

Measuring Oral Reading Fluency with a Dual-Pass ASR Architecture: Design, Instrument Characterization, and Synthetic Validation of the FRS Fluency Meter

Jasper Vande Moortel, Orion-Solution CommV Version 0.1 — 10 July 2026

Abstract

Oral reading fluency (ORF) is among the best-established observable indicators of reading development, but scoring it by hand is labor-intensive, and the automated tools now used at scale in classrooms do not publish their measurement methods. This paper specifies an open measurement architecture for ORF in Dutch and English on the Free Reading Standard (FRS) scale, and reports two validation studies of its mechanics. The architecture runs two passes of a commercial speech engine over one recording, a scripted pronunciation assessment and an unscripted transcription, and derives all miscue classifications from published rules that require the two passes to corroborate each other before an error is attributed to the reader. Study 1 characterizes the engine itself (the same engine that underlies Microsoft Reading Progress) with planted-error audio: it is deterministic on identical input, but its native miscue labeling is unreliable in ways that matter for reading assessment; it never emitted a repetition label in any run, and substitutions produced cascading misattributions. Study 2 validates the full measurement layer against a 144-item synthetic battery with position-seeded planted miscues, and reports a complete iteration cycle: initial precision/recall per miscue type, root-cause diagnosis, revised corroboration rules, and re-analysis on identical data. Detection is strong for omissions (precision 1.00/0.94, English/Dutch) and improved substantially where the initial rules failed, with all residual failure modes diagnosed and published. Criterion validity against annotated child recordings is registered as follow-up work. All parameters, thresholds, and rules are published in the appendices.

1. Introduction

A teacher listening to a child read aloud is running the most widely validated informal reading assessment that exists. Rate, accuracy, and expression during oral reading predict broader reading competence robustly enough that curriculum-based measurement built a forty-year practice on a one-minute sample (Deno, 1985; Sabatini, Wang, & O'Reilly, 2019). The cost is teacher time. Listening to one child is cheap; listening to thirty children every few weeks is not,

and the scoring itself (which words were skipped, substituted, self-corrected) demands attention that a classroom rarely affords.

Automated scoring is the obvious response, and it has a research lineage going back to Project LISTEN's Reading Tutor (Mostow & Duong, 2009) and production deployments reaching millions of students, most prominently Microsoft Reading Progress (Microsoft, n.d.-c). But the production tools are closed. Reading Progress reports accuracy rates, words per minute, and miscue categories through the Microsoft Graph API, yet the method that produces those numbers from audio is not published, cannot be independently checked, and can change without notice. For a school deciding whether to trust a fluency number, or a researcher measuring growth against it, that opacity is a real cost.

This paper does for fluency measurement what the Free Reading Standard did for text difficulty (Vande Moortel, 2026d): it publishes the whole method. The FRS fluency meter takes a recording of a reader reading a known text aloud, in Dutch or English, and returns rate (words correct per minute, WCPM), accuracy, a typed miscue inventory that includes self-corrections and repetitions, prosody features, and a pronunciation profile, each with stated definitions and published parameters.

The contributions are:

1. **An engine-independent measurement specification** for oral reading fluency whose reader-side and text-side measures share one open scale (FRS 0 to 100), with every reported number defined by published rules (Section 3).
2. **A dual-pass architecture**: a scripted pronunciation-assessment pass and an unscripted transcription pass over the same recording, combined under corroboration rules that require both passes to agree before the system accuses a reader of an error (Sections 3.4 and 3.5).
3. **An instrument characterization** of the commercial engine underlying Microsoft Reading Progress, documenting its locale gating, its determinism, and systematic failures of its native miscue labeling (Study 1, Section 4).
4. **A synthetic validation methodology** using text-to-speech audio with position-seeded planted miscues, reported as a full iteration cycle: initial results, root-cause diagnosis, revised rules, re-analysis on identical data, and a mechanism-level analysis of what remains wrong (Study 2, Section 5).
5. **Every parameter published** (appendices), consistent with the openness doctrine of the FRS project.

One scope statement belongs in the first section rather than the fine print. The two studies reported here validate the measurement layer's *mechanics*: whether the architecture correctly detects and classifies known miscues in clean speech, and how the engine behaves as an instrument. They do not establish agreement with human scorers on real children's reading. That

study, against the CHOREC corpus of annotated Flemish children's oral reading (Cleuren, Duchateau, Ghesquière, & Van hamme, 2008), is registered as follow-up work; the validation harness used in Study 2 was built behind a corpus-adaptor interface so the identical analysis re-runs on real recordings when corpus access is granted (Section 8).

2. Background

2.1 The construct: rate, accuracy, and prosody

Oral reading fluency in the assessment literature is a three-part construct: reading with adequate speed, with accuracy, and with appropriate expression (National Reading Panel, 2000). Curriculum-based measurement operationalized the first two as words correct per minute, and WCPM norms by grade and season are among the most replicated artifacts in reading research (Hasbrouck & Tindal, 2017). Accuracy conventions (independent, instructional, and frustration levels) descend from the informal reading inventory tradition; their specific percentage thresholds vary across sources and are best understood as conventions rather than empirically derived constants.

The miscue taxonomy this paper implements (omissions, insertions, substitutions, mispronunciations, repetitions, and self-corrections) is the one classroom practice itself uses and reports; the data schema of Microsoft Reading Progress distinguishes exactly these categories (Microsoft, n.d.-c). The treatment of self-corrections is a fixture of professional scoring practice: a word misread and then spontaneously corrected is counted as read correctly (e.g., Hasbrouck & Tindal, 2017), because self-correction indicates monitoring, a positive signal rather than a deficit. Repetitions similarly do not count against accuracy in most conventions, but they carry diagnostic information about fluency. Any automated system that cannot distinguish a self-correction from a substitution, or a repetition from an insertion, is not measuring the construct the literature defines.

Prosody is the least standardized dimension. The NAEP oral reading studies rate expression on a four-level scale ranging from word-by-word reading to reading in larger meaningful phrase groups with expressive interpretation (National Center for Education Statistics [NCES], 1995; Daane, Campbell, Grigg, Goodman, & Oranje, 2005), and the Multidimensional Fluency Scale rates expression, phrasing, smoothness, and pace (see the many-facet Rasch examination in "Reliability of Ratings of Multidimensional Fluency Scale," 2022). Two methodological details from that literature matter for the present work. First, NAEP's 1992 data suggest that rate and prosody, more than accuracy, separate fluent from non-fluent fourth-grade readers (fluent readers at 126 to 162 words per minute against 65 to 89, while accuracy differed by only a few points) (NCES, 1995). Second, NAEP scored passages the student had already read silently; curriculum-based measurement scores cold, first-sight reading. The FRS meter scores cold reads, and comparisons to practiced-reading norms must carry that caveat.

2.2 Automated ORF scoring

Automated oral reading assessment predates the current speech-technology cycle. Project LISTEN's Reading Tutor scored children's oral reading for two decades of research; its prosody work modeled a child's pitch, intensity, and timing contours against fluent adult narrations of the same text, correlating with human fluency ratings at $r = .87$ on the representative sample ($n = 55$; a widely quoted $r \geq .93$ figure comes from a small, deliberately stratified held-out subset of that study) (Mostow & Duong, 2009). The FLORA system reported automated WCPM within three to four words of expert human scorers, with automated expressiveness ratings whose agreement with human raters matched or exceeded human-human agreement (Bolaños et al., 2013). The Ordinate/Versant lineage brought psychometric validation discipline, including explicit bias testing across speaker populations, though its published validation targeted adult literacy assessment (Balogh et al., 2012). Commercial systems for children (SoapBox Labs, Amira Learning) circulate strong agreement figures, but those specific numbers are vendor-sourced rather than independently peer-reviewed, and where independent examination exists, consistency with human raters varied by task type ("Evaluation of the Consistency of a Speech Verification System," 2026).

For Dutch, the directly relevant validations are recent. Molenaar and colleagues evaluated six ASR systems for scoring Dutch children's oral reading accuracy against human raters (71 children, 1,455 sentences); performance ranged from near-zero to substantial agreement (MCC .01 to .65, with the best configurations at .63 to .65), and the spread itself carried the paper's central finding: systems whose language models encoded known reading-error patterns scored far better than the generic-LM baseline (MCC .24) (Molenaar, Tejedor-Garcia, Strik, & Cucchiari, 2023). Harmsen and colleagues validated a fifteen-measure ASR-based ORF protocol against measures derived from human orthographic transcripts of 244 recordings, finding twelve of fifteen measures correlating strongly ($r \geq .76$), though only four cleared the $r \geq .9$ bar the authors themselves cite for high-stakes use (Harmsen, van Hout, Cucchiari, & Strik, 2025); a Cito-affiliated team published a parallel validation line covering 653 children (van der Velde et al., 2025). All of these validated against real recordings. In the research pass behind this work, no published validation of an ORF scorer against synthetic audio was found in any language searched; we state this as a search finding rather than a proven absence, and it is the gap that Section 5's methodology and Paper 7 of this series address directly.

2.3 ASR on children's speech

Speech recognition degrades on children's voices. The degradation is driven by developmental variability in vocal-tract geometry and articulation stability, and it is large: recent zero-shot evaluations of modern systems on child speech corpora report word error rates several multiples of adult benchmarks, with fine-tuning narrowing but not closing the gap (Fan, Shankar, & Alwan, 2024). Recording conditions can dominate the child-speech penalty itself; a Dutch evaluation reported word error rates of 5.5 percent on clean child recordings and over 70

percent on a noisier set with the same model family (Lathouwers, Gao, Cucchiarini, & Strik, 2026). Two design consequences follow for any fluency meter: audio quality must be gated before scoring, and instrument error must be decomposed from reader error before miscue counts are interpreted (Section 3.7).

