SASTA Self Assessment: An efficient human-in-the-loop strategy for developmental and pathological language analysis

Jan Odijk, Jelte van Boheemen, Xander Vertegaal, Tessel Boerma, Marijn Schraagen
Utrecht University, the Netherlands
{j.odijk, j.vanboheemen, a.j.j.vertegaal, t.d.boerma, m.p.schraagen}@uu.nl

Abstract

This paper introduces SASTA self-assessment, i.e., a self-assessment procedure for the SASTA application for semiautomatic analysis of spontaneous language transcripts for Dutch. We introduce SASTA and the methods that it supports. These methods are used to assess the language development of young children and to assess the language skills of patients with aphasia. We illustrate this with typical example utterances. The performance of SASTA is good but not good enough for fully automatic use. The self-assessment procedure attempts to automatically identify utterances that require revision by a human expert. The self-assessment procedure gives promising results for datasets for the ASTA and STAP methods. Significant improvements are still needed for the TARSP method, but there is still potential for such improvements, and we sketch some directions to achieve such improvements.

Keywords: language acquisition, developmental language disorders, aphasia, parsing, grammar correction, self assessment, Dutch

1. Introduction

This paper introduces SASTA self-assessment, i.e., a self-assessment procedure for the SASTA application for semiautomatic analysis of spontaneous language transcripts for Dutch. We introduce SASTA (Section 2) and the methods that it supports. These methods use language measures to assess the language development of young children and to assess the language skills of patients with aphasia (Section 2.1). We illustrate this with typical example utterances in Section 2.2. The performance of SASTA (reported in Section 2.3) is good but not good enough for fully automatic use. We introduce a self-assessment procedure that attempts to automatically identify utterances that require revision by a human expert. We set an initial target for the F1 score that should be achieved for a sample after self-assessment and revision of the selected utterance by a human expert (Section 3) and observe that this target is in principle reachable for all samples from all datasets that we use in this research and for all methods. We describe the self-assessment procedure in more detail in Section 4, identifying two classes of criteria that can be used for this purpose. The results of the self-assessment procedure are reported in Section 5, both for the training data and for the test data. We discuss some relevant related work in Section 6. Section 7 summarises our findings and suggests concrete work and some directions that may lead to improvements of the self-assessment procedure.

2. SASTA

SASTA[1] (Odijk, 2020; Odijk et al., 2025) is an application for semiautomatic analysis of spontaneous language transcripts for Dutch. It supports established methods for the analysis of such transcripts, in particular TARSP (aimed at children 1-4 years of age) (Schlichting, 2005, 2017), STAP (aimed at children 4-6 years of age) (van Ierland et al., 2008; Verbeek et al., 2007) and ASTA (aimed at patients with aphasia) (Boxum et al., 2013).

SASTA automates the analysis of samples in accordance with these methods, with a focus on grammatical analysis, and a partial analysis of grammatical errors.

The application is continuously being improved but it is already used to support research into language acquisition, developmental language disorders, and aphasia at various universities, and as a tool to support diagnosis and treatment in several clinics.

2.1. Language Measures

The methods that SASTA supports provide recipes for analysing the transcript of a language sample. A sample consists of 35-50 utterances, with the exact size depending on the method. The methods define *language measures*, i.e counts of how often specific lexical and grammatical phenomena occur in the sample:

- # specific function words and function and content word classes:

[1] SASTA is the acronym for the Dutch version of Semi-automatic Analysis of Spontaneous Language.

Proceedings of the 8th Workshop on Clinical Natural Language Processing (Clinical NLP) @ LREC 2026, pages 103–112
16 May 2026. ©ELRA Language Resources Association (ELRA), 2026

  - # nouns, # auxiliary verbs, ...
- # specific morphological properties:
  - verbal inflection, noun plurals, diminutives, compounds, ...
- # specific syntactic constructions
  - various clause structures, NP structures, PP structures, ...
- (for some) # specific grammar error classes
  - subject verb agreement, determiner-noun agreement, overgeneralised verb forms, ...

The methods assign a code for each occurrence of these phenomena. The counts and some aggregations on the counts are summarised in a method-specific *language profile* form. The results are compared to normative results, which (often in combination with results from other tests) is used to make an assessment of the language skills and to set up a treatment plan. The importance of language sample analysis in clinical practice is emphasised by Klatte et al. (2022). Until recently, analysis of such samples had to be done fully manually. SASTA automates this analysis.

