0.001**	**0.711**	0.868	0.696
NARSSS	0.02	**0.001**	**0.583**	0.819	0.630
NARSR	0.04	**0.001**	**0.826**	0.818	0.643
NZPERNAR	0.01	**0.001**	0.138	0.862	0.689
NZSR	0.03	**0.001**	**0.861**	0.762	0.586

(c) Maze count

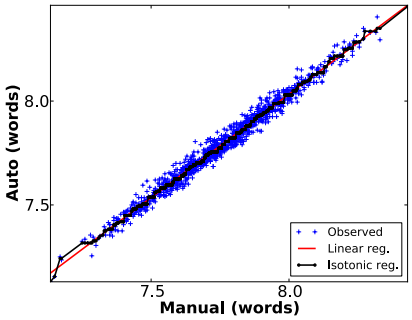
Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
ENNI	0.18	**0.001**	**0.781**	0.677	0.472
GILLAMNT	0.10	**0.001**	**0.436**	0.848	0.645
NARSSS	0.11	**0.001**	0.238	0.626	0.431
NARSR	0.15	**0.001**	**0.703**	0.621	0.430
NZPERNAR	0.12	**0.001**	**0.391**	0.778	0.570
NZSR	0.13	**0.001**	**0.814**	0.707	0.491

(d) Maze length

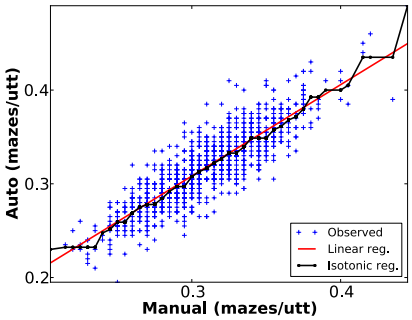
Table A.14: Reference statistic prediction accuracy using NARRATIVE maze detection model; all values of ρ and τ significant at $p \leq 0.001$ level



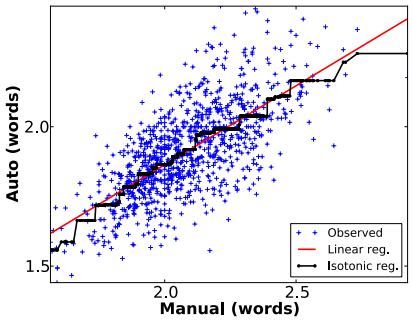
(a) Token count



(b) Type count

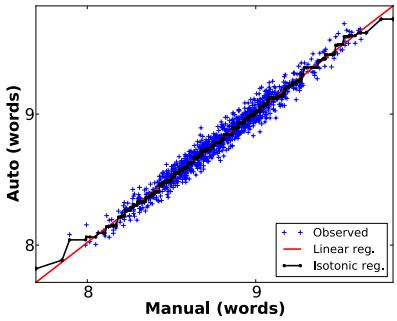


(c) Maze count

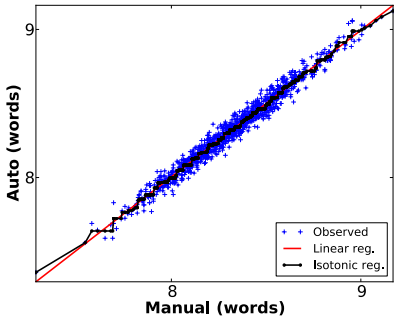


(d) Maze length

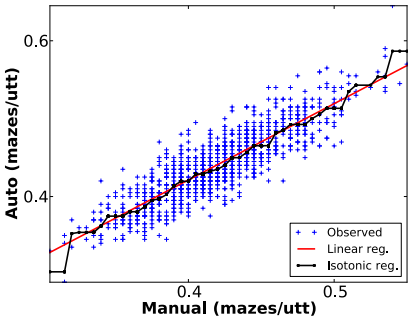
Figure A.25: Relationship between summary statistics computed from manual transcripts and FEDA NARRATIVE model; development fold of ENNI corpus



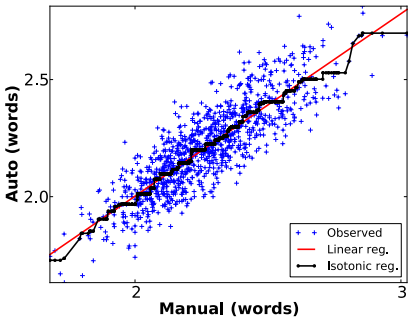
(a) Token count



(b) Type count

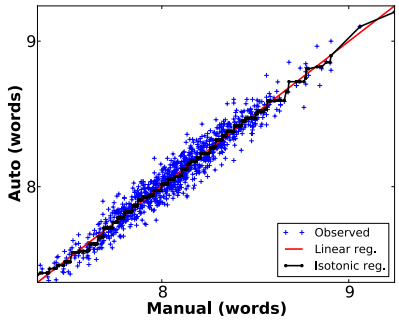


(c) Maze count

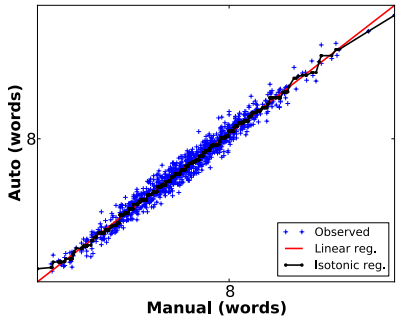


(d) Maze length

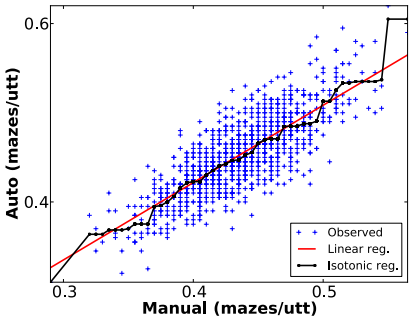
Figure A.26: Relationship between summary statistics computed from manual transcripts and FEDA NARRATIVE model; development fold of GILLAMNT corpus



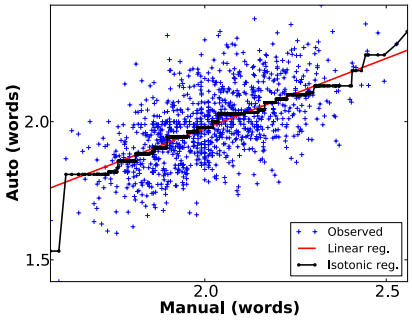
(a) Token count



(b) Type count

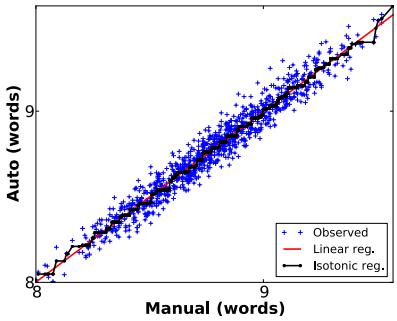


(c) Maze count

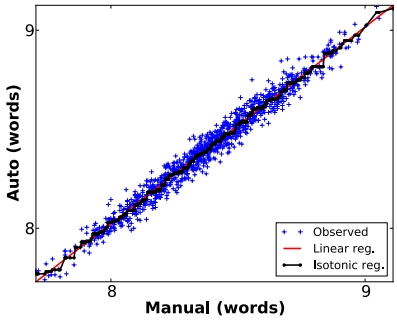


(d) Maze length

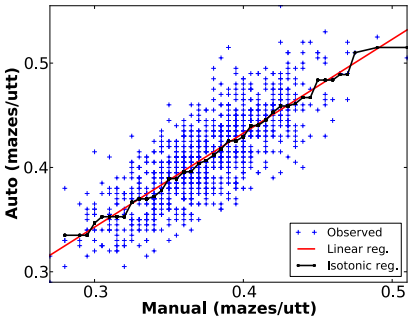
Figure A.27: Relationship between summary statistics computed from manual transcripts and FEDA NARRATIVE model; development fold of NARSSS corpus



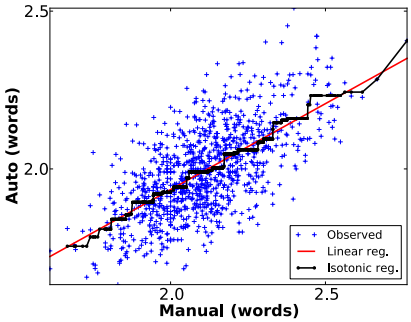
(a) Token count



(b) Type count

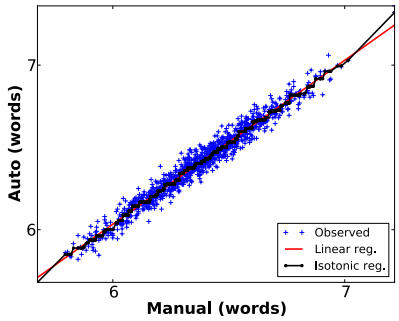


(c) Maze count

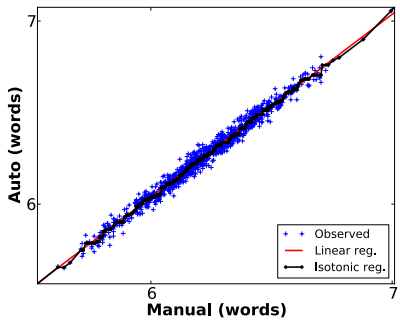


(d) Maze length

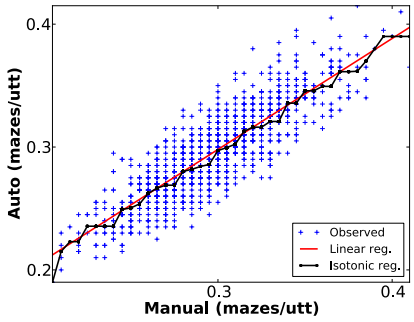
Figure A.28: Relationship between summary statistics computed from manual transcripts and FEDA NARRATIVE model; development fold of NARSR corpus



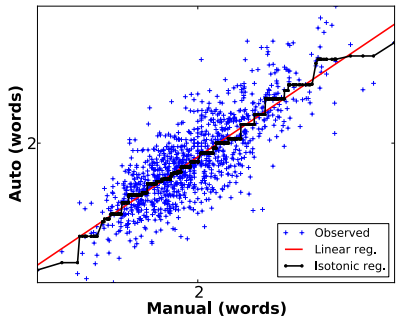
(a) Token count



(b) Type count

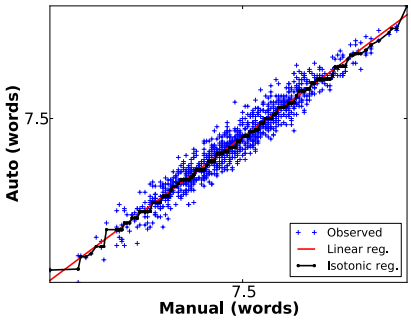


(c) Maze count

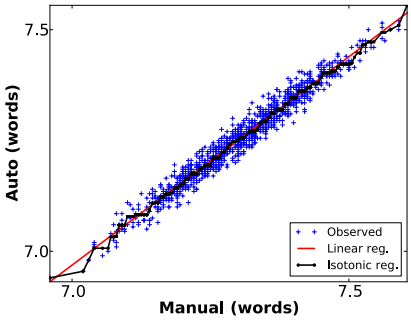


(d) Maze length

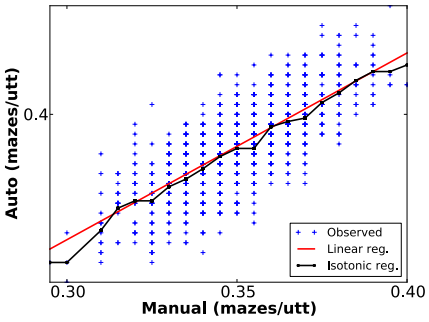
Figure A.29: Relationship between summary statistics computed from manual transcripts and FEDA NARRATIVE model; development fold of NZPERNAR corpus



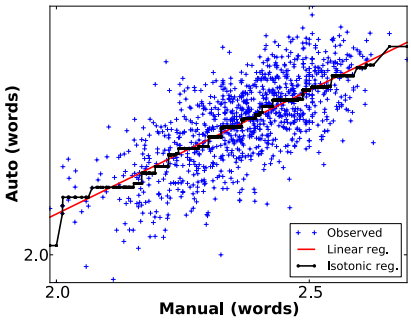
(a) Token count



(b) Type count



(c) Maze count



(d) Maze length

Figure A.30: Relationship between summary statistics computed from manual transcripts and FEDA NARRATIVE model; development fold of NZSR corpus

A.2.6 NZ model

Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
NZConv	0.04	0.466	0.023	0.977	0.872
NZPerNar	0.04	**0.001**	**0.600**	0.981	0.884
NZSR	0.05	**0.001**	**0.833**	0.966	0.844

(a) Token count

Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
NZConv	0.03	**0.001**	**0.520**	0.990	0.916
NZPerNar	0.04	**0.001**	**0.803**	0.990	0.920
NZSR	0.05	**0.001**	**0.861**	0.976	0.875

(b) Type count

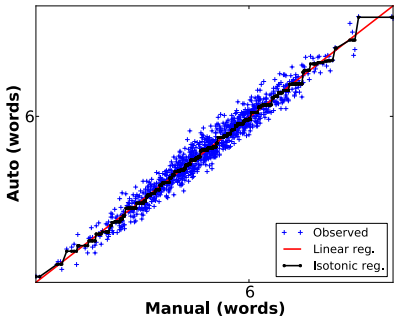
Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
NZConv	0.03	**0.001**	**0.792**	0.839	0.654
NZPerNar	0.02	**0.001**	0.103	0.855	0.678
NZSR	0.03	**0.001**	**0.860**	0.772	0.601

(c) Maze count

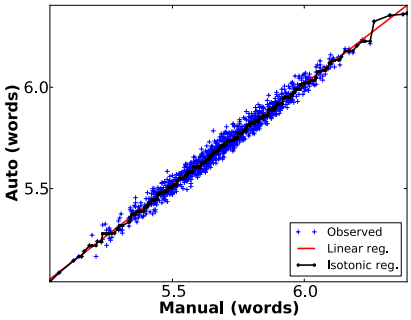
Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
NZConv	0.15	**0.001**	**0.546**	0.649	0.466
NZPerNar	0.11	**0.001**	**0.441**	0.804	0.608
NZSR	0.13	**0.001**	**0.816**	0.722	0.501

(d) Maze length

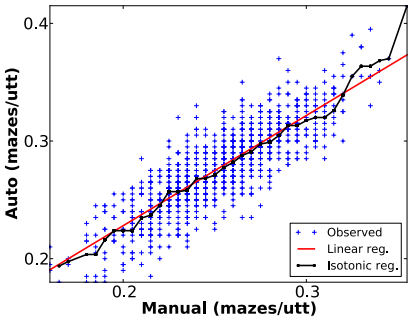
Table A.15: Reference statistic prediction accuracy using NZ maze detection model; all values of ρ and τ significant at $p \leq 0.001$ level



