Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Curriculum Learning for Structured Extraction from Historical Documents]{Curriculum Learning for Structured Extraction from Historical Documents: Fine-tuning Vision-Language Models on Dutch Housing Cards

Melvin Wevers

melvin.wevers@uva.nl

University of Amsterdam

Research Article

Keywords:

Posted Date: November 27th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11621135/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Curriculum Learning for Structured Extraction from Historical Documents: Fine-tuning Vision-Language Models on Dutch Housing Cards

Melvin Wevers[1*]

[1*]Department of History, University of Amsterdam, Amsterdam, The Netherlands.

Corresponding author(s). E-mail(s): melvin.wevers@uva.nl;

Abstract

Purpose: Historical administrative collections hold structured information that manual transcription cannot unlock at scale. We investigate whether a small, locally deployable vision-language model can be specialized to extract structured records from approximately 475 thousand housing cards (*woningkaarten*, 1954–1989) of the Amsterdam City Archives more accurately than frontier commercial models.

Methods: We fine-tune Qwen2-VL-7B with a two-stage curriculum: broad pre-training on ∼11K noisy automated extractions from multiple vision-language models, followed by refinement on 560 human-validated cards (of 700 corrected, with 140 held out). Extraction is segmented, treating the card header and the entry table as separate prediction tasks. We evaluate on a held-out 140-card split against GPT-4o, GPT-4o-mini, and Qwen 2.5 VL 32B, under identical prompts and inputs.

Results: The curriculum-trained 7B model attains the highest accuracy of all systems (entity F1 0.989, versus 0.967 for GPT-4o and 0.933 for the Qwen 32B model) and leads on every extracted field, while running on a single GPU at a compute cost of €0.17 per 1,000 cards, with no per-request API charge. The curriculum's noisy pre-training acts as a regularizer against hallucination: a gold-only model fabricates plausible but fake residents on empty cards (69 entries), the curriculum model almost never does (1), which is a distinction critical for archival use.

Conclusion: Curriculum fine-tuning turns a modest open model into the most accurate extractor for a fixed historical collection, cheaply and privately. The 700 annotations will be made available for scientific use under access conditions agreed with the source archive.

Keywords: Vision-Language Models, Curriculum Learning, Structured Information Extraction, Historical Document Processing, Fine-tuning, Document Understanding

1 Introduction

Historical documents contain valuable structured information that remains locked in image format. The Amsterdam City Archives holds 473,593 housing cards (*woningkaarten*) documenting residential occupancy from 1954 to 1989, containing structured data about residents, addresses, arrival and departure dates, and demographic information. Manual transcription of this collection would require thousands of hours of (expert) labor. Automating the extraction of structured records from such sources has clear precedent, mirroring earlier structured-extraction efforts on historical administrative documents [1].

The extraction target is analytic rather than archival: the cards feed a research database of residency spells used to study residential turnover, suburbanization, and mobility in postwar Amsterdam. This downstream purpose fixes which fields matter (dates, names, addresses, and destinations) and which trade-offs are acceptable: a misread date corrupts a turnover rate, while a misread name does not; whereas for record linkage, the sensitivities are reversed.[1] Hallucination is different. It is the one failure mode whose severity does not depend on which of these settings the extraction serves, and the archival case is, if anything, the graver of the two: in aggregate analysis, a handful of fabricated rows barely moves a rate, but when the same cards are published online and consulted one at a time, for example by historians or by families tracing relatives, an invented resident is read directly as fact, with no aggregate to absorb it. The asymmetry with omission holds in both settings. A missing entry leaves a visible gap; a fabricated one asserts, with the same confidence as the rest of the transcription, that a named and often still living person lived at an address they never occupied. That is a claim about a person, not merely a defect in a dataset.

Recent vision-language models (VLMs) have demonstrated strong performance on document understanding tasks, but their application to historical documents presents distinct challenges: (1) **domain-specific formatting**: housing cards follow a standardized but complex tabular structure with specific field semantics; (2) **transcription difficulties**: typewritten text from the 1950s to 1980s with varying quality, struck-through text, edits, handwritten annotations, and aged paper degradation; (3) **semantic understanding**: continuation lines, name overflow across rows, and context-dependent field interpretation; and (4) **cost and accessibility**: commercial API-based models incur substantial costs for large-scale extraction, while on-premises deployment requires specialized models.

We address these challenges through curriculum learning: a two-stage fine-tuning approach that first exposes the model to diverse but noisy examples from multiple automated extraction sources, then refines performance on human-validated data. Our

[1] Linking names with minor transcription errors, given associated constraints such as birth year, address, and dates of arrival and departure, is more feasible than linking on dates.

contributions are: (1) **curriculum learning for historical structured extraction**: a two-stage training methodology that leverages noisy automated extractions (Stage 1), followed by gold-standard refinement (Stage 2); (2) **a segmented extraction strategy**: separate prediction tasks for header fields and table entries that reduce context length and improve accuracy; (3) **a human-validated dataset**: 700 corrected housing-card annotations with detailed field-level ground truth; (4) **a comprehensive baseline comparison**: evaluation against GPT-4o, GPT-4o-mini, and Qwen 2.5 VL 32B; and (5) **practical deployment**: a 7B-parameter model LoRA-fine-tuned on a single 40 GB GPU that is deployable locally, with which we transcribed the complete archive.

2 Related Work

2.1 Vision-Language Models for Document Understanding

Pre-trained vision-language models such as GPT-4o [2], Qwen-VL [3], and LLaVA [4] have shown strong zero-shot performance on document understanding tasks. However, domain adaptation through fine-tuning remains essential for specialized historical document processing, where formatting conventions, vocabulary, and transcription challenges differ from modern web-scraped training data.

2.2 Curriculum Learning

Curriculum learning [5] proposes training models on progressively harder examples, mimicking human learning. Recent work has applied curriculum learning to vision tasks [6] and language models [7]. Our approach adapts curriculum learning to document understanding by using noisy automated extractions as "easier" diverse examples before refining on challenging human-validated cases.