2.2. Example Utterances

SASTA takes as input a language sample, which is a transcript of a recording of a conversation between the patient or child, and an interviewer. It consists of a sequence of utterances, together with metadata on the transcript and with metadata on individual utterances (at least the speaker). These utterances contain a lot of deviant language because the speakers are very young children who are still acquiring language, some of which may have a language disorder, or patients with aphasia. The transcribers usually transcribe faithfully what the speakers say, which results in utterances with many words that do not exist in Dutch or must be interpreted in an unusual way. Some examples of such utterances are in (1-3). For each example, we provide a gloss in which we try to reflect any deviant aspects of the utterance, the utterance as corrected by SASTA and its gloss, and a translation:

(1) TARSP examples

a. moe peelkaal toe
mus pay-all to

moet naar speelzaal toe
must to play-hall to

'(I) must go to the playing hall'

b. nu ga ik dees [: deze] pobreren
now go I thes [: this.one] ty
[: proberen].
[: try]

Nu ga ik deze proberen.
now go I this.one try

'Now I will try this one'

(2) STAP examples

a. die si o(p) wate(r).
that zi o(n) wate(r)

die zit op water
that sits on water

'That one sits on the water'

b. weet je waar ik woont?
know you where I lives?

weet je waar ik woon?
know you here I live?

'Do you know where I live?'

(3) ASTA examples

a. Op euh eventien zeventien okt ja
On eh eventeen seventeen Oct yeah
oktober is onze ja
October is our yeah

Op zeventien oktober is onze
On seventeen October is our

'On October 17th is our'

b. dan heeft het allemaal gebeurd
then gets it all happened

toen is het allemaal gebeurd
then is it all happened

'It all happened then'

The examples illustrate various kinds of deviancy, e.g., omitted sounds and syllables (*peelkaal* instead of ***s**peel**lo**kaal*), and informal spoken language variants (*moe* instead of *moet*) in (1a), wrong pronunciations (*pob**r**eren* instead of *p**r**oberen*) in (1b), agreement errors (*ik woon**t*** instead of *ik woon*) in (2b), wrong use of auxiliaries (*heeft* instead of *is* in (3b)), and filled pauses (*euh*), hesitations (*ja*) and false starts (*eventien*, *okt*) in an incomplete utterance (3a).

Transcribers can use inline CHAT annotations (MacWhinney, 2000) to clarify the transcript (e.g., *pobreren [: proberen]*, and *o(p)*), but these are used inconsistently (e.g. *o(p)* but not *si [: zit]* for *si* in (2a)) even by individual transcribers.

Table 1 provides the codes that have to be assigned to these utterances.

SASTA analyses all utterances by the target speaker in the sample. For each target speaker utterance, it attempts to detect any deviancy. If it detects any deviancy, it generates alternatives,

Method	Example	Codes
TARSP	moe peelkaal toe	BX, SamZn
TARSP	nu ga ik dees [: deze] pobreren [: proberen].	Avn,Hww i,Inv,OndWBVC,ik
STAP	die si o(p) wate(r).	BB p,N,PV,zelfst.vnw.
STAP	weet je waar ik woont?	OS,PV,PV
ASTA	Op euh eventien zeventien okt ja oktober is onze ja	MLUX,MLUX,MLUX,N,PV,SSZX, SSZX, lemma=oktober
ASTA	dan heeft het allemaal gebeurd	LEX,PV,lemma=gebeuren

Table 1: Example utterances with method and applicable codes

For example, in the utterance *moe peelkaal toe* it detects the following deviant or potentially deviant aspects:

- the word *moe* is an existing Dutch word (meaning *tired*) but it is also an informal pronunciation of the word *moet* (with meaning *must*). It therefore generates *moet* as an alternative to *moe*.
- the word *peelkaal* is not an existing Dutch word. SASTA generates an alternative using a dedicated child-language specific spelling corrector which has been developed as a component of SASTA. This generates the word *speelzaal* as an alternative.
- the word *toe* is an existing Dutch word. It is highly ambiguous: it can be an interjection (meaning 'please'), in certain regions an adjective (meaning *closed*), and it can be a postposition (meaning 'to'). The postposition *toe* never occurs on its own in the standard Dutch language (it usually occurs in the combination *naar ... toe*), but young children often use it on its own. SASTA adds the preposition *naar* before *peelkaal* as an alternative.

This results in 8 variants: all combinations of *moe* or *moet*, *peelkaal* or *speelzaal*, and presence or absence of *naar*. Each correction that is applied is recorded in correction metadata. SASTA parses all of these 8 variants (using the Alpino parser (Bouma et al., 2001; van Noord, 2006)). Then it evaluates the 8 parses using *parse evaluation criteria* and selects the utterance with the best parse.