2.4 Pronunciation assessment engines and Reading Progress

The engine used in this work is Azure AI Speech pronunciation assessment (Microsoft, n.d.-a). In scripted mode it force-aligns audio against a reference text and returns accuracy scores at word and phoneme level, fluency and completeness scores, and a word-level error type drawn from an enumeration that includes omission, insertion, and mispronunciation. Microsoft's Responsible AI documentation names Reading Progress as a deployment of this engine (Microsoft, n.d.-b), which makes its characterization doubly relevant: it is both this system's measurement instrument and the measurement core of the most widely deployed classroom fluency tool.

Microsoft's own research bibliography for Reading Progress cites the general fluency-pedagogy literature but no study validating the product's scoring accuracy (Microsoft, n.d.-d); an earlier paper in this series examined Reading Progress's classroom role as a growth instrument (Vande Moortel, 2026c).

Three properties of the engine, documented in Section 4, shape the architecture. Feature availability is locale-gated: Dutch (nl-NL) receives the core scores but no prosody assessment, no syllable detail, and no phoneme identities, all of which are restricted to English (en-US); Belgian Dutch (nl-BE) is not supported at all, so Flemish readers are assessed against Netherlands-Dutch pronunciation norms. Second, the engine exposes no self-correction or repetition concept; Reading Progress reports both, which means they are computed in an unpublished product layer above the engine, exactly the layer this paper publishes. Third, Microsoft's own documentation places the burden of validation on integrators and recommends loosening mispronunciation thresholds for children, while publishing no child-speech accuracy figures (Microsoft, n.d.-b).

3. The FRS fluency measurement architecture

3.1 Construct and reported measures

Table 1 defines the reported measures. Definitions are engine-independent: any engine that can produce a scripted word-level assessment and an unscripted transcription with word timings can implement the specification, and the engine in use is treated as a measurement instrument whose error is characterized separately.

Table 1. Reported measures per reading.

Measure	Definition	Dutch	English
WCPM	(reference words read correctly, incl. self-corrected) / reading time × 60	yes	yes
Accuracy	correct reference words / reference words	yes	yes
Miscues	counts by type: omission, insertion, substitution, mispronunciation, repetition, self-correction	yes	yes
Self-corrections	misread then corrected without prompting; reported separately, not counted as errors	yes	yes
Prosody features	pause placement relative to punctuation, articulation rate, pitch declination	yes	yes
Pronunciation profile	reference phonemes with systematically low engine scores	partial ¹	yes

¹ For Dutch the engine returns phoneme scores but not the identity of the phoneme actually produced; the Dutch profile therefore reports which target sounds scored poorly, not what was substituted for them.

Reading time runs from the onset of the first recognized word to the offset of the last. A reading is preceded by an audio quality gate (duration bounds, clipping ratio, near-silence detection); recordings that fail the gate are not scored, on the Section 2.3 evidence that recording conditions can dominate the error budget.

3.2 Dual-pass design

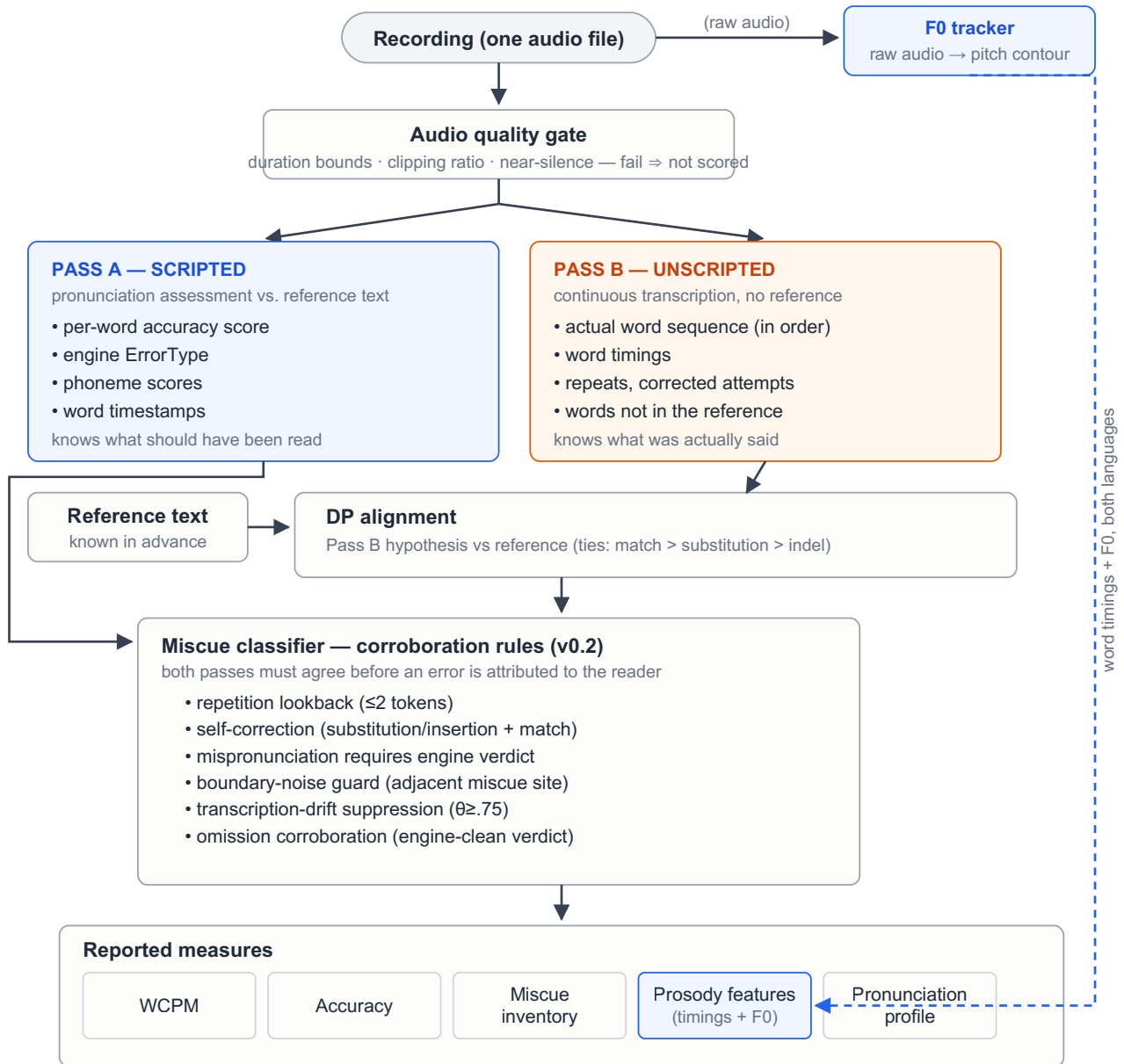
Each recording is processed twice by the engine (Figure 1):

- **Pass A (scripted):** pronunciation assessment against the reference text, returning per-word accuracy scores, engine error types, phoneme scores, and word timestamps. Pass A knows what *should* have been read.
- **Pass B (unscripted):** plain speech-to-text with word timings. Pass B reports what was *actually said*, in order, including material the scripted pass cannot represent: repeated words, corrected attempts, insertions of words not in the text.

The division of labor follows from what Study 1 shows each pass can and cannot see. Scripted force-alignment is anchored to the reference, so it is reliable about the reference words (was this word acceptably pronounced) and unreliable about sequence phenomena (it never emitted a repetition label in any of our runs, and substitutions produced cascading misattributions; Section 4). The unscripted transcript preserves sequence but introduces its own failure mode, recognizer normalization drift, in which correctly read words are transcribed as close variants (Section 5.3). Neither pass alone is a trustworthy miscue detector. The architecture therefore

treats agreement between the passes as the standard of evidence: each pass may veto the other's accusations.

Figure 1 — The dual-pass measurement architecture



Both passes run on every recording; the classifier accuses the reader of an error only when both passes agree (§3.2, §3.5). Prosody is computed outside the engine, from word timings and the F0 tracker, identically for Dutch and English (§3.6).

Both passes use continuous recognition. Single-shot recognition in the engine's SDK silently truncates multi-sentence audio at the first significant pause (in our testing, returning 2.5 seconds of a 15-second recording without any error indication), which would truncate exactly the readings that pause most, that is, the readings of the weakest readers.

3.3 Alignment

The reference text is tokenized with a punctuation map (which token precedes sentence-final or clause punctuation) and normalized token forms. The Pass B hypothesis sequence is aligned

to the reference by dynamic programming with unit costs for substitution, insertion, and deletion, and ties broken to prefer match over substitution over indel. The output is an operations sequence over reference positions, which the classifier then interprets.

3.4 Miscue classification

The classifier interprets alignment operations into the Table 1 taxonomy. The base rules (vo):

- an aligned match is **correct**;
- a hypothesis token re-matching the immediately preceding matched reference token (lookback window 2) is a **repetition**, not an insertion;
- a substitution or engine-flagged mispronunciation at reference position i , followed within two hypothesis tokens by a correct match of position i , is a **self-correction**; the word's final status is correct;
- remaining unmatched reference tokens are **omissions**; remaining unmatched hypothesis tokens are **insertions**;
- a matched word whose Pass A error type is mispronunciation, with an accuracy score below threshold θ_{mis} (default 60, the engine's own convention), is a **mispronunciation**.