2.3 Historical Document Processing and Digital Humanities

Prior work on historical document processing has focused on handwritten text recognition [8] and layout analysis [9], and recent VLM-based approaches have shown promise for end-to-end document understanding [10]. Within the digital humanities and cultural-heritage community, structured extraction from historical sources has become a substantial research program in its own right. Handwritten text recognition at scale, driven largely by the Transkribus platform, has moved from proof-of-concept to routine archival practice [11], and systematic reviews now map how the technology is applied across heritage contexts [12], while critical work interrogates what large-scale automated transcription does to our access to and knowledge of the past [13]. Closest to our task is the extraction of structured data from semi-structured administrative records: Lafia et al. combine OCR and deep learning to digitize and parse G.I. Bill mortgage records [1], a problem that mirrors our housing cards in its tabular regularity, degraded typescript, and archival provenance. At the collection level, Ali et al. assemble a computer-vision and machine-learning pipeline (document layout analysis, image similarity, and named entity recognition) to enrich the metadata and improve the searchability of the digitized newspapers of the Royal Library of Belgium [14]. On

the language side, named entity recognition and linking in historical documents have been benchmarked through shared tasks such as HIPE-2022 [15], and the arrival of general-purpose LLMs has prompted direct and often cautionary assessments of their zero-shot performance on historical material [16].

Taken together, this literature underscores a theme that also motivates our evaluation design: reported accuracy on small, heterogeneous samples is difficult to compare across systems, and the epistemic stakes of these pipelines, once collections are rendered searchable at scale, are not merely technical but bound up with the constructed, non-neutral character of the digital archive [17]. Few of these studies, however, address curriculum learning for a specialized historical collection or provide a held-out, baseline-controlled comparison of specialized against frontier models, which is the gap this paper addresses.

3 Dataset

3.1 Source Material

The Amsterdam City Archives housing-card collection contains 473,593 scanned images of standardized forms documenting residential occupancy. Cards were scanned at high resolution (typically $\sim$2600 $\times$ 3800 pixels) and distributed as JPEG 2000 via the archive's IIIF image service. Each card records:

- **Header section**: street name, house number, floor or position qualifier (*house appendix*, e.g. "I hoog"), district code (*codenummer*), neighborhood code (*buurtletter*);
- **Entry table**: chronological records of successive heads of household, with fields: arrival date (*datum*), head of household (*gezinshoofd*), year of birth (*jaar*), departure date (*datum vertrek*), destination (*waarheen*), remarks (*opmerkingen*);
- **Follow-up entries**: a second table imaging the reverse of the card, which records co-residents (*inwonenden*) rather than heads of household, under a correspondingly different set of columns.

Both tables are typewritten. Handwriting enters as a secondary layer of clerical annotation overlaid on the typed record: marginal notes in the *Aantekeningen* field, corrections, strike-throughs marking departures, and tick marks in the year strip along the lower edge. Cards also carry administrative fields that we do not target for extraction, among them male/female tallies ($M\,/\,V$), a voting-district number (*stemdistrict*), and references to continuation cards (*volgkaarten*).

3.2 Automated Extraction Pool and Gold Standard

We created a human-validated dataset through a multi-stage annotation process. First, we automatically extracted card information using vision-language models served through our institution's LLM proxy. Because the physical collection is organized alphabetically by street, bulk extraction jobs were dispatched in alphabetical street ranges (uninterrupted series of archival boxes) rather than as random samples; a range

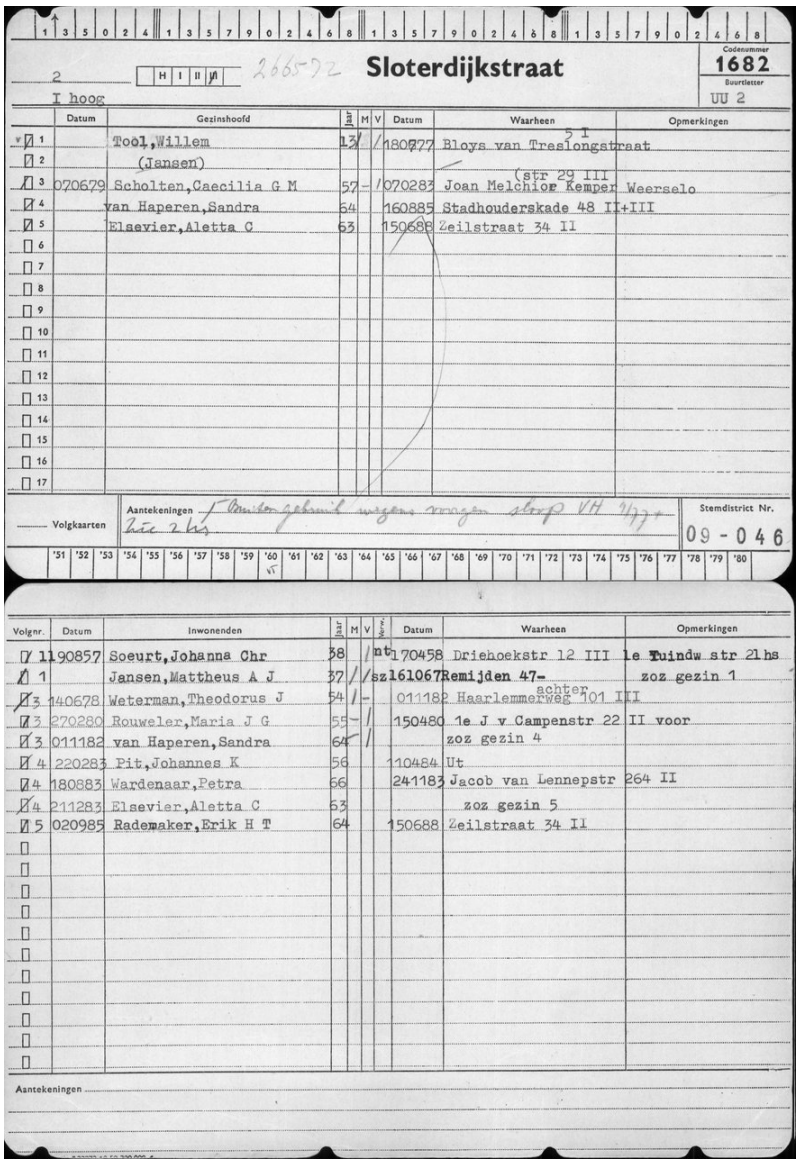

Fig. 1 A housing card from the 1954–1989 series. A single scan images both sides of the card: the front (upper half) carries the typed header identifying the dwelling and the main-entries table recording successive heads of household with arrival and departure dates and destinations; the reverse (lower half) carries the follow-up table of co-residents. Handwriting appears throughout as annotation over the typed record. Dashed boxes mark the two crop regions processed separately by our segmented extraction (§4.1): the header (3.2–9.0% of scan height) and the main-entries table (9.0–49.8%). The lower half is not processed (§8.4).

corresponds to a set of streets, and all cards for the addresses on them were downloaded. This yielded deep coverage of a few boxes but narrow coverage of the corpus:

the eight ranges extracted span only eight inventory numbers. To broaden the gold standard beyond these, we supplemented them with a random sample of 315 cards drawn corpus-wide, which spans 208 boxes. The two sources together give the 700-card corrected set its coverage of 315 streets across 216 boxes: the range-based cards supply depth within streets (including consecutive house numbers on the same street), and the random sample supplies breadth across the alphabet, scanning batches, and card variants.