There are currently around 30 parse evaluation criteria. Some are very generic, others are highly specific for particular grammatical configurations. An example of a parse evaluation criterion is the following: if the parser cannot analyze the utterance as a single utterance, it attempts to analyze it as a sequence of utterances. Each utterance in such a sequence gets the grammatical relation *dp* (discourse part). Since the string offered is a single utterance, presence of the grammatical relation *dp* in the parse tree is an indication that the parser could not analyze this utterance correctly. Such utterances will be less preferred when selecting the utterance with the best parse. We will discuss some more parse evaluation criteria in section 4.

The words in the best parse are then replaced again with the original words, and inserted words are deleted, by a procedure called *backplacement*. The result is that the utterance *moe peelkaal toe* has the parse tree of *moet naar speelzaal toe*, so *moe* is analysed as a finite modal verb, and *peelkaal toe* as an adverbial modifier.

The language measures have been implemented as queries on parse trees: these queries are applied to the original utterance with the selected best parse. The results of these queries are used to generate an annotated transcript and a digital version of the method-specific language profile form. For example (1a), it generates the codes *BX* (adverbial modifier and one other constituent) for the whole utterance, and *SamZn* (compound noun) for *peelkaal*,[2] exactly as required.

SASTA is, of course, not perfect. That is why SASTA has the term *semiautomatic* in its name. The user can edit the annotated transcript and resubmit it to SASTA, after which an updated language profile form is generated.

The SASTA application is publicly available[3], the source code is open software[4] and available on GitHub.[5]

2.3. Performance of SASTA

SASTA is being developed on the basis of a range of datasets. Some of these datasets are used for training, i.e. they are inspected to see what new alternatives must be generated, whether queries

[2]The backplacement procedure contains a specific procedure to determine whether the actually pronounced word (*peelkaal*) must be considered a compound if its correction (*speelzaal*) is one.

[3]https://sasta.hum.uu.nl/.

[4]BSD 3-Clause License, https://opensource.org/license/bsd-3-clause.

[5]https://github.com/CentreForDigitalHumanities/sasta, https://github.com/UUDigitalHumanitieslab/sastadev and https://github.com/CentreForDigitalHumanities/sasta-scripts.

that implement the language measures have to be adapted, etc. Other data sets are only used to test the system.

The performance of SASTA is measured in terms of recall (how many of the codes in the reference are actually generated by SASTA), precision (how many of the codes generated by SASTA occur in the reference), and by F1 score (the harmonic mean of recall and precision). Here we will discuss F1 scores only.

The performance of SASTA in terms of F1 scores is given in Table 2.

One can see that the performance is good with the F1 score on average between 88.5% and 94.4%. However, individual samples sometimes have a much lower score, with a minimum of 77.6%.

This level of performance still requires manual correction of the sample analysis to achieve the desired level of performance. If the whole sample has to be checked manually, the time gain by using SASTA is drastically reduced.[6] That is where SASTA self-assessment comes into play.

3. Self Assessment

In order to avoid that the whole sample has to be reviewed manually to check whether SASTA assigned the right codes, it would be ideal if SASTA could identify the utterances for which it thinks or suspects that its analysis is wrong, i.e., we need a form of SASTA Self-Assessment. That is the main topic of this paper.

The idea is that SASTA identifies a limited number of utterances that have to be manually checked. We assume that manually checking these utterances will lead to a 100% F1 score for these utterances.

Of course, we attempt to achieve as high an F1 score as possible, with as few revisions as possible, but as an initial (somewhat arbitrary) target, we aim to achieve a 95% F1 score for each sample by revising maximally 15 utterances per sample. 15 utterances is less than half of the utterances for an ASTA sample, 37.5% of the utterances for a TARSP sample and 30% of the utterances for a STAP sample.

Since we have expert-annotated reference data containing language measure codes for conversation transcripts, we can determine what this approach could achieve in the best possible case. This has been represented in Table 3. Note that this reference will change when SASTA is modified. The table represents the reference as determined on 2025-10-20.

One can see that all samples in all datasets reach an F1 score of at least 95% by revising at most 14 utterances. The average number of utterances that have to be modified over all samples in all datasets is 4.2 utterances per sample. Of course, this can only be achieved if the right utterances are selected for correction. For this selection, one needs assessment criteria.

4. Assessment Criteria

We identified two types of criteria that can be used for self-assessment: parse assessment criteria, and code assessment criteria. We will discuss each type in a separate subsection. In Section 4.3 we describe the strategy for testing these criteria against the training data and extending them.

4.1. Parse assessment criteria

If the parse of an utterance is wrong, it is very likely that the application of the queries that implement the language measures yields wrong results. For this reason, we assessed the parse evaluation criteria for their potential as a criterion for self-assessment. We identified 15 such criteria as having potential, and 17 as having no or low potential. These criteria determine the number of times that a specific configuration occurs in the parse tree. Some criteria are negative (so the lower the number of occurrences, the better), others are positive (so the higher the number of occurrences, the better).