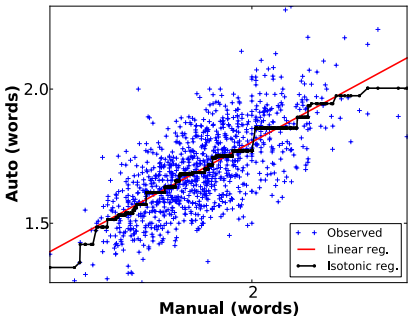
(a) Token count



(b) Type count

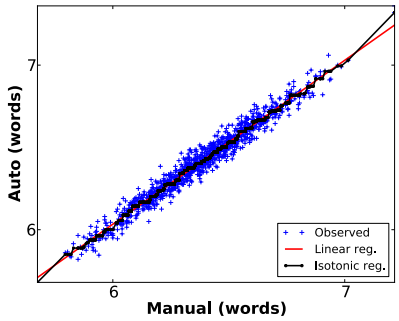


(c) Maze count

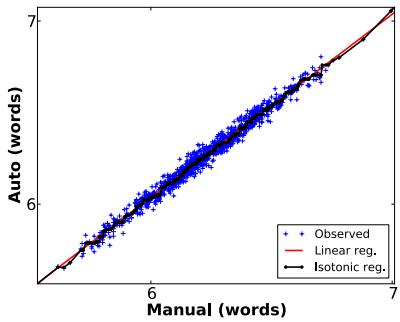


(d) Maze length

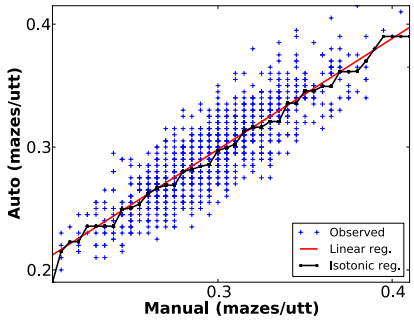
Figure A.31: Relationship between summary statistics computed from manual transcripts and FEDA NZ model; development fold of NZCONV corpus



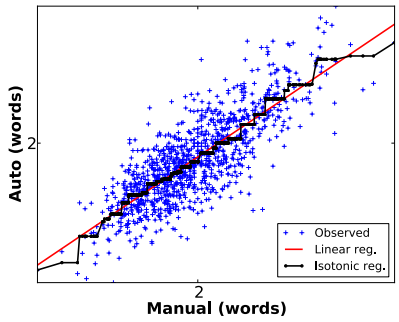
(a) Token count



(b) Type count

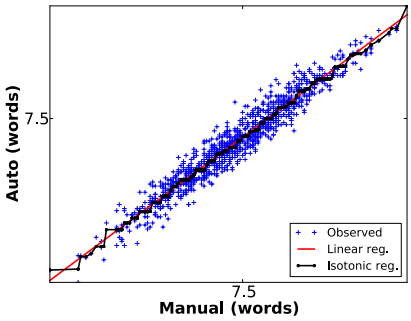


(c) Maze count

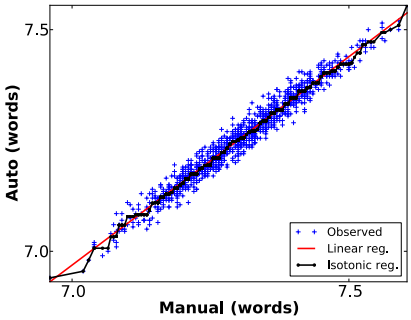


(d) Maze length

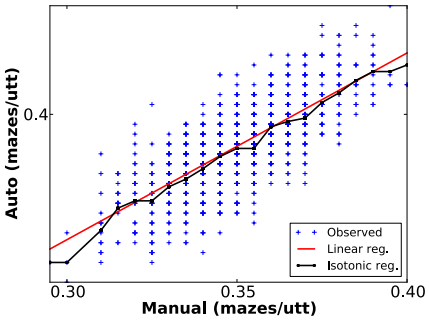
Figure A.32: Relationship between summary statistics computed from manual transcripts and FEDA NZ model; development fold of NZPERNAR corpus



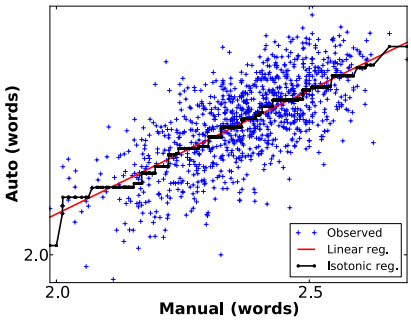
(a) Token count



(b) Type count



(c) Maze count



(d) Maze length

Figure A.33: Relationship between summary statistics computed from manual transcripts and FEDA NZ model; development fold of NZSR corpus

A.2.7 WI model

Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
CONV	0.05	**0.001**	0.266	0.982	0.882
EXPOSITORY	0.08	0.152	0.045	0.974	0.858
NARSSS	0.06	0.122	0.049	0.971	0.849
NARSR	0.05	**0.011**	0.080	0.974	0.860

(a) Token count

Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
CONV	0.03	0.792	0.008	0.990	0.914
EXPOSITORY	0.07	**0.001**	**0.586**	0.978	0.868
NARSSS	0.04	**0.001**	**0.568**	0.986	0.897
NARSR	0.04	**0.001**	**0.391**	0.984	0.892

(b) Type count

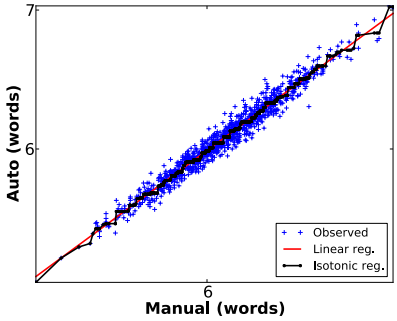
Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
CONV	0.02	**0.001**	**0.400**	0.829	0.642
EXPOSITORY	0.09	**0.001**	**0.866**	0.770	0.578
NARSSS	0.02	**0.001**	**0.585**	0.822	0.627
NARSR	0.04	**0.001**	**0.821**	0.780	0.596

(c) Maze count

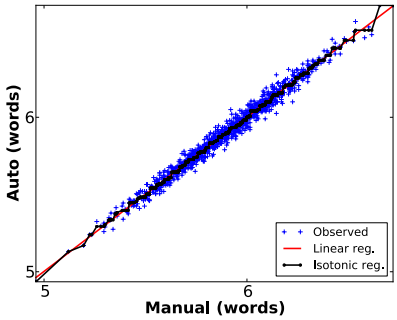
Corpus	$\Delta\mu$	$p \leq$	r	ρ	τ
CONV	0.15	**0.001**	**0.354**	0.596	0.417
EXPOSITORY	0.39	**0.001**	**0.866**	0.171	0.113
NARSSS	0.11	**0.001**	0.235	0.611	0.415
NARSR	0.15	**0.001**	**0.674**	0.599	0.412

(d) Maze length

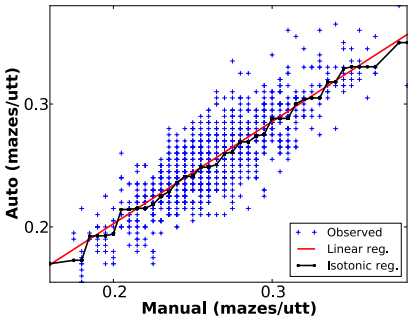
Table A.16: Reference statistic prediction accuracy using WI maze detection model; all values of ρ and τ significant at $p \leq 0.001$ level



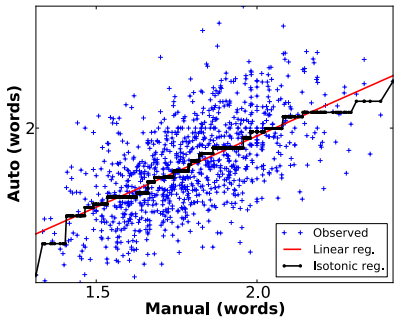
(a) Token count



(b) Type count

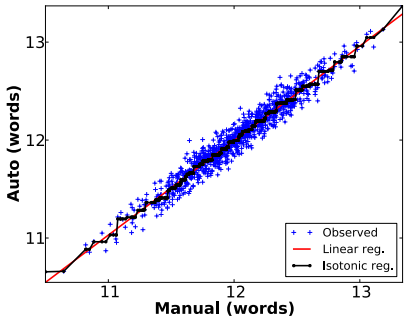


(c) Maze count

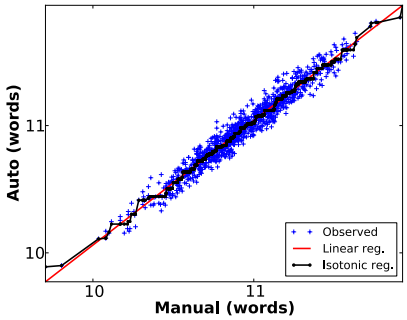


(d) Maze length

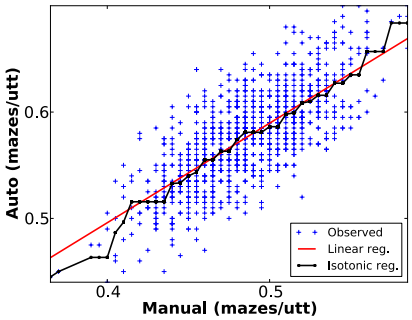
Figure A.34: Relationship between summary statistics computed from manual transcripts and FEDA WI model; development fold of CONV corpus



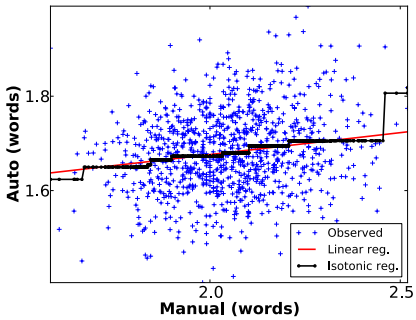
(a) Token count



(b) Type count

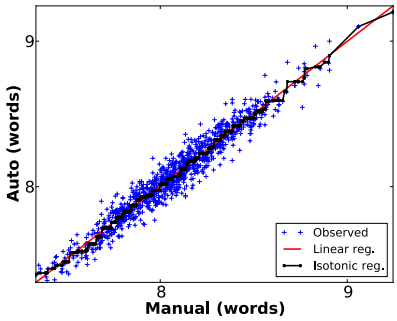


(c) Maze count

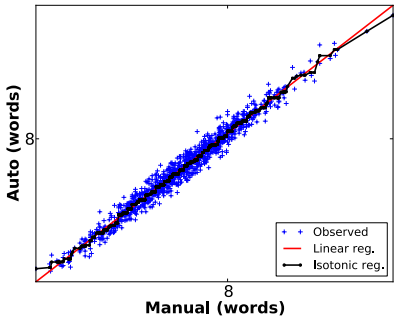


(d) Maze length

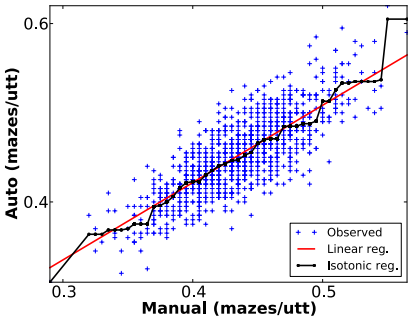
Figure A.35: Relationship between summary statistics computed from manual transcripts and FEDA WI model; development fold of EXPOSITORY corpus



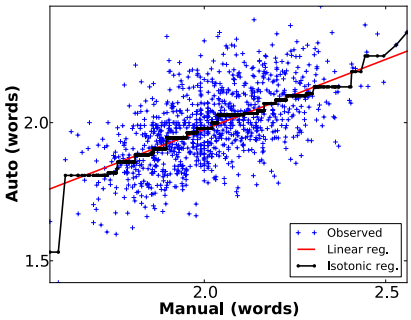
(a) Token count



(b) Type count

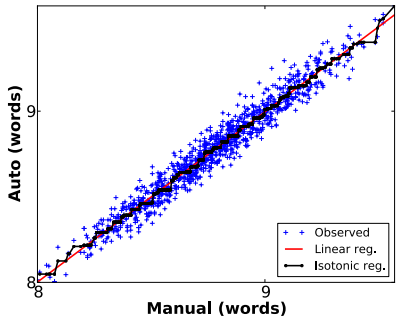


(c) Maze count

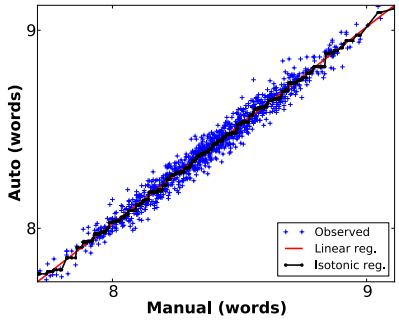


(d) Maze length

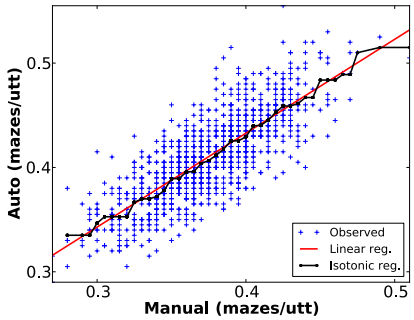
Figure A.36: Relationship between summary statistics computed from manual transcripts and FEDA WI model; development fold of NARSSS corpus



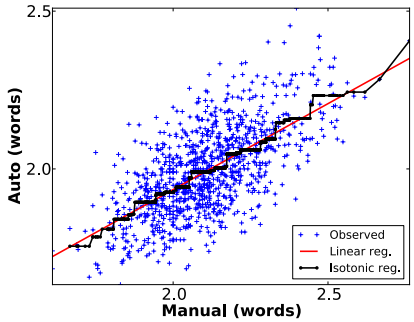
(a) Token count



(b) Type count



(c) Maze count



(d) Maze length

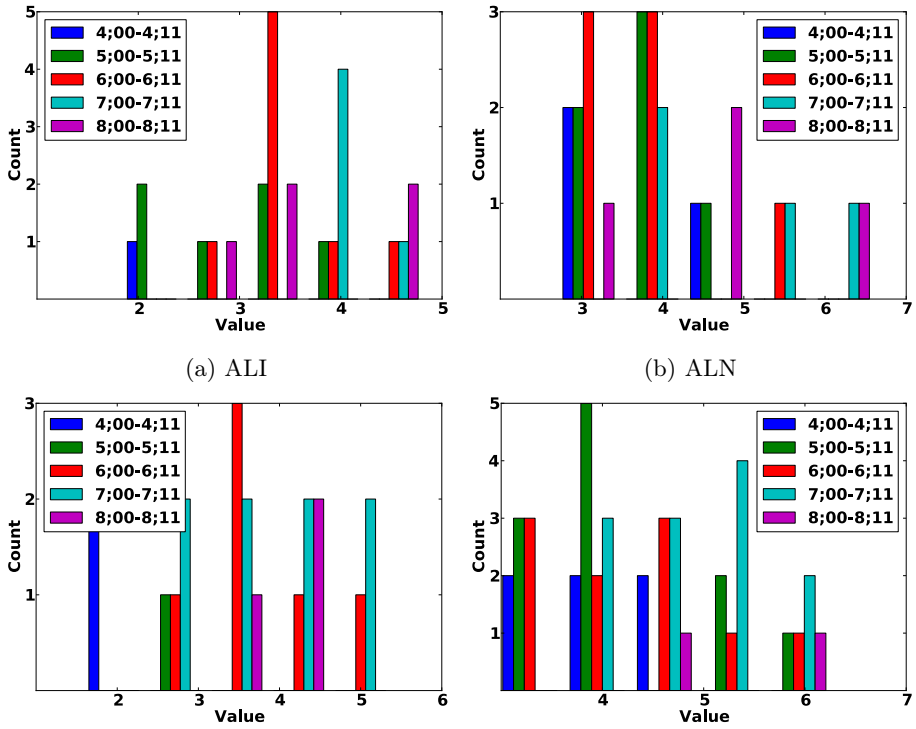
Figure A.37: Relationship between summary statistics computed from manual transcripts and FEDA WI model; development fold of NARSR corpus

Appendix A

Plots of features by age

These plots illustrate the distribution of the Baseline, Transcript, and SALT-1 features used in Chapter 7 for predicting scores on structured instruments and for the diagnostic discrimination task. We present each feature in turn. For each feature, we show eight plots: each of the individual diagnostic groups, along with the four composite groups. For reference, the four individual diagnostic groups are: autism with language impairment (ALI), autism no language impairment (ALI), specific language impairment (SLI), and typically developing (TD). The four composite groups are: autism (ASD, comprising ALI and ALN), no autism (nASD, comprising SLI and TD), unimpaired language (LN, comprising ALN and TD) and impaired language (LI, comprising ALI and SLI). Aside from the UmUhRat feature (ratio of 'um' to 'uh'), all features are normalized by the number of utterances spoken by the child.