A Qwen 2.5 VL 32B model served through the national infrastructure SURF processed all eight ranges; GPT-4o was additionally run over four of them and GPT-4o-mini over the remaining four, and GPT-4o-mini also extracted a separate corpus-wide random batch of ∼1,000 cards. The composition is practical rather than principled: the SURF endpoint is cheaply available for Dutch academic use and could sustain the volume, so it served as the primary annotator on the ranges, and the random batch broadened layout coverage beyond them. After discarding failed or incomplete extractions and deduplicating by card ID, keeping the first source encountered, we obtained 11,088 usable cards forming the Stage 1 noisy training pool: 10,195 from the 32B model and 893 from GPT-4o-mini, the latter being precisely its corpus-wide random batch, which no other model processed. Because SURF covered every range card, both API models' *range* extractions drop out of the pool entirely, which leaves GPT-4o available as an evaluation baseline that never contributed training data (§5.1); the same holds for GPT-4o-mini, whose 893 pool cards are disjoint from the 700-card corrected set. We return to the cost of that redundancy in §8.2. Second, we manually corrected a sample of 700 cards.[2] Expert annotators reviewed automated extractions and corrected errors using a custom annotation interface. The 700 corrected cards span 315 distinct street names across 216 archival boxes (*inventarisnummers*), giving broad coverage across the corpus, though the distribution is uneven: the single largest box (inv. 197, spanning "Gouden Leeuw — Govert Flinckstraat" alphabetically) contributes 196 cards. Corrections focused on date formatting (6-digit DDMMYY strings), name continuation across lines, correct street names and house numbers, and null handling for missing fields. The annotation covered only the header and main-entry (first-page) segments; the second-page follow-up table is out of scope for this work. That table contains cross-references to family members, but the accuracy of the original data there is uneven.

We split the corrected data, using a fixed seed, into a training set (560 cards, 80%) and a held-out test set (140 cards, 20%). All models (fine-tuned and baseline) are evaluated on the same 140 held-out cards, which no fine-tuned model was trained on, ensuring a leakage-free comparison.[3] We report single-split results (a fixed 80/20 split at seed 42) rather than cross-validation: each evaluation run requires re-extracting every baseline (including paid API calls) over the test fold, making k-fold prohibitively

[2] The corrected set was assembled in two waves. From an early 150-card snapshot, we built a 400-card annotation work package by adding 250 cards drawn from the automated extractions (201 GPT-4o, 49 SURF); 385 of those 400 cards are in the final set. Because that package was drawn only from the eight extracted ranges, it spans just eight boxes. The corpus-wide coverage reported here comes from the later random supplement.

[3] Stage 1's noisy pool is drawn from the same street ranges as the gold standard, so 70 of the 140 held-out cards appear in it. They enter Stage 1 paired with the source model's *uncorrected* extraction, not the human-validated target. Stage 2 and all evaluations use the fixed seed 42 split exclusively, so no fine-tuned model ever sees a held-out gold label.

expensive; the 140-card test set spans 82 distinct streets, giving reasonable coverage for a single held-out estimate.

3.3 Data Format

Each annotation contains structured JSON. The header carries five fields: `street`, `house_number`, `house_appendix` (floor/position qualifier, split from the raw house number; see §7.2), `codenummer` (a 4-digit code, stored as a zero-padded string), and `buurtletter`:

```
{
  "header": {
    "street": "Govert Flinckstraat",
    "house_number": "123A",
    "house_appendix": "II hoog",
    "codenummer": "1452",
    "buurtletter": "K2"
  },
  "main_entries": [
    {
      "datum": "250566",
      "gezinshoofd": "Johannes van der Berg-Smit",
      "year_of_birth": "21",
      "datum_vertrek": "101270",
      "waarheen": "Rotterdam",
      "remarks": "Met gezin"
    }
  ],
  "follow_up_entries": []
}
```

The cards also contain per-entry male/female occupant counts (the M and V columns). We deliberately exclude these from the extraction schema and evaluation: they are frequently blank or ambiguous on the scans, and they are not needed for the residency-turnover analyses this dataset targets (§1). The scope is thus the header plus the main-entries residency records.

4 Methodology

4.1 Segmented Extraction Approach

Rather than processing entire housing cards ($\sim 2600 \times 3800$ pixels) in a single pass, we employ a two-segment strategy (Fig. 1). Crop regions are defined as fractions of card height, not fixed pixel offsets, because scan heights vary across the collection. The header's crop region spans $y \in [0.0316, 0.0903]$ ($\approx$ pixels 120–345 on a $\sim$3800 px card). The main entries's crop region covers $y \in [0.0903, 0.4978]$ ($\approx$ pixels 345–1900). This

segmentation provides two benefits: (1) reduced context length: smaller crops require less VRAM and allow for higher-resolution processing; and (2) task-specific prompts: we can use tailored instructions for header and main entries extraction. This focused attention on relevant image regions improves accuracy, which we quantify directly against single-pass extraction in §8.3.

4.2 Curriculum Learning Strategy

We implement a two-stage curriculum.