We list some of the more generic assessment criteria:

dp # *dp* relations that the parse tree contains

unknown words # 'unknown' words that the parse tree contains[7]

extra-grammatical # content words that have no grammatical connection to the other words in the parse tree

Note that some parse evaluation criteria have to be adapted to function as a self-assessment criterion. In example (1a) discussed earlier, the utterance has the parse tree of *moet naar speelzaal toe* but it contains the words *moe peelkaal toe*, which includes the 'unknown' word *peelkaal*. The system only considers a word such as *peelkaal* as 'unknown' if it was not replaced with a known word

[6] SASTA is still useful even if the whole sample has to be checked, because SASTA increases recall (human annotators often overlook items that have to be coded) and because reviewing assigned codes is faster than assigning codes from scratch.

[7] We operationalised the notion of 'unknown word' by using some external lexicons, taking into account some very productive morphological processes.

Dataset	Use	Method	Avg F1	Min F1	Max F1
auristrain	train	tarsp	92.7	85.5	96.9
vkltarsp	train	tarsp	93.5	87.4	97.8
vklstap	train	stap	93.8	90.9	95.8
vklasta	train	asta	88.5	77.6	94.7
auristest	test	tarsp	92.6	83.5	97.3
elsdejong	test	stap	92.2	87.4	95.3
vklstap2	test	stap	93.3	90.2	96.2
vklasta2	test	asta	89.4	83.6	94.0
wrijpma	test	asta	94.9	90.6	98.2

Table 2: SASTA performance in terms of F1 score (measured 2025-10-20)

Use	Dataset	Method	#	$\geq$ target	Or. Avg F1	Avg F1	Min F1	Max F1	Avg #Utt	Min #Utt	Max # Utt
train	auristrain	tarsp	28	28	92.5	95.8	95.0	97.6	4.5	1	7
train	vkltarsp	tarsp	11	11	93.4	96.0	95.0	98.5	2.2	1	6
train	vklstap	stap	9	9	93.9	95.7	95.0	96.5	2.3	1	6
train	vklasta	asta	10	10	86.9	95.4	95.0	95.9	6.6	1	14
test	auristest	tarsp	8	8	91.9	96.0	95.4	98.4	3.0	1	7
test	vklstap2	stap	13	13	93.4	95.5	95.0	96.7	3.2	1	7
test	elsdejong	stap	10	10	92.2	95.4	95.0	96.1	4.1	1	10
test	vklasta2	asta	6	6	88.3	95.4	95.0	95.7	5.8	1	9
test	wrijpma	asta	25	25	94.8	96.2	95.2	98.6	1.4	1	4

Table 3: SASTA Self Assessment Reference (measured 2025-10-20), The table specifies for each dataset its use (for training or for testing), the method, the number of samples (in the column with header **#**), the number of samples for which the F1 target is reached, the average F1 score for each dataset (header **Or. Avg F1**), the average, minimal and maximal F1 score after correcting the relevant utterances, and the average, minimal and maximal number of utterances that must be revised.

by the correction mechanism: it can determine this by inspecting the correction metadata.

Currently, we have identified around 30 assessment criteria of this type. Table 4 provides some examples with a brief explanation. This table contains only verbal descriptions, but each criterion has been implemented as a query on the parse tree and/or the codes generated. Note that SASTA, using these criteria, can determine that the parse tree is very likely incorrect, but it cannot determine what is the correct one. Furthermore, many configurations that the criteria describe can be correct in special cases: what is important is that these criteria identify as many utterances with truly incorrect parses and identify as little as possible utterances with correct parses. Two different criteria can have overlapping results: in a next version of the system we will try to reduce any superfluous redundancy.

4.2. Code assessment criteria

A second type of criterion for self-assessment involves the queries that are used to implement the language measures. For example, under the TARSP-method, if an utterance contains a verb and at least one other real word (i.e., not interpunction, filled pause or interjection), then a code from a certain subset must have been assigned. If no such code is present, the analysis is very likely wrong. The cause for this can be a wrong parse, but it could also be an error or omission in the query. If the cause is in the query, we of course try to improve the query, but that is not always possible: criteria that identify utterances in which this occurs can be used as self-assessment criteria. We currently have three such criteria.