A.1 Baseline features



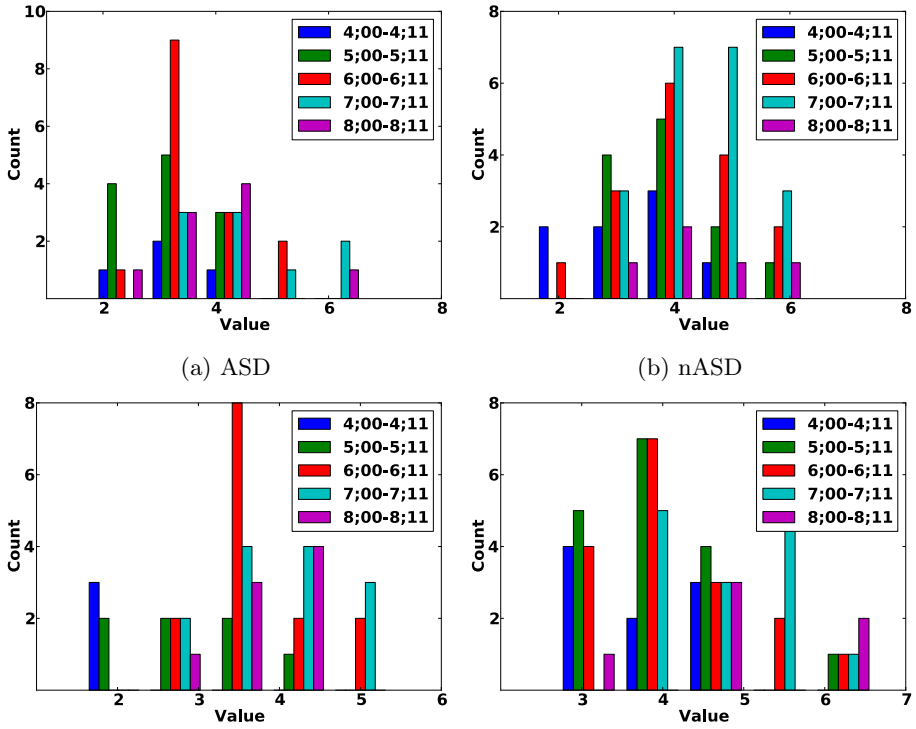
(a) ALI

(b) ALN

(c) SLI

(d) TD

Figure A.1: TkCt — token count, individual diagnostic groups



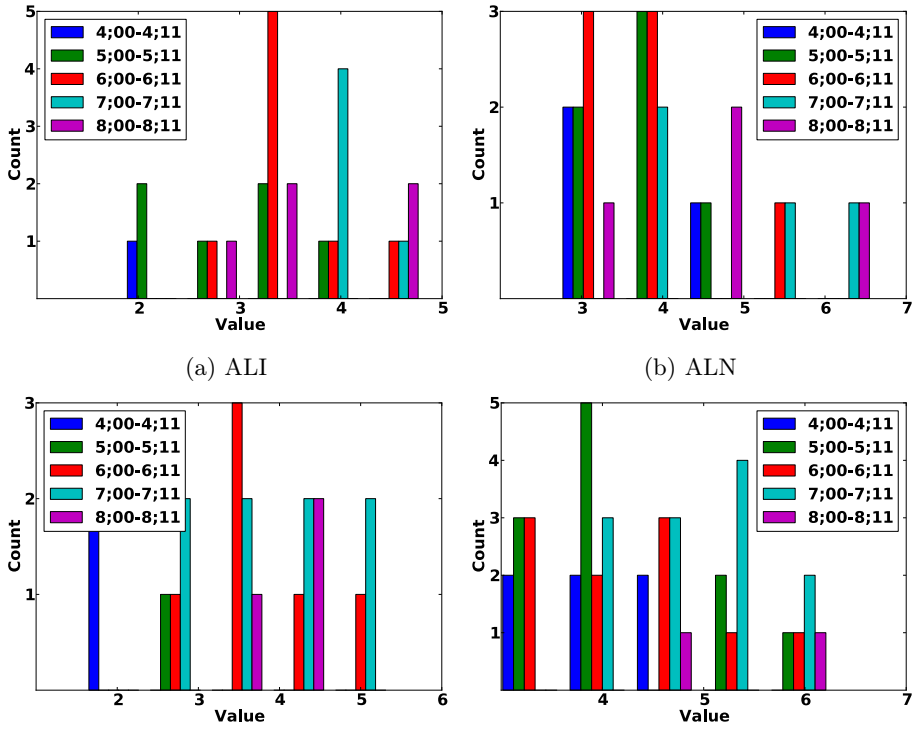
(a) ASD

(b) nASD

(c) LI

(d) LN

Figure A.2: TkCt — token count, composite diagnostic groups



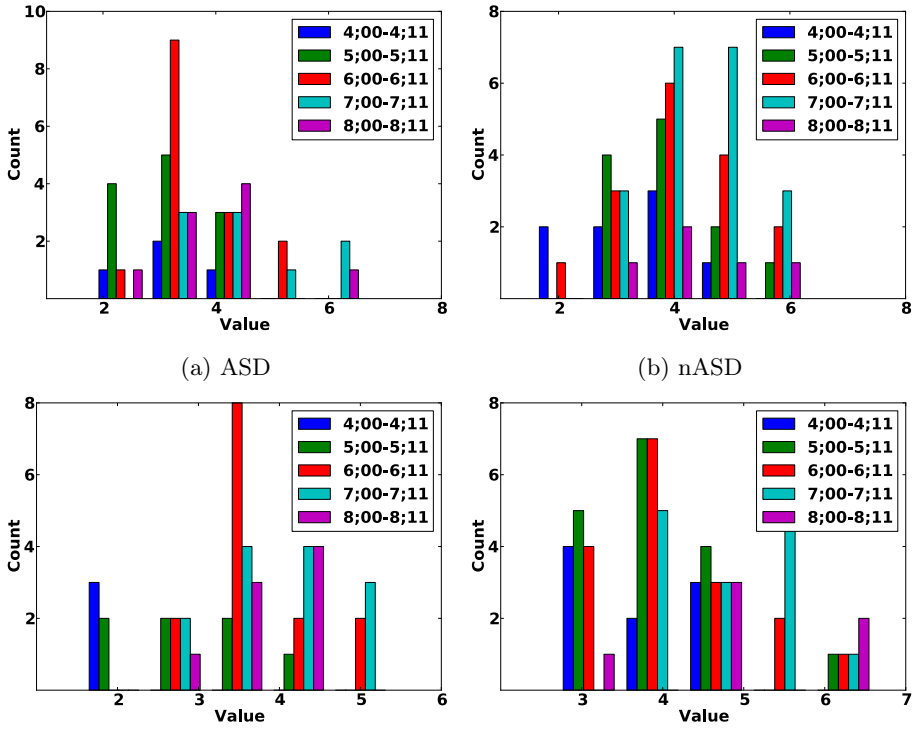
(a) ALI

(b) ALN

(c) SLI

(d) TD

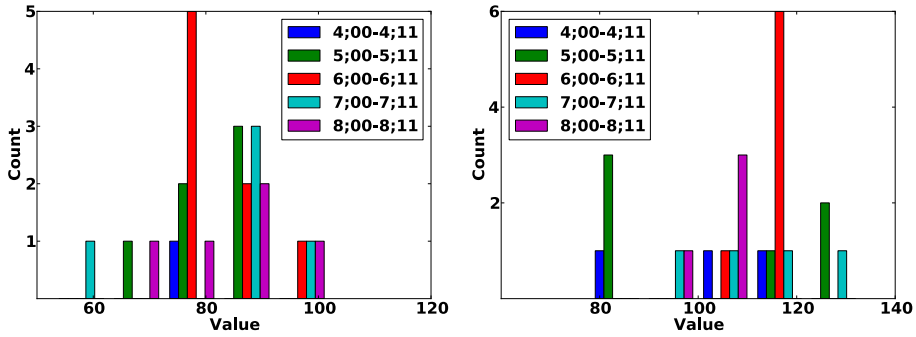
Figure A.3: TpCt — type count, individual diagnostic groups



(a) ASD (b) nASD

(c) LI (d) LN

Figure A.4: TpCt — type count, composite diagnostic groups



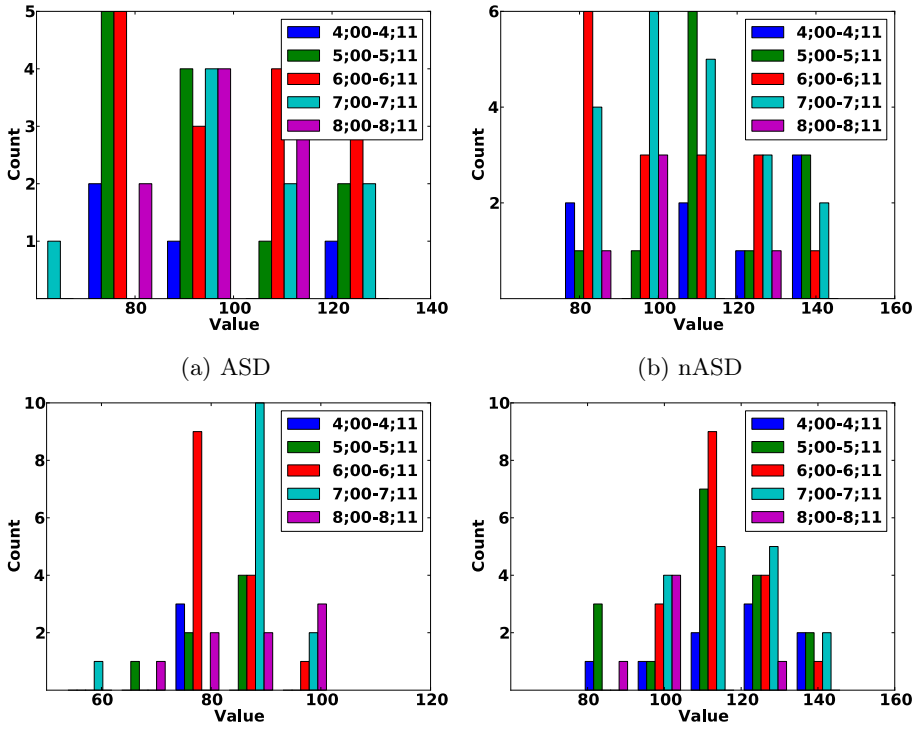
(a) ALI

(b) ALN

(c) SLI

(d) TD

Figure A.5: VIQ — Concatenated WPPSI3 & WISC4 VIQ standard scores, individual diagnostic groups



(a) ASD

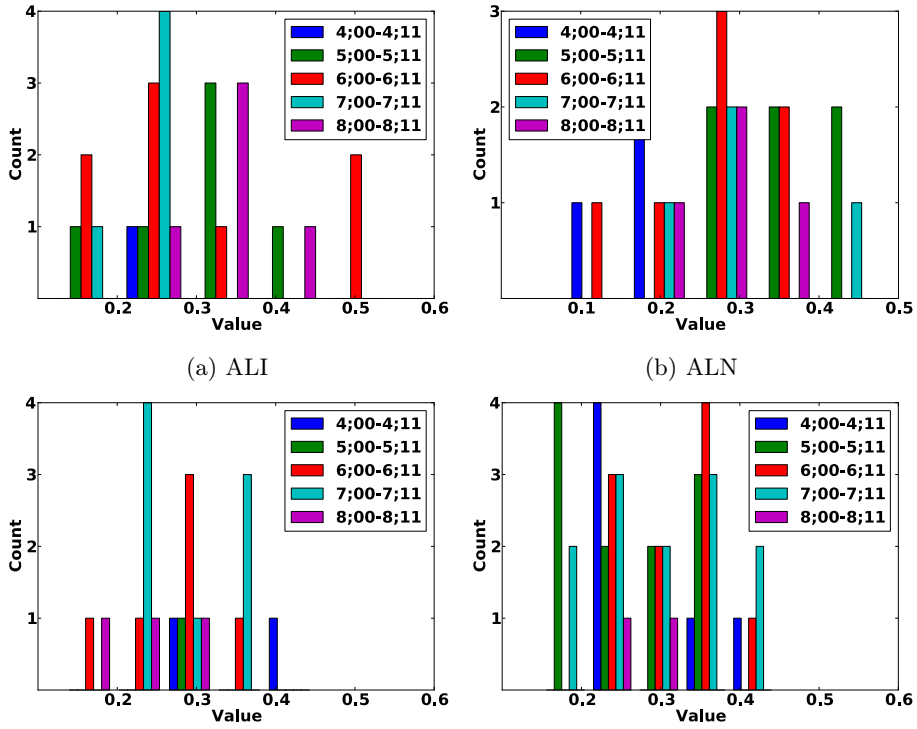
(b) nASD

(c) LI

(d) LN

Figure A.6: VIQ — Concatenated WPPSI3 & WISC4 VIQ standard scores, composite diagnostic groups

A.2 Transcript features



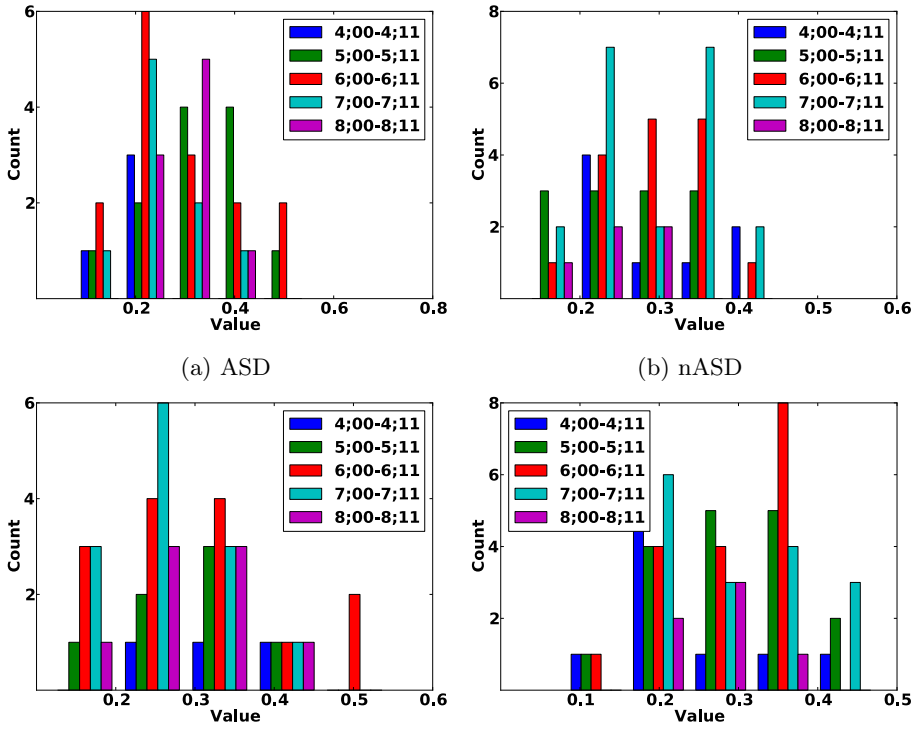
(a) ALI

(b) ALN

(c) SLI

(d) TD

Figure A.7: CEOLP — # of times examiner speaks while child is talking, individual diagnostic groups