4.2.1 Stage 1: Noisy Pre-training (Broad Coverage)

In this stage, the objective is to learn diverse document layouts and transcription patterns from large-scale automated extractions. The training data consist of the 11,088 extracted cards described in §3.2, deduplicated by card ID so that each card contributes at most once. We deduplicate rather than keep every model's noisy label for a card because retaining conflicting labels for the same image would inject contradictory supervision; the goal of Stage 1 is broad layout and format coverage rather than per-card consensus, which Stage 2's gold refinement supplies. The rationale is that noisy data provides broad coverage of document variation (different streets, scanning conditions, and formatting conventions) without expensive human annotation, while a low epoch count prevents the model from memorizing the errors in the noisy input data.[4]

4.2.2 Stage 2: Gold Standard Refinement (Precision Correction)

The objective of the second stage is to correct systematic errors and improve field-level accuracy on human-validated data. The data source is the 700 human-corrected cards, with 140 held out for testing, yielding 1,120 training samples (560 cards $\times$ 2 segments). Starting from the Stage 1 checkpoint, Stage 2 performs precision refinement on verified ground truth; the lower learning rate prevents forgetting the broad document coverage acquired in Stage 1.[5]

4.3 Model Architecture and Training

The base model is Qwen2-VL-7B-Instruct [18], fine-tuned with LoRA [19] on ∼161M trainable parameters. Both training stages were conducted on a single NVIDIA A100 (40 GB) using a single-device training loop with no distributed or model-parallel machinery. Together, the two stages required approximately 6 GPU-hours: 5 for Stage 1 and 1 for Stage 2. The hyperparameters include gradient accumulation of 16 steps; optimizer AdamW with a weight decay of 0.01; warmup ratio of 0.05; images resized to 768×768 from the crops; and a maximum sequence length of 2816 tokens, which ensures no entry is truncated (§8.5). That sequence budget is what fixes the hardware floor: at 2816 tokens, a training step on the longest card peaks at ∼23.4 GB,

[4] Configuration: epochs: 2 (low, to avoid overfitting to errors); learning rate: 2×10^{-5}; LoRA parameters: $r = 64$, $\alpha = 128$, dropout 0.05.

[5] We initialize using the Stage 1 checkpoint and continue finetuning using the following configuration: epochs: 5; learning rate: 1×10^{-5} (lower than Stage 1 for stability); LoRA parameters: inherited from Stage 1.

which leaves no safe margin on a 24 GB consumer card, so a 40 GB card is required for training even though the model is 7B and inference is far lighter. For the prompts, we used task-specific prompts that were identical across all models compared (§5): a header prompt that targets the five header fields, the typed-text convention, and the `house_number`/`house_appendix` split; and a main-entries prompt that specifies the JSON entry schema together with the continuation rules for names spanning two lines and for distinguishing genuine date-less entries from continuation lines. Both prompts are reproduced in full in Appendix A.

5 Baseline Comparisons

We compare our curriculum-trained Qwen2-VL-7B against three baselines. All three are run zero-shot on the same 140-card held-out test set as our fine-tuned model, using identical two-segment crops and prompts (§7.1). Two of these models also contributed extractions to the Stage 1 noisy training pool (§4.2); we note each model's contribution below.

5.1 GPT-4o (OpenAI)

We accessed this model via our institutional LLM proxy, which served `gpt-4o-2024-11-20`. The specific parameters are not disclosed. OpenAI lists pricing at the time of writing as ∼$2.50 per 1M input tokens and ∼$10 per 1M output tokens. However, actual costs via the institutional proxy may differ, and we did not meter costs for this model.

5.2 GPT-4o-mini (OpenAI)

This model was also accessed via the institutional LLM proxy (served as `gpt-4o-mini-2024-07-18`). The parameters are also not disclosed, but they are smaller than GPT-4o. The listed pricing at the time of writing is ∼$0.15 per 1M input tokens and ∼$0.60 per 1M output tokens; as mentioned above, actual costs via the institutional proxy may differ.

5.3 Qwen 2.5 VL 32B (SURF)

Again, access via the institutional LLM proxy; however, this model is hosted through SURF. The proxy resolves to an AWQ-quantized build `Qwen/Qwen2.5-VL-32B-Instruct-AWQ` [20]. This baseline is a 4-bit-quantized 32B model, not the full-precision weights. This model can be accessed without API costs through an academic account.

6 Experiments

6.1 Inference and Scoring

For each model, we run inference on the 140-card held-out test split and score the predictions with a common evaluation script. All systems (our fine-tuned models, the

base 7B, and the API-served baselines (§5)) are scored by the same script against the same unfiltered gold standard, so the only difference between conditions is the model that produced the predictions.

One scoring convention needs to be highlighted because all reported results depend on it. Before scoring, we apply a *blank-row filter* to every system's predictions: any predicted entry whose data fields (date, name, year of birth, destination, remarks) are *all* empty is discarded. Such rows carry no transcribed content, they arise when a model reproduces the pre-printed row numbers of an empty entry grid, so counting them as false positives would penalize an artifact that any production pipeline would strip in post-processing.

The filter is applied uniformly and to predictions only: the gold standard is read unfiltered, so the reference is identical across all conditions, and no model is advantaged by the convention. It affects only the entity-level counts of Table 1; the field-level results of Tables 2 and 4 are unchanged since blank rows never survive entity alignment to become matched pairs. In practice, the filter is consequential for one system only, for reasons we take up in §7.3, where we also quantify its effect.

6.2 Evaluation Metrics

We evaluate at three granularities. Scoring a field first requires knowing which entries to compare. A header field is unambiguous, as each card has exactly one *street*, so the predicted value is compared with the gold value. Entries are not: a model may return five rows where the card holds six, in a different order, with a name garbled badly enough that its identity is unclear. Nothing in the output says which predicted row is meant to be which gold row, so we must decide that before any entry- or card-level metric can be computed. We describe that step first.

Entity alignment: A predicted entry is compared against the gold entry it is most plausibly a transcription of, determined by *optimal bipartite assignment* (the Hungarian algorithm) over the full matrix of pairwise similarities rather than by a greedy nearest-match. Pairwise similarity is a weighted mean of per-field string similarities (Ratcliff/Obershelp), weighting the identifying fields most heavily: *gezinshoofd* 2.0, *year of birth* 1.5, the two dates and *waarheen* 1.0 each, and *remarks* 0.5. Assignments scoring below 0.5 are rejected, leaving the gold entry unmatched (a false negative) and the predicted entry unmatched (a false positive). Global assignment matters here because entries on a card are frequently near-duplicates, e.g., the same surname recurring across successive tenancies, where a greedy pass can consume the wrong partner and cascade the error down the table.