Two rules were revised during Study 2 into the vo.2 corroboration set (Section 5.4); the version identifier of the rule set is embedded in every result.

3.5 Merge rules: corroboration before accusation

The two passes are merged under rules with a single organizing principle: **both passes must agree before the system accuses the reader of an error**. Figure 2 shows the full decision flow for one reference word. The vo.2 rule set:

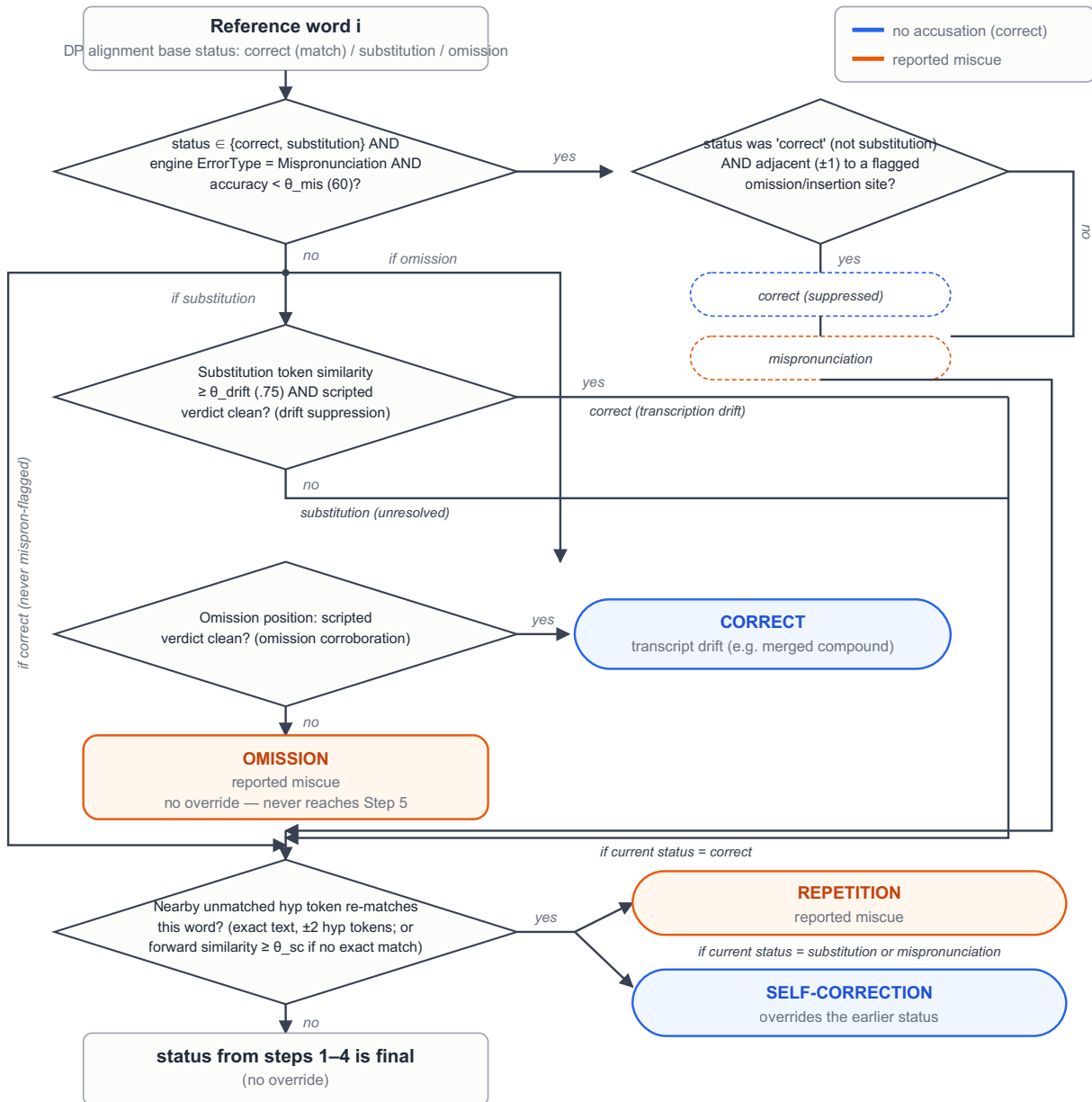
1. **Mispronunciation requires the engine's verdict.** A low accuracy score alone does not classify a word as mispronounced; the scripted pass must carry the explicit mispronunciation error type (Study 1 shows scores alone fluctuate on correctly read neighbors of true miscues). The verdict may *override* a textual substitution: a severe decoding error is transcribed by Pass B as a different real word (a recognizer can only emit words), and the scripted verdict is what distinguishes "read a different word" from "failed to decode this word."
2. **Boundary-noise guard.** An engine mispronunciation verdict on a word whose alignment status is correct is suppressed when the word is directly adjacent to an engine-flagged omission or insertion site, because the engine's forced alignment is demonstrably disrupted at miscue boundaries (Section 4). Suppressed verdicts are counted and marked per word rather than silently discarded.
3. **Transcription-drift suppression.** A textual substitution whose token is near-identical to the reference (similarity $\geq \theta_{\text{drift}}$, default 0.75) *and* whose scripted verdict is

clean is reclassified as correct: the divergence is recognizer normalization (spelling variants, inflectional drift), not reading behavior. Suppressions are counted (Section 5.4).

4. **Omission corroboration.** A textual omission whose scripted verdict is clean is likewise suppressed: Study 1 shows the engine detects true omissions reliably in both languages, so text-missing plus engine-clean indicates transcript drift (for example, two words merged into a compound by the recognizer).
5. **Self-correction via similarity.** An insertion whose token is similar ($\geq \theta_{sc}$, default 0.6) to a reference token matched within the next two hypothesis positions is a self-correction of that reference word.

The rules are listed here by conceptual role. The implementation's exact execution order differs in one respect that matters for reproduction: the sequence-level pass (repetition and self-correction) runs last and may override a word's earlier substitution or mispronunciation status. Appendix B gives the order exactly.

Figure 2 — Miscue classification: the rule cascade for one reference word



Rules shown in implementation execution order (Appendix B); the sequence-level pass runs last and may override earlier statuses. The boundary-noise guard applies only to correct-status words; a substitution-status word matching the mispronunciation trigger is reclassified directly (substitution-override), never guarded. D3 (omission corroboration) is the only true dead end — an omission never consumed a hypothesis token to re-match at Step 5.

Every threshold is a named, published parameter (Appendix A); the defaults were tuned on the Study 2 battery and are explicitly subject to recalibration on real child recordings.

3.6 Prosody layer

Because the engine's prosody assessment is unavailable for Dutch, prosody is computed outside the engine, identically for both languages, from word timings and the raw audio: pause placement relative to the punctuation map (pauses at punctuation versus mid-phrase), articulation rate excluding pauses, mean pause duration, and two pitch features (overall range and sentence-final declination) from a dependency-free autocorrelation Fo tracker with published parameters (Appendix A). The tracker's accuracy target is prosody-level trend detection, not

phonetic measurement; on synthetic tones its error is under 1.5 percent, and its range statistic is percentile-trimmed because isolated transient frames at consonant boundaries otherwise inflate raw ranges. This mirrors the design tradition of contour-based prosody scoring (Mostow & Duong, 2009) while remaining fully published.

3.7 Determinism, versioning, and the instrument stance

The measurement layer is deterministic: identical audio and identical engine responses produce identical scores, and the rule-set version is embedded in every result, consistent with the deterministic-serving and versioned-calibration doctrine of the FRS scale itself (Vande Moortel, 2026b). Engine nondeterminism is therefore confined to the engine itself, and Study 1 measures it directly (it is zero on identical audio). The broader stance is that the engine is a *measurement instrument*: its error contribution is characterized separately from reader behavior, and every downstream claim is bounded by the instrument characterization. This is also the honest response to the engine vendor's own position, which explicitly leaves validation to integrators (Microsoft, n.d.-b).

4. Study 1: Instrument characterization

4.1 Method

Two reference texts (45 words Dutch, 42 words English), each rendered to audio by a neural TTS voice in six variants: clean, one omission, one substitution, one insertion, one repetition, and one forced mispronunciation produced by an SSML phoneme override (a wrong vowel in one word). Each of the twelve clips was processed by both passes; the clean clips were additionally submitted five times each for repeat-consistency measurement. Forty-six engine calls in total; raw responses are published with the code.

4.2 Results

Locale gating, verified field-by-field. The Dutch responses contain no prosody score, no prosody feedback, and empty syllable and phoneme identity fields, while phoneme *scores* are populated; the English responses contain all of them. This confirms the documented gating (Microsoft, n.d.-a) empirically. The engine's composite pronunciation score reproduces the documented weighting formulas exactly in both configurations (hand-recomputed from sub-scores).

Determinism. Across five identical submissions per language, the standard deviation of every reported score was 0.0. The engine is deterministic on identical audio; observed variance on real recordings is therefore attributable to the audio, not the instrument.