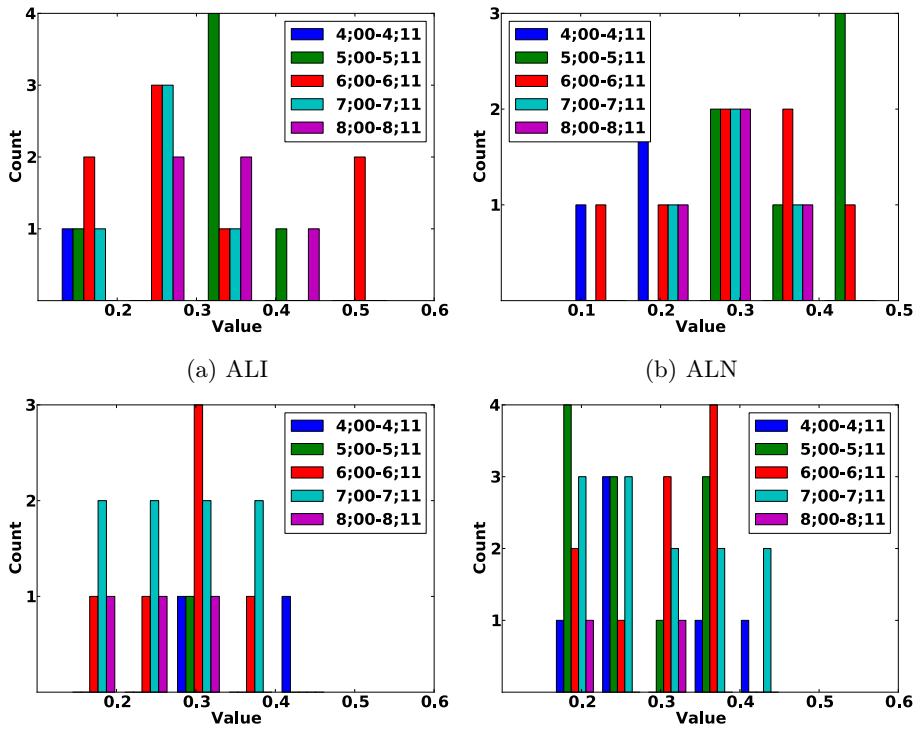
(a) ASD

(b) nASD

(c) LI

(d) LN

Figure A.8: CEOLP — # of times examiner speaks while child is talking, composite diagnostic groups



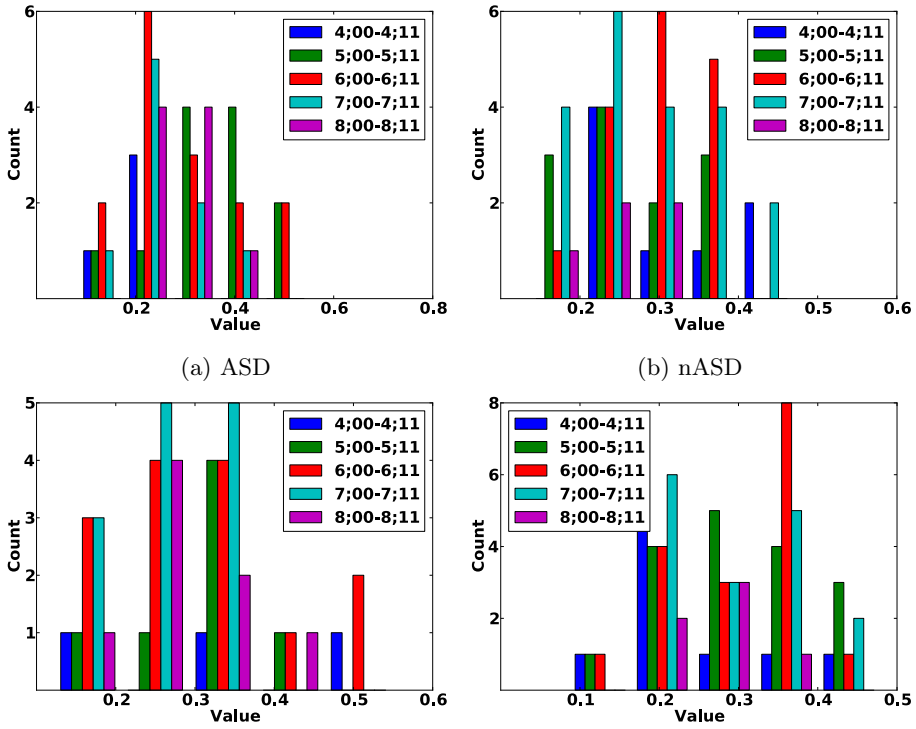
(a) ALI

(b) ALN

(c) SLI

(d) TD

Figure A.9: ECOLP — # of times child speaks while examiner is talking, individual diagnostic groups



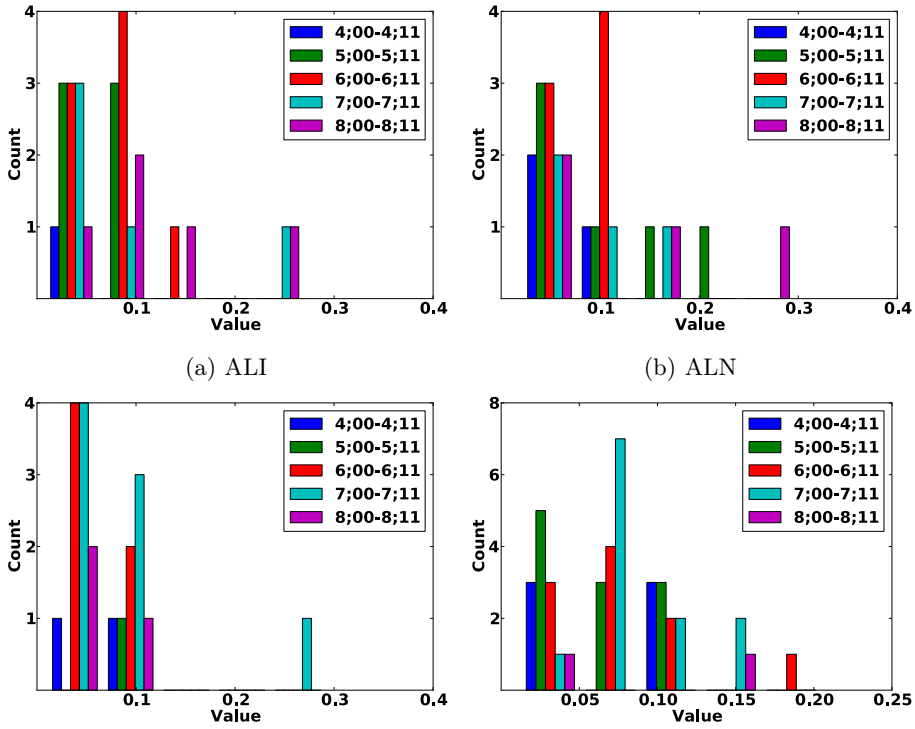
(a) ASD

(b) nASD

(c) LI

(d) LN

Figure A.10: ECOLP — # of times child speaks while examiner is talking, composite diagnostic groups



