Published in final edited form as:

Proc Conf. 2015 June 5; 2015: 108–116.

Automated morphological analysis of clinical language samples

Kyle Gorman, **Steven Bedrick**, **Géza Kiss**, **Eric Morley**, **Rosemary Ingham**, **Metrah Mohammad**, **Katina Papadakis**, and **Jan P.H. van Santen**

Center for Spoken Language Understanding, Oregon Health & Science University, Portland, OR, USA

Abstract

Quantitative analysis of clinical language samples is a powerful tool for assessing and screening developmental language impairments, but requires extensive manual transcription, annotation, and calculation, resulting in error-prone results and clinical underutilization. We describe a system that performs automated morphological analysis needed to calculate statistics such as the mean length of utterance in morphemes (MLUM), so that these statistics can be computed directly from orthographic transcripts. Estimates of MLUM computed by this system are closely comparable to those produced by manual annotation. Our system can be used in conjunction with other automated annotation techniques, such as maze detection. This work represents an important first step towards increased automation of language sample analysis, and towards attendant benefits of automation, including clinical greater utilization and reduced variability in care delivery.

1 Introduction

Specific language impairment (SLI) is a neurodevelopmental disorder characterized by language delays or deficits in the absence of other developmental or sensory impairments (Tomblin, 2011). A history of specific language impairment is associated with a host of difficulties in adolescence and adulthood, including poorer quality friendships (Durkin and Conti-Ramsden, 2007), a greater risk for psychiatric disturbance (Durkin and Conti-Ramsden, 2010), and diminished educational attainment and occupational opportunities (Conti-Ramsden and Durkin, 2012). SLI is common but remains significantly underdiagnosed; one large-scale study estimates that over 7% of kindergarten-aged monolingual English speaking children have SLI, but found that the parents of most of these children were unaware that their child had a speech or language problem (Tomblin et al., 1997).

Developmental language impairments are normally assessed using standardized tests such as the Clinical Evaluation of Language Fundamentals (CELF), a battery of norm-referenced language tasks such as Recalling Sentences, in which the child repeats a sentence, and Sentence Structure, in which the child points to a picture matching a sentence. However, there has been a recent push to augment norm-referenced tests with language sample analysis (Leadholm and Miller, 1992; Miller and Chapman, 1985), in which a spontaneous

language sample collected from a child is used to compute various statistics measuring expressive language abilities.

Natural language processing (NLP) has the potential to open new frontiers in language sample analysis. For instance, some recent work has applied NLP techniques to quantify clinical impressions that once were merely qualitative (e.g., Rouhizadeh et al. 2013, van Santen et al. 2013) and other work has proposed novel computational features for detecting language disorders (e.g., Gabani et al. 2011). In this study, our goal is somewhat simpler: we attempt to apply novel NLP techniques to assist the clinician by automating the computation of firmly established spontaneous language statistics.

Quantitative analysis of language samples is a powerful tool for assessing and screening developmental language impairments. Measures derived from naturalistic language samples are thought to be approximately as sensitive to language impairment as are decontextualized tests like those that make up the CELF (Aram et al., 1993); they may also be less biased against speakers of non-standard dialects (Stockman, 1996). Despite this, language sample analysis is still underutilized in clinical settings, in part due to the daunting amount of manual transcription and annotation required.

Clinicians may avail themselves of software like Systematic Analysis of Transcripts (SALT; Miller and Iglesias 2012), which partially automates the language sample analysis. But this tool (and others like it) require the clinician to provide not only a complete orthographic transcription, but also detailed linguistic annotations using a complex and unforgiving annotation syntax that itself takes significant effort to master. In what follows, we describe a system which automates a key part of this annotation process: the tedious and error-prone annotation of morphological structure.

In the next section, we describe *mean length of utterance in morphemes* (MLUM), a widely used measure of linguistic productivity, and associated morphological annotations needed to compute this measure. We then outline a computational model which uses a cascade of linear classifiers and finite-state automata to generate these morphological annotations; this allows MLUM to be computed directly from an orthographic transcription. Our evaluation demonstrates that this model produces estimates of MLUM which are very similar to those produced by manual annotation. Finally, we outline directions for future research.