4.3. Developing criteria

We applied an initial set of assessment criteria to the various training datasets and analysed the mismatches, in particular the utterances that were not identified by the self-assessment criteria but that should have been. The analysis of these mismatches revealed that each case can be classified in one of three types:

1. the manual annotation in the reference contains an error. This actually happened quite often. Some of these errors were real errors in the reference data, but most were caused because for certain types of constructions specific conventions for annotating them were fixed during the development of SASTA. The

Short Description	Explanation
Bad categories	# categories that occur mostly if the utterance was parsed incorrectly
Basic replacement not applied	For a list of frequent incorrect or informal pronunciations replacements are generated as an alternative. But if such a word does not get replaced, the parse will very likely be wrong
Non-neuter determiner with neuter noun	This concerns an agreement error that can sometimes be corrected by SASTA, but if it is not corrected, the parse is very likely wrong
Infinitive probably wrongly parsed as subject	Infinitives as a subject are very rare in young children's language
Multiple main clauses	The presence of multiple main clauses makes it very likely that the utterance has been parsed incorrectly
Word unknown to the parser	Alpino attempts to guess the properties of words that it does not know. It does this often successfully, but not always (e.g. for the word *peelkaal* Alpino guessed that it is an adjective, which is understandable since it ends in *-aal*, a very common derivational suffix to form adjectives, but in utterance (1a) this is not correct.
Likely wrong indirect object	In certain configurations indirect objects are unlikely
Mismatching conjunct	The Alpino parser is (like all parsers) bad at analysing conjuncts correctly; if the conjuncts have a mismatch in category, it is very likely that the parse is wrong
Likely wrong analysis as locative and nominal predicate	The combination of a locative and a nominal predicate usually means that the utterance has been parsed incorrectly
Direct object with a (preferably) intransitive verb	If a direct object co-occurs with a verb that is (preferably) intransitive, it is highly likely that the parse is wrong
Wrong part of speech	Certain words are ambiguous (e.g., as noun or as verb), but one of these readings is very unlikely to occur in the samples. For example, the word *kijk* 'look' can be a verb or a noun, but it is very unlikely that *kijk* occurs as a noun in young children's language samples
Semantic incompatibility	An argument of a verb is semanticallly incompatible with the semantic selection properties of this verb for that argument.
Ellipsis applied incorrectly	Alpino is (like all parsers) not very good at analysing cases of ellipsis. This criterion selects the most likely wrong parses of ellipted words and phrases

Table 4: Example assessment criteria with a brief explanation

original reference data were made without these conventions in place, so were inconsistent with regard to them. We discuss such inconsistencies with the original data providers, and adapt the reference in accordance with the annotation convention agreed upon.[8]

2. the criteria did not cover them: in these cases we improved the criteria or added new criteria

3. the utterance was correctly not selected but the SASTA queries contain an error or omission: in these cases we try to improve these queries, and where we do not succeed we try to add new criteria.

[8]This concerns, for example, decisions on certain cases that could be analysed as a complement or as a modifier; which sequences of two adverbs form one constituent, and which sequences form two constituents, and similar cases.

5. Self Assessment Results

The results of self-assessment for the training data as measured on 2025-10-20 have been summarised in Table 5.

SASTA produces an F1 score $\geq$ 95% without SASTA self-assessment for some samples. These are included in the results, but they are few, as is clear from Table 6.

We observe that the results for datasets *vklstap* and *vklasta* with 8 out of 9 and 10 out of 10 samples reaching the target are good. The average number of utterances per sample that have to be revised (10.3 and 13.4) is much higher than the ideal case (2.2 and 6.6), but we did not optimise for this yet. The results for the TARSP datasets *auristrain* and *vkltarsp* are more mixed, with 20 out of 28 and 9 out of 11 samples reaching the target set. The criteria used are clearly not able to select all the right utterances for revision yet. These require

dataset	method	#	OK	Orig. Avg f1	Avg F1	Min F1	Max F1	Avg #Utt	Min #Utt	Max #Utt	Min f1 Δ	Max f1 Δ	Avg f1 Δ
auristrain	tarsp	28	20	89.9	93.9	88.2	99.7	6.0	2	12	0.0	6.9	3.2
vkltarsp	tarsp	11	9	93.4	96.1	91.4	98.8	6.1	2	15	0.5	8.9	2.8
vklstap	stap	9	8	93.9	96.3	94.3	97.6	10.3	6	15	0.8	4.3	2.5
vklasta	asta	10	10	86.9	96.1	89.0	98.8	13.4	8	15	2.0	18.7	10.5

Table 5: Self-assessment results for the training data (2025-10-20).The table provides columns for the dataset, the method, the number of samples (#), the number of samples for which the target of 95% F1 score is reached (OK), the original average F1 score, the average, minimal and maximal F1 score after application of self-assessment, the average, minimal and maximal number of utterances that have to be revised, and the minimal, maximal and average increase in F1 score after self-assessment.

use	dataset	method	#	# $\geq$ 95%
train	auristrain	TARSP	28	5
train	vkltarsp	TARSP	11	3
train	vklstap	STAP	9	4
train	vklasta	ASTA	10	0
test	auristest	TARSP	8	2
test	vklstap2	STAP	13	2
test	vklasta2	ASTA	6	0
test	elsdejong	STAP	11	1
test	wrijpma	ASTA	25	11

Table 6: Number of samples per dataset and the number of samples with an F1 score $\geq$ 95% by SASTA without self-assessment per dataset (2025-10-20)

more work.