Field-level accuracy: This measures exact matches per field after field-appropriate normalization, i.e., case- and punctuation-insensitive for all fields; whitespace-insensitive for compact codes (`house_number`, `buurtletter`), where "67 B" ≡ "67B"; and leading zero-aware for the 4-digit `codenummer` (a value stored as an integer loses a leading zero, so both sides are zero-padded to 4 digits before comparison). This measures transcription correctness rather than incidental formatting differences. As a soft, partial-credit companion metric, we use ANLS (Average Normalized Levenshtein Similarity) [21]. Exact match is all-or-nothing: a name read as "van der Berg" against a gold "van den Berg" scores 0, exactly like unrelated output. ANLS instead

scores $1 -$ (edit distance/length of the longer string), so near-misses are visible, which matters here because most residual errors are single-character slips rather than wholesale misreads. Following the standard ICDAR/DocVQA formulation, we zero any score below $\tau = 0.5$, so a prediction less than half-right earns no credit. Note that ANLS is a *reported* metric over already-aligned fields and uses edit distance, whereas the alignment step above uses Ratcliff/Obershelp merely to decide which entries correspond; the two serve different purposes and are deliberately not the same function.

Entry-level accuracy: Complete entry match (all fields correct); partial match ($\geq$50% of fields correct).

Card-level accuracy: Perfect extraction (header + all entries correct); header accuracy (header fields only); entry recall (percentage of true entries detected); entry precision (percentage of predicted entries that are correct).

7 Results

7.1 Overall Performance

Table 1 reports entity-level extraction on the 140-card held-out test split. Our curriculum-trained 7B model attains the highest F_1 (0.989), ahead of GPT-4o (0.967 ± 0.004), the Qwen-32B (0.933 ± 0.002), and GPT-4o-mini (0.905 ± 0.007). GPT-4o is the strongest baseline, but the specialized 7B leads in both precision and recall and can be run locally without any additional API costs.

Table 1 Entity-level extraction on the 140-card held-out test split (precision / recall / F_1 for main-entry rows). All models see identical two-segment inputs (768 px header + main-entries crops), are decoded at the same max-token budget, and are scored after the blank-row filter (§6.1). "Resid. F1" is F1 restricted to the 134 residential cards, excluding the six the card-type classifier routes away in production (§8.4): five that carry no entries at all and one that carries entries but no person, such as a card recording only an office tenancy. API-baseline figures are the mean of three same-day runs ($\pm$ standard deviation on F_1); the fine-tuned models decode greedily and are deterministic, so they are reported from a single run.

Model	Params	Prec.	Rec.	Entity F_1	Resid. F_1
GPT-4o	(undiscl.)	0.986	0.950	0.967 $\pm$0.004	0.968
GPT-4o-mini	(undiscl.)	0.979	0.842	0.905 $\pm$0.007	0.906
Qwen 2.5 VL 32B	32B	0.941	0.925	0.933 $\pm$0.002	0.935
Qwen2-VL-7B (base)	7B	0.900	0.798	0.846	0.847
+ gold-only FT	7B	0.858	**0.994**	0.921	0.983
+ **curriculum (ours)**	7B	**0.988**	0.990	**0.989**	**0.990**

The three API baselines were run via the institutional LLM proxy in a single same-day batch under prompts and crops identical to the fine-tuned models, with the served snapshot logged from each response: GPT-4o as

`gpt-4o-2024-11-20`, GPT-4o-mini as `gpt-4o-mini-2024-07-18`, and the 32B row as `Qwen/Qwen2.5-VL-32B-Instruct-AWQ` (an AWQ-quantized build; §5.3).[6]

7.2 Field-Level Breakdown

Exact match, as reported in Table 2, is an unforgiving measure: a single mistyped character scores the same as a wholly unrelated reading. This penalty falls hardest on the long free-text fields, where there is simply more surface on which to slip. To separate near-misses from genuine failures, Table 3 rescores our model with per-field ANLS (§6.2) alongside exact match.[7]

Table 2 Per-field exact-match accuracy on the 140-card held-out set. Header fields are scored per card over all 140 cards; the six entry fields are scored over matched entry pairs only (§6.2), whose count is given in the final row. That count differs by model because each is scored only on the entries it recovered and transcribed closely enough to align: of the 510 gold entries in the split, ours aligns 505 and the base 7B only 406, and the entries a model missed are charged as recall in Table 1 rather than as field errors here.

Field	GPT-4o	GPT-4o mini	Qwen 32B	Qwen 7B	Gold only	**Curriculum (ours)**
`street`	95.2	89.0	95.7	92.8	95.7	**97.1**
`house_number`	86.2	75.7	79.8	83.5	95.0	**95.7**
`house_appendix`	74.3	71.2	80.0	76.3	85.0	**89.3**
`codenummer`	96.9	96.4	97.1	89.2	97.1	**97.1**
`buurtletter`	81.0	44.5	79.3	50.4	84.3	**95.7**
`datum`	79.7	74.0	80.8	36.1	92.5	**94.1**
`gezinshoofd`	71.4	15.9	70.6	52.5	75.0	**77.7**
`year_of_birth`	85.4	83.8	86.8	56.0	95.1	**96.0**
`datum_vertrek`	81.9	74.0	82.3	42.9	92.3	**93.5**
`waarheen`	77.7	65.6	70.3	43.8	87.4	**88.6**
`remarks`	74.7	61.5	75.2	44.1	86.5	**89.5**
Matched pairs	484	430	472	406	507	**505**

The two measures diverge sharply by field type. On the short, structured fields, the difference is marginal, with *year of birth* gaining 0.1 points and *house appendix* 0.6. It should be noted that little else is possible on fields this short, where a single wrong character already consumes a large fraction of the string. On the free-text fields, the gap opens wide: *waarheen* gains 5.4 points, and *gezinshoofd* rises from 77.7 to 92.8, a gain of 15.1. Names, in other words, are rarely misread outright. Inspecting the 108 mismatched names among the 505 aligned entries directly confirms this: half (50.9%) differ from the gold string by a single character and 69.4% by at most two, a class

[6] We batch the baselines within one day because the same API alias, queried three days apart with an identical prompt and image, produced a systematic shift in output, with every GPT-4o field moving in one direction, well beyond the ∼1-point run-to-run variance. This seems to be due to the alias being repointed to a different snapshot.