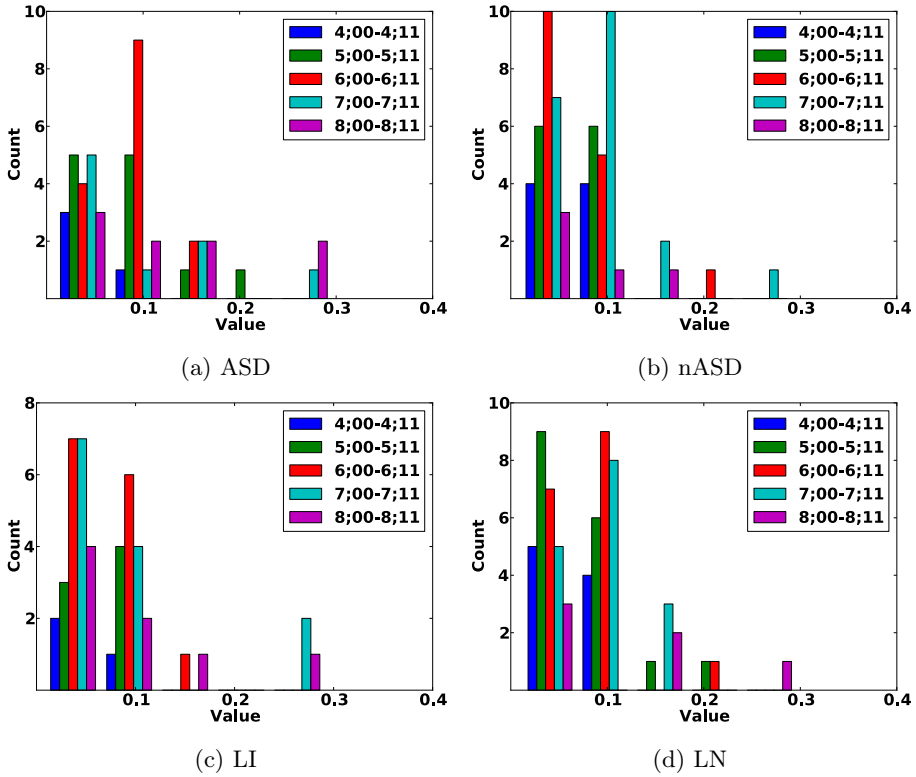
(a) ALI

(b) ALN

(c) SLI

(d) TD

Figure A.11: IncCt — Incomplete word count, individual diagnostic groups



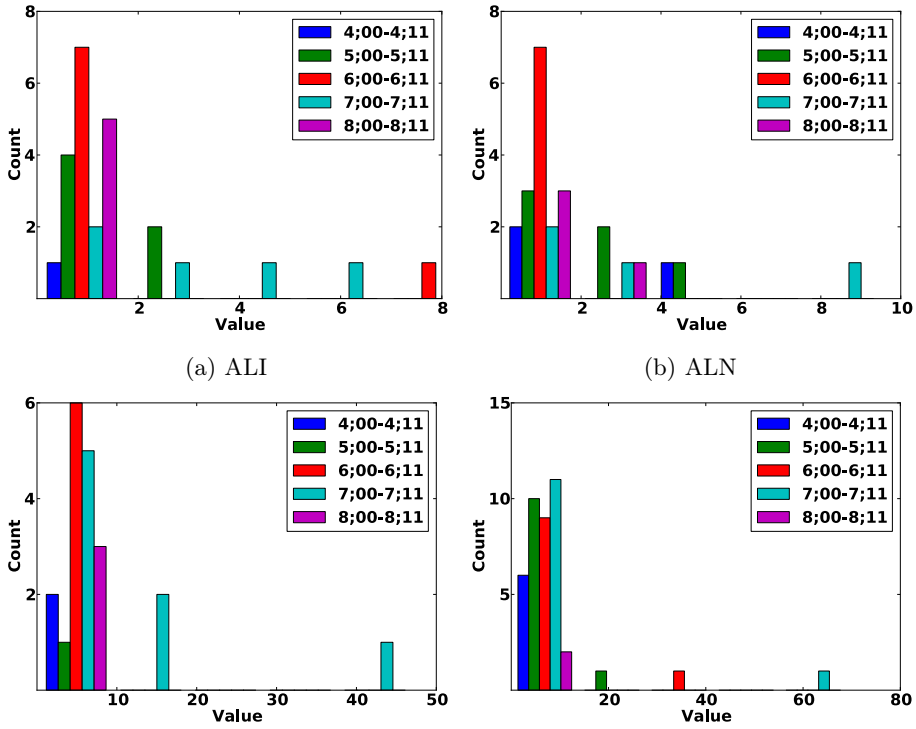
(a) ASD

(b) nASD

(c) LI

(d) LN

Figure A.12: IncCt — Incomplete word count, composite diagnostic groups



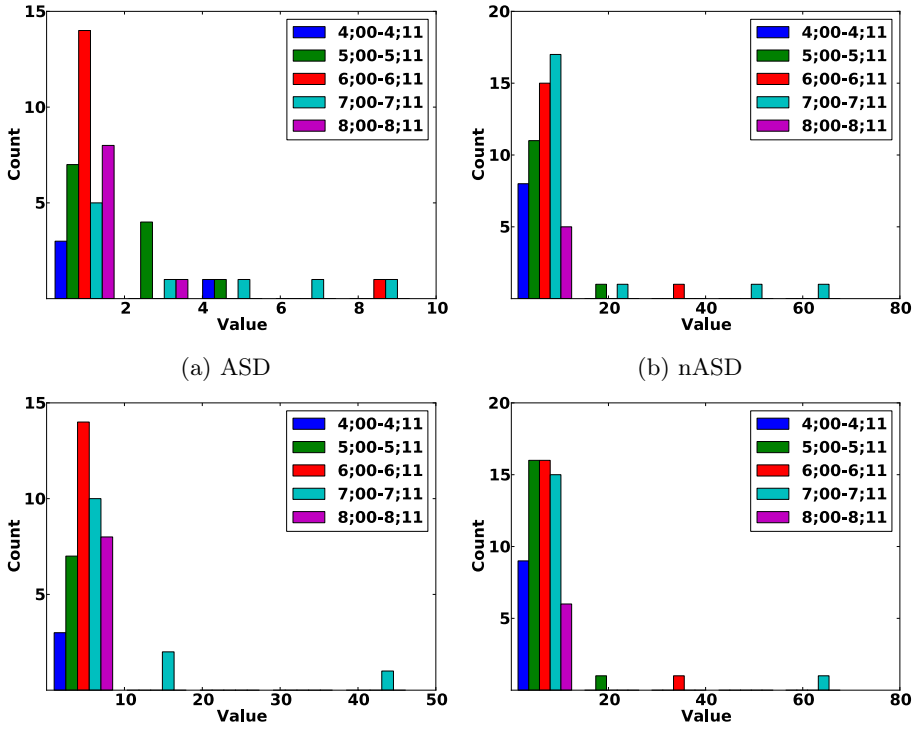
(a) ALI

(b) ALN

(c) SLI

(d) TD

Figure A.13: UmUhRat — Ratio of 'um' to 'uh', individual diagnostic groups



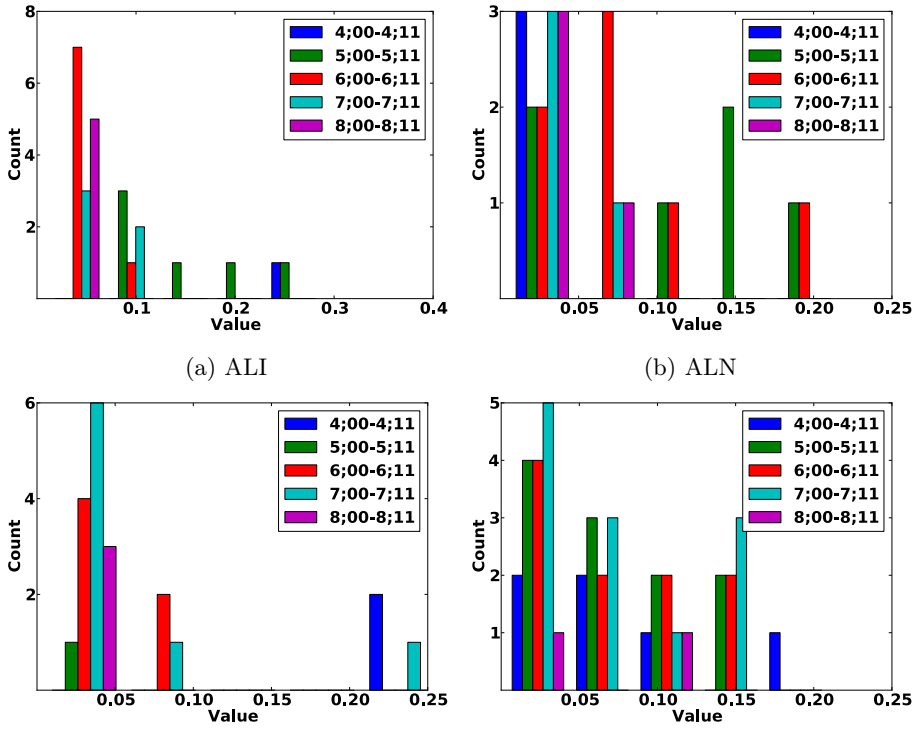
(a) ASD

(b) nASD

(c) LI

(d) LN

Figure A.14: UmUhRat — Ratio of 'um' to 'uh', composite diagnostic groups



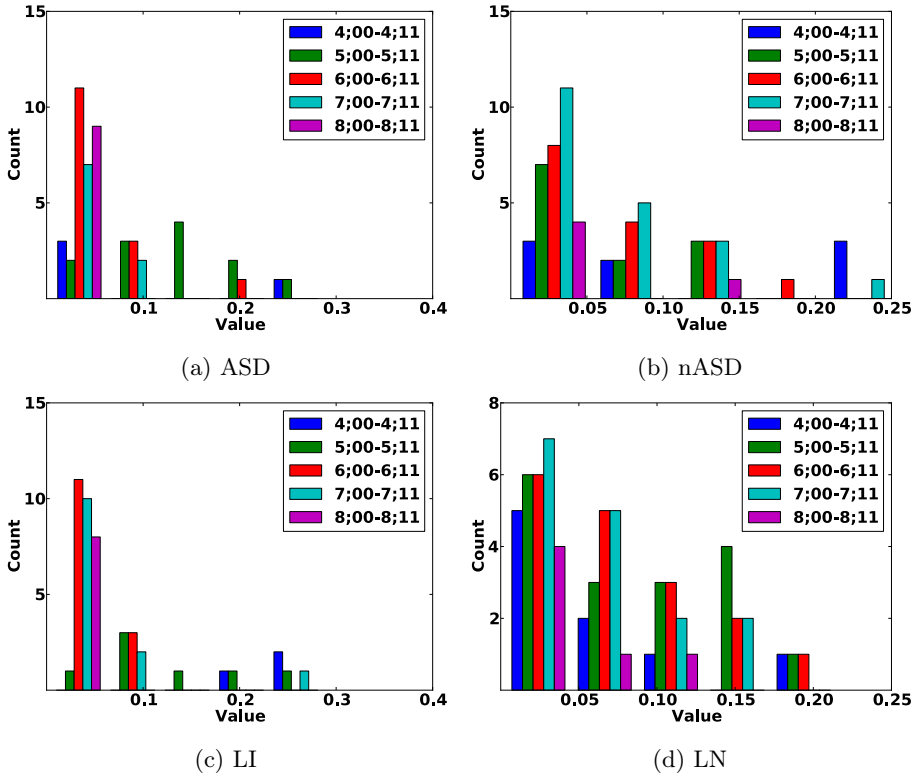
(a) ALI

(b) ALN

(c) SLI

(d) TD

Figure A.15: UNINTCT — Unintelligible word count, individual diagnostic groups

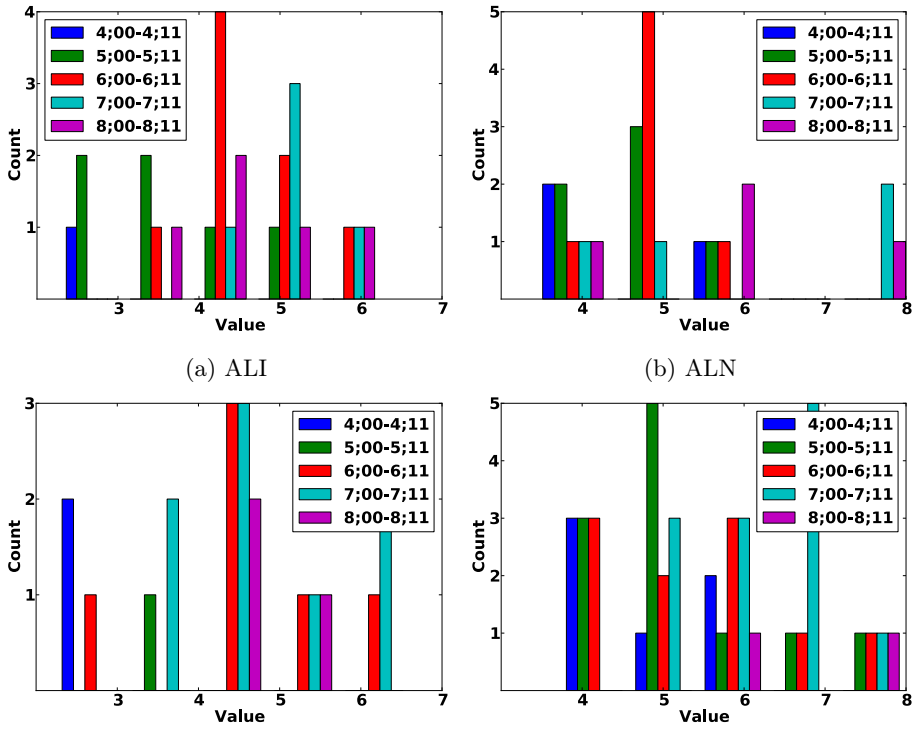


(a) ASD (b) nASD

(c) LI (d) LN

Figure A.16: UnintCt — Unintelligible word count, composite diagnostic groups

A.3 SALT-1 features



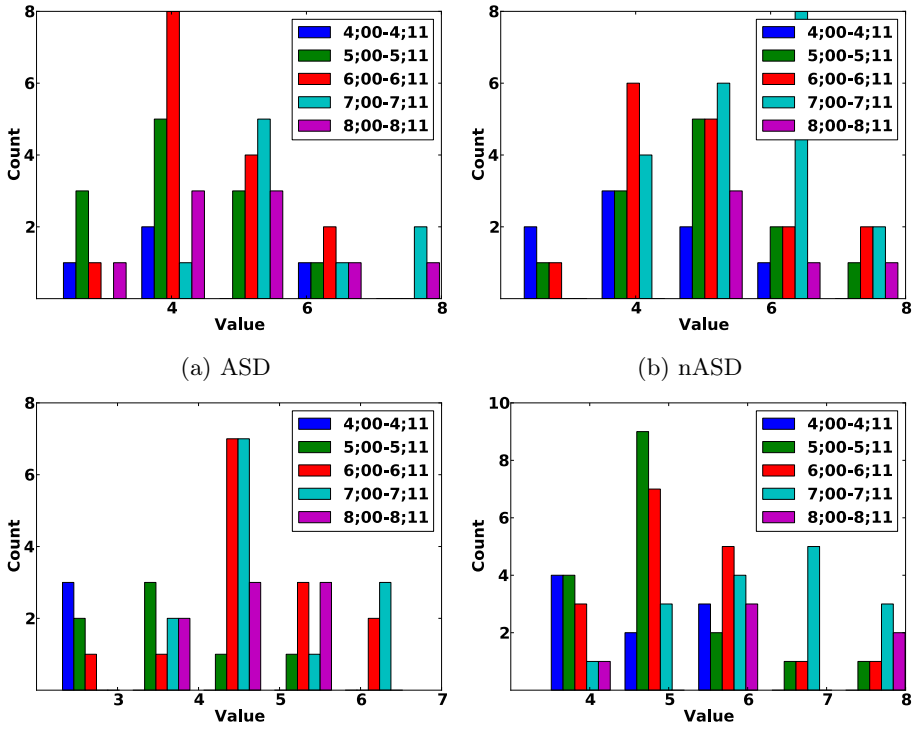
(a) ALI

(b) ALN

(c) SLI

(d) TD

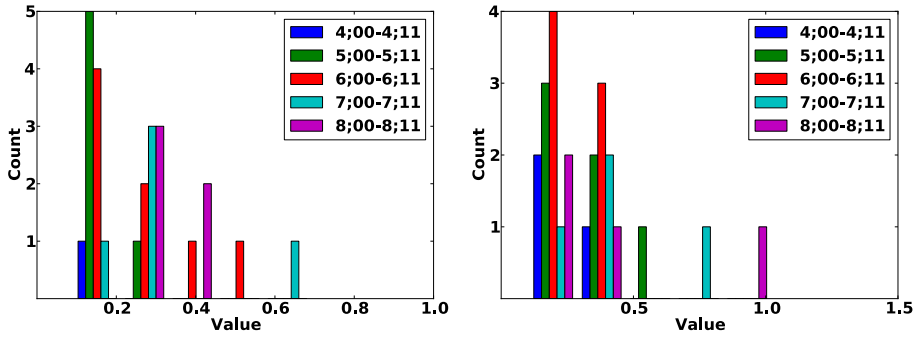
Figure A.17: MpCt — Morpheme count, individual diagnostic groups



