Human–AI Annotation Error Auditing for Hebrew Diacritization with Frontier LLMs

Hillel Gershuni and **Avi Shmidman**
Bar-Ilan University and DICTA
gershuni@gmail.com, avi.shmidman@biu.ac.il

Abstract

Large annotated datasets inevitably contain errors that are costly to identify via manual review. We study a human–AI annotation error auditing workflow using frontier Large Language Models (LLMs), focusing on Hebrew *nikud* (diacritization). We take the the EACL 2023 Hebrew Homograph Challenge Set as our test case. In a focused evaluation on 12 of the homograph sets with 271 confirmed errors (verified through exhaustive manual review of all 7,241 sentences), Gemini 3 Pro achieves 83.6% recall (95% confidence interval: [79.3%, 88.2%]) and 99.1% precision - substantially higher than other frontier LLMs. Two independent human experts achieved 62.4% and 42.8% recall respectively, a 20-percentage-point spread that reflects the difficulty of sparse-target error search. Even the union of both experts' findings (73.4% recall) falls short of a single LLM run (83.6%), while LLM-aided auditing reduces review effort by over 95%. We analyze the trade-offs between batch size and recall, and release both a human-verified Gold Standard with per-error difficulty annotations and a globally corrected version of the Challenge Set.

1 Introduction

High-quality annotations are foundational for evaluating and training NLP systems, but manual error auditing is slow and expensive. This challenge is particularly acute in the case of annotated datasets of Hebrew *nikud* (diacritization). Hebrew is typically written without vowel diacritics; in an annotated Hebrew corpus for *nikud*, the words are annotated with the relevant diacritic vowels. However, the determination of the correct *nikud* is not a simple dictionary lookup; rather, the task often requires disambiguating homographs using deep contextual cues. Even expert-curated resources can accumulate subtle errors that impact downstream model reliability. A case in point is the EACL 2023 Hebrew Homograph Challenge Set (Shmidman et al., 2023), upon which we will focus herein. This Challenge Set is comprised of ~150,000 sentences, each containing a Hebrew homograph which is annotated with diacritics to disambiguate its meaning. Naturally, in a human-curated dataset of this size, a certain number of errors will persist. The question is how to locate them.

We propose using frontier LLMs as a human–AI auditing tool to flag likely *nikud* errors. Unlike the traditional diacritization scenario, in which computational systems are built to diacritize non-diacritized texts (see below, 3.1), our auditing approach focuses on validating *existing* diacritization labels with a prioritized **recall-centric objective**, surfacing as many real issues as possible for subsequent human verification.

Our key contributions are as follows:

- We define a practical annotation error auditing procedure and quantify precision/recall trade-offs across multiple frontier LLMs.
- We quantify the impact of batch size on error detection sensitivity, identifying a practical sweet spot that balances recall and cost.
- We validate the pipeline through a workflow comparison with two independent human experts, reporting inter-annotator agreement ($\kappa = 0.57$) and showing that even the union of both experts' findings (73.4% recall) falls short of a single LLM run (83.6%), while LLM-aided auditing reduces review effort by over 95%.
- We provide a human-verified gold standard covering 7,241 sentences across 15 homograph files (12 homograph groups), with 271 confirmed diacritization errors and per-error difficulty annotations derived from cross-system detection rates, which can be used as a benchmark for future experiments in this area.

Proceedings of the 20th Linguistic Annotation Workshop (LAW XX), pages 33–46

- We apply our pipeline to the entirety of the aforementioned EACL 2023 Hebrew Homograph Challenge Set, and we release a new globally corrected version of this Challenge Set for the Hebrew NLP community.[1]

2 Task and Linguistic Background

The primary challenge in Hebrew diacritization auditing lies in the morphological density of the language. A single consonantal string can represent multiple parts of speech, tenses, or complex syntactic constructions involving suffixes.

Motivating Example. Consider the string אהבה (/ʔhbh/). Depending on the context, it may be diacritized as:

1. **Noun:** אַהֲבָה (love, /ʔa.hă.vaː/).
2. **Verb (3fs Past):** אָהֲבָה (she loved, /ʔaː.hă.vaː/).
3. **Verb + Accusative Suffix:** אֲהֵבָהּ (he loved her, /ʔă.he.'vaːh/).[2]

In these examples, the *nikud* distinction, for example between *patah* (אַ, historically [a]), *qamatz* (אָ, historically [aː]), and *hataf-patah* (אֲ, historically [ă]), is contextually governed by gender agreement or the presence of an accusative pronominal suffix (marked by a *mapiq* dot in the final ה). While the Tiberian diacritization system originally distinguished the quality and length of these vowels (with *qamatz* representing an intermediate [ɔ] sound), Modern Hebrew follows a Sephardic reading tradition in which these distinctions have merged as [a] in the relevant contexts. The *mapiq*, too, is consistently unpronounced in standard speech. Human auditors therefore cannot rely on phonetic intuition, as the orthography preserves historical-grammatical layers that have no auditory counterpart in the modern language. This complexity makes manual review at scale prohibitively costly.

[1]Released artifacts are at https://github.com/Dicta-Israel-Center-for-Text-Analysis/EACL_2023. The original EACL 2023 Challenge Set is preserved under git tag `v1.0`; the corrected Challenge Set, Gold Standard, and difficulty annotations are released under tag `v2.0`.

[2]This particular form combines a verb with a cliticized pronominal suffix, and would in principle be resolvable through morphological segmentation. However, Hebrew digital texts carry no segmentation markup: the entire orthographic string is encountered as a single token, indistinguishable from the homographic noun and verb forms above. Determining the correct diacritization therefore requires precisely the kind of contextual analysis described in this paper, and the same is true for other homographs in our evaluation set.

3 Related Work

3.1 Hebrew diacritization.

Hebrew diacritic restoration has been approached using rule-based systems (Choueka and Neeman, 1995) and hidden markov models (Gal, 2002), as well as neural-network-based systems such as *Dicta Nakdan* (Shmidman et al., 2020) and *Nakdimon* (Gershuni and Pinter, 2022), which take consonantal texts as input and predict the diacritics. In contrast, our work targets auditing: given a diacritization label, we ask an LLM to judge its contextual correctness.

3.2 Annotation error detection.

Automated detection of labeling errors has been studied using training dynamics (Swayamdipta et al., 2020) or probabilistic methods (Northcutt et al., 2021). LLMs provide a complementary tool: they can draw on broad knowledge and generate rationales without task-specific training. Our work contributes an empirical comparison of frontier LLM models in a realistic human-in-the-loop workflow.

4 Methodology

4.1 Source Dataset

We utilize the EACL 2023 Hebrew Homograph Challenge Set (Shmidman et al., 2023), containing 75 homographs across ~150,000 sentences. The corpus consists exclusively of **written Modern Hebrew**, drawn from a mixture of newspapers, Wikipedia, literature, and social media. Per-homograph sentence counts (typically 1,000 sentences for the primary analysis and 250–500 for secondary analyses) are tabulated in Appendix A of Shmidman et al. (2023). All auditing experiments reported in this paper use the original release of the Challenge Set; the corrected version is released as an output of this work.