2 Mean length of utterance and morphological annotations

Mean length of utterance in morphemes is a widely-used measure of linguistic productivity in children, consisting essentially of the average number of morphemes per utterance. Brown (1973), one of the first users of MLUM, describes it as a simple, face-valid index of language development simply because nearly any linguistic feature newly mastered by the child—be it obligatory morphology, more complex argument structure, or clausal recursion —results in an increase in the average utterance length. MLUM has also proven useful in diagnosing developmental language impairments. For instance, typically-developing children go through a stage where they omit affixes and/or function words which are obligatory in their target language (e.g., Harris and Wexler 1996; Legate and Yang 2007).

Children with language impairment are thought to omit obligatory morphemes at a higher rate than their typically-developing peers (Eisenberg et al., 2001; Rice and Wexler, 1996; Rice et al., 1998; Rice et al., 2006), and differences in omission rate can be detected, albeit indirectly, with MLUM.

SALT (Miller and Chapman, 1985) provides specific guidelines for estimating MLUM. These guidelines are concerned both with what utterances and tokens "count" towards MLUM, as well as which tokens are to be considered morphologically complex. The SALT guidelines require that complex words be written by writing the free stem form of the word, followed by a forward-slash (/) and an unambiguous signature representing the suffix. SALT recognizes 13 "suffixes", including the noun plural (dog/s), possessive (mom/z), preterite/past participle (walk/ed), progressive/future (stroll/ing), and various enclitics (I/'m, we/'re, is/n't); some SALT suffixes can also be combined (e.g., the plural possessive boy/s/z). Each SALT suffix is counted as a single morpheme, as are all stems and simplex words. Irregulars affixes (*felt*), derivational affixes (*un-lock*, *write-r*), and compounds (*break-fast*) are not annotated, and words bearing them are counted as a single morpheme unless these words happen to contain one of the aforementioned SALT suffixes.

In the next section, we propose a computational model which generates SALT-like morphological annotations. Our highest priority is to be faithful to the SALT specification, which has proved sufficient for the creators' well-defined, clinically-oriented aims. We do not claim that our system will generalize to any other linguistic annotation scheme, but only that we have successfully automated SALT-style morphological annotations. We recognize the limitations of the SALT specification: it draws little inspiration from linguistic theory, and furthermore fails to anticipate the possibility of the sort of automation we propose. As it happens, there is a large body of work in natural language processing on automated methods for morphological segmentation and/or analysis, which could easily be applied to this problem. Yet, the vast majority of this literature is concerned with unsupervised learning (i.e., inducing morphological analyses from unlabeled data) rather than the (considerably easier) task of mimicking morphological analyses produced by humans, our goal here. (For one exception, see the papers in Kurimo et al. 2010.) While it would certainly be possible to adapt existing unsupervised morphological analyzers to implement the SALT specification, the experiments presented below demonstrate that simple statistical models, trained on a small amount of data, achieve near-ceiling performance at this task. Given this result, we feel that adapting existing unsupervised systems to this task would be a purely academic exercise.

3 The model

We propose a model to automatically generate SALT-compatible morphological annotations, as follows. First, *word extraction* identifies words which count towards MLUM. Then, *suffix prediction* predicts the most likely set of suffixes for each word. Finally, *stem analysis* maps complex words back to their stem form. These three steps generate all the information necessary to compute MLUM. We now proceed to describe each step in more detail.

3.1 Word extraction

The SALT guidelines excludes any speech which occurs during an incomplete or abandoned utterance, speech in utterances that contain incomprehensible words, and speech during *mazes*—i.e., disfluent intervals, which encompass all incomplete words and fillers—for the purpose of computing MLUM and related statistics. A cascade of regular expressions are used to extract a list of eligible word tokens from individual lines of the orthographic transcript.

3.2 Suffix prediction

Once unannotated word tokens have been extracted, they are input to a cascade of two linear classifiers. The first classifier makes a binary prediction as to whether the token is morphologically simplex or complex. If the token is predicted to be complex, it is input to a second classifier which attempts to predict which combination of the 13 SALT suffixes is present.