[7] Per-field ANLS for the baselines is available from the evaluation script; we report our model's here to characterize its residual errors, not to re-rank the systems, since the exact-match ranking of Table 2 already establishes that.

of error that exact match discards entirely. The residue is orthographic rather than semantic (*Franciscus*/*Francisus*, *Schoenmaker*/*Schoemaker*, *Visscher*/*Vischer*), and 33 of the mismatches (30.6%) are in fact identical once punctuation and spacing are disregarded, differing only in, for example, the comma separating surname from given names. Which measure to prefer depends on the downstream use. For record linkage, where a name is a key that must match precisely, exact match is the honest figure. For reading and search, where a human or a fuzzy index tolerates a single-character slip, ANLS better reflects what the extraction actually delivers.

Table 3 Exact match versus ANLS for our curriculum model, per field. Δ is the partial credit that exact match discards; it is negligible on short, structured fields and largest on free text.

Header fields				*Entry fields*			
Field	Exact	ANLS	Δ	Field	Exact	ANLS	Δ
`street`	97.1	99.7	+2.6	`datum`	94.1	96.0	+1.9
`house_number`	95.7	98.2	+2.5	`gezinshoofd`	77.7	92.8	**+15.1**
`house_appendix`	89.3	89.9	+0.6	`year_of_birth`	96.0	96.1	+0.1
`codenummer`	97.1	99.3	+2.2	`datum_vertrek`	93.5	95.6	+2.1
`buurtletter`	95.7	98.0	+2.3	`waarheen`	88.6	94.0	+5.4
				`remarks`	89.5	93.9	+4.4

The address is represented as two fields: `house_number` (the bare unit, e.g. "34c") and `house_appendix` (floor/position qualifiers, e.g. "I hoog", "huis", "hs"). Splitting these, rather than packing them into one string, raised our model's `house_number` accuracy from 0.83 to 0.96, since the inconsistently transcribed qualifiers no longer corrupt the primary address key.

Our fine-tuned 7B leads in every field against every baseline, including GPT-4o, with a single tie on codenummer at 97.1, shared with the 32B model and the gold-only ablation. In the case of the `buurtletter` field, where all fine-tuned models dramatically outperform every zero-shot model (95.7 / 84.3 vs 81.0 GPT-4o, 79.3 Qwen-32B, 44.5 GPT-4o-mini): this neighborhood code sits at the edge of the header crop and is easily confused with other numbers in its vicinity, but fine-tuning teaches the model where to look.

Name conventions vs. transcription errors

A subset of the residual `gezinshoofd` errors are not misreadings at all: the model recovers every character and disagrees only about *convention* — how the surname is punctuated from the given names, whether the maiden name leads or sits in parentheses, and whether the `echtg` (*echtgenote/echtgenoot*, "spouse of") prefix is retained. Because record linkage normalizes name order and affixes before matching anyway, these inflate an error rate a downstream user would not care about. To quantify them, we re-scored names after a deliberately simple normalization applied identically to gold and prediction: strip the `echtg` prefix, drop parenthetical maiden names, and sort the remaining tokens. This resolves 38 of our model's 108 non-exact pairs: 22 differ

from gold only in punctuation or spacing (a subset of the 33 punctuation-only mismatches noted above; the other 11 differ in an affix or parenthetical as well), 9 only in `echtg` handling,[8] 6 only in the placement of a parenthetical maiden name, and 1 in a combination of these. Only a single pair differs purely by token order. Table 4 reports `gezinshoofd` before and after.

Table 4 Head-of-household (`gezinshoofd`) scores, as transcribed and after downstream name normalization. Scored per card over its matched entry pairs and averaged across all 140 cards, identically to Table 2, whose `gezinshoofd` row the "Exact" column reproduces; **Pairs** varies by model for the reason given there. SeqRatio ("SeqR.", Ratcliff/Obershelp) is shown alongside ANLS because the two diverge precisely on the reordering errors at issue here. API-baseline rows are the mean of the same three runs as Table 1; the fine-tuned rows are the single deterministic run.

		As transcribed			*Normalized*	
Model	Pairs	Exact	ANLS	SeqR.	Exact	ANLS
GPT-4o	484	0.714	0.846	0.857	0.772	0.854
GPT-4o-mini	430	0.159	0.773	0.815	0.711	0.811
Qwen 2.5 VL 32B	472	0.706	0.823	0.842	0.732	0.829
Qwen2-VL-7B (base)	406	0.525	0.636	0.656	0.579	0.646
+ gold-only FT	507	0.750	0.918	0.934	0.804	0.923
+ **curriculum (ours)**	505	**0.777**	**0.928**	**0.940**	**0.835**	**0.934**

The same caution applies, far more strongly, to GPT-4o-mini, whose `gezinshoofd` score of 0.159 is not primarily a reading failure: the model consistently omits the space after the comma separating the surname from the given names, emitting `van der Ham,Joris` where the gold reads `van der Ham, Joris`. Averaged over the same three runs, 270 of its 341 non-exact names (79.3%) are character-identical to gold once punctuation and spacing are disregarded, which is why its ANLS (0.773) sits so far above its exact match, and why normalization lifts it to 0.711: tokenizing discards the offending comma incidentally, rather than absorbing the name-order convention the step was designed for. Only a single one of its mismatches is a genuine reordering; the remainder are ordinary transcription errors (character confusions such as *Lenardus*/*Ienardus*) and occasional truncation after the comma (2.6% of mismatches). We report the strict figure in Tables 2 and 4 because the convention is genuinely wrong with respect to the archival source and would have to be repaired before use, but readers should not take the 0.159 as evidence of an inability to read the names. Normalization moves the other systems far less (+2.6 to +5.8 points of exact match), so the ranking is unchanged; the gap to the weakest baseline simply narrows once the comparison is restricted to what a linkage pipeline would actually consume.

One caveat on the normalized column: because it sorts tokens, it cannot tell a harmless convention from a real error, and would credit a model that confused "Jansen,

[8] The `echtg` cases are partly an annotation artifact rather than a model error. Within the test split the gold standard spells the abbreviation four ways across its 22 occurrences (`echtg` 16, `echt` 3, `echtgenote` 2, `echt g` 1), whereas the model writes `echtg` in all 24 of the cases it produces, so some mismatches are scored against an inconsistent reference. The artifact does not account for all of them: of the 14 `echtg`-related mismatches, roughly half are spelling differences of this kind, while the rest are genuine errors in which the model supplies a prefix the card does not carry or omits one it does.