4.2 Request Structure and Context Enrichment

Every processing request, regardless of batch size, included a fixed header consisting of:

- **System Prompt:** A linguist-persona prompt prioritizing high recall of *nikud* errors.
- (optionally) **Morphological Data:** A list of valid diacritized forms, lemmas, and grammatical features for the target homograph.

- (optionally) **Dictionary Definitions:** Semantic definitions and common usage examples for each morphological variant.[3]

This prompt was prepended to every batch of 50, 100, 200, or 500 sentences, allowing us to measure how batch size affects model performance while amortizing the fixed prompt cost across more examples. The full API request is given in Appendix D).

4.3 Experimental Design

We curated a **Focused Evaluation Subset** of 15 files (7,241 sentences total, gathered from 12 homograph groups), from across the EACL 2023 Hebrew Homograph Challenge Set. The selection proceeded as follows: an initial exploratory phase screened 25 homograph files from the alphabetical extremes of the dataset (homographs beginning with א–ב and ת) using multiple LLMs to flag suspected errors. From the results, we selected homograph groups with the *fewest* flagged errors, i.e., those where exhaustive human adjudication was feasible, to ensure a rigorous, fully verified Gold Standard. This criterion naturally filtered out high-noise homographs such as אם (see §6.4), whose large volume of prescriptive-norm disputes would have made complete adjudication impractical. The selection criteria were fixed before any focused evaluation runs were conducted.

This subset represents a diverse range of linguistic challenges including part-of-speech ambiguity, construct-state vs. attributive distinctions and inflectional errors.

Models and Parameters. We evaluate five frontier LLMs: Gemini 3 Pro, Gemini 3 Flash, GPT-5.2, and Claude Opus 4.5/4.6, all accessed via OpenRouter APIs in a zero-shot setting with the same prompt template (Appendix D) and structured JSON output. All runs used default temperature (T=1.0), following Google's recommendations for reasoning-centric generation (Google, 2026); temperature sensitivity is left for future work. No system-level seed was set; stochastic variation between runs is part of our analysis. A total of 50 valid focused runs were conducted.[4] Full model identifiers, access dates, and additional configuration details are provided in Appendix G.

4.4 Gold Standard (GS) Construction

The Gold Standard for the focused subset was established through a multi-stage process designed to maximize recall:

1. **Flag collection:** Candidate errors were flagged independently by multiple LLMs across different configurations and by two independent human experts (both native Hebrew speakers with graduate-level linguistics training), all without access to each other's output.

2. **Union aggregation:** All unique flags were pooled into a candidate set.

3. **Expert adjudication:** The first author reviewed every candidate, classifying each as *Error*, *Correct*, or *Other*. This yielded 265 confirmed errors across 863 adjudicated sentences.

4. **Exhaustive dormant-error audit:** To guard against errors absent from the candidate set, the first author manually reviewed all 6,378 remaining sentences, uncovering 6 additional errors (dormant rate: 0.09%).

5. **Adjudication robustness check:** Each expert's findings were compared against the final GS. Of 178 findings by Expert 1 and 121 by Expert 2, only 9 and 5 respectively targeted sentences that were labelled *Correct* in the Gold Standard (13 unique cases, as one sentence was flagged by both). All 13 were re-adjudicated; every Gold Standard label was confirmed, leaving the Gold Standard unchanged.

The final Gold Standard contains **271** confirmed errors across all 7,241 sentences; all recall figures below are computed against this fully verified denominator.

Error Difficulty Annotations. Each error is annotated with a difficulty label based on its cross-system detection rate (proportion of 50 runs that flagged it): *Easy* ($\geq$80%; 117 errors), *Medium* (40–80%; 109), *Hard* (<40%; 40), or *Undetected* (5). These labels are included in the released Gold Standard to support stratified evaluation.

[3] Definitions are sourced from the Academy of the Hebrew Language: https://hebrew-academy.org.il

[4] This count includes 10 exploratory runs not analyzed individually: 3 with Grok 4.1 Fast (recall <22%), 1 with Claude Sonnet 4.5 (recall <4%), and 6 early Gemini Pro runs using alternative prompt variants. All contribute to the aggregate union and difficulty statistics reported in §7.2 and §4.4. Results from two Gemini-Pro runs were discarded due to an incorrect reasoning-mode configuration.

5 Experiments and Results

Before reporting results on the focused subset, we conducted a sanity check using contaminated files with 308 synthetically injected errors (Appendix F); all three top models achieved ≥97% precision and ≥79% recall, confirming basic detection capability.[5]

Table 1 compares all models across three batch sizes. Consistent with our auditing objective, we prioritize **Recall** as the primary metric. Gemini 3 Pro achieves the highest recall at every batch size, with 83.0%–83.6% recall and ≥99% precision across batch sizes 50–200 (95% confidence interval for batch size 200: [79.3%, 88.2%]). All pairwise recall differences between Gemini 3 Pro and other models are statistically significant (McNemar's test with continuity correction, $p < 0.001$ for all comparisons at every batch size). GPT-5.2 ranks second with reasonable recall at batch size 50 (69.4%) but substantially lower precision (85.1%).

	Recall (%) ↑			Precision (%)		
Model	**B50**	**B200**	**B500**	**B50**	**B200**	**B500**
Gemini 3 Pro	**83.0**	**83.6**	**72.1**	**99.6**	**99.1**	**99.5**
GPT-5.2	69.4	55.2	43.7	85.1	94.9	98.3
Claude Opus 4	45.6	43.5	40.0	56.4	94.4	68.8
Gemini 3 Flash	59.0	45.2	27.1	39.7	40.4	57.9

Table 1: Model comparison on focused evaluation set (271 errors), no morphology, no dictionary (–m–d). Gemini 3 Pro values are averaged over 2 runs. B50, B200, etc. indicate batch sizes.

5.1 Batch Size Trends and Efficiency

As shown in Tables 1 and 3 and Figure 1, Gemini 3 Pro recall remains stable across batch sizes 50, 100, and 200 (83.0%–83.6% averaged across configurations); McNemar's tests confirm no statistically significant differences among these three sizes ($p > 0.8$). The 200-sentence batch thus represents a practical "sweet spot" for large-scale auditing: equivalent sensitivity at substantially lower cost (Table 2).

However, a **sharp recall drop** occurs at batch size 500, where average recall falls to 71.4% (−11 pp from batch size 200). This drop is highly significant (McNemar's test, $p < 0.001$ for all B200 vs. B500 run pairs). The cause is likely multifactorial: longer prompts, output length constraints, or reduced per-sentence attention. But the practical implication is clear: batches beyond 200 sentences incur a significant sensitivity penalty that outweighs cost savings.[6]

Cost and Review Burden. Table 2 summarizes the cost–recall trade-off. At batch size 200, a single Gemini 3 Pro run costs approximately $7 for our 7,241-sentence evaluation set ($0.96 per 1,000 sentences) and completes in ∼75 minutes of wall-clock time. In terms of *review burden*, i.e. the number of flagged findings a human reviewer must adjudicate, Gemini 3 Pro at batch size 200 produces ∼225 findings on average, yielding 31.0 findings per 1,000 sentences. This reduces the human task from scanning 7,241 sentences to verifying ∼225 items, a 96.9% reduction in review volume.