(a) ASD (b) nASD

(c) LI (d) LN

Figure A.18: MpCt — Morpheme count, composite diagnostic groups



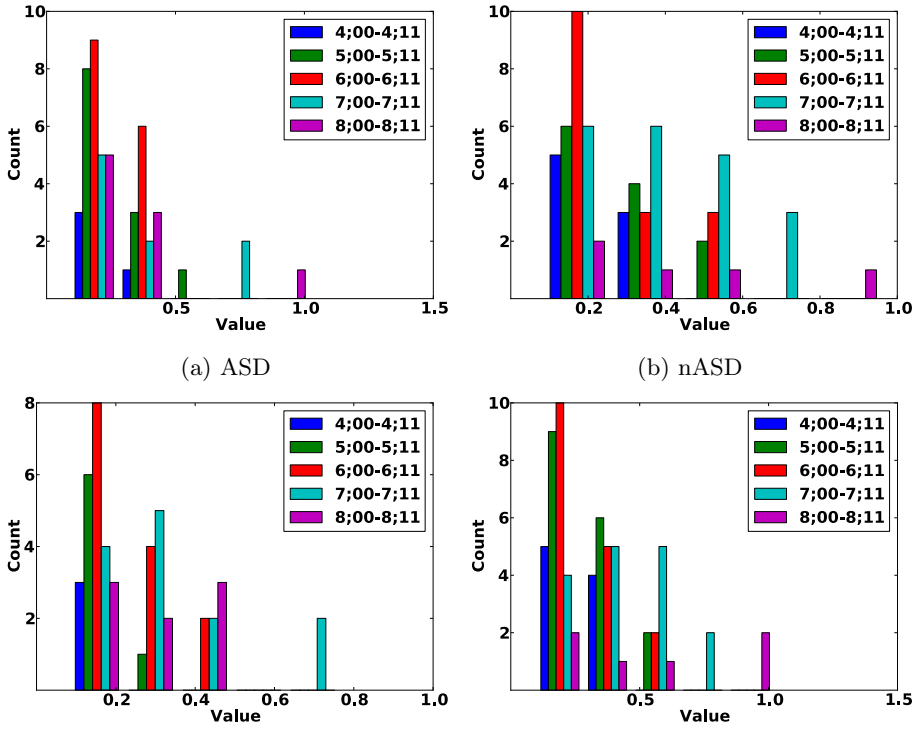
(a) ALI

(b) ALN

(c) SLI

(d) TD

Figure A.19: MazeCt — Maze count, individual diagnostic groups



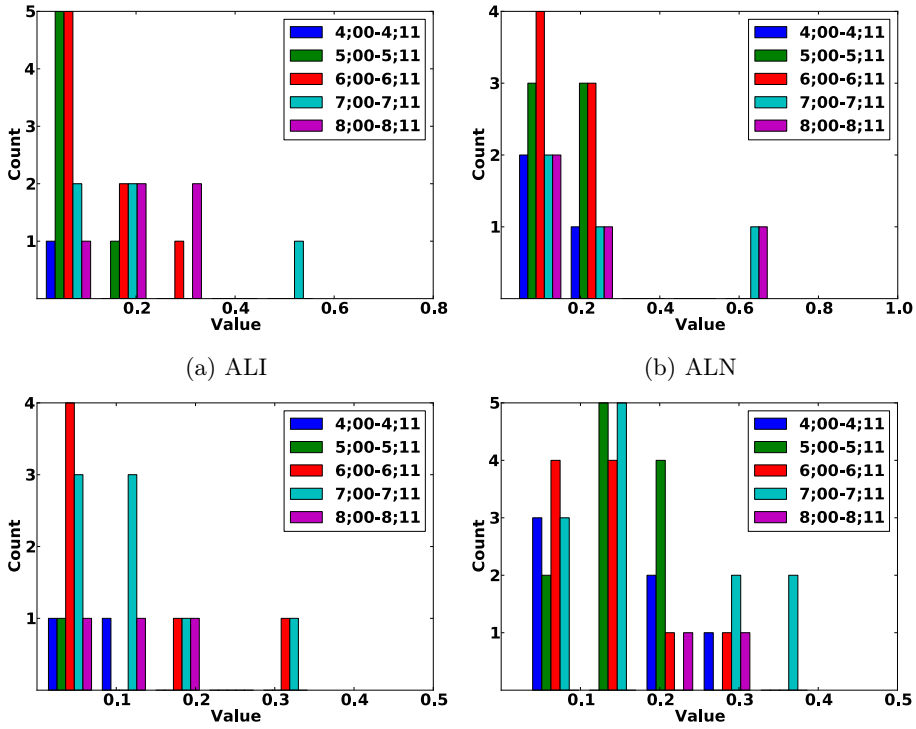
(a) ASD

(b) nASD

(c) LI

(d) LN

Figure A.20: MazeCt — Maze count, composite diagnostic groups



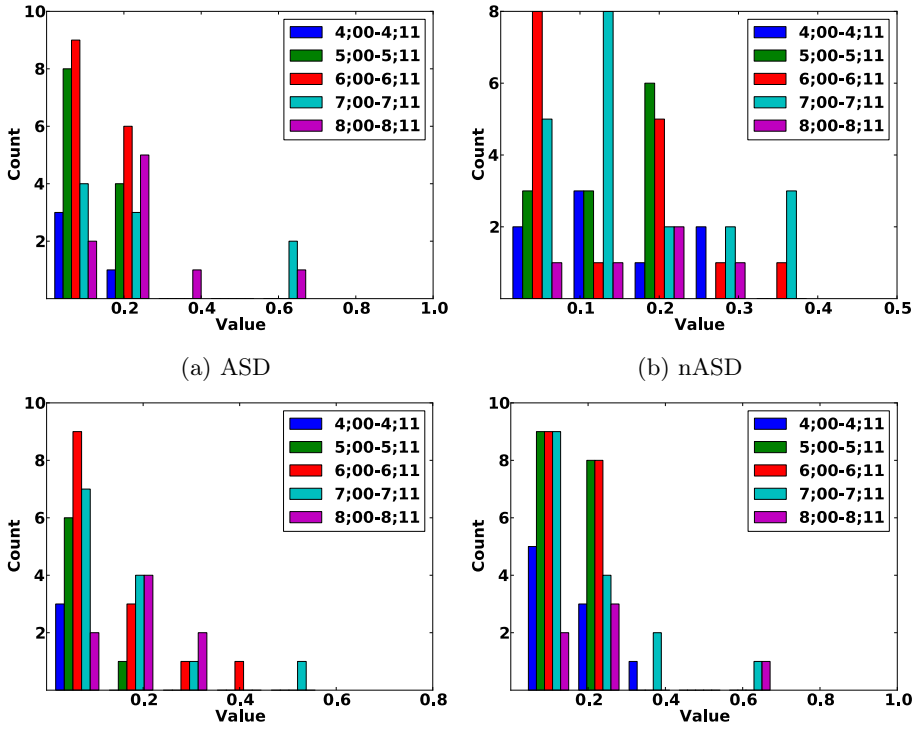
(a) ALI

(b) ALN

(c) SLI

(d) TD

Figure A.21: MazeTkCt — Token count within mazes, individual diagnostic groups



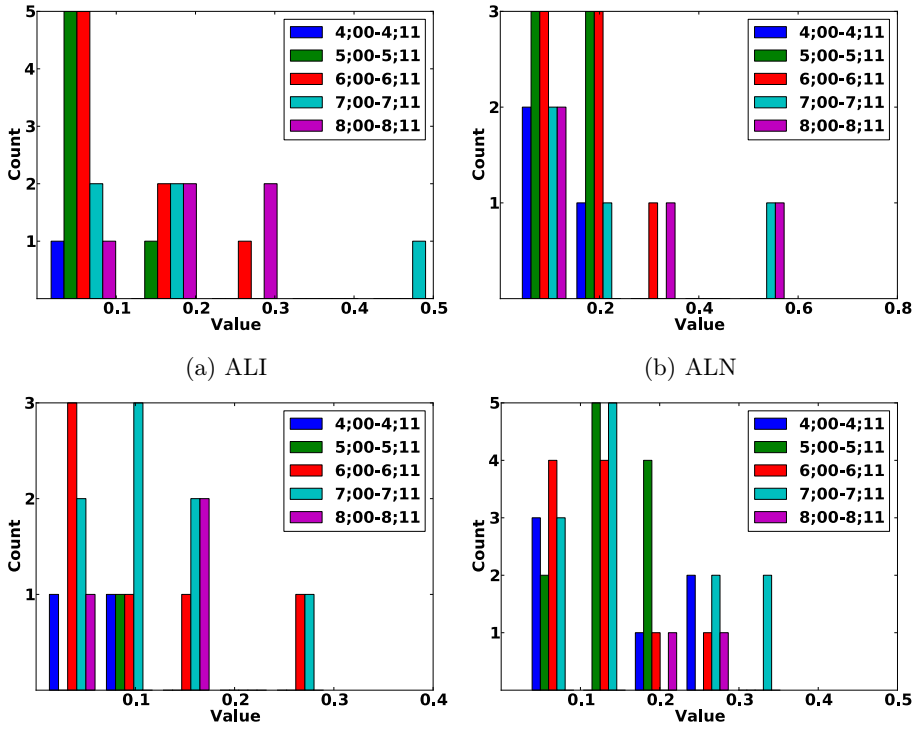
(a) ASD

(b) nASD

(c) LI

(d) LN

Figure A.22: MazeTkCt – Token count within mazes, composite diagnostic groups



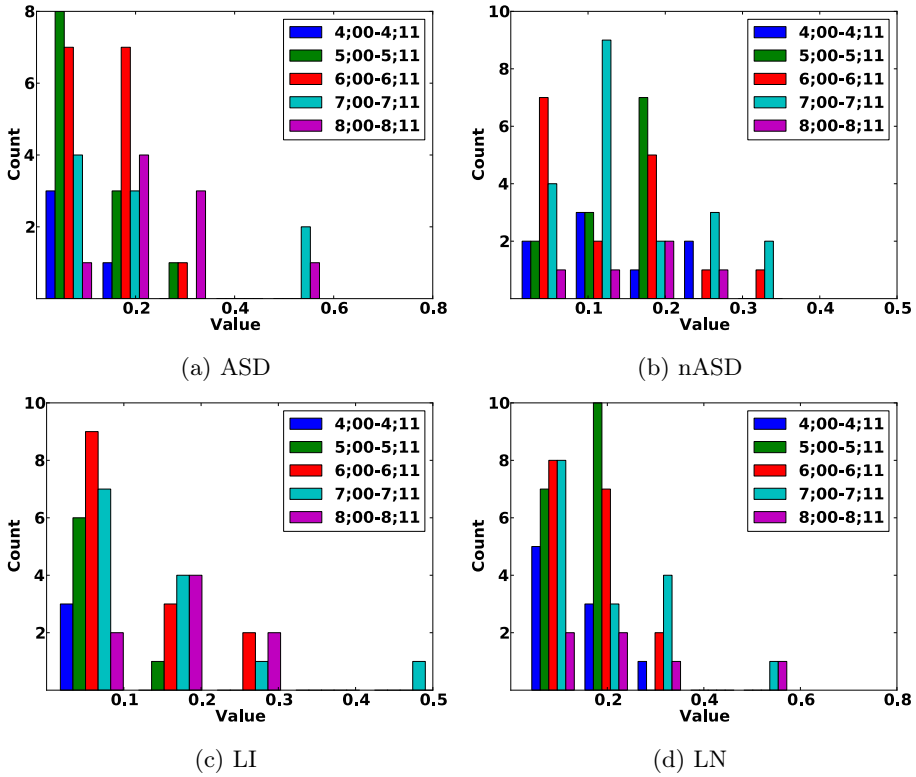
(a) ALI

(b) ALN

(c) SLI

(d) TD

Figure A.23: MazeTpCt — Type count within mazes, individual diagnostic groups



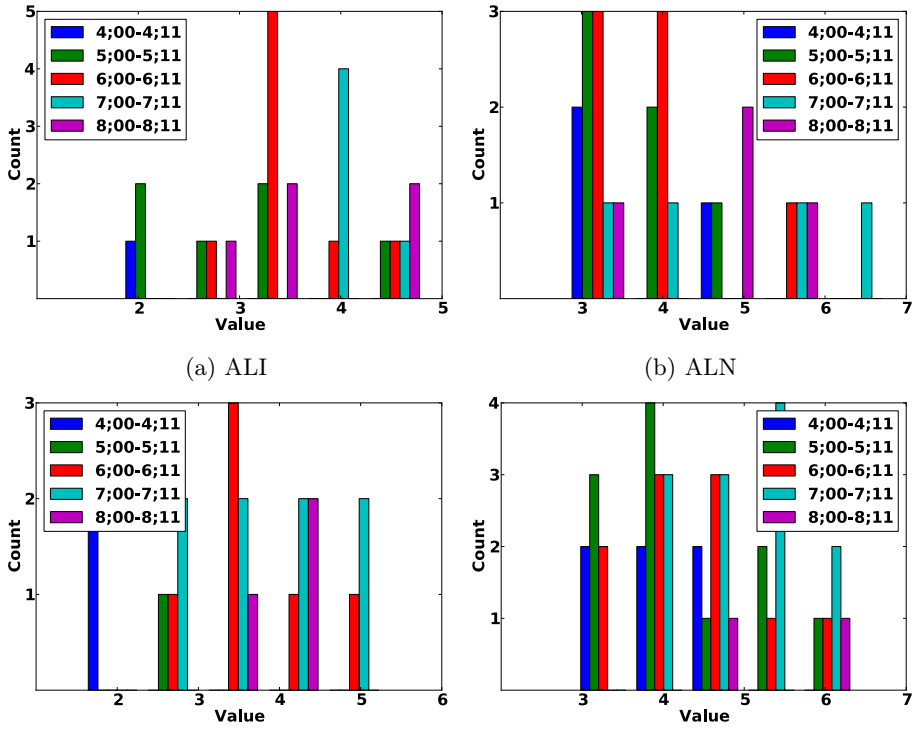
(a) ASD

(b) nASD

(c) LI

(d) LN

Figure A.24: MazeTpCt – Type count within mazes, composite diagnostic groups



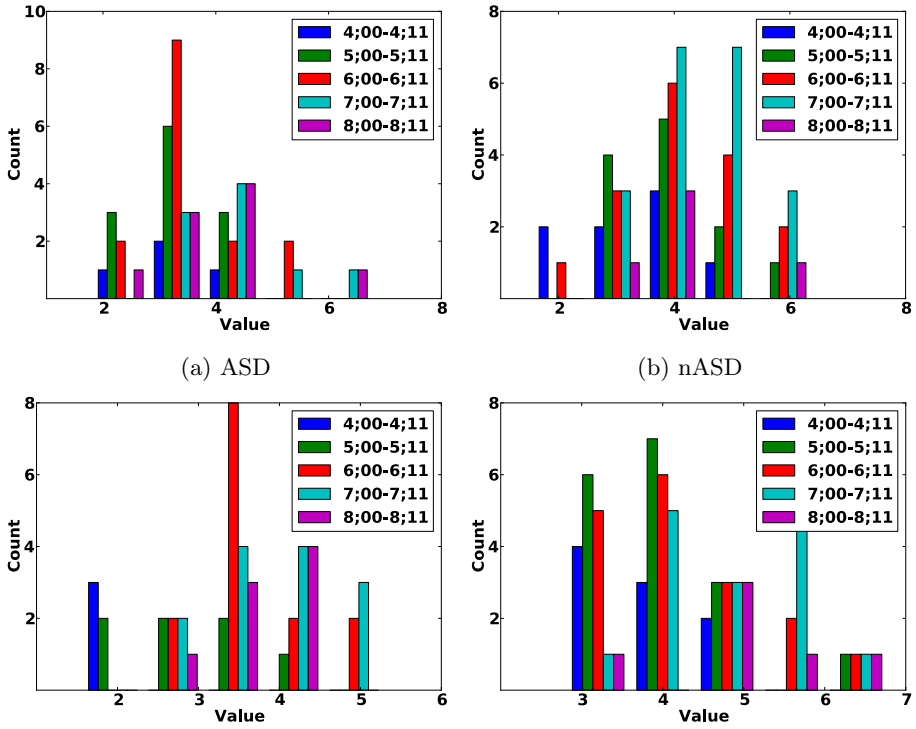
(a) ALI

(b) ALN

(c) SLI

(d) TD

Figure A.25: NoMazeTkCt — Token count outside of mazes, individual diagnostic groups



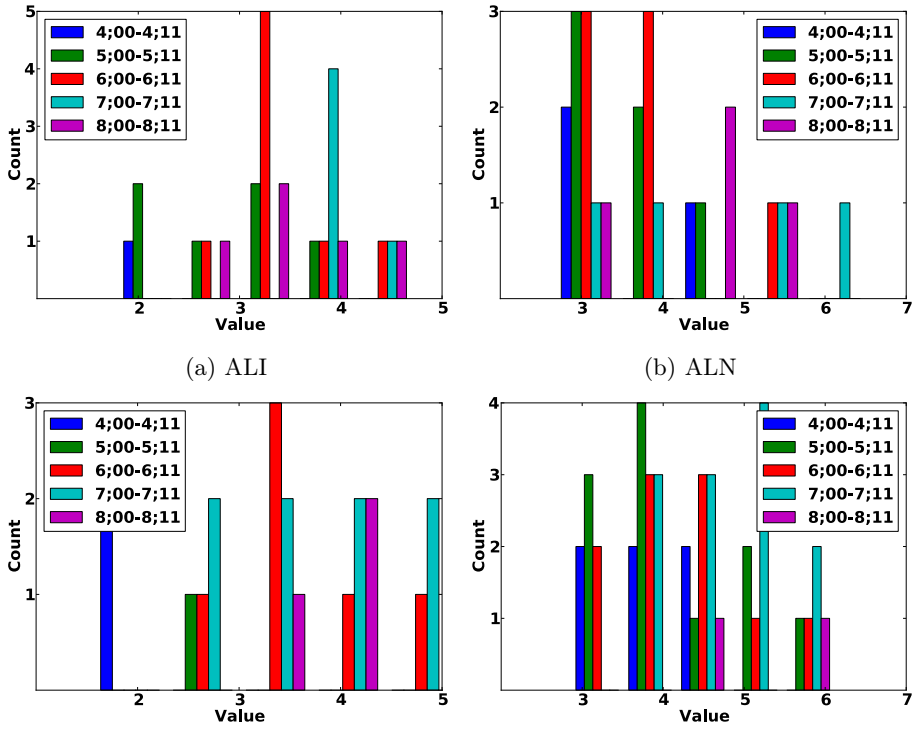
(a) ASD

(b) nASD

(c) LI

(d) LN

Figure A.26: NoMazeTkCt – Token count outside of mazes, composite diagnostic groups



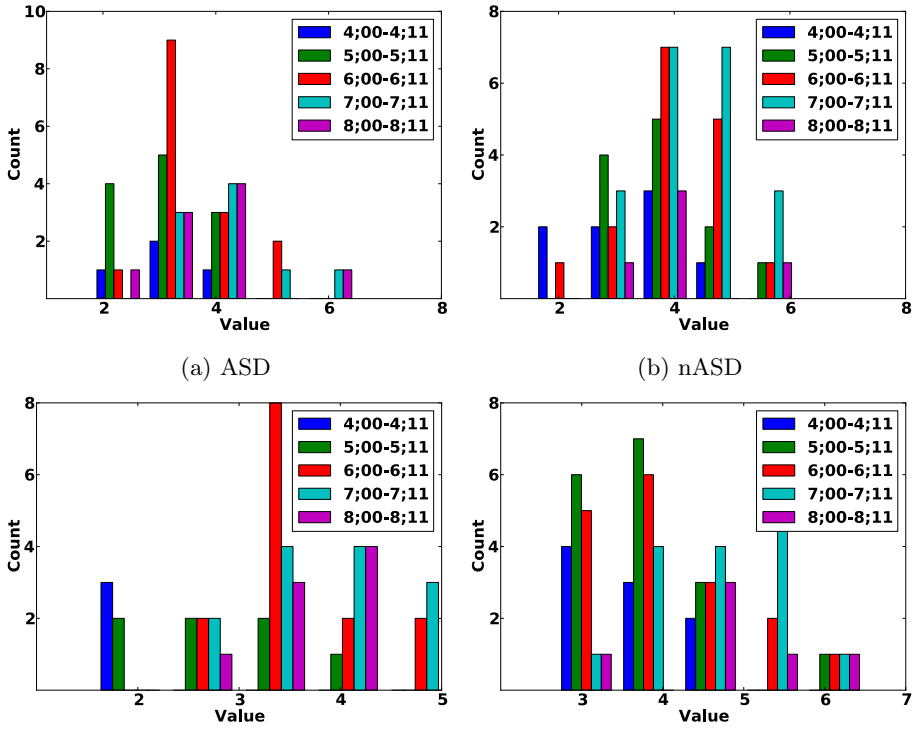
(a) ALI

(b) ALN

(c) SLI

(d) TD

Figure A.27: NoMazeTpCt — Type count outside of mazes, individual diagnostic groups



(a) ASD

(b) nASD

(c) LI

(d) LN

Figure A.28: NoMazeTpCt – Type count outside of mazes, composite diagnostic groups

Bibliography

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, Portland, Oregon, USA, June 2011. The Association for Computational Linguistics. ISBN 978-1-932432-88-6.