The results of self-assessment for the test data as measured on 2025-10-20 have been summarised in Table 7. This table has the same structure as Table 5.

The results for the STAP data *vklstap2*, with 11 out of 13 is promising, but the results for the dataset *elsdejong*), with 6 out of 11 samples reaching the 95% F1 target still requires improvement. The ASTA dataset *wrijpma* shows a perfect score with 25 out of 25 samples reaching the target, but the original average F1 score for samples in this dataset was very high to begin with (94.8%), as is the number of samples that start with an F1 score $\geq$ 95% (11 out of 25). It is very likely that SASTA self-assessment can reach a higher target for this dataset. In the *vklasta2* dataset 3 out of 6 samples reach the target, so an improvement is clearly desirable here.

The *auristest* data, with 5 out of 8 samples reaching the target, also still require improvement. The average number of utterances to be revised is quite low (5.1), which suggests that the current selection criteria are not yet able to select the right utterances for revision.

6. Related Work

The focus of this paper is on SASTA self-assessment. For this reason we will not discuss work that relates to automating methods to assess the language skills of young children and patients with aphasia. For this , we refer to (Odijk, 2020).

The code assessment criteria are, to our knowledge, unique, though they are instantiations of a general technique, viz. formulating a less specific query to determine whether a code from a specific subset must be present, raise an error if this is not the case, and suggest the members of the subset as candidate codes.

The other method used to do self-assessment is by attempting to automatically detect parsing errors. This makes a lot of research dealing with this topic or with automatically detecting annotation errors in treebanks relevant for this paper.

Ambati et al. (2010) created a tool to detect parsing and annotation errors in Hindi treebanks The tool contains rules implementing checks on annotation conventions and statistical checks on local tree configurations. For a different project, one of the authors made a similar tool for Dutch (Odijk et al., 2018),[9] with the statistics obtained from the manually verified Spoken Dutch Corpus (Oostdijk et al., 2002) and Lassy treebanks (van Noord et al., 2013).

Many researchers, inter alia (van Halteren, 2000; Boyd et al., 2008; Dickinson and Meurers, 2003, 2005; Dickinson, 2011; Dickinson and Smith, 2011; Krasnowska and Przepiórkowski, 2013) use variation in annotations as a basis for automatically

[9] https://github.com/JanOdijk/ace

dataset	meth	#	OK	Orig. Avg F1	Avg F1	Min F1	Max F1	Avg #Utt	Min #Utt	Max #Utt	Min F1 Δ	Max F1 Δ	Avg F1 Δ
auristest	tarsp	8	5	91.9	94.9	91.6	97.8	5.1	2	8	0.6	8.1	2.9
vklstap2	stap	13	11	93.4	96.4	93.3	97.8	12.5	6	15	1.0	7.4	3.2
elsdejong	stap	11	6	91.8	95.8	93.7	98.6	13.5	9	15	0.9	7.6	4.5
vklasta2	asta	6	3	88.4	95.1	91.6	98.2	14.7	13	15	3.1	13.6	8.0
wrijpma	asta	25	25	94.8	97.9	95.9	100.0	12.6	6	15	1.1	6.3	3.2

Table 7: Self-assessment results for the test data. (2025-10-20)

detecting likely errors.

The work on error mining in treebanks to identify elements that are the cause for wrongly parsed sentences is clearly also relevant (van Noord, 2004; Sagot and de la Clergerie, 2006; de Kok et al., 2009; Suzuki et al., 2016). This research is usually aimed at improving a treebank and/or a parser. These works attempt to determine the 'suspicion' of elements (e.g. words) or combinations of elements (e.g., n-grams) based on their frequencies in wrongly parsed sentences in comparison to their frequencies in correctly parsed sentences. Some of these methods work iteratively, e.g. running the same procedure on a continuously improving treebank until the suspicions stabilize.