Batch	**Rec.**	**Cost**	**$/1K**	**Time**	**Flags./1K**
B50	83.0%	$12.67	$1.75	139m	31.4
B100	83.6%	$9.09	$1.26	104m	31.6
B200	82.2%	$6.96	$0.96	75m	31.0
B500	71.4%	$4.28	$0.59	41m	26.7

Table 2: Cost–efficiency comparison for Gemini 3 Pro (averaged across configurations). $/1K = cost per 1,000 sentences. Flags/1K = flagged findings per 1,000 sentences, representing the human review burden. B = batch size. B200 achieves comparable recall at 77% of B100 cost and 55% of B50 cost.

5.2 Ablation Study: Context Robustness and Semantic Noise

We conducted an ablation study on the auxiliary context features (Morphology and Dictionary) to understand their impact on model performance. As shown in Table 3, the results indicate that explicit morphological or dictionary data does not yield consistent improvements under our tested zero-shot conditions.

Auxiliary Context Does Not Help Consistently. No configuration consistently outperforms the others across batch sizes (Table 3): the bare –m–d setting (neither morphology nor dictionary) wins

[5] Because injected errors are often more salient than naturally occurring mistakes, these results should be interpreted as a sanity check rather than a proxy for real-world recall.

[6] Note that because source files rarely divide evenly by the batch size, the final batch of each file is typically shorter (e.g., 100–174 sentences at batch size 200). It would have been defensible to discard these tail batches altogether, since they do not strictly conform to the nominal batch size of the experimental condition. We chose to retain them, however, because (i) discarding them would have reduced the effective evaluation set and (ii) we verified empirically that they do not inflate recall: the mean detection rate for errors in these tail batches (67.9%) was within 2.2 pp of the rate in full-sized batches (65.7%). The effect is therefore negligible.

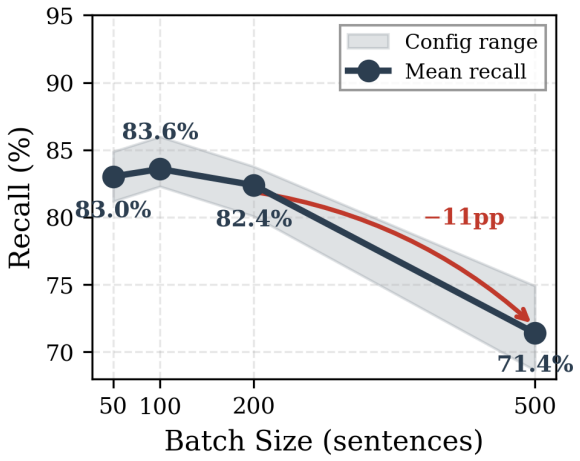

Figure 1: Recall vs. batch size for Gemini 3 Pro across all four context configurations (with and without morphological and dictionary info). Recall is stable from batch size 50 to 200, then drops sharply at 500. Context configuration has minimal effect relative to batch size.

at batch size 200, while +m–d (morphology only) wins at other batch sizes: 50, 100, and 500. The largest observed effect is at batch size 100, where +m–d achieves 233 TP vs. 223 for –m–d (+10 TP), though this is based on single runs and may reflect stochastic variation. McNemar's tests confirm no statistically significant pairwise differences at batch sizes 50, 100, and 200 (all $p > 0.28$); the inter-configuration gaps (5–10 TP) fall within run-to-run variance. These results suggest that the LLM possesses sufficient internal linguistic knowledge for disambiguation, and that it is not in need of extra morphological/dictionary info in order to make its determination.

Config	B50	B100	B200†	B500
–m–d (bare)	225	223	**226.5**	194
–m+d (dict)	220	226	225.5	186
+m–d (morph)	**230**	**233**	221.5	**203**
+m+d (both)	225	223.5†	217.5	184.5
GT total	*271*	*271*	*271*	*271*

Table 3: Ablation: True Positives across batch sizes for Gemini 3 Pro, varying whether morphological (m) or dictionary (d) data is provided within the prompt regarding the target homograph. B = batch size. (†Averaged over 2 runs per configuration.) Auxiliary context does not yield consistent improvement; at batch size 100, morphology shows the largest single-configuration effect (+10 TP), but this is based on single runs.

6 Error Analysis

The focused evaluation subset (15 files, 7,241 sentences) contains 271 confirmed errors, an error rate of 3.7%. Sections 6.1–6.2 analyze error patterns within this verified subset; §6.3–6.4 then discuss observations from the broader Challenge Set (332 files).

6.1 Syntactic Ambiguity: Construct vs. Adjective

Errors that were not found by most models are characterized by a high level of obscurity. Edge cases often involve the morphological distinction between construct-state nouns and attributive adjectives. In phrases like התפרצויות אלימות, the same consonantal string admits two competing analyses, distinguished only by the diacritization of the second word:

(a) הִתְפָּרְצוּיוֹת אַלִּימוּת
hitparṣuyot *'alimut*
outburst.F.PL violence.F.SG
'outbursts of violence' (N + N.CSTR)

(b) הִתְפָּרְצוּיוֹת אַלִּימוֹת
hitparṣuyot *'alimot*
outburst.F.PL violent.F.PL
'violent outbursts' (N + ADJ)

Determining which reading is intended is a high-level syntactic challenge that can perplex even expert human annotators.

6.2 Errors Undetectable by LLMs

Of the 271 gold-standard errors, **five** (2%) were never flagged by any of the 50 valid runs. Two are construct-vs.-standalone ambiguities related to the syntactic challenges described in §6.1; the remaining three are dormant תְּנָאִי singular-vs.-plural cases uncovered only by exhaustive manual review (§4.4), that no model and neither human expert detected. Three additional dormant errors were each detected by only 1–2 runs, and seven more errors were each flagged by only a single run. All fifteen of these highly challenging cases share a common profile: the error is not clearly wrong in isolation, and disambiguation requires pragmatic or collocational knowledge that goes beyond syntactic parsing.

6.3 World Knowledge in Error Detection

Beyond the focused 15-file subset, exploratory runs on the broader Challenge Set (332 files) revealed error patterns that require encyclopedic, rather than linguistic, knowledge. Our analysis highlights a clear advantage of LLMs over rule-based or purely

Homograph	Context	Original	Correction	Error Type
אהבה /ʔhbh/	...אמא שלי «אַהֲבָה» רק גבר אחד... 'My mother **loved** only one man...'	אַהֲבָה (N) 'love' /ʔa.ha.ˈva/	אָהֲבָה (V) 'she loved' /ʔa.ha.ˈva/	Part-of-Speech (Noun → Verb)
אלימות /ʔljmwt/	...ישירות ובלתי «אַלִּימוּת»... '...direct and non-**violent**'	אַלִּימוּת (N) 'violence' /ʔa.li.ˈmut/	אַלִּימוֹת (Adj) 'violent' /ʔa.li.ˈmot/	Syntax (Noun → Adj)
תנאי /tnʔj/	...«תְּנָאֵי» בסיסי לחברה... 'a basic **condition** for society...'	תְּנָאֵי (Pl.C) 'conditions of' /tna.ˈe/	תְּנַאי (Sg) 'condition' /t.ˈnai/	Inflection (Pl → Sg)
בחיי /bxjj/	...«בְּחַיֵּי» הקדושים... '**in the life of** the holy...'	בְּחַיַּי (Poss) 'in my life' /be.xa.ˈjai/	בְּחַיֵּי (Cst) 'in the life of' /be.xa.ˈjej/	Morphology (Poss → Cst)
אמרה /ʔmrh/	...«אַמָּרֶה» סטודמאייר הוא שחקן... '**Amar'e** Stoudemire is a player...'	אִמְרָה (N) 'saying' /ʔim.ˈra/	אָמָרֶה Amar'e (name)	World Knowledge (Named Entity)