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

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

American Psychiatric Association. *DSM-V Diagnostic and Statistical Manual of Mental Disorders*. American Psychiatric Publishing, Washington, DC, 5th edition, 2013.

American Speech-Language-Hearing Association. Asha's recommended revisions to the dsm-5. `http://www.asha.org/uploadedFiles/DSM-5-Final-Comments.pdf`, 2012. "[Online; accessed 13-August-2015]".

Yigal Attali and Jill Burstein. Automated essay scoring with e-rater® v. 2.0. *ETS Research Report Series*, 2004(2):i–21, 2004.

Miguel Ballesteros and Joakim Nivre. Going to the roots of dependency parsing. *Computational Linguistics*, 39(1):5–13, 2013.

Richard Beaufort, Sophie Roekhaut, Louise-Amélie Cougnon, and Cédrick Fairon. A hybrid rule/model-based finite-state framework for normalizing sms messages. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 770–779, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P10-1079`.

Dorothy VM Bishop and Courtenay Frazier Norbury. Exploring the borderlands of autistic disorder and specific language impairment: a study using standardised diagnostic instruments. *Journal of Child Psychology and Psychiatry*, 43 (7):917–929, 2002.

Dorothy VM Bishop and F Volkmar. *The Children's Communication Checklist: CCC-2*. ASHA, 2003.

Heather Bortfeld, Silvia D Leon, Jonathan E Bloom, Michael F Schober, and Susan E Brennan. Disfluency rates in conversation: Effects of age, relationship, topic, role, and gender. *Language and speech*, 44(2):123–147, 2001.

Mari I Bowden and Richard K Fox. 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*, 2002.

Ulrik Brandes. A faster algorithm for betweenness centrality*. *Journal of mathematical sociology*, 25(2):163–177, 2001.

Ronald L Breiger, Scott A Boorman, and Phipps Arabie. An algorithm for clustering relational data with applications to social network analysis and

comparison with multidimensional scaling. *Journal of mathematical psychology*, 12(3):328–383, 1975.

Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *COMPUTER NETWORKS AND ISDN SYSTEMS*, pages 107–117. Elsevier Science Publishers B. V., 1998.

Noel Burton-Roberts. *Analysing sentences*. Routledge, 2013.

Andrew Caines and Paula Buttery. 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, 2010.

Alexandra Canavan, David Graff, and George Zipperlen. Callhome american english speech. *Linguistic Data Consortium*, 1997.

Monique J Charest and Laurence B Leonard. Predicting tense: Finite verb morphology and subject pronouns in the speech of typically-developing children and children with specific language impairment. *Journal of child language*, 31 (01):231–246, 2004.

Eugene Charniak and Mark Johnson. Edit detection and parsing for transcribed speech. In *NAACL*, 2001.

J Clifford, J Reilly, and B Wulfeck. Narratives from children with specific language impairment: An exploration in language and cognition. *University of California, San Diego: Center for Research in Language*, 1995.

Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics, 2002.

Corinna Cortes, Mehryar Mohri, and Ashish Rastogi. An alternative ranking problem for search engines. In *Experimental Algorithms*, pages 1–22. Springer, 2007.

Richard F Cromer. Developmental language disorders: Cognitive processes, semantics, pragmatics, phonology, and syntax. *Journal of Autism and Developmental Disorders*, 11(1):57–74, 1981.

Hal Daumé III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P07-1033`.

Marie-Catherine De Marneffe and Christopher D Manning. Stanford typed dependencies manual. Technical report, Technical report, Stanford University, 2008.

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

Neeraj Deshmukh, Aravind Ganapathiraju, Andi Gleeson, Jonathan Hamaker, and Joseph Picone. Resegmentation of switchboard. In *ICSLP*. Syndey, 1998.

Patricia Ann Eadie, ME Fey, JM Douglas, and CL Parsons. Profiles of grammatical morphology and sentence imitation in children with specific language impairment and down syndrome. *Journal of Speech, Language, and Hearing Research*, 45(4):720–732, 2002.

Jay Earley. An efficient context-free parsing algorithm, 1970.

Sigmund Eldevik, Svein Eikeseth, Erik Jahr, and Tristram Smith. Effects of low-

intensity behavioral treatment for children with autism and mental retardation. *Journal of Autism and Developmental Disorders*, 36(2):211–224, 2006.

Janice Ellsworth and Akiko Fuse. CELF-4: Potential for Bias Against Speakers of African American English. Presented as the 2008 convention of American Speech-Language-Hearing Association, Chicago, IL, 2008.

Paul E Engelhardt, Fernanda Ferreira, and Joel T Nigg. Language production strategies and disfluencies in multi-clause network descriptions: a study of adult attention-deficit/hyperactivity disorder. *Neuropsychology*, 25(4):442, 2011.

Paul E Engelhardt, Joel T Nigg, and Fernanda Ferreira. Is the fluency of language outputs related to individual differences in intelligence and executive function? *Acta psychologica*, 144(2):424–432, 2013.

Anton J Enright, Stijn Van Dongen, and Christos A Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic acids research*, 30(7):1575–1584, 2002.

Günes Erkan and Dragomir R Radev. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, pages 457–479, 2004.

Julia L Evans and Holly K Craig. Language sample collection and analysisinterview compared to freeplay assessment contexts. *Journal of Speech, Language, and Hearing Research*, 35(2):343–353, 1992.

Shimon Even. *Graph algorithms*. Cambridge University Press, 2011.

James Ferguson, Greg Durrett, and Dan Klein. Disfluency detection with a semi-markov model and prosodic features. In *Proceedings of NAACL*, Denver, Colorado, USA, June 2015.

Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *Knowledge and data engineering, ieee transactions on*, 19(3):355–369, 2007.

Yoav Freund and Robert E Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296, 1999.

Michael Gamon, Claudia Leacock, Chris Brockett, William B Dolan, Jianfeng Gao, Dmitriy Belenko, and Alexandre Klementiev. Using statistical techniques and web search to correct ESL errors. *Calico Journal*, 26(3):491–511, 2013.

William J Gavin and Lisa Giles. Sample size effects on temporal reliability of language sample measures of preschool children. *Journal of Speech, Language, and Hearing Research*, 39(6):1258–1262, 1996.

Kallirroi Georgila. Using integer linear programming for detecting speech disfluencies. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 109–112. Association for Computational Linguistics, 2009.

Ronald Bradley Gillam and Nils A Pearson. Test of narrative language: Examiner's manual, 2004.

Gail T Gillon. The efficacy of phonological awareness intervention for children with spoken language impairment. *Language, Speech, and Hearing Services in Schools*, 31(2):126–141, 2000.

Gail T Gillon. Follow-up study investigating the benefits of phonological awareness intervention for children with spoken language impairment. *International Journal of Language & Communication Disorders*, 37(4):381–400, 2002.

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

Yoav Goldberg and Joakim Nivre. Training deterministic parsers with non-deterministic oracles. *Transactions of the Association for Computational Linguistics*, 1:403–414, 2013. ISSN 2307-387X. URL `https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/145`.

Andrew Golding. A bayesian hybrid method for context-sensitive spelling correction. In *Proceedings of the Third Workshop on Very Large Corpora*. Association for Computational Linguistics, 1995. URL `http://www.aclweb.org/anthology/D14-1001`.

Sieneke M Goorhuis-Brouwer and Barbara J Wijnberg-Williams. Specificity of specific language impairment. *Folia Phoniatrica et Logopaedica*, 48(6):269–274, 1996.

Kyle Gorman. *ldamatch: Multivariate Condition Matching by Backwards Elimination Using Linear Discriminant Analysis*, 2015. URL `http://CRAN.R-project.org/package=ldamatch`. R package version 0.6.3.

Kyle Gorman, Steven Bedrick, Geza Kiss, Eric Morley, Rosemary Ingham, Metrah Mohammed, Katina Papadakis, and Jan van Santen. Automated morphological analysis of clinical language samples. pages 108–116, June 5 2015. URL `http://www.aclweb.org/anthology/W15-1213`.

Katherine Gotham, Susan Risi, Andrew Pickles, and Catherine Lord. The autism diagnostic observation schedule: revised algorithms for improved diagnostic validity. *Journal of autism and developmental disorders*, 37(4):613–627, 2007.

Bernard G Grela and Laurence B Leonard. The influence of argument-structure complexity on the use of auxiliary verbs by children with sli. *Journal of Speech, Language, and Hearing Research*, 43(5):1115–1125, 2000.

Ulrike Grömping et al. Relative importance for linear regression in r: the package relaimpo. *Journal of statistical software*, 17(1):1–27, 2006.

Ling-yu Guo, J Bruce Tomblin, and Vicki Samelson. Speech disruptions in the narratives of english-speaking children with specific language impairment. *Journal of Speech, Language, and Hearing Research*, 51(3):722–738, 2008.

Louis Guttman. A basis for analyzing test-retest reliability. *Psychometrika*, 10 (4):255–282, 1945.

Pamela A Hadley. Language sampling protocols for eliciting text-level discourse. *Language, Speech, and Hearing Services in Schools*, 29(3):132–147, 1998.

Khairun-nisa Hassanali and Yang Liu. 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, 2011.

Peter A Heeman and James F Allen. The TRAINS spoken dialog corpus. *Linguistics Data Consortium*, 1995.

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

John Heilmann, Jon F Miller, Aquiles Iglesias, Leah Fabiano-Smith, Ann Nockerts, and Karen Digney Andriacchi. Narrative transcription accuracy and reliability in two languages. *Topics in Language Disorders*, 28(2):178–188, 2008.

Antje Helfrich and Bradley Music. Design and evaluation of grammar checkers in multiple languages. In *COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics*, 2000. URL `http://aclweb.org/anthology/C00-2153`.

Charles T Hemphill, John J Godfrey, and George R Doddington. The atis spoken language systems pilot corpus. In *Proceedings of the DARPA speech and natural language workshop*, pages 96–101, 1990.

Alison Presmanes Hill, Jan van Santen, Kyle Gorman, Beth Hoover Langhorst, and Eric Fombonne. Memory in language-impaired children with and without autism. *Journal of neurodevelopmental disorders*, 7(1):19, 2015.

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

Matthew Honnibal, Yoav Goldberg, and Mark Johnson. A non-monotonic arc-eager transition system for dependency parsing. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 163–172, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W13-3518`.

Richard A Hudson. *English word grammar*, volume 108. Basil Blackwell Oxford, 1990.

Karen Jensen, George E. Heidorn, Lance A. Miller, and Yael Ravin. Parse fitting and prose fixing: getting a hold on ill-formedness. *Computational Linguistics*, 9(3-4):147–160, 1983.

Zhongye Jia, Peilu Wang, and Hai Zhao. Grammatical error correction as multiclass classification with single model. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages

74–81, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-3610.

Mark Johnson and Eugene Charniak. A TAG-based noisy-channel model of speech repairs. In Donia Scott, Walter Daelemans, and Marilyn A. Walker, editors, *ACL*, pages 33–39, Barcelona, Spain, July 2004. The Association for Computational Linguistics.

Martin Kay. Guides and oracles for linear-time parsing. In *Proceedings of the 6th International Workshop on Parsing Technologies (IWPT)*, pages 6–9, 2000.

Susan Kemper, Donna Kynette, and Suzanne Norman. Age differences in spoken language. In *Everyday memory and aging*, pages 138–152. Springer, 1992.

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

Ami Klin, Jason Lang, Domenic V Cicchetti, and Fred R Volkmar. Brief report: Interrater reliability of clinical diagnosis and dsm-iv criteria for autistic disorder: Results of the dsm-iv autism field trial. *Journal of autism and developmental disorders*, 30(2):163–167, 2000.

Cyndie Koning and Joyce Magill-Evans. Social and language skills in adolescent boys with asperger syndrome. *Autism*, 5(1):23–36, 2001.

Jacqueline C Kowtko and Patti J Price. Data collection and analysis in the air travel planning domain. In *Proceedings of the workshop on Speech and Natural Language*, pages 119–125. Association for Computational Linguistics, 1989.

William Kruskal. Relative importance by averaging over orderings. *The American Statistician*, 41(1):6–10, 1987.

Taku Kudo. CRF++: Yet another CRF toolkit. `http://taku910.github.io/crfpp/`, 2005.

Claudia Leacock, Martin Chodorow, Michael Gamon, and Joel Tetreault. Automated grammatical error detection for language learners. 2014.

Matthew Lease, Mark Johnson, and Eugene Charniak. Recognizing disfluencies in conversational speech. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(5):1566–1573, 2006.

John Lee and Stephanie Seneff. Automatic grammar correction for second-language learners. In *INTERSPEECH*, Pittsburgh, Pennsylvania, USA, September 2006. ISCA.

John Lee and Stephanie Seneff. Correcting misuse of verb forms. In *Proceedings of ACL-08: HLT*, pages 174–182, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P08/P08-1021`.

Li-Ching Lee, Angeline B David, Julie Rusyniak, Rebecca Landa, and Craig J Newschaffer. Performance of the social communication questionnaire in children receiving preschool special education services. *Research in Autism Spectrum Disorders*, 1(2):126–138, 2007.

Laurence B Leonard. *Language, Speech, and Communication : Children with Specific Language Impairment (2nd Edition)*. A Bradford Book, Cambridge, MA, USA, 2014.

Laurence B Leonard, Julia A Eyer, Lisa M Bedore, and Bernard G Grela. Three accounts of the grammatical morpheme difficulties of english-speaking children with specific language impairment. *Journal of Speech, Language, and Hearing Research*, 40(4):741–753, 1997.

Willem JM Levelt. *Speaking: From intention to articulation*, volume 1. MIT press, Cambridge, MA, 1993.

Ovsanna T Leyfer, Helen Tager-Flusberg, Michael Dowd, J Bruce Tomblin, and Susan E Folstein. Overlap between autism and specific language impairment: Comparison of autism diagnostic interview and autism diagnostic observation schedule scores. *Autism Research*, 1(5):284–296, 2008.

Fei Liu, Fuliang Weng, Bingqing Wang, and Yang Liu. Insertion, deletion, or substitution? normalizing text messages without pre-categorization nor supervision. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 71–76, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-2013`.

Fei Liu, Fuliang Weng, and Xiao Jiang. A broad-coverage normalization system for social media language. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1035–1044, Jeju Island, Korea, July 2012a. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P12-1109`.

Xiaohua Liu, Ming Zhou, Xiangyang Zhou, Zhongyang Fu, and Furu Wei. Joint inference of named entity recognition and normalization for tweets. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 526–535, Jeju Island, Korea, July 2012b. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P12-1055`.

Walter Loban. Stages, velocity, and prediction of language development kindergarten through grade twelve. *Final Report, Project*, (7-0061), 1970.

Walter Loban. *Language development: Kindergarten through grade twelve*, volume 18. National Council of Teachers of English Urbana, IL, 1976.

Catherine Lord. Autism and the comprehension of language. In *Communication problems in autism*, pages 257–281. Springer, 1985.