Volokh and Neumann (2011) parse a manually verified treebank with two parsers trained on the treebank and use the differences between these results as likely sources of annotation errors, and they even try to correct these automatically. (Agrawal et al., 2013) also use a parser to identify errors in a manually verified treebank, and they emphasize that using a parser will identify inconsistencies in human annotation.

These methods are usually applied to find errors in large treebanks that contain enough data to work with frequencies and derivatives of frequencies, or to train parsers on. In our research, the total amount of data (training and testing) is relatively small (currently 2300 utterances for training, 2600 for testing), so that it is not obvious that these methods can be usefully applied. (cf. (van Noord, 2004): 'this technique obtains very good results if it is applied to large sets of sentences'.

For this reason we have so far used a rule-based approach to identify likely parsing errors. In a future version, we will also try to use statistics obtained from the manually verified Van Kampen CHILDES treebank (Odijk et al., 2018) to detect likely parse errors, at least for the data involving young children. This treebank contains 109k utterances, 47k of which are utterances made by young children.

7. Conclusions and Future Work

We have introduced a mechanism of self-assessment for SASTA. The current version of the self-assessment procedure yields promising results for ASTA and STAP datasets, but clearly still requires improvement for the TARSP datasets.

We introduced two types of assessment criteria: parse assessment criteria, which attempt to identify wrong parses, and code assessment criteria, which attempt to identify incorrectly missing codes.

The work on SASTA self-assessment is still ongoing, and there is still potential for further improvement. We finished the analysis of the utterances that were incorrectly not selected but we did not implement all new criteria or SASTA improvements yet. We also aim to investigate some other criteria, in particular whether we can somehow use confidence values for language measures in new criteria.

We will also try to make use of the statistics obtained from the manually verified Van Kampen CHILDES treebank (Odijk et al., 2018) to detect likely parse errors, at least for the data involving young children.

We also want to analyse the utterances that have incorrectly been selected, to try to reduce their number as much as possible. Though some criteria are already specific for a particular method, this might be necessary for other criteria that are currently applied for all methods as well.

We have inventoried a number of criteria that can be used to sort the selected utterances, so that the utterances that increase the F1 score most come earlier. Examples of such criteria are the utterance length relative to the number of codes assigned and the average F1 increase in the training data. However, we have not experimented with these criteria yet.

In short, there is still a lot of potential for improvement, and we hope to report on better results of new versions of the self-assessment procedure in the near future.

Acknowledgements

The research reported on in this paper was partially financed by the Utrecht University Human-Centered AI focus area. We thank the anonymous reviewers for their comments.

8. Bibliographical References

Bhasha Agrawal, Rahul Agarwal, Samar Husain, and Dipti M. Sharma. 2013. An automatic approach to treebank error detection using a dependency parser. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing. CICLing 2013. Lecture Notes in Computer Science, vol. 7816*, pages 294–303. Springer, Berlin, Heidelberg.

Bharat Ram Ambati, Mridul Gupta, Samar Husain, and Dipti Misra Sharma. 2010. A high recall error identification tool for Hindi treebank validation. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Gosse Bouma, Gertjan van Noord, and Robert Malouf. 2001. Alpino: Wide-coverage computational analysis of Dutch. *Language and Computers*, 37(1):45–59.

Elsbeth Boxum, Fennetta van der Scheer, and Mariëlle Zwaga. 2013. *Analyse voor Spontane Taal bij Afasie. Standaard in samenwerking met de VKL*. VKL. https://klinischelinguistiek.nl/uploads/201307asta4eversie.pdf.

Adriane Boyd, Markus Dickinson, and W. Detmar Meurers. 2008. On detecting errors in dependency treebanks. *Research on Language and Computation*, 6(2):113–137.

Daniël de Kok, Jianqiang Ma, and Gertjan van Noord. 2009. A generalized method for iterative error mining in parsing results. In *Proceedings of the 2009 Workshop on Grammar Engineering Across Frameworks (GEAF 2009)*, pages 71–79, Suntec, Singapore. Association for Computational Linguistics.

Markus Dickinson. 2011. Detecting ad hoc rules for treebank development. *Linguistic Issues in Language Technology (LiLT)*, 4(3).

Markus Dickinson and W. Detmar Meurers. 2003. Detecting inconsistencies in treebanks. In *Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT 2003)*, pages 45–56, Växjö, Sweden. Växjö University Press.

Markus Dickinson and W. Detmar Meurers. 2005. Detecting errors in discontinuous structural annotation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 322–329, Ann Arbor, Michigan. Association for Computational Linguistics.

Markus Dickinson and Amber Smith. 2011. Detecting dependency parse errors with minimal resources. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 241–252, Dublin, Ireland. Association for Computational Linguistics.