Table 4: Representative errors identified by the auditing pipeline. *Original* is the incorrect diacritization in the source dataset; *Correction* is the LLM's suggestion. IPA transcriptions reflect Modern Hebrew pronunciation; note that the אהבה pair is **phonetically identical**; the distinction is purely orthographic (§2). N=Noun, V=Verb, Adj=Adjective, Pl=Plural, Pl.C=Plural Construct, Sg=Singular, Poss=Possessive, Cst=Construct.

morphological systems: the integration of **broad world knowledge**. This advantage is most apparent in the detection of transliterated names that function as homographs to common Hebrew words, a class of errors that is virtually invisible to traditional Hebrew diacritizers lacking encyclopedic knowledge.

Named Entities as Homographs. The homograph אמרה (/ʔmrh/; in Hebrew usually /amˈra/ "she said" or /imˈra/ "saying") frequently appears in sentences about people whose names are transliterated with the same consonantal form: athletes, politicians, and artists such as Amar'e Stoudemire, Emre Belözoglu, and Fadela Amara (see Table 4, last row). In the homograph challenge set they were diacritized as the Hebrew noun, producing contextually impossible readings. The LLM detected 36 such errors by recognizing the named entities from its training data and flagging the semantic incongruity.

In a complementary pattern, the homograph די (/dj/; in Hebrew usually "enough"/"quite") encompasses foreign names where the correct vowel depends on the actual pronunciation of the person's name: Ruby *Dee* and Chuck *D* require דִּי (/di/, with *hiriq*), while Daniel *Day*-Lewis requires דֵּי (/dej/, with *tsere*). Across two files, 29 such errors were confirmed, each requiring the model to identify the specific individual and the phonetics of the individual's name.

6.4 The אם Case Study: Asymmetric Precision

A major source of false positives is the consonantal string אם (/ʔm/), corresponding to אֵם (/ʔem/, "mother") or אִם (/ʔim/, "if").

In our analysis, the two אֵם files were outliers with notably low precision of the LLM runs (36.5% and 33.1% respectively), accounting for the bulk of spurious flags. The main issue is the Hebrew transliteration of the Latin letter *M*. The LLM correctly identified occurrences of "M" as semantically distinct from the Hebrew word for "mother," but suggested diacritizing it with *segol* (אֶם) rather than *tzere* (אֵם). The two vowels are phonetically identical in Modern Hebrew; the distinction is purely orthographic and governed by the Academy of Hebrew Language, which prescribes *tzere*. We note that the morphology list supplied to the model contained only the canonical Hebrew readings ("mother", "if") and not the Latin letter-name use, leaving the model without an in-prompt anchor for the prescribed form. While such an entry could in principle be added, our ablation (§5.2) shows that morphological context yields only marginal gains, and the deeper issue, such as disagreement among prescriptive norms regarding loan words, would persist regardless.

This case illustrates a fundamental challenge in auditing evaluation: the "correctness" of a diacritization depends on the prescriptive norm adopted. Under a strict Academy norm, the LLM's suggestion is wrong (FP), but the *segol* may be considered acceptable by other audiences. Despite the low precision, the model's ability to distinguish the foreign-letter meaning from the Hebrew word demonstrates genuine semantic understanding, even though its specific vowel suggestion does not match the prescribed form.

6.5 Summary of Detected Errors

In the focused subset (15 files, 7,241 sentences), the 271 confirmed errors correspond to an error rate of 3.7%. The union of all 50 focused runs detected 266 of these (98.2%), with the 5 remaining errors discussed in §6.2. Two full-corpus runs (Gemini 3 Pro, B200 –m–d, and B500, +m+d) covering all 332 files in the Challenge Set flagged a combined 4,731 instances outside the current GS. Our human domain expert has adjudicated each of these instances, thus producing a new and improved version of the EACL 2023 Challenge Set. We hereby release this new version of the Challenge Set, reflecting all errors confirmed to date.

6.6 Workflow Comparison: LLM-Aided vs. Manual Auditing

To validate our Gold Standard and quantify the practical value of LLM-based auditing, we compare two annotation-review workflows on the same 15 focused files (7,241 sentences, 271 confirmed errors):

- **Manual auditing:** Two independent native Hebrew speakers with graduate-level linguistics training (Expert 1 and Expert 2) each reviewed all 7,241 sentences sequentially (XLSX files, one row per sentence, target diacritization highlighted), without access to model output or each other's annotations. Each review required approximately 30 hours over one week; both experts were compensated at standard rates.
- **LLM-aided auditing:** Gemini 3 Pro (batch size 200, without morphological or dictionary data, averaged over 2 runs) processed the same sentences via API. Each run completed in ∼75 minutes of wall-clock time at a cost of ∼$7; the human reviewer then verified only the flagged findings (∼225 items per run).

Expert 1 achieved 94.9% precision and 62.4% recall; Expert 2 achieved 95.9% precision but only 42.8% recall. Both maintained high precision (>94%), yet their recall differed by 20 percentage points, highlighting the inherent inconsistency of manual error detection in sparse-target settings. In contrast, the LLM-aided workflow achieved 83.6% recall with 99.1% precision (Figure 2). The recall gaps of 21.2 pp (vs. Expert 1) and 40.8 pp (vs. Expert 2) are both statistically significant (McNemar's test, $p < 0.001$).

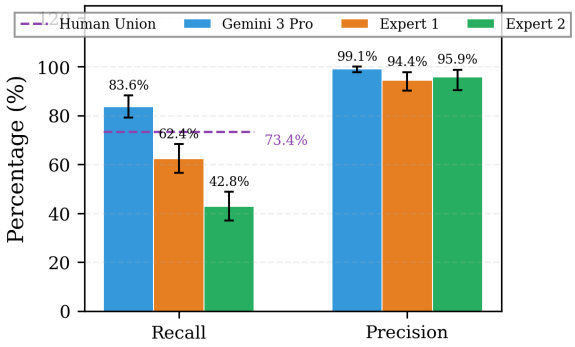

Figure 2: Workflow comparison: LLM-aided auditing (Gemini 3 Pro, batch size 200, no auxiliary context, averaged over 2 runs) vs. two independent human experts. Recall and precision are measured against our 271-error gold standard. The dotted line marks the union recall of both experts combined (73.4%), which still falls short of the LLM (83.6%). Error bars indicate 95% confidence intervals.

Even the **union** of both experts' findings reaches only 73.4% recall (199/271), still 10.2 pp below the LLM's 83.6%. This result demonstrates that the recall gap is not attributable to one annotator's idiosyncratic weakness: combining two qualified reviewers still fails to match the automated pipeline.