Both classifiers are trained with held-out-data using the perceptron learning algorithm and weight averaging (Freund and Schapire, 1999). We report results using four feature sets. The baseline model uses only a bias term. The ϕ_0 set uses orthographic features inspired by "rare word" features used in part-of-speech tagging (Ratnaparkhi, 1997) and intended to generalize well to out-of-vocabulary words. In addition to bias, ϕ_0 consists of six orthographic features of the target token (w_i), including three binary features ("w_i contains an apostrophe", "w_i is a sound effect", "w_i is a hyphenated word") and all proper string suffixes of w_i up to three characters in length. The ϕ_1 feature set adds a nominal attribute, the identity of w_i. Finally, ϕ_2 also includes four additional nominal features, the identity of the nearest tokens to the left and right (w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2}). Four sample feature vectors are shown in Table 1.

3.3 Stem analysis

Many English stems are spelled somewhat differently in free and bound (i.e., bare and inflected) form. For example, stem-final usually changes to *i* in the past tense (e.g., *buried*), and stem-final *e* usually deletes before the progressive (e.g., *bouncing*). Similarly, the SALT suffixes have different spellings depending on context; the noun plural suffix is spelled *es* when affixed to stems ending in stridents (e.g., *mixes*), but as *s* elsewhere. To model these spelling changes triggered by suffixation, we use finite state automata (FSAs), mathematical models widely used in both natural language processing and speech recognition. Finite state automata can be used implement a cascade of context-dependent rewrite rules (e.g., "α goes to β in the context δ__γ") similar to those used by linguists in writing phonological rules. This makes FSAs particularly well suited for dealing with spelling rules like the ones described above.

This spell-out transducer can also be adapted to recover the stem of a wordform, once morphological analysis has been performed. If I is the input wordform, S is the spell-out transducer, and D is a simple transducer which deletes whatever suffixes are present, then the output-tape symbols of $I \circ S^{-1} \circ D$ contain the original stem.[1] However, there may be multiple output paths for many input wordforms. For instance, a doubled stem-final

consonant in the inflected form could either be present in the bare stem (e.g., *guess* → *guessing*) or could be a product of the doubling rule (e.g., *run* → *running*); both are permitted by S^{-1}. To resolve these ambiguities, we employ a simple probabilistic method. Let W be a weighted finite-state acceptor in which each path represents a stem, and the cost of each path is proportional to that stem's frequency.[2] Then, the most likely stem given the input wordform and analysis is given by the output-tape symbols of

$$\text{ShortestPath}(I \circ S^{-1} \circ D \circ W).$$

Both the spell-out transducer and the stemmer were generated using the Thrax grammar-compilation tools (Roark et al., 2012); a full specification of both models is provided in the appendix.

4 Evaluation

We evaluate the model with respect to its ability to mimic human morphological annotations, using three intrinsic measures. *Suffix detection* refers to agreement on whether or not an eligible word is morphologically complex. *Suffix classification* refers to agreement as to which suffix or suffixes are borne by a word which has been correctly classified as morphologically complex by the suffix detector. Finally, *token agreement* refers agreement as to the overall morphological annotation of an eligible word. We also evaluate the model extrinsically, by computing the Pearson product-moment correlation between MLUM computed from manual annotated data to MLUM computed from automated morphological annotations. In all evaluations, we employ a "leave one child out" cross-validation scheme.

4.1 Data

Our data comes from a large-scale study of autism spectrum disorders and language impairment in children. 110 children from the Portland, OR metropolitan area, between 4–8 years of age, took part in the study: 50 children with autism spectrum disorders (ASD), 43 typically-developing children (TD), and 17 children with specific language impairment (SLI). All participants had full-scale IQ scores of 70 or higher. All participants spoke English as their first language, and produced a mean length of utterance in morphemes (MLUM) of at least 3. During the initial screening, a certified speech-language pathologist verified the absence of speech intelligibility impairments. For more details on this sample, see van Santen et al. 2013.

The ADOS (Lord et al., 2000), a semi-structured autism diagnostic observation, was administered to all children in the current study. These sessions were recorded and used to generate verbatim transcriptions of the child and examiner's speech. Transcriptions were

[1] An anonymous reviewer asks how this "stemmer" relates to familiar tools such as the Porter (1980) stemmer. The stemmer described here takes morphologically annotated complex words as input and outputs the uninflected ("free") stem. In contrast, the Porter stemmer takes unannotated words as input and outputs a "canonical" form—crucially, not necessarily a real word—to be used in downstream analyses.