Inge Klatte, Vera Van Heugten, Rob Zwitserlood, and Ellen Gerrits. 2022. Language sample analysis in clinical practice: Speech-language pathologists' barriers, facilitators, and needs. *Language, Speech, and Hearing Services in Schools*, 53:1–16.

Katarzyna Krasnowska and Adam Przepiórkowski. 2013. Detecting syntactic errors in dependency treebanks for morphosyntactically rich languages. In *Language Processing and Intelligent Information Systems – 20th International Conference, IIS 2013 (selected papers)*, pages 69–79. Springer.

Brian MacWhinney. 2000. *The CHILDES Project: Tools for Analyzing Talk*, 3 edition. Lawrence Erlbaum Associates, Mahwah, NJ.

Jan Odijk. 2020. Towards semi-automatic analysis of spontaneous language for Dutch. In *Selected papers from the CLARIN Annual Conference 2020*, volume 180 of *Linköping Electronic Conference Proceedings*, pages 165–175. Linköping University Press, Linköping, Sweden.

Jan Odijk, Alexis Dimitriadis, Martijn Van der Klis, Marjo Van Koppen, Meie Otten, and Remco Van der Veen. 2018. The AnnCor CHILDES Treebank. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Jan Odijk, Margo Zwitserlood-Nijenhuis, Rob Zwitserlood, Jelte van Boheemen, Anouk Scheffer, and Frank Wijnen. 2025. Semi-automatische spontane taalanalyse (SASTA): Op weg naar een efficiënte methode voor de bepaling van het taalprofiel van kinderen met TOS. *Stem-, Spraak- en Taalpathologie*, 30:79–104.

N. Oostdijk, W. Goedertier, F. Van Eynde, L. Boves, J.P. Martens, M. Moortgat, and H. Baayen. 2002. Experiences from the Spoken Dutch Corpus project. In M. González Rodriguez and C. Paz Suárez Araujo, editors, *Proceedings of the third International Conference on Language Resources and Evaluation (LREC-2002)*, pages 340–347. ELRA, Las Palmas.

Benoît Sagot and Éric de la Clergerie. 2006. Error mining in parsing results. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting*

of the Association for Computational Linguistics, pages 329–336, Sydney, Australia. Association for Computational Linguistics.

Liesbeth Schlichting. 2005. *TARSP: Taal Analyse Remediëring en Screening Procedure. Taalontwikkelingsschaal van Nederlandse kinderen van 1-4 jaar*, 7th edition. Pearson, Amsterdam.

Liesbeth Schlichting. 2017. *TARSP: Taalontwikkelingsschaal van Nederlandse kinderen van 1-4 jaar met aanvullende structuren tot 6 jaar*, 8th edition. Pearson, Amsterdam.

Kanta Suzuki, Yoshihide Kato, and Shigeki Matsubara. 2016. Correcting errors in a treebank based on tree mining. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1540–1545, Portorož, Slovenia. European Language Resources Association (ELRA).

Hans van Halteren. 2000. The detection of inconsistency in manually tagged text. In *Proceedings of the COLING-2000 Workshop on Linguistically Interpreted Corpora*, pages 48–55, Centre Universitaire, Luxembourg. International Committee on Computational Linguistics.

Margreet van Ierland, Jeannette Verbeek, and Leen van den Dungen. 2008. *Spontane Taal Analyse Protocol. Handleiding van het STAP-instrument*. UvA, Amsterdam.

Gertjan van Noord. 2004. Error mining for wide-coverage grammar engineering. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 446–453, Barcelona, Spain.

Gertjan van Noord. 2006. At last parsing is now operational. In *Actes de la 13ème conférence sur le Traitement Automatique des Langues Naturelles. Conférences invitées*, pages 20–42, Leuven, Belgique. ATALA.

Gertjan van Noord, Gosse Bouma, Frank Van Eynde, Daniël de Kok, Jelmer van der Linde, Ineke Schuurman, Erik Tjong Kim Sang, and Vincent Vandeghinste. 2013. Large scale syntactic annotation of written Dutch: Lassy. In Peter Spyns and Jan Odijk, editors, *Essential Speech and Language Technology for Dutch*, Theory and Applications of Natural Language Processing, pages 147–164. Springer Berlin Heidelberg.

Jeannette Verbeek, Leen van den Dungen, and Anne Baker. 2007. *Spontane Taal Analyse Protocol. Verantwoording van het STAP-instrument, ontwikkeld door Margreet van Ierland*. UvA.

Alexander Volokh and Günter Neumann. 2011. Automatic detection and correction of errors in dependency treebanks. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (short papers)*, pages 346–350, Portland, Oregon, USA. Association for Computational Linguistics.