The gap (Figure 2) does not imply that the LLM "understands" Hebrew better than the experts; rather, it reflects the inherent difficulty of sustaining attention across thousands of mostly-correct sentences. At the observed error density of 3.7% (271/7,241), even a diligent reviewer encounters one genuine error roughly every 27 sentences. This is a sparse-target search task where human fatigue is a well-documented limitation (Swayamdipta et al., 2020).

The LLM-aided workflow shifts the reviewer's task from *open-ended search* (scanning 7,241 sentences for anomalies) to *targeted verification* (adjudicating ∼230 flagged items), reducing review effort by over 95% while improving recall. This supports the practical recommendation that LLM-based auditing is most valuable not as a replacement for human expertise, but as a **triage layer** that focuses human attention where it is most needed.

6.7 Inter-Annotator Reliability

On the binary sentence-level task (error or not error, Cohen's $\kappa = 0.571$ ("moderate"), but this low value is an artifact of extreme class imbalance: with only 3.7% error prevalence, raw agreement is 98.3%, a well-known prevalence paradox where κ underestimates true agreement (Feinstein and Cicchetti,

1990; Cicchetti and Feinstein, 1990).

Of the 271 confirmed errors, only 86 (31.7%) were found by both experts; 83 were found only by Expert 1, 30 only by Expert 2, and 72 (26.6%) by neither. This low overlap reinforces that human error detection in sparse-target settings depends less on linguistic competence than on sustained vigilance. In contrast, repeated LLM runs exhibit much lower variance: two Gemini 3 Pro runs at identical settings yielded 225 and 228 TP respectively (recall spread: 1.1 pp vs. 20 pp for humans).

7 Post-Hoc Quality Filters

Given the substantial gap in false-positive rates across models (Table 1), we investigated several post-hoc filtering strategies. Since our task is recall-centric (surfacing as many real errors as possible for human review), filtering is primarily useful when precision is low enough to impose a significant manual review burden.

7.1 Finding-Level Filtering

We investigated two post-hoc filters for reducing false positives. **Confidence filtering**, i.e. retaining only High-confidence findings, has negligible effect on Gemini 3 Pro (which already labels nearly all findings as High) but raises Flash precision from 39.7% to 64.0% at a steep recall cost (−15.5 pp). **Trivially invalid findings**, where the suggested correction is identical to the original diacritization (after Unicode normalization) or empty, account for 25.3% of Flash output but under 1% of Gemini Pro output; filtering them improves Flash precision with near-zero recall loss. For the best-performing model, neither filter has a meaningful effect, as its precision already exceeds 99%.

7.2 Multi-Run Union: Trading Precision for Recall

Since our workflow prioritizes recall, surfacing errors for human review, we analyze how combining the output of multiple runs affects coverage. Taking the **union** of findings from two or more runs increases recall, but the precision cost depends strongly on which runs are combined. Same-model Gemini Pro unions preserve high precision, while cross-model and all-run unions surface many more false positives.

Table 5 and Figure 3 present practical multi-run strategies that a user could employ without prior knowledge of which specific runs are optimal. A single Gemini 3 Pro run achieves up to 86.0% recall (batch size 100, with morphology data but without dictionary data). Adding a second run, whether a repeat of the same configuration, a different configuration of the same model, or a different model entirely, raises recall to ∼89%. A third run of the same model in a different configuration reaches 90.4% recall while maintaining 98.8% precision. With batch size 50, all three diversification strategies yield similar marginal gains (mean +9–11 TP across Gemini Pro B50 pairs), suggesting that the stochastic variation between runs is the primary driver of union gains, not the specific form of diversity.

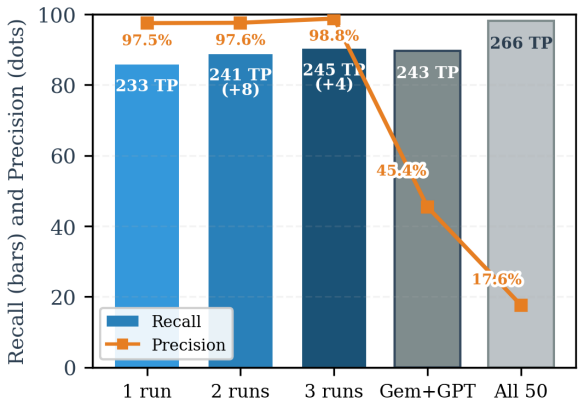

Figure 3: Diminishing returns of multi-run union. Bars show recall with True Positive counts; the orange line shows precision.

Strategy	TP	Recall	Prec.	F1
Single best run (Gem. Pro B100)	233	86.0%	97.5%	91.4%
2× same config (repeat)	241	88.9%	97.6%	93.1%
2× diff. config (same model)	241	88.9%	97.6%	93.1%
Gem. Pro + GPT-5.2 (B50)	243	89.7%	45.4%	60.3%
3× Gem. Pro (mixed B & config)	245	90.4%	98.8%	94.4%
All 50 runs (theoretical ceiling)	266	98.2%	17.6%	29.9%

Table 5: Recall, precision, and F1 from union of multiple runs (out of 271 Gold Standard errors). Same-model Gemini Pro unions raise recall to ∼89–90% while maintaining high precision; cross-model and all-run unions recover additional errors but introduce many more false positives. Five errors (including 3 dormant) were never detected by any of 50 runs. Gem. Pro = Gemini Pro; B = batch size.

The cross-model union with GPT-5.2 adds coverage beyond Gemini Pro, detecting 13 errors missed by the Gemini Pro B50 run and reaching 89.7% recall, but at a substantial precision cost (45.4%). The theoretical ceiling across all 50 runs remains 98.2%; however, its precision is only 17.6%, so

this setting should be interpreted as a recall upper bound rather than a practical review strategy. The 5 remaining errors consist of 2 ambiguous cases discussed in §6 plus 3 dormant errors that eluded all models and both human experts alike.

For practical deployment, we recommend **two to three runs of the same model** in different configurations as a cost-effective strategy: this raises recall to ∼89–90% while maintaining precision above 97%.

8 Future Directions

The auditing pipeline presented in this work operates on a focused confusion-set structure: each sentence contains a single pre-identified homograph that the model is asked to evaluate. This setup maximizes per-query precision and allows controlled evaluation, but it presupposes that the set of candidate homographs is known in advance. Three natural extensions suggest themselves: **running-text auditing**, in which errors are flagged in arbitrary text without pre-specified target words; **homograph diacritization**, in which LLMs are used to predict the correct diacritized form of a homograph in context rather than merely validating existing labels; and **full running-text diacritization**, applying LLMs to the general task of adding *nikud* to undiacritized Hebrew text, which subsumes both of the preceding extensions and is likely the hardest of the three. Preliminary single-run probes for all three extensions are detailed in Appendix H; they suggest that frontier LLMs retain high precision on running text but pay a substantial recall cost without the focusing effect of specific words, that diacritization-from-scratch is already strong on easier homograph cases, and that on full running-text diacritization a frontier LLM trails the specialist *Dicta* auto-diacritizer overall (92.0% vs. 95.4%) yet edges it at homograph positions. Full characterization of these extensions is left for future work.