Piet" with "Piet, Jansen". In practice this does not occur—re-running the normalization with the sorting step removed changes no model's score by even a single pair, so every gain comes from stripping the affix, the parentheses, and the punctuation. The column is nonetheless an optimistic estimate of what a linkage pipeline would recover rather than a stricter measurement of transcription, and it does not supersede the exact-match figures of Table 2.

7.3 Empty-Card Hallucination and the Blank-Row Filter

Misreading text remains the most frequent source of error (for our model, 263 incorrect field values against 6 spurious rows) but the most consequential one is *inventing* rows that the card does not contain. A wrong character in a name is visible as noise to a downstream user; a fabricated resident is not. We distinguish two variants, which matter differently in an archival context. First, blank skeleton rows, which are entries whose every data field (date, name, year, destination, remarks) is empty. In effect, the model transcribes the pre-printed row numbers, but it adds no content. These additions are harmless and removable. Second, fabricated content, which includes entries with invented names and dates that pass JSON validation. The latter is the dangerous outcome since a fabricated resident is indistinguishable from a real one to a downstream user.

These behaviors are strongly model-dependent, and they are concentrated on the empty cards rather than distributed across the collection. Across the five empty cards in the test set, the **gold-only** model emitted 69 spurious rows, each carrying fabricated content. The **curriculum** model emitted 37, of which 36 are blank skeleton rows and only **one** carries fabricated content. The two GPT-4o baselines fabricated **zero**; the Qwen-32B model fabricated **one**, and, tellingly, that one is the few-shot name from the prompt itself (`Johannes van der Berg-Smit`, `250566`) echoed onto an empty card, reproduced identically in all three of its runs. Fabrication is thus a symptom of narrow gold-only fine-tuning that the curriculum's broad Stage 1 nearly eliminates ($69 \rightarrow 1$). We attribute the residual invention shared by both fine-tuned models to the training mix itself: the gold set contains almost no empty cards, so a fine-tuned model learns that a housing-card image implies a populated table, which is a limitation we return to, with mitigations, in §8.4.

On populated cards, the picture is different in both frequency and kind. Spurious rows are rare there for every system except the base 7B (5 for our model, 5 for GPT-4o, 29 for the 32B model, 45 for the base 7B), and they are not appended after the last genuine entry, as the empty-card behavior might suggest: they are almost all interleaved among the real rows (for our model, 5 of 5; for the base 7B, 38 of 43). Their causes also differ from empty-card fabrication. Our model's five are fragments rather than inventions: two are orphaned parenthetical maiden names (`(Koopmans)`, `(Schoo)`) split off from a name that runs across two lines, and two transcribe non-person annotations on the card (`Kantoor`, "office"; `In gebruik bij...`, "in use by") that the gold standard does not record as entries. The base 7B's include verbatim echoes of the few-shot example from its own prompt and duplicated rows emitted twice under different dates.

The blank-row filter of §6.1 removes the harmless skeleton-row artifact without touching real entries: the curriculum model's entity precision rises from 0.923 to 0.988, and its F_1 from 0.956 to 0.989 (empty-card false positives 37 → 1), while every baseline scores identically with the filter on or off. The convention therefore does not flatter our system relative to the alternatives; it removes a self-inflicted penalty that only our system incurs. The single residual fabrication and the gold-only model's un-filterable fabricated content are what the residential-only column and §8.4's card-type filter address.

7.4 Errors on Populated Cards

Empty cards are a small and separable problem. On the 135 populated cards, the errors are of a different character, and three categories account for most of them.

Presence versus content. A field can be wrong because it is misread or because one side records a value while the other records nothing. The second kind is where the systems differ most sharply. Such presence/absence disagreements make up 26.5% of the 32B model's field errors and 19.9% of GPT-4o's, but only 7.2% of ours (19 of 263). The baselines most often leave *remarks* and *waarheen* empty on cards that fill them (GPT-4o accounts for 37 of its 87 *remarks* errors this way), which is the signature of a model that transcribes what it reads confidently and skips what it does not. Fine-tuning on in-domain cards largely removes this behavior, since the model learns that these columns are usually populated and what kind of content to expect in them.

Date misreads. Where both sides record a date, disagreements are mostly single-digit slips on degraded scans rather than wholesale misreadings: 27 of our model's 35 such cases (77%) differ by one character, compared to 66% for GPT-4o and 54% for the 32B model. The six-digit DDMMYY format offers no redundancy, so a single struck-over digit is unrecoverable from context; this is the one category where we see no route to improvement other than training a specific model for data extraction.

Name continuation. Names frequently run across two printed lines, and a model that treats the second line as a separate record loses half the name. This is rare in absolute terms, affecting 3 matched pairs for our model, 4 for GPT-4o, and 6 for the 32B model. However, it is the failure of the continuation rules in our prompt (§4.1) that is written to prevent it. Our three residual cases are instructive: two drop a parenthetical maiden name carried on the second line (`van Rijn, Cornelia J M (Koopmans)` returned without the parenthesis), and this is the same construction that elsewhere appears as a spurious extra row (§7.3). The second line of a name is thus either absorbed, dropped, or split off as its own record, depending on the card.

Across all three categories, the ranking is consistent with Table 2, but the mechanisms differ: the 32B model recognizes characters well, while over-segmenting the table more than any other system (29 stray predictions; Table 6), whereas our model's residual errors are concentrated in single-character near-misses on names where the typescript is degraded, overtyped, or struck through (§7.2).

7.5 Curriculum Learning Ablation

Fine-tuning on gold data alone lifts F_1 from 0.846 to 0.921; adding the Stage 1 noisy pre-training raises it further to 0.989 (+0.068 over gold-only; Table 5), confirming that the curriculum contributes well beyond gold fine-tuning. The gain is not free. The gold-only variant is the same Stage 2 run started from the base model instead of the Stage 1 checkpoint, so the curriculum's extra cost is exactly Stage 1. That stage dominates: it processes 44,352 training samples (11,088 cards × 2 segments × 2 epochs) against Stage 2's 5,600 ($560 \times 2 \times 5$), a 7.9:1 ratio, and both stages run on the same hardware at the same sequence length. The curriculum therefore costs several times the training of the gold-only variant—about 5 of the 6 GPU-hours. Against a one-off budget of well under a GPU-day, and a collection of 473,593 cards over which that cost is amortized, this is a favorable exchange, but it would not be for a collection of a few hundred.