[2] To prevent composition failure with out-of-vocabulary stems, the acceptor W is also augmented with additional arcs permitting it to accept, with some small probability, the closure over the vocabulary.

generated using SALT guidelines. Conversational turns were segmented into individual utterances (or “C-units”), each of which consisted of (at most) a main clause and any subordinate clauses modifying it.

4.2 Interannotator agreement

Manual annotation quality was assessed using a stratified sample of the full data set, consisting of randomly-selected utterances per child. These utterances were stripped of their morphological annotations and then re-annotated by two experienced transcribers, neither of whom participated in the initial transcription efforts. The results are shown in Table 2. On all three intrinsic measures, the original and retrospective annotators agreed an overwhelming amount of the time; the K (chance-adjusted agreement) values for the former two indicate “almost perfect” (Landis and Koch, 1977) agreement according to standard qualitative guidelines.

4.3 Results

Table 3 summarizes the intrinsic evaluation results. The baseline system performs poorly both in suffix detection and suffix classification. Increasingly complex feature sets result in significant increases in both detection and classification. Even though most eligible words are not morphologically complex, the full feature set (ϕ_2) produces a good balance of precision and recall and correctly labels nearly 99% of all eligible word tokens. MLUMs computed using the automated annotations and the full feature set are almost identical to MLUMs derived from manual annotations ($R = .9998$).

This table also shows accuracies for two particularly difficult morphological distinctions, between the noun plural S and the 3rd person active indicative suffix 3s (*seeks*), and between the possessive ‘S and Z (the contracted form of *is*), respectively. These distinctions in particular appear to benefit in particular from the contextual features of the ϕ_2 feature set.

In the above experiments, the data contained manually generated annotations of mazes. These are required for computing measures like MLUM, as speech in mazes is ignored when counting the number of morphemes in an utterance. Like morphological annotations, human annotation of mazes is also tedious and time-consuming. However, some recent work has attempted to automatically generate maze annotations from orthographic transcripts (Morley et al., 2014a), and automatic maze annotation would greatly increase the utility of the larger system described here.

We thus performed a simple “pipeline” evaluation of the morphological annotation system, as follows. First, maze annotations are automatically generated for each transcript. We then feed the maze-annotated transcripts into the morphological analyzer described above, which is then used to compute MLUM. The maze annotation system used here was originally developed by Qian and Liu (2013) for detecting fillers in Switchboard as an early step in a larger disfluency detection system; Morley et al. (2014a) adapted it for maze detection. This system is trained from a dataset of transcripts with manually-annotated mazes; here we depart from the prior work in training it using a leave-one-child-out strategy. Features used are derived from tokens and automatically generated part-of-speech tags. This system treats

maze detection as a sequence labeling task performed using a max-margin Markov network (Taskar et al., 2004); for more details, see Morley et al. 2014a.

We hypothesized that the errors introduced by automated maze annotation would not greatly affect MLUM estimates, as maze detection errors do not necessarily impact MLUM. For example, an utterance like *I went to I go to school* might be bracketed as either (I went to) I go to school and I went to (I go to) school, but either analysis results in the same MLUM. And in fact, MLUMs computed using the combined maze detection/morphological annotation system are competitive with MLUMs derived from manual annotations ($R = .9991$).

4.4 Discussion

Our results show that the proposed morphological analysis model produces accurate annotations, which then can be used to compute relatively precise estimates of MLUM. Furthermore, automation of other SALT-style annotations (such as maze detection) does not negatively impact automatic MLUM estimates.

We experimented with other feature sets in the hopes of approving accuracy and generalizability. We hypothesized that suffix classification would benefit from part-of-speech features. Since our data was not manually part-of-speech tagged, we extracted these features using an automated tagger similar to the one described in (Collins, 2002).[3] The tagger was trained on a corpus of approximately 150,000 utterances of child-directed speech (Pearl and Sprouse, 2013) annotated with a 39-tag set comparable to the familiar PTB tagset. Additional POS features were also generated by mapping the 39-tag set down to a smaller set of 11 “universal” tags (Petrov et al., 2012). However, neither set of POS features produced any appreciable gains in performance. We speculate that these features are superfluous given the presence of the ϕ_2 word context features.