9 Conclusion

We presented a human–AI auditing workflow for identifying Hebrew *nikud* errors, achieving high precision and recall with frontier LLMs. Our findings demonstrate that performance is primarily sensitive to batch size, whereas auxiliary morphological and dictionary context does not yield consistent improvements under zero-shot conditions, suggesting that frontier LLMs possess sufficient internal linguistic knowledge for this task. A workflow comparison with two independent human experts shows that manual recall varies widely (42.8%–62.4%) and that even the union of both experts' findings (73.4%) falls short of a single LLM run (83.6%). In contrast, our LLM-aided auditing pipeline reduces review effort by over 95%, shifting the human role from open-ended search to targeted verification, improving both recall and cost. Given the manageable inference costs and high effectiveness, this pipeline is economically viable for large-scale dataset maintenance. We release our Gold Standard dataset, including per-error difficulty annotations derived from cross-system detection rates, as well as the full corrected EACL 2023 Challenge Set to support further work within the Hebrew NLP community.

Limitations

While our auditing workflow demonstrates high precision and recall, several factors constrain the interpretation of these metrics.

First, the definition of a diacritization error is subject to **prescriptive linguistic norms**. As shown in the case of the consonantal string מא (serving as the letter *M*), a single decision, such as whether a loanword requires *tzere* or *segol*, can shift the measured precision of specific files by over 50 percentage points. Our evaluation assumes the conventions of the source dataset, yet we acknowledge that alternative normative decisions would lead to different performance profiles. This is further compounded by **orthographic noise** (typos) in the source corpus; both human and AI auditors often identify inconsistencies that stem from spelling errors rather than diacritization issues, which technically fall outside the scope of our task but impact the measured precision.

The second factor is the **dormant-error problem**. Although we tried to mitigate this through exhaustive manual review (§4.4), and through the validation of the Gold Standard via a second independent annotator (§6.7), the final decisions remain those of a single adjudicator. The fact that Expert 2's 5 disagreements were all resolved in favor of the existing ground truth provides evidence of adjudication robustness, but errors subtle enough to elude both annotators and all models may remain undiscovered; the 0.09% dormant rate we report is therefore itself an upper bound.

A related concern is **Gold Standard circularity**: since model runs contributed to the initial candidate pool used to construct the Gold Standard

(§4.4), errors that no model detects are structurally underrepresented in the denominator, potentially inflating measured recall. We mitigate this in three ways: (1) two independent human experts flagged errors without access to model output, contributing candidates that models missed; (2) exhaustive manual review of all 6,378 non-flagged sentences uncovered 6 additional dormant errors; and (3) the Gold Standard explicitly includes 5 errors that no model detected across all 50 runs, confirming that the denominator is not limited to model-detectable cases. Nevertheless, we acknowledge that if a class of errors exists that is systematically invisible to both humans and models, it would remain absent from the Gold Standard, and true recall would be lower than reported.

Finally, our reliance on **proprietary frontier LLMs** poses challenges for long-term reproducibility. API updates and shifting cost structures may affect the consistency of auditing results over time. In future work we plan to explore the viability of fine-tuned open-source models to ensure transparent and repeatable dataset maintenance.

10 Acknowledgments

This work has been funded by the Israel Science Foundation (grant No. 2617/22) and by the European Union (ERC, MiDRASH, Project No. 101071829; principal investigators: Avi Shmidman, Bar-Ilan University; Daniel Stökl, EPHE-PSL; Nachum Dershowitz, Tel Aviv University; and Judith Olszowy-Schlanger, EPHE-PSL), for which we are grateful. Views and opinions expressed are, however, those of the authors only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

References

Yaacov Choueka and Yoni Neeman. 1995. Nakdan-text,(an in-context text-vocalizer for modern hebrew). In *BISFAI-95, The Fifth Bar Ilan Symposium for Artificial Intelligence*.

Domenic V. Cicchetti and Alvan R. Feinstein. 1990. High agreement but low kappa: II. Resolving the paradoxes. *Journal of Clinical Epidemiology*, 43(6):551–558.

Alvan R. Feinstein and Domenic V. Cicchetti. 1990. High agreement but low kappa: I. The problems of two paradoxes. *Journal of Clinical Epidemiology*, 43(6):543–549.

Ya'akov Gal. 2002. An hmm approach to vowel restoration in arabic and hebrew. In *Proceedings of the ACL-02 workshop on Computational approaches to semitic languages*, pages 1–7. Association for Computational Linguistics.

Elazar Gershuni and Yuval Pinter. 2022. Restoring Hebrew diacritics without a dictionary. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1010–1018, Seattle, United States. Association for Computational Linguistics.

Google. 2026. Gemini 3 — API documentation. Accessed: 2026-02-16.

Curtis G. Northcutt, Lu Jiang, and Isaac L. Chuang. 2021. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 70:1373–1411.

Avi Shmidman, Cheyn Shmuel Shmidman, Dan Bareket, Moshe Koppel, and Reut Tsarfaty. 2023. Do pre-trained contextual language models distinguish between Hebrew homograph analyses? In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 831–843, Dubrovnik, Croatia. Association for Computational Linguistics.

Avi Shmidman, Shaltiel Shmidman, Amir Eyal, and Moshe Koppel. 2020. Nakdan: Professional Hebrew diacritizer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 197–203. Association for Computational Linguistics.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, Online. Association for Computational Linguistics.

A Full-Sentence Examples from the Dataset

This appendix presents four representative sentences from the focused evaluation set, along with English translations.

Example 1: Part-of-Speech (Noun → Verb).

> אמא שלי «אַהֲבָה» רק גבר אחד מכל מי שהיה לי והוא כרגע בן 24 בדרך לחתונה והיא עדיין מחכה שנחזור.

Original: אַהֲבָה (*'ahava*, noun "love")
Correction: אָהֲבָה (*'ahava*, verb 3FS.PST "she loved")
Translation: "My mother **loved** only one man out of all those I had, and he is currently 24, on his way to a wedding, and she is still waiting for us to come back."
Notes: The two forms are phonetically identical in Modern Hebrew; only the contextual reading (subject-verb-object) reveals the verbal interpretation.

Example 2: Syntax (Adjective → Construct Noun).

> הוא מוכר למשטרה מעבירות «אַלִּימוֹת» הקשורות לפעילותו בימין הקיצוני.

Original: אַלִּימוֹת (*'alimot*, adjective F.PL "violent")
Correction: אַלִּימוּת (*'alimut*, noun F.SG "violence")
Translation: "He is known to the police for **violence** offenses connected to his activity on the far right."
Notes: The standard legal term is the construct compound "offenses of violence" (N+N), not the attributive "violent offenses" (N+Adj).

Example 3: Inflection (Plural Construct → Singular Absolute).

> אבל אחדות נגד אויב משותף זה «תְּנָאֵי» בסיסי לחברה, אבל זה לא מספיק...

Original: תְּנָאֵי (*tna'ei*, plural construct "conditions of")
Correction: תְּנַאי (*tnai*, singular absolute "a condition")
Translation: "But unity against a common enemy is a basic **condition** for society, but it is not enough..."
Notes: The agreement cues, singular copula זה and singular adjective בסיסי, require a singular noun.