Native miscue labeling fails on sequence phenomena. The planted repetition was never labeled as a repetition in any run: the Dutch response labeled the duplicated word an insertion; the English response absorbed it entirely, reporting a word list that matched the reference one-to-one even though the unscripted transcript correctly contained the duplicate. The planted substitution produced cascading misattribution in both languages: in the Dutch case, the aligner attributed an insertion to an unrelated later word that happened to rhyme with the spoken substitute. Omissions and insertions were flagged cleanly. The conclusion for architecture is direct: the engine's error-type output alone cannot implement the Section 2.1 taxonomy, and any system reporting repetitions or self-corrections on top of this engine (as Reading Progress does) is necessarily computing them in a layer above the engine.

Forced mispronunciations work, and are severe. The SSML phoneme override was honored in both languages: the Dutch word rendered with a wrong vowel was scored 31 of 100 and flagged as a mispronunciation, and the unscripted pass independently transcribed it as a different real word; the English case scored 4. This establishes the mechanism used at scale in Study 2 and examined as a methodology in Paper 7, and it previews the transcription behavior that motivates merge rule 1: severe decoding errors surface in transcripts as *other words*, never as phonetic garbage.

A silent truncation hazard. Single-shot recognition returned the first sentence fragment of multi-sentence audio (2.5 of 15 seconds) with no error signal. All production paths use continuous recognition.

5. Study 2: Synthetic validation battery

5.1 Method

Eight new reference texts (four per language, 40 to 80 words, child-appropriate register, varied punctuation), each rendered in nine variants: clean, one-word omission, three-word omission, substitution, insertion, repetition, self-correction (a wrong-phoneme rendering of a word immediately followed by its correct form), and two mispronunciation severities (vowel swap, consonant swap). Planted positions were seeded per text and recorded as ground truth. Each variant was rendered by two neural voices per language: 144 items, 432 engine calls (TTS, scripted, unscripted), zero call failures. The analysis consumes items through a corpus-adapter interface: a validation item is (audio, reference text, ground-truth miscue annotations) regardless of origin, so the identical analysis re-runs on real corpus recordings when access is granted.

Detection is scored at exact planted position, per miscue type and language, with a secondary ± 1 -position tolerance analysis.

5.2 Results with the v0 rules

Table 2. Precision / recall (exact position), v0 rules.

Type	English P / R	Dutch P / R
omission	.97 / .97	.45 / .91
substitution	.20 / 1.00	.05 / 1.00
insertion	.53 / 1.00	.38 / .75
repetition	.78 / .88	.60 / .38
mispronunciation	.36 / .56	.33 / .38
self-correction	— / .00	— / .00

Recall of planted miscues was high for omissions, substitutions, and insertions, but precision was poor to catastrophic for substitutions, and self-correction detection failed entirely. WCPM behaved as expected: single planted errors depressed WCPM by 2.9 words per minute on average against the clean baseline, and three-error variants by 5.6, scaling roughly linearly with planted-error count.

5.3 Root-cause diagnosis

Spot-checking raw responses located two mechanisms, neither a classifier bug in the narrow sense.

Recognizer normalization drift drives the substitution false positives: the unscripted pass transcribes correctly read words as close variants, recurring systematically (a proper name transcribed in its more common spelling; a verb transcribed in a different inflection). The classifier correctly reports that the transcript differs from the reference; the transcript is simply not a faithful record at that grain. This is an instrument ceiling with a product consequence: uncorroborated text-diffing would tell children they misread words they read correctly.

Self-correction misses decompose into two mechanisms: the wrong-phoneme first attempt is transcribed as an unrelated word, so no textual lookback can associate it with its target; or it is transcribed as the correct word itself, in which case the sequence looks like a repetition. Both were pre-documented limitations of the v0 text-only lookback; the battery quantified them.

5.4 The v0.2 corroboration rules, on identical data

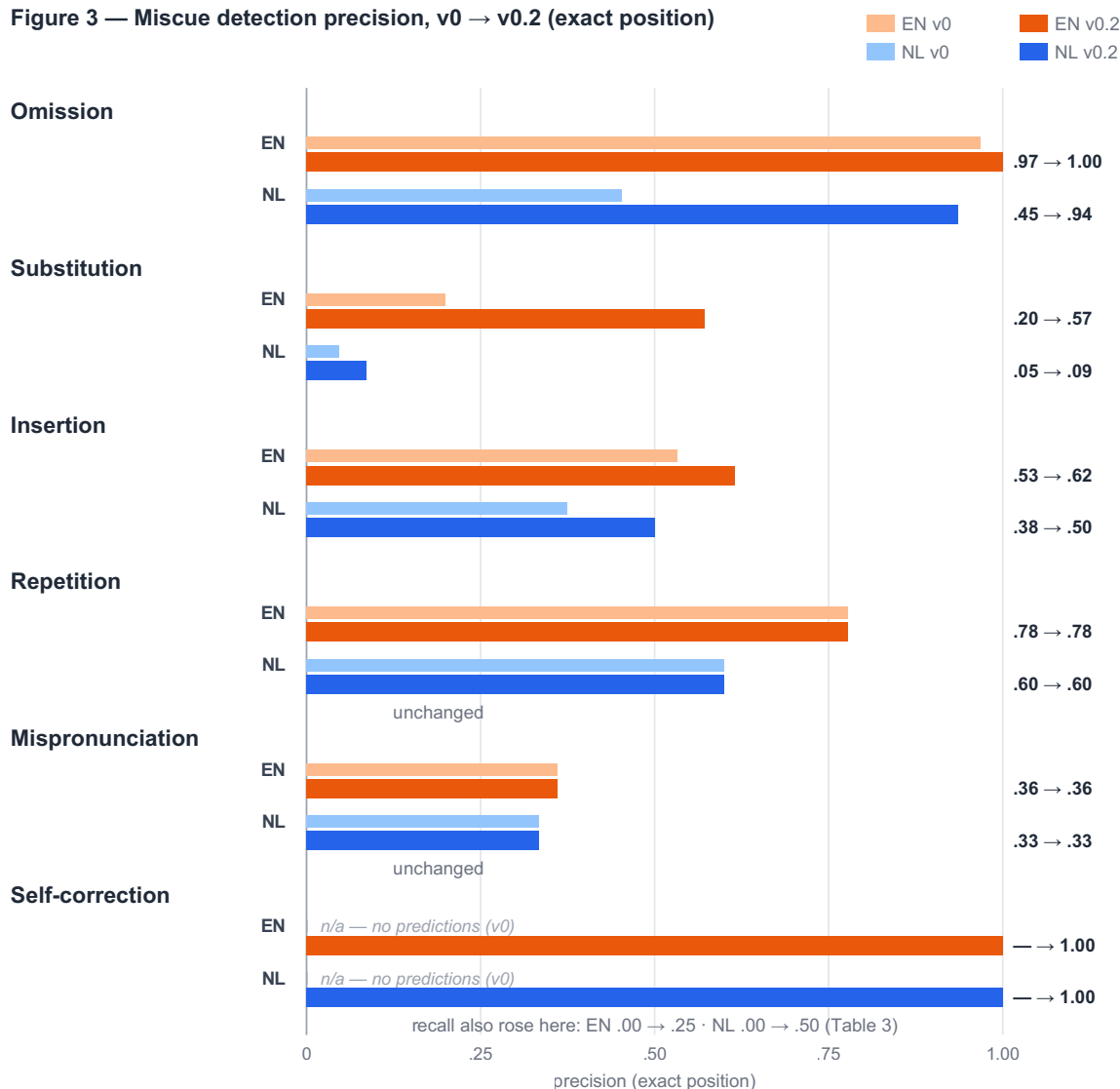
The Section 3.5 rules 3 to 5 were added and the entire battery re-analyzed offline against the same 432 stored responses. Figure 3 shows the precision changes per miscue type and language.

Table 3. vo → vo.2 (exact position).

Type	English P	English R	Dutch P	Dutch R
omission	.97 → 1.00	.97	.45 → .94	.91
substitution	.20 → .57	1.00	.05 → .09	1.00
self-correction	— → 1.00	.00 → .25	— → 1.00	.00 → .50
insertion	.53 → .62	1.00	.38 → .50	.75
repetition / mispronunciation	unchanged		unchanged	

Insertion precision improved as a side effect of the similarity-based self-correction rule: hypothesis tokens that vo mislabeled as insertions were reclassified as the self-corrections they are, shrinking the insertion false-positive count without touching insertion recall. No planted miscue detection was lost at exact position.

Figure 3 — Miscue detection precision, v0 → v0.2 (exact position)



Precision at exact planted position, v0 vs v0.2 rules, 144-item synthetic battery (§5.2–5.4). Bars: lighter = v0, solid = v0.2.

The only worsened cells appear in the ± 1 -tolerance analysis of omissions, and tracing them shows the tolerance matcher had previously credited a spurious near-miss detection by coincidence; v0.2 removed the spurious detection and the coincidental credit with it.

One candidate rule was designed, implemented, and **rejected during this iteration**: triggering self-correction reclassification from the engine's insertion-tagging of consecutive identical words. Re-analysis showed it collapsed genuine repetition recall to zero, because the engine tags every literal repeat that way, not only self-corrections. The rule was reverted and the failed design is reported here deliberately: on this engine, self-corrections whose first attempt transcribes as the correct word are not distinguishable from disfluent repetitions by any signal we could find, including the engine's raw alignment metadata. They remain a documented limitation.