5 Conclusions

We have described a principled and accurate system for automatic calculation of widely-used measures of expressive language ability in children. The system we propose does *not* require extensive manual annotation, nor does it require expensive or difficult-to-use proprietary software, another potential barrier to use of these measures in practice. It is trained using a small amount of annotated data, and could easily be adapted to similar annotation conventions in other languages.

We view this work as a first step towards increasing the use of automation in language assessment and other language specialists. We foresee two benefits to automation in this area. First, it may reduce time spent in manual annotation, increasing the amount of time clinicians spend interacting with patients face to face. Second, increased automation may lead to decreased variability in care delivery, a necessary step towards improving outcomes (Ransom et al., 2008).

[3]The tagger was tested using the traditional “standard split” of the Wall St. Journal portion of the Penn Treebank, with sections 0–18 for training, sections 19–21 for development, and sections 22–24 for evaluation. The tagger correctly assigned 96.69% of the tags for the evaluation set.

One remaining barrier to wider use of language sample analysis is the need for manual transcription, which is time-consuming even when later annotations are generated automatically. Future work will consider whether transcripts derived from automatic speech recognition are capable of producing valid, unbiased estimates of measures like MLUM.

Our group has made progress towards automating other clinically relevant annotations, including grammatical errors (Morley et al., 2014b) and repetitive speech (van Santen et al., 2013), and we are actively studying ways to integrate our various systems into a full suite of automated language sample analysis utilities. More importantly, however, we anticipate collaborating closely with our clinical colleagues to develop new approaches for integrating automated assessment tools into language assessment and treatment workflows—an area in which far too little research has taken place.

Acknowledgments

All experiments were conducted using OpenFst (Allauzen et al., 2007), OpenGrm-Thrax (Roark et al., 2012), and Python 3.4. A demonstration version of the system can be viewed at the following URL: http://sonny.cslu.ohsu.edu:8080.

Thanks to the other members of the CSLU autism research group, and to Emily Prud'hommeaux and Mabel Rice.

This material is based upon work supported by the National Institute on Deafness and Other Communication Disorders of the National Institutes of Health under awards R01DC007129 and R01DC012033, and by Autism Speaks under Innovative Technology for Autism Grant 2407.

References

Allauzen, Cyril, Riley, Michael, Schalkwyk, Johan, Skut, Wojciech, Mohri, Mehryar. OpenFst: A general and efficient weighted finite-state transducer library. Proceedings of the 9th International Conference on Implementation and Application of Automata; 2007. p. 11-23.

Aram, Dorothy M., Morris, Robin, Hall, Nancy E. Clinical and research congruence in identifying children with specific language impairment. Journal of Speech and Hearing Research. 1993; 36(3): 580–591. [PubMed: 8331914]

Brown, Roger. A first language: The early stages. Harvard University Press; Cambridge: 1973.

Collins, Michael. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. EMNLP. 2002:1–8.

Conti-Ramsden, Gina, Durkin, Kevin. Postschool educational and employment experiences of young people with specific language impairment. Language, Speech, and Hearing Services in Schools. 2012; 43(4):507–520.

Durkin, Kevin, Conti-Ramsden, Gina. Language, social behavior, and the quality of friendships in adolescents with and without a history of specific language impairment. Child Development. 2007; 78(5):1441–1457. [PubMed: 17883441]

Durkin, Kevin, Conti-Ramsden, Gina. Young people with specific language impairment: A review of social and emotional functioning in adolescence. Child Language Teaching and Therapy. 2010; 26(2):105–121.

Eisenberg, Sarita L., McGovern Fersko, Tara, Lundgren, Cheryl. The use of MLU for identifying language impairment in preschool children: A review. American Journal of Speech-Language Pathology. 2001; 10(4):323–342.

Freund, Yoav, Schapire, Robert E. Large margin classification using the perceptron algorithm. Machine Learning. 1999; 37(3):277–296.

Gabani, Keyur, Solorio, Thamar, Liu, Yang, Hassanali, Khairun-nisa, Dollaghan, Christine A. Exploring a corpus-based approach for detecting language impairment in monolingual English-speaking children. Artificial Intelligence in Medicine. 2011; 53(3):161–170. [PubMed: 21937203]

Harris, Tony, Wexler, Kenneth. The optional-infinitive stage in child English: Evidence from negation. In: Clahsen, Harald, editor. Generative perspectives on language acquisition: Empirical findings. John Benjamins; Amsterdam: 1996. p. 1-42.