Example 4: Morphology (Construct → Possessive Suffix).

> וכך, גם אם ניסיתי שלא לערב את עבודתי «בְּחַיֵּי» האישיים, נשזר סיפור חוויותיי המקצועיות באופן טבעי בתולדות חיי.

Original: בְּחַיֵּי (*be-ḥayyei*, prep. + construct plural "in the lives of")
Correction: בְּחַיַּי (*be-ḥayyai*, prep. + plural with 1SG possessive "in my life")
Translation: "Even though I tried not to involve my work **in my** personal **life**, the story of my professional experiences became naturally woven into my life history."

B System Prompt

The models were prompted using a linguist-persona template designed to prioritize recall over precision (Appendix D). Output was constrained to a structured JSON schema (Appendix E).

C API Request Structure

This appendix presents a concise overarching view of the request structure; the full text of the prompt is given below in Appendix D.

— System Prompt (Appendix D) —
You are an expert linguist specializing in Hebrew grammar, morphology, syntax, and particularly *nikud*…

— Morphological Data (optional) —
אָהֲבָה: Lemma=אהב, `<Verb Bareinfinitive Paal>`
אָהֲבָה: Lemma=אהב, `<Verb Fem Sg P3 Past Paal>`
אַהֲבָה: Lemma=אַהֲבָה, `<Noun Fem Sg Absolute>`
...(31 diacritized forms total)

— Dictionary Definitions (optional) —
– אַהֲבָה
רגש עמוק של משיכה לאדם או לדבר
– אהב (Qal)
נמשך ברגש עמוק לאדם או לדבר
...(3 entries total)

— Sentences (batch of 50–500) —
S1: הוא חוצה את… שאשתו «אָהֲבָה» במיוחד…
S2: היא «אָהֲבָה» את הבתולה מריה…
…

Figure 4: Structure of a single API request, illustrated with the homograph אהבה. Each batch includes the system prompt, optional morphological and dictionary context, and 50–500 sentences. The target word is marked with guillemets.

D Prompt Text

The following is the complete system prompt sent to each model. The prompt was authored in Markdown; formatting conventions (`**bold**`→**bold**, `*italic*`→*italic*, `` `code` ``→`code`) are rendered here as their visual equivalents.

> You are an expert linguist specializing in Hebrew grammar, morphology, syntax, and particularly *nikud* (vocalization). Your task is to act as a quality assurance agent for Hebrew sentences where one specific word form has been vocalized.
>
> **Task Context:**
> You will receive a list of sentences. In **all** these sentences, the same string of letters has been vocalized in the **exact same way** (marked with `` `«...»` ``). Your goal is to validate if this specific vocalization is contextually correct for each sentence. Since Hebrew homographs (words with same letters but different vowels/meanings) are common, the provided vocalization might be correct for some sentences but incorrect for others (where a different vocalization of the same letters is required).
>
> **Key Objective:**
> Achieve high recall of actual *nikud* errors. It is acceptable if up to 20% of your reported errors are false positives. If you are uncertain but suspect an error based on context, report it with `Medium` or `Low` confidence.
>
> **Input Structure:**
>
> 1. **Morphological Data:** This data lists unvocalized words, their possible *nikud* forms, and corresponding lemma/grammatical information (e.g., `<Lemma=..., Gram=...>`). This data is extensive but not necessarily complete (especially for loanwords or rare forms) and may, very rarely, contain errors. Use this list as a reference for possible alternative vocalizations if the provided one seems wrong.
> 2. **Sentences:** A list of sentences (e.g., S1, S2, ...), each with the target word marked by `` `«...»` ``.
>
> **Processing Instructions for Each Sentence:**
>
> 1. Identify the unvocalized letters of the word within `` `«...»` ``.
> 2. Carefully analyze the full sentence context surrounding the marked word. Pay attention to syntax, semantics, and grammatical agreement.
> 3. **Validate the Provided Nikud:**
>   - Does the specific form provided in `` `«...»` `` fit the sentence grammatically and semantically?
>   - **If YES:** Move to the next sentence.
>   - **If NO:** This is an error. You must determine the correct vocalization.
>   – Consult the **Morphological Data**: Does another form in the list fit the context perfectly?
>   – If the correct form is not in the data (rare), derive it based on your linguistic knowledge.
> 4. **Contextual Validation – Specific Checks:**
>   - **Part of Speech & Tense:** This is the most common error in this task. For example, if the input is `` `«אוֹכֵל»` `` (Present/Participle), but the sentence says "מחר אני...", the context requires Future tense (`` `«אֹכַל»` `` or `` `«אוּכַל»` ``).
>   - **Gender and Number Agreement:** Does the *nikud* agree with other elements in the sentence?
>   - **Construct State (*Smichut*) vs. Absolute State:** Is the word part of a *smichut*? Is the vowel pattern correct for that state?
>   - **Definiteness with Prefixes (ל, ב, ב):** Check if the vowel under the prefix indicates the correct definiteness (e.g., `` `בְּסֵפֶר` `` vs `` `בַּסֵּפֶר` ``).
>   - **Loanwords and Proper Nouns:** Use world knowledge for names or foreign terms not fully covered in morphology.
>   - **Original orthography:** If the sentence includes `[[[original: ...]]]`, consider the spelling changes when determining the correct vocalization.
> 5. **What NOT to Flag as Errors:**
>   - **Shva vs. Hataf:** Do NOT report differences like `` `מַעְיָן` `` vs `` `מַעֲיָן` `` unless strictly necessary.
>   - **Minor Stylistic Variants:** Acceptable variants are not errors.
>   - **The Unvocalized Word Itself:** Assume the base letters are correct.
> 6. **Reporting Errors:**
>   - If the provided *nikud* is contextually incorrect:
>   – Populate the JSON object.
>   – In `suggested_correct_nikud_form`, provide the correct vocalization (preferably from the Morphological Data if a match exists).
>   – In `reason_for_error`, explain the mismatch (e.g., "Context implies Future tense 1st person, but word is vocalized as Present tense").
>
> **Output Format:**
> You must output a JSON object strictly conforming to the following schema. The top-level object should have a single key: `"errors"`, which is an array of error objects. If no errors are found, this array should be empty.

E Output JSON Schema

Each model was instructed to return a structured JSON object conforming to the following schema.

```
{
  "type": "object",
  "properties": {
    "errors": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "sentence_id": {
            "type": "string",
            "description": "Sentence ID
              (e.g., S1, S2)"
          },
          "full_sentence": {
            "type": "string",
```

```
            "description": "Complete Hebrew
              sentence as provided"
          },
          "marked_text_unvocalized": {
            "type": "string",
            "description": "Unvocalized letters
              of the marked word"
          },
          "provided_nikud_form": {
            "type": "string",
            "description": "Vocalized word as
              it appeared in input"
          },
          "suggested_correct_nikud_form": {
            "type": "string",
            "description": "Suggested correct
              vocalized form"
          },
          "reason_for_error": {
            "type": "string",
            "description": "Explanation of why
              the nikud is incorrect"
          },
          "confidence_in_error": {
            "type": "string",
            "enum": ["High","Medium","Low"]
          },
          "morphological_data_feedback": {
            "type": "string",
            "description": "Optional feedback
              on morphological data gaps"
          }
        },
        "required": [
          "sentence_id",
          "full_sentence",
          "marked_text_unvocalized",
          "provided_nikud_form",
          "reason_for_error",
          "confidence_in_error"
        ]
      }
    }
  },
  "required": ["errors"]
}
```


F Recall on Injected Errors

To validate recall measurement under controlled conditions, we constructed contaminated files by injecting synthetic errors into cleaned homograph files. Starting from 5 files spanning 3 homograph groups (2,360 sentences total), we swapped 10% of homograph tokens with a competing diacritized alternative selected at random, producing 308 injected errors (Table 7).