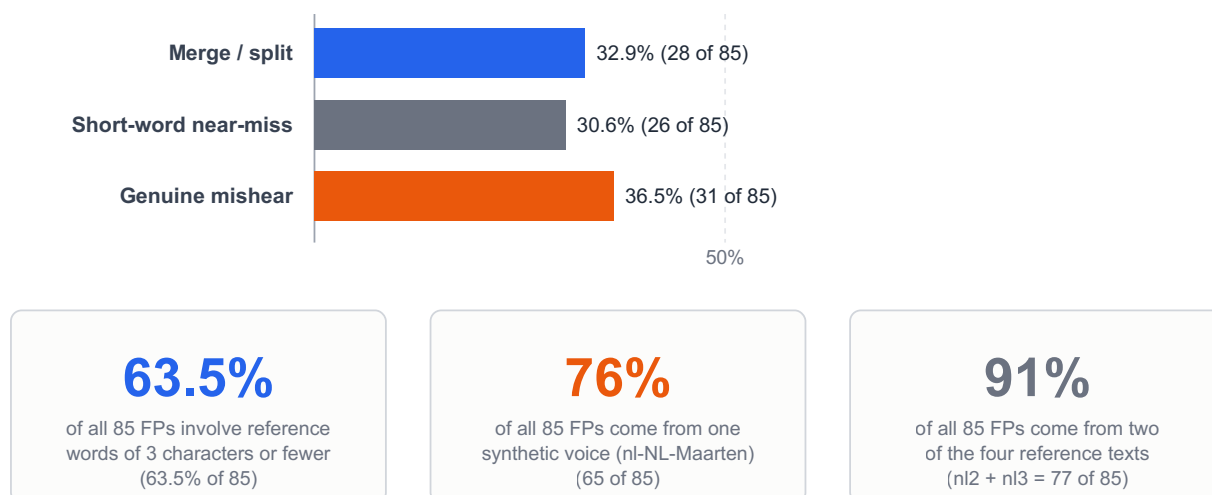
5.5 The residual Dutch substitution false positives, diagnosed

Dutch substitution precision remained low (.09, 85 residual false positives) after drift suppression. A mechanism-level analysis of all 85 (Figure 4):

- **32.9 percent are recognizer merges and splits:** the transcript fuses two reference words into a compound or splits one into two, and the alignment reads the debris as substitutions.
- **30.6 percent are near-misses below the similarity threshold,** concentrated on short words: 63.5 percent of all false positives involve reference words of three characters or fewer, where a single edit already drops the similarity ratio below the 0.75 threshold. The similarity measure loses its dynamic range exactly where Dutch function words live.
- **36.5 percent are genuine recognizer mishears** of correctly read words, with no textual signal of their innocence.

Figure 4 — NL substitution false positives: mechanism and concentration

85 residual false positives after v0.2 drift suppression (NL substitution precision .09) — §5.5



Three mutually exclusive mechanism classes sum to the residual 85 (§5.5). The distribution is not structural to Dutch: false positives concentrate in specific short words, one voice, and two of four texts. Source: nl_substitution_diagnosis.json.

The distribution is not structural to Dutch: four specific words account for the large majority, two of the four texts carry 91 percent of the false positives, and one of the two synthetic voices carries 76 percent. Because every genuinely planted substitution in the battery carries a non-clean scripted verdict, threshold changes could not cost true positives *on this data*; that guarantee is a property of the battery, not of reading, and the thresholds were therefore left unchanged pending real child-speech data. The practical conclusion: substitution precision claims for Dutch are not supportable from this battery, and the number is published as a limitation rather than tuned away.

6. Discussion

What the numbers do and do not mean. The Study 2 figures are mechanics validation: they measure whether the published rules recover known miscues from clean synthetic speech through a real engine. They are not comparable to human-agreement statistics from systems validated on real children (Bolaños et al., 2013; Molenaar et al., 2023), and no such comparison is claimed. The relationship is complementary: mechanics validation with machine-known ground truth catches failures that human-agreement studies cannot even see (a scorer could agree with humans on aggregate WCPM while misclassifying miscue types systematically), and it is repeatable at every rule change for the cost of compute alone.

The false-accusation framing. For a formative classroom tool, the product-critical error is not a missed miscue but a false accusation: a child told they misread a word they read correctly. The corroboration architecture is asymmetric by design in exactly this direction; every rule added in vo.2 suppresses accusations that one pass makes and the other contradicts, every suppression is counted and inspectable, and the residual Dutch false-positive rate is published rather than smoothed over. We suggest false-accusation rate per correctly-read word as a standard reporting metric for automated ORF scorers; it is rarely reported in the literature, vendor claims notwithstanding.

Implications for closed products. Study 1 establishes that the engine underlying Reading Progress cannot natively produce repetition or self-correction labels and misattributes substitution cascades. Reading Progress reports repetitions and self-corrections; whatever product layer produces them faces the same evidence base documented here, and its behavior on these phenomena is unpublished. Schools using such reports for decisions have, at present, no way to know how those categories behave. This is the credibility argument for open measurement layers, made concrete.

Cost. The dual-pass design doubles engine calls per reading; at current pricing this is approximately six euro cents per two-minute reading, and the language-independent prosody layer avoids both the engine's prosody surcharge and its English-only gate. Openness and economy point the same way here.

7. Limitations

1. **Synthetic, fluent, adult speech.** Both studies use TTS audio. The battery validates measurement mechanics, not performance on real children's voices, whose acoustic properties degrade recognition substantially (Section 2.3). No child-validity claim is made anywhere in this paper.

2. **Discrete planted errors.** Real miscues are messier than one clean planted error per variant: partial attempts, drawn-out decoding, dialect features. The battery's error model is a lower bound on difficulty.
3. **Flemish readers are scored against Netherlands-Dutch norms**, because the engine supports nl-NL only. The size of this penalty is unknown and is a primary question for the CHOREC follow-up.
4. **Dutch substitution precision is .09** on this battery, fully diagnosed but unresolved; Dutch substitution counts should be treated as indicative, not exact, until corroborated on real recordings.
5. **Self-correction recall is partial** (.25 English, .50 Dutch, at precision 1.0), and one confusion class (first attempt transcribed as the correct word) is undetectable on this engine without sacrificing repetition detection.
6. **Thresholds are battery-tuned.** All defaults are published and versioned, and recalibration on real data is a stated part of the method, not an admission against it.
7. **Single engine.** The specification is engine-independent; the evidence in this paper is engine-specific. Replication on a second engine is future work.

8. Conclusion and registered future work

This paper published a complete, parameterized measurement architecture for oral reading fluency in Dutch and English, characterized its commercial engine as a measurement instrument, and validated its mechanics against planted ground truth through a fully reported iteration cycle. The measurement layer detects omissions near-perfectly, recovers self-corrections that the engine cannot represent at all, and, most importantly for classroom use, refuses to accuse readers of errors that only one of its two evidence sources supports.

Registered follow-up work, in order:

1. **Criterion validity on real children's oral reading** against the miscue-annotated CHOREC corpus (Cleuren, Duchateau, Ghesquière, & Van hamme, 2008) (license requested July 2026), through the same adapter interface and analysis as Study 2, including quantification of the Flemish/nl-NL penalty (Limitation 3).
2. **Reader-level ability estimation** on the FRS scale from fluency observations, specified and simulated in Paper 6 of this series.
3. **The synthetic-validation methodology itself**, its transfer validity and its limits, examined in Paper 7.
4. **Threshold recalibration and the batched-revision doctrine** applied to the classifier, mirroring the FRS scale's versioned recalibration rounds (Vande Moortel, 2026a). The CHOREC study will also treat annotation agreement itself with care: a recent re-examination of an extended CHOREC-style protocol reported substantially

lower inter-annotator agreement than the corpus's own headline figures (Gao, Tejedor-García, Cucchiarini, & Strik, 2025).

References

- Bolaños, D., Cole, R. A., Ward, W. H., Tindal, G. A., Hasbrouck, J., & Schwanenflugel, P. J. (2013). Human and automated assessment of oral reading fluency. *Journal of Educational Psychology*, 105(4), 1142–1151. <https://doi.org/10.1037/a0031479>
- Cleuren, L., Duchateau, J., Ghesquière, P., & Van hamme, H. (2008). Children's Oral Reading Corpus (CHOREC): Description and assessment of annotator agreement. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC 2008)*. European Language Resources Association. <https://aclanthology.org/L08-1205/>
- Daane, M. C., Campbell, J. R., Grigg, W. S., Goodman, M. J., & Oranje, A. (2005). *Fourth-grade students reading aloud: NAEP 2002 special study of oral reading* (NCES 2006-469). National Center for Education Statistics. <https://nces.ed.gov/NATIONSREPORTCARD/pubs/studies/2006469.aspx>
- Deno, S. L. (1985). Curriculum-based measurement: The emerging alternative. *Exceptional Children*, 52(3), 219–232.
- Evaluation of the consistency of a speech verification system with human raters in early literacy screening assessments. (2026). *Frontiers in Education*. <https://www.frontiersin.org/journals/education/articles/10.3389/feduc.2026.1671946/full>
- Fan, R., Shankar, N., & Alwan, A. (2024). Benchmarking children's ASR with supervised and self-supervised speech foundation models. In *Proceedings of INTERSPEECH 2024*. <https://arxiv.org/abs/2406.10507>
- Gao, L., Tejedor-García, C., Cucchiarini, C., & Strik, H. (2025). Rethinking reading miscue annotation protocol: Insights from re-examining Dutch child reading annotation. In *Proceedings of the 10th Workshop on Speech and Language Technology in Education (SLaTE 2025)* (pp. 116–117). https://www.isca-archive.org/slate_2025/gao25_slate.pdf
- Harmsen, W., van Hout, R., Cucchiarini, C., & Strik, H. (2025). Can ASR generate valid measures of child reading fluency? In *Proceedings of INTERSPEECH 2025* (pp. 2395–2399). <https://doi.org/10.21437/Interspeech.2025-306>
- Hasbrouck, J., & Tindal, G. (2017). *An update to compiled ORF norms* (ERIC ED594994).
- Lathouwers, N., Gao, L., Cucchiarini, C., & Strik, H. (2026). *Transcribing children's speech: ASR performance and obtaining reliable orthographic transcriptions* [Preprint]. arXiv. <https://arxiv.org/abs/2605.28833>
- Microsoft. (n.d.-a). *How to configure pronunciation assessment* [Azure AI Speech documentation]. Microsoft Learn. Retrieved July 10, 2026, from <https://learn.microsoft.com/azure/ai-services/speech-service/how-to-pronunciation-assessment>
- Microsoft. (n.d.-b). *Transparency note: Pronunciation assessment* [Azure AI Foundry Responsible AI documentation]. Microsoft Learn. Retrieved July 10, 2026, from <https://learn.microsoft.com/azure/foundry/responsible-ai/speech-service/pronunciation-assessment/transparency-note-pronunciation-assessment>