Kurimo, Mikko, Virpioja, Sami, Turunen, Ville T. Technical Report TKK-ICS-R37. Aalto University School of Science and Technology; 2010. Proceedings of the Morpho Challenge 2010 workshop.

Richard Landis J, Koch Gary G. The measurement of observer agreement for categorical data. Biometrics. 1977; 33(1):159–174. [PubMed: 843571]

Leadholm, Barbara J., Miller, Jon F. Language sample analysis: The Wisconsin guide. Wisconsin Department of Public Instruction; Madison, WI: 1992.

Legate, Julie A., Yang, Charles. Morphosyntactic learning and the development of tense. Language Acquisition. 2007; 14(3):315–344.

Lord, Catherine, Risi, Susan, Lambrect, Linda, Jr, Cook, Edwin H., Leventhal, Bennett L., DiLavore, Pamela C., Pickles, Andrew, Rutter, Michael. The Autism Diagnostic Observation Schedule-Generic: A standard measure of social and communication deficits associated with the spectrum of autism. Journal of Autism and Developmental Disorders. 2000; 30(3):205–223. [PubMed: 11055457]

Miller, Jon F., Chapman, Robin S. Systematic Analysis of Language Transcripts. University of Wisconsin; Madison, WI: 1985.

Miller, Jon F., Iglesias, Aquiles. Systematic Analysis of Language Transcripts, Research Version 2012. SALT Software, LLC; Middleton, WI: 2012.

Morley, Eric, Hallin, Anna Eva, Roark, Brian. Challenges in automating maze detection. ACL CLPsych. 2014a:69–77.

Morley, Eric, Hallin, Anna Eva, Roark, Brian. Data-driven grammatical error detection in transcripts of children's speech. EMNLP. 2014b:980–989.

Pearl, Lisa, Sprouse, Jon. Syntactic islands and learning biases: Combining experimental syntax and computational modeling to investigate the language acquisition problem. Language Acquisition. 2013; 20(1):23–68.

Petrov, Slav, Das, Dipanjan, McDonald, Ryan. A universal part-of-speech tagset. LREC. 2012:2089–2096.

Porter, Martin F. An algorithm for suffix stripping. Program. 1980; 14(3):130–137.

Qian, Xian, Liu, Yang. Disfluency detection using multi-step stacked learning. NAACL-HLT, pages. 2013; 820–825

Ransom, Elizabeth R., Joshi, Maulik S., Nash, David B., Ransom, Scott B. The healthcare quality book: Vision, strategy, and tools. 2. Health Administration Press; Chicago: 2008.

Ratnaparkhi, Adwait. A maximum entropy model for part-of-speech tagging. EMNLP. 1997:133–142.

Rice, Mabel L., Wexler, Kenneth. Towards tense as a clinical marker of Specific Language Impairment in English-speaking children. Journal of Speech and Hearing Research. 1996; 39(6):1239–1257. [PubMed: 8959609]

Rice, Mabel L., Wexler, Kenneth, Hershberger, Scott. Tense over time: The longitudinal course of tense acquisition in children with Specific Language Impairment. Journal of Speech, Language, and Hearing Research. 1998; 41(6):1412–1431.

Rice, Mabel L., Redmond, Sean M., Hoffman, Lesa. Mean length of utterance in children with specific language impairment and in younger control children shows concurrent validity, stable and parallel growth trajectories. Journal of Speech, Language, and Hearing Research. 2006; 49(4):793–808.

Roark, Brian, Sproat, Richard, Allauzen, Cyril, Riley, Michael, Sorensen, Jeffrey, Tai, Terry. The OpenGrm open-source finite-state grammar software libraries. ACL. 2012:61–66.

Rouhizadeh, Masoud, Prud'hommeaux, Emily, Roark, Brian, vanSanten, Jan. Distributional semantic models for the evaluation of disordered speech. NAACL-HLT. 2013:709–714.

Stockman, Ida J. The promises and pitfalls of language sample analysis as an assessment tool for linguistic minority children. Language, Speech, and Hearing Services in Schools. 1996; 27(4): 355–366.