Model	Prec.	Rec.
Gemini 3 Pro	100.0	78.9–100.0
GPT-5.2	99.3–99.7	94.5–98.4
Gemini 3 Flash	96.8–97.7	82.5–96.8

Table 6: Contaminated-set results (5 files, 308 injected errors).

G Model Identifiers and Configuration

Table 8 lists the API identifiers for each model evaluated in our study. All models were accessed via the OpenRouter API between January 25 and February 16, 2026; exact version snapshots are not available through the API. The prompt was not individually optimized per model architecture; performance differences may therefore partly reflect prompt compatibility rather than intrinsic model capability.

Homograph file	Sentences	Injected
אֲחֹרוֹת_1	489	46
אֲחֹרוֹת_2	514	64
אם_1	497	44
אם_2	508	59
אִמְרָה	352	95
Total	2,360	308

Table 7: Contaminated-set composition: sentences per file and injected errors (10% swap rate). Suffixes _1/_2 denote sub-files partitioned from the original dataset for batch processing.

Model	API Identifier
Gemini 3 Pro	`gemini-3.0-pro-preview`
Gemini 3 Flash	`gemini-3.0-flash-preview`
GPT-5.2	`gpt-5.2`
Claude Opus 4.5/4.6	`claude-opus-4-*`

Table 8: Model API identifiers. All models accessed via OpenRouter; exact version snapshots are not available through the API.

Token Usage. Table 9 reports the total input and output token counts for each model–batch-size combination on the focused evaluation set. Input counts include the system prompt repeated once per batch; larger batch sizes amortize this overhead across more sentences, resulting in lower total input tokens. These counts allow readers to recompute costs under updated pricing.

Model	Batch	Input	Output	Total
Gemini 3 Pro	50	663,600	945,103	1,608,703†
	200	526,530	491,989	1,018,520†
	500	485,055	283,539	768,594†
GPT-5.2	50	608,436	326,395	934,831
	200	451,248	130,335	581,583
Claude Opus 4	50	812,421	60,935	873,356
	200	693,808	68,053	761,861
	500	680,159	24,277	704,436
Gemini 3 Flash	50	616,893	172,923	789,816
	200	492,533	65,929	558,462
	500	478,431	27,571	506,002

Table 9: Token usage per run on the focused evaluation set (7,241 sentences). Input and output counts are totals across all API calls in a single run. †Averaged over multiple configurations. These counts allow cost recomputation under updated pricing schedules.

H Preliminary Probes

This appendix reports three single-run probe studies that empirically anchor extensions discussed in the Future Directions section. We report these as single-run probe numbers; full characterization is left for future work. The main experiments in this paper were conducted in January–February 2026 with the `gemini-3.0-pro-preview` snapshot, which has since been retired; the probes below were run in May 2026 using its closest currently-available successor, `gemini-3.1-pro-preview`.

H.1 Running-Text Auditing

We ran a single-run probe on 77 sentences drawn from the focused corpus (20 each from the Easy/Medium/Hard error tiers and 20 randomly-sampled Correct sentences, with three removed for duplication or off-domain content), stripped of guillemets and diacritic marks, and re-diacritized by *Dicta Nakdan* (Shmidman et al., 2020) as an off-the-shelf auto-diacritized baseline. The first author manually corrected Dicta Nakdan's output to produce a blind reference; one additional Gemini-flagged error missed in the blind pass was promoted post-hoc, giving 71 blind and 72 adjudicated gold errors. Dicta Nakdan resolved most originally-marked homographs without guillemets (only $\sim$10 of the original homograph positions remained wrong post-diacritization), but introduced 62 new errors elsewhere in the text. Using `gemini-3.1-pro-preview` with morph and dictionary headers disabled, single-run inference yields whole-text $P = 100\%$, $R = 41.7\%$, $F_1 = 0.59$ over 30 flagged errors and 72 gold; correction-exact-match among true positives is 76.7%. Stratified, recall is 40% (4/10) at original homograph positions, a substantial drop from the focused pipeline's 83.6%, and 41.9% (26/62) on Nakdan-introduced errors elsewhere, suggesting that the recall cost of removing guillemets is comparable in both regions of the text. Failure modes cluster around foreign-name transliteration, definiteness on ב/ל prefixes, and recurring focused-corpus homographs flagged in some sentences but missed at other running-text occurrences (תְּנָאִי, בְּחַיֵּי). Two caveats temper this picture: the gold is human-adjudicated against model output and is not an exhaustive independent benchmark, and the homograph-position subset ($n = 10$) and per-tier counts are too small for reliable per-tier conclusions.

H.2 Homograph Diacritization

We ran a single-run probe on 542 sentences (all 271 confirmed errors plus a per-file matched Correct sample, randomly ordered across mixed-confusion-set batches) using `gemini-3.1-pro-preview` at the same B200 batching budget. Without morph context, the model produces the gold diacritized form on 88.6% of the 271 error positions and preserves the published form on 98.5% of matched correct positions, with 100% consonant preservation and no missing predictions. Two caveats temper this picture: a batch-level union morph header, which substitutes for the per-file header used in our auditing setup (§5.2) given the mixed batches, in fact *hurts* error-correction by 10.7 pp, with the regression concentrated in prefix-letter homographs (e.g., בְּ vs. בַּ in בחיי, בהמשך) where the union header appears to bias the model away from definite-article-bearing forms; and the Hard difficulty band stalls at 51.2% in both configurations, marking cases where context-dependent decisions remain robustly hard.

H.3 Full Running-Text Diacritization

We ran a single-run probe on the same 80 sentences used in §H.1, asking `gemini-3.1-pro-preview` (without morph or dictionary headers) to diacritize each token from scratch as plene undiacritized Hebrew text, and compared the result token-by-token to the human-adjudicated gold and to *Dicta Nakdan* as an off-the-shelf auto-diacritized baseline. Gemini reaches 92.0% across $n = 1{,}551$ active tokens versus Dicta Nakdan's 95.4%; at the 76 original homograph positions Gemini's 88.2% slightly edges Dicta Nakdan's 86.8%, and on the Hard tier Gemini retains a substantial lead (81.8% vs. 68.2%, $n = 22$), the same context-dependent pattern as in §H.2. Two caveats temper this picture: the 80-sentence, 1,551-token sample is small and shares provenance with §H.1, so token-level scores are not an independent benchmark of running-text *nikkud*; and the Hard-tier subset ($n = 22$) is suggestive rather than conclusive. A larger, independently-curated full-text gold, evaluated blind to both systems, would be needed for a robust Nakdan-vs.-LLM verdict.