Catherine Lord, Susan Risi, Linda Lambrecht, Edwin H Cook Jr, Bennett L Leventhal, Pamela C DiLavore, Andrew Pickles, and Michael Rutter. The autism diagnostic observation scheduleâĂŤgeneric: A standard measure of social and communication deficits associated with the spectrum of autism. *Journal of autism and developmental disorders*, 30(3):205–223, 2000.

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

O Ivar Lovaas and Tristram Smith. Early and intensive behavioral intervention in autism. 2003.

Barbara G MacLachlan and Robin S Chapman. Communication breakdowns in normal and language learning-disabled children's conversation and narration. *Journal of Speech and Hearing Disorders*, 53(1):2, 1988.

Brian MacWhinney. The childes project: Tools for analyzing talk. *Child Language Teaching and Therapy*, 8(2):217–218, 1992.

Brian MacWhinney. The CHILDES project tools for analyzing talk âĂŞ electronic edition part 1: The CHATtranscription format. Technical report, Carnegie Mellon University, Pittsburgh, PA, September 2015. URL `http://childes.psy.cmu.edu/manuals/CHAT.pdf`.

Brian MacWhinney and Catherine Snow. The child language data exchange system: An update. *Journal of child language*, 17(02):457–472, 1990.

Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 188–191. Association for Computational Linguistics, 2003.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics, 2005.

Ryan McDonald, Keith Hall, and Gideon Mann. Distributed training strategies for the structured perceptron. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 456–464. Association for Computational Linguistics, 2010. URL `http://aclweb.org/anthology/N10-1069`.

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

Microsoft. Select grammar and writing style options, 2015. URL `https://support.office.microsoft.com/en-us/article/Select-grammar-and-writing-style-options-86dd1e89-cfb5-4405-94df-48c284af9dbd?CorrelationId=5a7bbad2-6037-4ae1-84c2-2f56d5ff1957&ui=en-US&rs=en-US&ad=US`.

Rada Mihalcea. Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *Proceedings of the conference on Human Language Technology and Empirical Methods in*

Natural Language Processing, pages 411–418. Association for Computational Linguistics, 2005.

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

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

Tim Miller. Improved syntactic models for parsing speech with repairs. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 656–664. Association for Computational Linguistics, 2009.

Tim Miller, Luan Nguyen, and William Schuler. Parsing speech repair without specialized grammar symbols. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 277–280. Association for Computational Linguistics, 2009.

Eric Morley, Brian Roark, and Jan van Santen. 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 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W13-1701`.

Eric Morley, Anna Eva Hallin, and Brian Roark. Challenges in automating maze detection. In *Proceedings of the Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*, pages 69–77, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W14-3209`.

Linda Mortensen, Antje S Meyer, and Glyn W Humphreys. Age-related effects on speech production: A review. *Language and Cognitive Processes*, 21(1-3): 238–290, 2006.

Christian Müller, Barbara Großmann-Hutter, Anthony Jameson, Ralf Rummer, and Frank Wittig. Recognizing time pressure and cognitive load on the basis of speech: An experimental study. In *Proceedings of the 8th International Conference on User Modeling 2001*, UM '01, pages 24–33, London, UK, UK, 2001. Springer-Verlag. ISBN 3-540-42325-7. URL `http://dl.acm.org/citation.cfm?id=647664.733413`.

Christine H. Nakatani and Julia Hirschberg. A speech-first model for repair detection and correction. In Lenhart K. Schubert, editor, *ACL*, pages 46–53, Columbus, Ohio, USA, June 1993. The Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. The conll-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W13-3601`.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2014 Shared Task)*, pages 1–14, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W14/W14-1701`.

Marilyn A Nippold, Tracy C Mansfield, Jesse L Billow, and J Bruce Tomblin.

Expository discourse in adolescents with language impairments: Examining syntactic development. *American Journal of Speech-Language Pathology*, 17 (4):356–366, 2008.

Joakim Nivre. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT*. Citeseer, 2003.

Joakim Nivre. *Inductive dependency parsing*. Springer, 2006.

Joakim Nivre. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 2008.

Joakim Nivre. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 351–359, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-45-9. URL `http://dl.acm.org/citation.cfm?id=1687878.1687929`.

Joakim Nivre, Marco Kuhlmann, and Johan Hall. An improved oracle for dependency parsing with online reordering. In *Proceedings of the 11th International Conference on Parsing Technologies*, IWPT '09, pages 73–76, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1697236.1697250`.

Jorge Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782, 1980.

Eric W Noreen. Computer intensive methods for testing hypotheses. an introduction. 1989. *John Wiley & Sons*, 2(5):33, 1989.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: bringing order to the web. 1999.

Jack L Paradise, Thomas F Campbell, Christine A Dollaghan, Heidi M Feldman, Beverly S Bernard, D Kathleen Colborn, Howard E Rockette, Janine E Janosky, Dayna L Pitcairn, Marcia Kurs-Lasky, et al. Developmental outcomes after early or delayed insertion of tympanostomy tubes. *New England Journal of Medicine*, 353(6):576–586, 2005.

Pearson Education, Inc. Clinical evaluation of language fundamentals fourth edition, 2008. URL `http://images.pearsonassessments.com/images/tmrs/tmrs_rg/CELF_4_Tech_Report.pdf`.

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. Scikit-learn: Machine learning in python. *CoRR*, abs/1201.0490, 2012.

Carole Peterson and Allyssa McCabe. *Developmental psycholinguistics*. Springer Science & Business Media, 1983.

Albert Postma and Herman Kolk. The covert repair hypothesis: Prearticulatory repair processes in normal and stuttered disfluencies. *Journal of Speech and Hearing Research*, 36(3):472, 1993.

Emily T. Prud'hommeaux, Brian Roark, Lois M. Black, and Jan van Santen. Classification of atypical language in autism. In *Proceedings of the 2Nd Workshop on Cognitive Modeling and Computational Linguistics*, CMCL '11, pages 88–96, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-95-4. URL `http://dl.acm.org/citation.cfm?id=2021096.2021106`.

Xian Qian and Yang Liu. 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 2013. The Association for Computational Linguistics.

Scott B Ransom, Maulik Joshi, and David B Nash. *The healthcare quality book: vision, strategy, and tools*. Health Administration Press, 2005.

Mohammad Sadegh Rasooli and Joel R. Tetreault. Joint parsing and disfluency detection in linear time. In *EMNLP*, pages 124–129, Seattle, Washington, USA, October 2013. The Association for Computational Linguistics. ISBN 978-1-937284-97-8.

Mohammad Sadegh Rasooli and Joel R. Tetreault. Non-monotonic parsing of fluent umm i mean disfluent sentences. In Gosse Bouma and Yannick Parmentier, editors, *EACL*, pages 48–53, Gothenburg, Sweden, April 2014. The Association for Computational Linguistics. ISBN 978-1-937284-78-7.

Christina Reuterskiöld Wagner, Ulrika Nettelbladt, Birgitta Sahlén, and Claes Nilholm. Conversation versus narration in pre-school children with language impairment. *International Journal of Language & Communication Disorders*, 35(1):83–93, 2000.

Mabel L Rice, Filip Smolik, Denise Perpich, Travis Thompson, Nathan Rytting, and Megan Blossom. Mean length of utterance levels in 6-month intervals for children 3 to 9 years with and without language impairments. *Journal of Speech, Language, and Hearing Research*, 53(2):333–349, 2010.

Matthew Rispoli, Pamela Hadley, and Janet Holt. Stalls and revisions: A developmental perspective on sentence production. *Journal of Speech, Language, and Hearing Research*, 51(4):953–966, 2008.

Brian Roark and Richard Sproat. Hippocratic abbreviation expansion. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 364–369, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P14-2060`.

Brian Roark and Richard William Sproat. *Computational approaches to morphology and syntax*. Oxford University Press Oxford, 2007.

Jenny A Roberts, Mabel L Rice, and Helen Tager-Flusberg. Tense marking in children with autism. *Applied Psycholinguistics*, 25(03):429–448, 2004.

Sally J Rogers. Empirically supported comprehensive treatments for young children with autism. *Journal of clinical child psychology*, 27(2):168–179, 1998.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Alla Rozovskaya, Kai-Wei Chang, Mark Sammons, and Dan Roth. The university of illinois system in the conll-2013 shared task. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 13–19, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W13-3602`.

Alla Rozovskaya, Kai-Wei Chang, Mark Sammons, Dan Roth, and Nizar Habash. The Illinois-Columbia system in the CoNLL-2014 shared task. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 34–42, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W14-1704`.

Michael Rutter, Anthony Bailey, and Cathrine Lord. *The Social Communication Questionnaire: Manual*. Western Psychological Services, 2003.

LLC SALT Software. Course 1306: Transcription - Conventions Part 3. `http://saltclasses.saltsoftware.com/course/view.php?id=11`, 2014a. [Online; accessed 29-December-2014].

LLC SALT Software. Course 1304: Transcription - Conventions Part 1. `http://saltclasses.saltsoftware.com/course/view.php?id=9`, 2014b. [Online; accessed 29-December-2014].

LLC SALT Software. Conversation Reference Database. `http://saltsoftware.com/media/wysiwyg/reference_database/ConRDBDoc.pdf`, 2014c. [Online; accessed 12-December-2014].

LLC SALT Software. SALT software. `http://saltclasses.saltsoftware.com/`, 2014d. [Online; accessed 4-August-2015].

Hollis Scarborough, Janet Wyckoff, and Robin Davidson. A reconsideration of the relation between age and mean utterance length. *Journal of Speech, Language, and Hearing Research*, 29(3):394–399, 1986.

Phyliss Schneider, Rita Vis Dubé, and Denyse Hayward. The Edmonton Narrative Norms Instrument: Description of the Normative Study. `http://www.rehabresearch.ualberta.ca/enni/manual/description-of-the-normative-study`, 2014. [Online; accessed 1-December-2014].

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

Cheryl M Scott and Jennifer Windsor. General language performance measures in spoken and written narrative and expository discourse of school-age children with language learning disabilities. *Journal of Speech, Language & Hearing Research*, 43(2), 2000.

Eleanor Messing Semel, Elisabeth Hemmersam Wiig, and Wayne Secord. *Clinical evaluation of language fundamentals*. The Psychological Corporation, San Antonio, TX, fourth edition, 2003.

Eleanor Messing Semel, Elisabeth Hemmersam Wiig, and Wayne Secord. *Clinical evaluation of language fundamentals Preschool*. The Psychological Corporation, San Antonio, TX, second edition, 2004.

Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141. Association for Computational Linguistics, 2003.

Mark D Shermis and Jill Burstein. *Handbook of automated essay evaluation: Current applications and new directions*. Routledge, 2013.

Elizabeth Ellen Shriberg. *Preliminaries to a theory of speech disfluencies*. PhD thesis, University of California, Berkeley, 1994.

Lawrence D Shriberg, Rhea Paul, Jane L McSweeny, Ami Klin, Donald J Cohen, and Fred R Volkmar. Speech and prosody characteristics of adolescents and adults with high-functioning autism and asperger syndrome. *Journal of Speech, Language, and Hearing Research*, 44(5):1097–1115, 2001.

Daniel D. Sleator and Davy Temperley. Parsing english with a link grammar, 1991.

Robert L Spitzer and Bryna Siegel. The dsm-iii-r field trial of pervasive developmental disorders. *Journal of the American Academy of Child & Adolescent Psychiatry*, 29(6):855–862, 1990.

Drahomíra "johanka" Spoustová, Jan Hajič, Jan Raab, and Miroslav Spousta. Semi-supervised training for the averaged perceptron POS tagger. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 763–771, Athens, Greece, March 2009. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/E09-1087`.

Richard Sproat, Alan W Black, Stanley Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards. Normalization of non-standard words. *Computer Speech & Language*, 15(3):287–333, 2001.

Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. *Introduction to statistical relational learning*, pages 93–128, 2006.

E Swan. Ko au na galo (ana gets lost). *Wellington, NZ: Learning Media, Ministry of Education*, 1992.

Helen Tager-Flusberg. On the nature of linguistic functioning in early infantile autism. *Journal of Autism and Developmental Disorders*, 11(1):45–56, 1981.

Helen Tager-Flusberg. Psycholinguistic approaches to language and communication in autism. In *Communication problems in autism*, pages 69–87. Springer, 1985.

Helen Tager-Flusberg, Rhea Paul, Catherine Lord, et al. Language and communication in autism. *Handbook of autism and pervasive developmental disorders*, 1:335–364, 2005.

Helen Tager-Flusberg, Sally Rogers, Judith Cooper, Rebecca Landa, Catherine Lord, Rhea Paul, Mabel Rice, Carol Stoel-Gammon, Amy Wetherby, and Paul Yoder. 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, 2009.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Maximum-margin markov networks. In *Neural Information Processing Systems (NIPS)*, 2003.

Lucien Tesnière. *Eléments de syntaxe structurale*. Librairie C. Klincksieck, 1959.

Elin T Thordardottir and Susan Ellis Weismer. Content mazes and filled pauses in narrative language samples of children with specific language impairment. *Brain and cognition*, 48(2-3):587–592, 2001.

J Bruce Tomblin, Nancy L Records, Paula Buckwalter, Xuyang Zhang, Elaine Smith, and Marlea O'Brien. Prevalence of specific language impairment in kindergarten children. *Journal of Speech, Language, and Hearing Research*, 40 (6):1245–1260, 1997.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, volume 1, pages 173–180. Association for Computational Linguistics, 2003.

Antal van den Bosch and Peter Berck. Memory-based grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 102–108, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W13-3614`.

Joanne Volden and Catherine Lord. Neologisms and idiosyncratic language in autistic speakers. *Journal of autism and developmental disorders*, 21(2): 109–130, 1991.

Wolfgang Wahlster. *Verbmobil: foundations of speech-to-speech translation*. Springer Science & Business Media, 2000.

David Wechsler. The wechsler primary and preschool scale of intelligenceâĂŤ, 2002.

David Wechsler. *Wechsler Intelligence Scale for Children: WISC-IV*. Psychological Corporation, 2003.

Danielle Wetherell, Nicola Botting, and Gina Conti-Ramsden. Narrative in adolescent specific language impairment (sli): A comparison with peers across two different narrative genres. *International Journal of Language & Communication Disorders*, 42(5):583–605, 2007.

B Wheatley, G Doddington, C Hemphill, J Godfrey, EC Holliman, J McDaniel, and D Fisher. Switchboard: A userâĂŹs manual, 1995.

Alexander S. Yeh. More accurate tests for the statistical significance of result differences. In *COLING*, pages 947–953, Saarbrücken, Germany, July 2000. The Association for Computational Linguistics.

Daniel H Younger. Recognition and parsing of context-free languages in time n 3. *Information and control*, 10(2):189–208, 1967.

F Benjamin Zhan and Charles E Noon. Shortest path algorithms: an evaluation using real road networks. *Transportation Science*, 32(1):65–73, 1998.

Yue Zhang and Stephen Clark. Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, 37(1):105–151, 2011.

Yue Zhang and Joakim Nivre. Transition-based dependency parsing with rich non-local features. In *ACL (Short Papers)* DBL (2011), pages 188–193. ISBN 978-1-932432-88-6.

Simon Zwarts and Mark Johnson. The impact of language models and loss functions on repair disfluency detection. In *ACL* DBL (2011), pages 703–711. ISBN 978-1-932432-88-6.

Arnold M Zwicky. Heads. *Journal of linguistics*, 21(01):1–29, 1985.