Taskar, Ben, Guestrin, Carlos, Koller, Daphne. Max-margin Markov networks. NIPS. 2004:25–32.

Bruce Tomblin J, Records Nancy L, Buckwalter Paula, Zhang Xuyang, Smith Elaine, O'Brien Marlea. Prevalence of specific language impairment in kindergarten children. Journal of Speech, Language, and Hearing Research. 1997; 6:1245–1260.

Bruce Tomblin J. Co-morbidity of autism and SLI: Kinds, kin and complexity. International Journal of Language and Communication Disorders. 2011; 46(2):127–137. [PubMed: 21401812]

van Santen, Jan, Sproat, Richard, Hill, Alison Presmanes. Quantifying repetitive speech in autism spectrum disorders and language impairment. Autism Research. 2013; 6(5):372–383. [PubMed: 23661504]

Appendix

```
# spellout.grm: finite-state spell-out transducers for English
## alphabets
symbol = "%" | "_" | "-" | "/" | "\\" | "'" | "3";
V = "A" | "E" | "I" | "O" | "U";
C = "B" | "C" | "D" | "F" | "G" | "H" | "J" | "K" | "L" | "M" |
    "N" | "P" | "Q" | "R" | "S" | "T" | "V" | "W" | "X" | "Y" | "Z";
letter = V | C;
stemsym = Optimize[letter | symbol];
## suffixes
s  = "/S" | "/3S"; # since these always behave exactly the same
z  = "/Z";
ed  = "/ED";
ing  = "/ING";
_d  = "/'D";
_m  = "/'M";
_s  = "/'S";
_t   = "/'T";
_ll  = "/'LL";
_re  = "/'RE";
_ve = "/'VE";
n_t  = "/N'T";
suffix = Optimize[s | z | ed | ing | _d | _m | _s | _t | _ll | _re |
              _ve | n_t];
alphabet = Optimize[(stemsym | suffix)*];
## stem change rules
# y -> i, as in bury -> buried
y2i_suffix = s | ed;
y2i = Optimize[CDRewrite["Y": "IE", C, y2i_suffix, alphabet]];
# ie -> y, as in die -> dying
ie2y = Optimize[CDRewrite["IE": "Y", C, ing, alphabet]];
# e -> 0 / __ ing
e_exc = "[BOS]" ("BE" | "EYE");
e_before_ing = Optimize[CDRewrite["": "_", e_exc, ing, alphabet] @
                    CDRewrite["E": "", C,    ing, alphabet] @
```

```
                    CDRewrite["_": "", e_exc, ing, alphabet]];
# f -> ve, as in wolf -> wolves
f2ve_words = "CAL" | "EL" | "DWAR" | "HAL" | "HOO" | "LEA" | "LOA" |
            "SCAR" | "SEL" | "SHEL" | "STAF" | "THIE" | "WOL";
f2ve = Optimize[CDRewrite["F"{1,2}: "VE", f2ve_words, s, alphabet]];
# doubling rules
doubling_left = (C | "QU") V;
doubling_suffix = ing | ed;
func CopyRule[X, left_context, right_context, alphabet] {
    return CDRewrite[X: X X, left_context, right_context, alphabet];
}
double = Optimize[CopyRule["B", doubling_left, doubling_suffix, alphabet] @
                CDRewrite["C": "CK", V, doubling_suffix,    alphabet] @
                CopyRule["D", doubling_left, doubling_suffix, alphabet] @
                CopyRule["F", doubling_left, doubling_suffix, alphabet] @
                CopyRule["G", doubling_left, doubling_suffix, alphabet] @
                CopyRule["M", doubling_left, doubling_suffix, alphabet] @
                CopyRule["N", doubling_left, doubling_suffix, alphabet] @
                CopyRule["P", doubling_left, doubling_suffix, alphabet] @
                CopyRule["S", doubling_left, doubling_suffix, alphabet] @
                CopyRule["T", doubling_left, doubling_suffix, alphabet] @
                CopyRule["M", doubling_left, doubling_suffix, alphabet] @
                CopyRule["N", doubling_left, doubling_suffix, alphabet] @
                CopyRule["P", doubling_left, doubling_suffix, alphabet] @
                CopyRule["S", doubling_left, doubling_suffix, alphabet] @
                CopyRule["T", doubling_left, doubling_suffix, alphabet] @
                # now, exceptions
                CDRewrite["TT": "T", "VISI", doubling_suffix, alphabet] @
                CDRewrite["NN": "N", "E",   doubling_suffix, alphabet]];
stem_rules = y2i @ ie2y @ f2ve @ double @ e_before_ing;
## suffix rules
es_cntx = ("CH" | "SH" | "S" | "Z" | "X" | "GO");
s_spellout = Optimize[CDRewrite["": "S", "", s, alphabet] @
                  CDRewrite["": "E", es_cntx, "S" s, alphabet]];
sz_coalescence = Optimize[CDRewrite["": "'", "", s z, alphabet]];
z_spellout = Optimize[CDRewrite["": "'S", "", z, alphabet] @
                  # this overgenerates when /z is preceded by an /s,
                  # so we just undo that
                  CDRewrite["'S": "", s, z, alphabet]];
ed_cntx = C | ("I" | "O");
ed_spellout = Optimize[CDRewrite["": "D", "", ed, alphabet] @
                   CDRewrite["": "E", ed_cntx, "D" ed, alphabet]];
func SpelloutRule[string, suffix, alphabet] {
    return CDRewrite["": string, "", suffix "[EOS]", alphabet];
```

```
}
other_spellout = Optimize[SpelloutRule[“ING”, ing, alphabet] @
                      SpelloutRule[“‘D”,  _d,  alphabet] @
                      SpelloutRule[“‘M”,  _m, alphabet] @
                      SpelloutRule[“‘S”,  _s,  alphabet] @
                      SpelloutRule[“‘T”,  _t,  alphabet] @
                      SpelloutRule[“‘LL”, _ll,  alphabet] @
                      SpelloutRule[“‘RE”, _re, alphabet] @
                      SpelloutRule[“‘VE”, _ve, alphabet] @
                      SpelloutRule[“N’T”, n_t, alphabet]];
suffix_rules = s_spellout @ sz_coalescence @ z_spellout @ ed_spellout @
other_spellout;
## putting it all together
export spellout = Optimize[stem_rules @ suffix_rules];
```