- Microsoft. (n.d.-c). *readingAssignmentSubmission resource type* [Microsoft Graph v1.0 documentation]. Microsoft Learn. Retrieved July 10, 2026, from <https://learn.microsoft.com/graph/api/resources/readingassignmentsubmission>
- Microsoft. (n.d.-d). *Research behind Reading Progress and Reading Coach* [Educator Center documentation]. Microsoft Learn. Retrieved July 7, 2026, from <https://learn.microsoft.com/training/educator-center/product-guides/reading-progress/research>
- Molenaar, B., Tejedor-Garcia, C., Strik, H., & Cucchiarini, C. (2023). Automatic assessment of oral reading accuracy for reading diagnostics. In *Proceedings of INTERSPEECH 2023* (pp. 5232–5236). <https://doi.org/10.21437/Interspeech.2023-1681>
- Mostow, J., & Duong, M. (2009). Automated assessment of oral reading prosody. In *Proceedings of the 14th International Conference on Artificial Intelligence in Education (AIED 2009)* (pp. 189–196). <https://doi.org/10.3233/978-1-60750-028-5-189>
- National Center for Education Statistics. (1995). *NAEPfacts: Listening to children read aloud: Oral fluency* (NCES 95-762). U.S. Department of Education. <https://nces.ed.gov/pubs95/web/95762.asp>
- National Reading Panel. (2000). *Teaching children to read: An evidence-based assessment of the scientific research literature on reading and its implications for reading instruction*. National Institute of Child Health and Human Development.
- Reliability of ratings of Multidimensional Fluency Scale with many-facet Rasch model. (2022). *International Journal of Assessment Tools in Education*. <https://dergipark.org.tr/en/pub/ijate/article/974214>
- Sabatini, J., Wang, Z., & O'Reilly, T. (2019). Relating reading comprehension to oral reading performance in the NAEP fourth-grade special study of oral reading. *Reading Research Quarterly*, 54(2), 253–271.
- van der Velde, M., Harmsen, W., Veldkamp, S., Feskens, R., Keuning, J., & Swart, N. M. (2025). Speech enabled reading fluency assessment: A validation study. *International Journal of Artificial Intelligence in Education*. <https://pmc.ncbi.nlm.nih.gov/articles/PMC12686063/>
- Vande Moortel, J. (2026a). *A living standard: Crowdsourced comparative judgment and batched recalibration of an open reading-difficulty scale (vo.2)* [Working paper]. Orion-Solution. <https://frs.orion-solution.net/research>
- Vande Moortel, J. (2026b). *Calibrating an open reading-difficulty scale without human-labeled corpora: LLM pairwise judgments, deterministic features, and their composition (vo.1)* [Working paper]. Orion-Solution. <https://frs.orion-solution.net/research>
- Vande Moortel, J. (2026c). *Measuring reading growth: The Free Reading Standard and Microsoft Reading Progress in the classroom (vo.1)* [Working paper]. Orion-Solution. <https://frs.orion-solution.net/research>
- Vande Moortel, J. (2026d). *The Free Reading Standard: Definition and construction of an open, continuous scale for technical reading difficulty in Dutch and English (vo.2)* [Working paper]. Orion-Solution. <https://frs.orion-solution.net/research>
- Balogh, J., Bernstein, J., Cheng, J., Van Moere, A., Townshend, B., & Suzuki, M. (2012). Validation of automated scoring of oral reading. *Educational and Psychological Measurement*, 72(3), 435–452. <https://doi.org/10.1177/0013164411412590>

Appendix A. Published parameters

Parameter	Default	Role	Provenance
θ_{mis} (mispron accuracy threshold)	60	mispronunciation classification (with engine verdict)	engine convention
θ_{drift} (drift similarity)	0.75	transcription-drift suppression	battery-tuned
θ_{sc} (self-correction similarity)	0.60	self-correction association	battery-tuned
lookback window	2 tokens	repetition/self-correction search	design
boundary guard adjacency	± 1 reference position	verdict suppression at miscue boundaries	Study 1 evidence
pause threshold	300 ms	pause detection	ORF convention
F0 frame / hop	40 / 10 ms	pitch tracking	design
F0 search band	60–500 Hz	pitch tracking	speech range
F0 voicing threshold	0.30	voiced-frame gate	design
F0 range trim	5th–95th percentile (≥ 20 samples)	transient robustness	pitch-tracker evaluation
audio gate	2–120 s; clipping $> 1\%$; RMS floor	pre-scoring quality gate	Section 2.3 evidence

Appendix B. Miscue classification rules (pseudocode)

Two stages: alignment produces a reference-versus-hypothesis operations sequence; classification turns those operations, plus the scripted pass's own per-word verdicts, into a final status for every reference word and a list of miscue events. Named thresholds (θ_{mis} , θ_{drift} , θ_{sc} , lookback window L , boundary guard adjacency) are defined in Appendix A. Source: `frs_core/fluency/align.py` and `frs_core/fluency/miscues.py` (rule-set version v0.2).

B.1 Tokenization and alignment

```

tokenize_reference(text):
    for each whitespace-delimited chunk in text:
        display = chunk with leading/trailing punctuation stripped
        norm = casefold(display)
        punct = the trailing sentence/clause mark (. , ! ? : ;), if any, else None
        append {index, text: display, norm, punct}
    # a chunk that is pure punctuation attaches its mark to the
    # previous word's punct field instead of producing its own record

tokenize_hypothesis(transcript_words):
    for each word in transcript_words:
        display = word with leading/trailing punctuation stripped
        append {text: display, norm: casefold(display)}

align_tokens(ref_norms, hyp_norms):
    # classic Levenshtein DP: substitution cost 1, indel cost 1
    dp[0][0] = 0
    dp[i][0] = i for all i in 1..n_ref
    dp[0][j] = j for all j in 1..n_hyp
    for i in 1..n_ref, for j in 1..n_hyp:
        sub_cost = 0 if ref_norms[i] == hyp_norms[j] else 1
        dp[i][j] = min(
            dp[i-1][j-1] + sub_cost,    # match or substitution
            dp[i-1][j] + 1,              # omission: ref word unmatched
            dp[i][j-1] + 1,              # insertion: hyp word unmatched
        )
    # traceback from dp[n_ref][n_hyp] to dp[0][0]. Tie-break priority
    # MATCH > SUBSTITUTION > OMISSION > INSERTION whenever more than one
    # predecessor reaches the minimum -- this fixed priority is what makes
    # the alignment deterministic on identical input (Section 3.7)
    ops = []
    i, j = n_ref, n_hyp
    while i > 0 or j > 0:
        if i>0 and j>0 and ref_norms[i-1]==hyp_norms[j-1] and dp[i][j]==dp[i-1][j-1]:
            prepend {op: MATCH, ref_idx: i-1, hyp_idx: j-1} to ops; i -= 1; j -= 1
        elif i>0 and j>0 and dp[i][j]==dp[i-1][j-1]+1:
            prepend {op: SUBSTITUTION, ref_idx: i-1, hyp_idx: j-1} to ops; i -= 1; j -= 1
        elif i>0 and dp[i][j]==dp[i-1][j]+1:
            prepend {op: OMISSION, ref_idx: i-1, hyp_idx: None} to ops; i -= 1
        elif j>0 and dp[i][j]==dp[i][j-1]+1:
            prepend {op: INSERTION, ref_idx: None, hyp_idx: j-1} to ops; j -= 1
    return ops

```

B.2 Merge the scripted pass's per-word verdicts

The scripted pass echoes every reference word once, in reference order, plus extra entries for words it believes were inserted (those carry no reference index of their own and must be resolved to adjacent reference positions here, at merge time, from raw sequence order).

```
merge_scripted_scores(ref_tokens, scripted_words):
    scored = {} # ref_idx -> {accuracy_score, error_type}
    insertion_adjacent = {} # ref indices next to a scripted-pass Insertion entry
    ref_i = 0
    pending_insertion = False
    for w in scripted_words:
        if w.error_type == "Insertion":
            if ref_i - 1 >= 0: insertion_adjacent.add(ref_i - 1)
            pending_insertion = True
            continue # consumes no reference index
        if ref_i >= len(ref_tokens): break
        if pending_insertion:
            insertion_adjacent.add(ref_i)
            pending_insertion = False
        scored[ref_i] = {accuracy_score: w.accuracy_score, error_type: w.error_type}
        ref_i += 1
    return scored, insertion_adjacent
```