Table 1

Sample features for the utterance *I'm looking for one dinosaur*; each column represents a separate feature vector.

	I'm	**looking**	**for**	**one**	**dinosaur**
ϕ_0	*apostrophe*				
	suf1="M"	suf1="G"	suf1="R"	suf1="E"	suf1="R"
	suf2="'M"	suf2="NG"	suf2="OR"	suf2="NE"	suf2="UR"
		suf3="ING"			suf3="AUR"
ϕ_1	w_i="I'M"	w_i="LOOKING"	w_i="FOR"	w_i="ONE"	w_i="DINOSAUR"
ϕ_2	*initial*	*peninitial*	w_i−2="I'M"	w_i−2="LOOKING"	w_i−2="FOR"
		w_i−1="I'M"	w_i−1="LOOKING"	w_i−1="FOR"	w_i−1="ONE"
	w_i+1="LOOKING"	w_i+1="FOR"	w_i+1="ONE"	w_i+1="DINOSAUR"	*ultimate*
	w_i+2="FOR"	w_i+2="ONE"	w_i+2="PET"	*penultimate*	

Table 2

Interannotator agreement statistics for suffix detection, suffix identity, and overall token-level agreement; the K values indicate “almost perfect agreement” (Landis and Koch, 1977) according to qualitative guidelines.

	Anno. 1	Anno. 2
Suffix detection K	.9207	.9529
Suffix classification K	.9135	.9452
Token agreement	.9803	.9869

Table 3

Intrinsic analysis results on suffix detection, suffix classification, and overall token accuracy.

	Baseline	ϕ_0	ϕ_1	ϕ_2
Suffix detection				
Accuracy	.8122	.9667	.9879	.9913
Precision		.8710	.9508	.9610
Recall		.8393	.9451	.9644
F_1		.8549	.9479	.9627
Suffix classification				
Overall accuracy	.1917	.8916	.9689	.9880
S vs. 3S accuracy		.7794	.9478	.9788
‘S vs. Z accuracy		.9341	.9469	.9923
Token accuracy	.8267	.9663	.9878	.9899