B.3 Classification, in execution order

Given: `ref_tokens`, `hyp_tokens`, `ops` (from B.1), `scripted_words`. Thresholds: `theta_mis` = 60, `theta_drift` = 0.75, `theta_sc` = 0.60, `L` = 2 hypothesis tokens, boundary guard adjacency = +-1 reference position.

```
# Step 1 -- base statuses from the DP alignment. INSERTION ops carry no
# ref_idx and are not placed here; they are handled in Step 7.
words = array of length n_ref, initially None
for op in ops:
    if op.op == MATCH:        words[op.ref_idx] = {status: "correct", hyp_idx: op.hyp_idx}
    if op.op == SUBSTITUTION: words[op.ref_idx] = {status: "substitution", hyp_idx:
op.hyp_idx}
    if op.op == OMISSION:     words[op.ref_idx] = {status: "omission", hyp_idx: None}

# Step 2 -- merge scripted-pass scores (B.2)
scored, insertion_adjacent = merge_scripted_scores(ref_tokens, scripted_words)

# Step 3 -- boundary-noise guard candidate set: reference indices next to
# a scripted-pass Insertion entry (from Step 2), unioned with reference
# indices next to a scripted-pass Omission entry
boundary_ref_idxxs = copy(insertion_adjacent)
for j, info in scored:
    if info.error_type == "Omission":
        if j - 1 >= 0:        boundary_ref_idxxs.add(j - 1)
        if j + 1 < n_ref:    boundary_ref_idxxs.add(j + 1)

# Step 4 -- mispronunciation reclassification: requires the engine's own
# ErrorType verdict AND a below-threshold score (an AND, never either
# alone); applies to "correct"- and "substitution"-status words only
boundary_suppressed_count = 0
for i, w in words (skip None):
    info = scored.get(i, {})
    w.accuracy_score = info.accuracy_score
    w.boundary_suppressed = False
    is_low_score = info.accuracy_score is not None and info.accuracy_score < theta_mis
    if w.status in ("correct", "substitution")
    and info.error_type == "Mispronunciation"
    and is_low_score:
        if w.status == "correct" and i in boundary_ref_idxxs:
            # the engine's own forced alignment is noisy at this boundary
            # (Section 4.2's cascade finding) -- do not accuse a word the
            # independent DP alignment already calls "correct"
            w.boundary_suppressed = True
            boundary_suppressed_count += 1
        else:
            # substitution-override: a "substitution"-status word is NEVER
            # suppressed here, even adjacent to a boundary -- a severe
            # decoding error surfaces in the transcript as a real-word
            # substitution, and the engine's own Mispronunciation verdict
            # is what recovers "failed to decode this word" from "read a
            # different word" (Section 3.5, rule 1)
            w.status = "mispronunciation"

# Step 5 -- v0.2: STT-drift suppression (substitution) / omission
# corroboration. Runs after Step 4, so a word already promoted to
# "mispronunciation" there is never reconsidered here.
stt_drift_suppressed_count = 0
for i, w in words (skip None):
```

```

w.stt_drift_suppressed = False
info = scored.get(i, {})
scripted_clean = (info.error_type in (None, "None")
                  and info.accuracy_score is not None
                  and info.accuracy_score >= theta_mis)
if not scripted_clean: continue
if w.status == "substitution":
    if token_similarity(ref_tokens[i].norm, hyp_tokens[w.hyp_idx].norm) >= theta_drift:
        w.status = "correct"
        w.stt_drift_suppressed = True
        stt_drift_suppressed_count += 1
elif w.status == "omission":
    # no similarity check is possible here -- an omission has no
    # hypothesis token to compare against; the scripted pass's clean
    # verdict is the only corroborating signal available
    w.status = "correct"
    w.stt_drift_suppressed = True
    stt_drift_suppressed_count += 1

# Step 6 -- candidate set for self-correction rule (b), a documented dead
# letter against the 144-item battery: never observed to fire (Section
# 5.4), kept for a future capture that carries this exact verdict shape
mispron_like_repeat_idx = { i : scored[i].error_type == "Mispronunciation"
                           and not words[i].boundary_suppressed }

# Step 7 -- repetition / self-correction pass over unmatched hypothesis
# tokens (the DP's INSERTION ops)
matched_ops = [op in ops : op.op in (MATCH, SUBSTITUTION)]
match_only_ops = [op in ops : op.op == MATCH]
repetition_events = []
insertion_events = []
for op in ops where op.op == INSERTION:
    hyp_idx = op.hyp_idx
    hyp_norm = hyp_tokens[hyp_idx].norm

    # 7a. exact-text lookback, BOTH directions, within L hypothesis
    # positions, closest first. A repeated or self-corrected word usually
    # lands on the INSERTION side of the DP tie-break, not the MATCH side
    # (empirically verified direction: "cat cat" assigns MATCH to the
    # LAST occurrence, INSERTION to the earlier one).
    candidates = [m in matched_ops : |m.hyp_idx - hyp_idx| <= L], sorted by distance ascending
    matched = first candidate in candidates where ref_tokens[candidate.ref_idx].norm ==
hyp_norm

if matched is not None:
    ref_idx = matched.ref_idx
    target = words[ref_idx]
    if target.status in ("substitution", "mispronunciation"):
        target.status = "correct-self-corrected"          # v0 self-correction
    elif ref_idx in mispron_like_repeat_idx:
        target.status = "correct-self-corrected"          # v0.2 rule (b), dead letter
    else:
        repetition_events.append({ref_idx, hyp_idx})
    continue

```

```
# 7b. no exact match nearby -- v0.2 rule (a), the similarity variant.
# FORWARD only, MATCH ops only (never SUBSTITUTION, so a genuinely
# different, unrelated reference word nearby cannot be mistaken for
# the self-correction target).
selfcorr_candidates = [m in match_only_ops : 0 < (m.hyp_idx - hyp_idx) <= L],
                      sorted by distance ascending
target = first candidate in selfcorr_candidates
      where token_similarity(hyp_norm, ref_tokens[candidate.ref_idx].norm) >= theta_sc
if target is not None:
    words[target.ref_idx].status = "correct-self-corrected"
else:
    insertion_events.append({hyp_idx})          # real, unattached insertion

# Step 8 -- assemble events. Every reference word ends with exactly one
# final status: correct, correct-self-corrected, omission, substitution,
# or mispronunciation. Repetitions and unattached insertions are separate
# events, not word statuses, since one reference word can be repeated
# more than once.
events = []
for w in words (skip None):
    event_type = {
        "omission": "omission",
        "substitution": "substitution",
        "mispronunciation": "mispronunciation",
        "correct-self-corrected": "self_correction",
    }.get(w.status) # "correct" maps to no event
    if event_type: events.append({event_type, ref_idx: w.index, hyp_idx: w.hyp_idx})
events += repetition_events
events += insertion_events

return {words, events, boundary_suppressed_count, stt_drift_suppressed_count}
```

Appendix C. Battery reference texts

Eight reference texts, four Dutch and four English, authored for the Study 2 battery (Section 5.1). Source: `experiments/fluency_synthetic_battery/texts.py`. Word counts are `len(text.split())` on the exact strings below. FRS scores were computed once against the local FRS scoring service (POST /v1/analyze, live calibration, FRS 1.0.1) on 2026-07-10 and are cross-checked against the identical strings embedded in `frs-site/site/src/data/readingSamples.ts`; both sources agree exactly.

n11 (49 words, FRS 25.4)

Sara had een kleine grijze kat. Elke ochtend speelde ze buiten met een rode bal. Op een dag rolde de bal onder het hek. De kat rende er meteen achteraan! Ze zocht tussen de struiken, maar kon de bal niet vinden. Gelukkig vond papa hem terug naast de schutting.

Gloss: a girl's cat loses its ball under the garden fence, and after searching among the bushes her father finds it again next to the fence.

nl2 (49 words, FRS 31.1)

In de tuin van oma woonde een klein bruin konijn. Het at graag verse sla en zoete wortels. Op zonnige dagen groef het konijn kleine gaatjes in de grond. Waar ging het konijn toch elke avond naartoe? Oma volgde het stilletjes en zag dat het onder de heg sliep.

Gloss: a young rabbit living in grandma's garden eats lettuce and carrots and digs small holes, until grandma quietly follows it and discovers it sleeps every night under the hedge.

nl3 (51 words, FRS 27.7)

Bij de vijver achter het huis zwom een vrolijke gouden vis. Elke middag kwamen de kinderen kijken hoe hij rondjes zwom. De vis at graag kleine stukjes brood, die zachtjes op het water dreven. Soms sprong hij hoog uit het water, alsof hij wilde spelen. De kinderen klaptten in hun handen.

Gloss: a cheerful golden fish in a pond behind the house eats floating bread crumbs and sometimes leaps out of the water as if it wants to play, while the children watch and clap.

nl4 (50 words, FRS 30.5)

Hoog in de oude boom bouwde een kleine vogel haar nest. Ze verzamelde takjes, veren en zachte mos uit de tuin. Elke ochtend zong ze een vrolijk liedje voor de zon opkwam. Op een dag legde ze drie kleine blauwe eitjes. De hele buurt wachtte spannend tot de vogeltjes uitkwamen!

Gloss: a small bird builds a nest high in an old tree, sings each morning before sunrise, and lays three eggs that the whole neighborhood eagerly awaits hatching.

en1 (50 words, FRS 16.4)

Sara had a small gray cat. Every morning she played outside with a red ball. One day the ball rolled under the fence. The cat ran after it right away! She searched between the bushes, but could not find the ball. Luckily her dad found it near the garden wall.

en2 (44 words, FRS 21.9)

In grandma's garden lived a small brown rabbit. It loved eating fresh lettuce and sweet carrots. On sunny days the rabbit dug little holes in the ground. Where did the rabbit go every evening? Grandma followed quietly and saw it sleeping under the hedge.

en3 (55 words, FRS 20.1)

Beside the pond behind the house swam a cheerful golden fish. Every afternoon the children came to watch it swim in circles. The fish loved eating small pieces of bread that floated gently on the water. Sometimes it jumped high out of the water, as if it wanted to play. The children clapped their hands.

en4 (51 words, FRS 20.8)

High in the old tree a small bird built her nest. She gathered twigs, feathers, and soft moss from the garden. Every morning she sang a happy song before the sun came up. One day she laid three tiny blue eggs. The whole street waited eagerly for the eggs to hatch!

Appendix D. Per-cell battery results

Source: `experiments/fluency_synthetic_battery/results.json` (vo rules) and `results_v02.json` (vo.2 rules), field `precision_recall_table`. Both files report 144 items, 0 failed. TP/FN/FP are raw integer counts from the JSON; precision, recall, and F1 are copied from the JSON's own computed fields, rounded to two decimals (a null precision, where TP+FP is 0, is shown as "-"). Both files carry the identical set of tallies at tolerance 0 (exact planted position) and tolerance 1 (+-1 position); only the omission row changes between the two tolerances in either rule set.

Table D1. vo rules, per miscue type and language.

Type	Lang	TP	FN	FP	P	R	F1	TP (+-1)	FN (+-1)	FP (+-1)	P (+-1)	R (+-1)	F1 (+-1)
insertion	en	8	0	7	.53	1.00	.70	8	0	7	.53	1.00	.70
insertion	nl	6	2	10	.38	.75	.50	6	2	10	.38	.75	.50
mispronunciation	en	9	7	16	.36	.56	.44	9	7	16	.36	.56	.44
mispronunciation	nl	6	10	12	.33	.38	.35	6	10	12	.33	.38	.35
omission	en	31	1	1	.97	.97	.97	32	0	0	1.00	1.00	1.00
omission	nl	29	3	35	.45	.91	.60	32	0	32	.50	1.00	.67
repetition	en	7	1	2	.78	.88	.82	7	1	2	.78	.88	.82
repetition	nl	3	5	2	.60	.38	.46	3	5	2	.60	.38	.46
selfcorrection	en	0	8	0	-	.00	-	0	8	0	-	.00	-
selfcorrection	nl	0	8	0	-	.00	-	0	8	0	-	.00	-

Type	Lang	TP	FN	FP	P	R	F1	TP (+1)	FN (+1)	FP (+1)	P (+1)	R (+1)	F1 (+1)
substitution	en	8	0	32	.20	1.00	.33	8	0	32	.20	1.00	.33
substitution	nl	8	0	165	.05	1.00	.09	8	0	165	.05	1.00	.09

Table D2. vo.2 rules, per miscue type and language.

Type	Lang	TP	FN	FP	P	R	F1	TP (+1)	FN (+1)	FP (+1)	P (+1)	R (+1)	F1 (+1)
insertion	en	8	0	5	.62	1.00	.76	8	0	5	.62	1.00	.76
insertion	nl	6	2	6	.50	.75	.60	6	2	6	.50	.75	.60
mispronunciation	en	9	7	16	.36	.56	.44	9	7	16	.36	.56	.44
mispronunciation	nl	6	10	12	.33	.38	.35	6	10	12	.33	.38	.35
omission	en	31	1	0	1.00	.97	.98	31	1	0	1.00	.97	.98
omission	nl	29	3	2	.94	.91	.92	30	2	1	.97	.94	.95
repetition	en	7	1	2	.78	.88	.82	7	1	2	.78	.88	.82
repetition	nl	3	5	2	.60	.38	.46	3	5	2	.60	.38	.46
selfcorrection	en	2	6	0	1.00	.25	.40	2	6	0	1.00	.25	.40
selfcorrection	nl	4	4	0	1.00	.50	.67	4	4	0	1.00	.50	.67
substitution	en	8	0	6	.57	1.00	.73	8	0	6	.57	1.00	.73
substitution	nl	8	0	85	.09	1.00	.16	8	0	85	.09	1.00	.16

Cross-check: at exact position, TP+FN for each type/language cell is identical between Table D1 and D2 (the total number of planted miscues did not change), and matches the planted-item arithmetic in Section 5.1: omission 32/32 (en/nl; 8 omission₁ items x 1 + 8 omission₃ items x 3), substitution 8/8, insertion 8/8, repetition 8/8, self-correction 8/8, mispronunciation 16/16 (8 vowel-swap + 8 consonant-swap items). Only FP moves between vo and vo.2, which is what "no planted miscue detection was lost" (Section 5.4) means concretely.

Table D3. vo.2 suppression counters, per language.

Source: `results_v02.json`, per-item fields `flu_boundary_suppressed_count` and `flu_stt_drift_suppressed_count`, summed across the 72 items of each language. The JSON's own `boundary_suppressed_distribution` and `stt_drift_suppressed_distribution` fields are item-count histograms (how many items

had 0, 1, 2, ... suppressions) and are not split by language; the per-language totals below are derived by summing `per_item` and cross-checked against those histograms (boundary: 133 items with 0, 11 items with 1, sum = 11 = the English total below, since Dutch had zero boundary suppressions in this battery; STT-drift: 63/35/35/10/1 items at 0/1/2/3/5 suppressions, weighted sum = 140 = the en+nl total below).

Language	boundary_suppressed (total)	stt_drift_suppressed (total)	items (n)
en	11	27	72
nl	0	113	72

Appendix E. Engine response field inventory, nl-NL vs en-US

Source: Study 1 raw responses in `experiments/fluency_instrument/runs/` (`nl_clean_pa.json` compared field-by-field against `en_clean_pa.json`; the Omission- and Insertion-status word shapes are cross-checked against `nl_omission_pa.json`, `en_omission_pa.json`, and `nl_insertion_pa.json`). Every row below is a field actually observed in these raw responses, not a documentation claim.

Table E1. Utterance-level scores (`NBest[0].PronunciationAssessment`).

Field	nl-NL	en-US	Note
AccuracyScore	present	present	
FluencyScore	present	present	
ProsodyScore	absent	present (90.7 in the clean run)	locale-gated; the fifth top-level score is en-US only
CompletenessScore	present	present	
PronScore	present	present	different weighting formula per language, hand-recomputed exactly from the sub-scores (Section 4.2): nl sorts {Accuracy, Fluency, Completeness} and applies weights 0.6/0.2/0.2; en sorts {Accuracy, Fluency, Prosody, Completeness} and applies weights 0.4/0.2/0.2/0.2

Table E2. Word-level fields (`NBest[0].Words[]`).

Field	nl-NL	en-US
Word (text)	present	present

Field	nl-NL	en-US
Offset / Duration	present, except absent on Omission-status entries	same
PronunciationAssessment.ErrorType	present; observed values None, Omission, Insertion, Mispronunciation	same four values observed
PronunciationAssessment.AccuracyScore	present on every non-Omission word	same

Field	nl-NL	en-US
PronunciationAssessment.Feedback.Prosody.Break	absent	present on every word (ErrorTypes list plus BreakLength)
PronunciationAssessment.Feedback.Prosody.Intonation.Monotone	absent	present on every word (SyllablePitchDeltaConfidence)
Syllables[]	present, one entry per syllable	present
Syllables[].Syllable (name)	empty string on every syllable	populated, e.g. "dog"
Syllables[].PronunciationAssessment.AccuracyScore	present	present
Syllables[].Offset / Duration	present	present
Phonemes[]	present; empty list on Omission-status entries	same
Phonemes[].Phoneme (name)	empty string on every phoneme	populated, e.g. "d", "a", "g"

Field	nl-NL	en-US
Phonemes[].PronunciationAssessment.AccuracyScore	present, real per-phoneme scores	present
Phonemes[].PronunciationAssessment.NBestPhonemes[].Phoneme (name)	empty string	populated
Phonemes[].PronunciationAssessment.NBestPhonemes[].Score	present	present
Phonemes[].Offset / Duration	present	present

An Insertion-status word entry (the scripted pass's own claim that the reader added a word not in the reference) carries Offset, Duration, and PronunciationAssessment.AccuracyScore in both languages, unlike an Omission entry: something was actually said, so the engine can time and score it, just against no reference word of its own (Appendix B.2's `insertion_adjacent` bookkeeping exists specifically because these entries have no reference index). No SelfCorrection value was observed for ErrorType in either language, consistent with Section 4.2's finding that the engine's own miscue labeling cannot represent sequence phenomena and with the architectural decision (Section 3.2) to compute self-correction and repetition in a layer above the engine.