Table 5 Impact of curriculum learning stages. All rows blank-row filtered, as in Table 1. The final two columns describe what each variant cost to produce rather than how it performed: GPU-hours are wall-clock on a single A100 (40 GB).

	Held-out performance			*Training cost*	
Model Variant	Prec.	Rec.	F_1	Training Data	GPU-h
Base Qwen2-VL-7B (no FT)	0.900	0.798	0.846	—	0
Gold-only fine-tuning	0.858	**0.994**	0.921	560 gold cards	∼1
Curriculum (Stage 1→2)	**0.988**	0.990	**0.989**	11K noisy + 560 gold	∼6

Crucially, this gain is entirely in precision (0.858 → 0.988, +0.130); recall is essentially unchanged (0.994 → 0.990). The gold-only model recovers nearly all true entries but over-produces false positives that the blank-row filter cannot remove, dominated by fabricated content on empty cards (§7.3). On non-empty cards, the curriculum model is also cleaner on ordinary over-segmentation (5 stray predictions vs 15). It fuses rows slightly more often than the gold-only variant (2 merges vs 0), a much smaller effect than the false positives it avoids. The Qwen-32B baseline shows the opposite profile from the fine-tuned models: it has the **lowest precision of any system (0.941)** and the most over-segmentation (29 stray predictions on non-empty cards; Table 6), even though it rarely fuses rows (7 merges) and invents almost nothing on empty cards. Table 2 details the per-field breakdown; Table 6 shows the false-positive decomposition.

7.6 Computational Efficiency

Our model enables: (1) near-zero and, crucially, *fixed* inference cost: no per-request charge and no per-token billing, so the marginal cost of a card is compute time rather than API spend (the bootstrap that produced the training data did incur API costs; §8.2); (2) data privacy: records containing personal data are processed on-premises,

Table 6 False-positive decomposition on the 140-card held-out set, after the blank-row filter (§6.1), as in Table 1. The empty-card and populated-card columns partition Total FP exactly. "Merges" = correct rows fused into a single prediction; it is counted on the *gold* side (how many true rows were swallowed), so it is a separate diagnostic and not part of that sum. Only our model is affected by the filter: before it, our Total FP is 42, of which 36 are blank skeleton rows.

			Where they occur	
Model	Total FP	Merges	Empty cards	Populated cards
Qwen 2.5 VL 32B	30	7	1	29
Qwen2-VL-7B (base)	45	7	0	45
+ gold-only FT	84	0	69	15
+ **curriculum (ours)**	**6**	2	1	**5**

which matters for historical archives operating under the GDPR; and (3) scalability. We processed all 473,593 cards in ∼2h25m on 16 A100 GPUs with vLLM, at ∼0.29 s/card, without API rate limits or per-request costs (§8.5).

Table 7 Deployment and cost comparison. API costs are list price for the two-segment run (2 calls/card); the local row is the measured HPC charge for the full-archive run (§8.5), which is additional to the one-time ∼6 GPU-hours of training.

Model	Deployment	VRAM	Cost per 1000 cards
GPT-4o	Cloud API	N/A	∼$10–20[1]
GPT-4o-mini	Cloud API	N/A	∼$0.60–1.20[1]
SURF Qwen 32B	Cloud API	N/A	Free[2]
Ours (Stage 2)	**On-prem.**	**40 GB**[3]	**€0.17**[4]

[1]Order-of-magnitude list-price estimate from typical per-card token counts (∼1–2k input incl. image tiles, ∼0.3–0.5k output × 2 segments); not metered directly in this study.

[2]Dutch academic access only.

[3]The A100 (40 GB) we trained and served on. Inference is lighter than training and we did not measure a minimum-VRAM floor, so we report the hardware used rather than the smallest card the model would fit on.

[4]Not free, but metered: the full-archive run was billed as HPC compute time, and €0.17 is that measured charge expressed per 1,000 of the 473,593 cards (§8.5). On self-owned hardware the marginal cost is electricity alone; we report the figure we can substantiate rather than assuming free capacity.

8 Discussion

8.1 Curriculum Learning Effectiveness

The curriculum outperforms direct gold-only fine-tuning: F1 0.989 vs 0.921 (+0.068; Table 5). More informative than the headline number, however, is *where* the gain resides. Recall is essentially unchanged between the two, so the improvement is almost

entirely in precision (§7.5). Stage 1's ∼11K diverse but noisy cards do not primarily teach the model to *read* better (per field gains over gold-only are modest, a few points per field); they teach it when not to emit an entry.

The clearest evidence is the empty card behavior (§7.3). When exposed in Stage 1 to the messiness of the archive, including sparse and empty cards, the curriculum model almost never fabricates entries, whereas the gold-only model invents entire residencies. For an archival tool, this is the difference that matters: gold-only's errors are fabricated residents indistinguishable from real ones, whereas the curriculum model's residual errors are droppable blank rows and single-character near-misses. Broad noisy pre-training thus acts less as a knowledge source than as a regularizer against over-confident hallucination.

8.2 Model Size vs. Performance, and What "No API Cost" Means

The headline comparison (§7.1) shows that the parameter count is not the bottleneck for this task: our 7B model outperforms GPT-4o, GPT-4o-mini, and the Qwen-32B model on entity F_1 and on every one of the 11 fields (Table 2), even though GPT-4o is far larger and the 32B model comes from the same architecture family as our base. [9]

The decisive factor is **task-specific adaptation**: the layout conventions (e.g. the edge-of-crop `buurtletter` code, where zero-shot models score 44–81% and ours 96%), the 6-digit date format, and the archive's continuation and null-handling rules are learned from a few hundred gold cards, not recovered from scale.

Two qualifications keep the cost claim honest. First, the curriculum itself was bootstrapped with API-served models: Stage 1's noisy labels came from the SURF academic endpoint (free, but not free to operate) and from GPT-4o-mini, and GPT-4o contributed to the annotation work package from which part of the gold standard was corrected (§3.2). In principle those labels could have come from a smaller local model or from annotation from scratch. In our case, "zero API cost" describes *inference*: API dependence is confined to a one-time labeling step that is never repeated, and that step cost approximately €64 at the proxy's own rate table.[10] Roughly 99% of that spend was redundant by construction: the paid models were run over ranges the free endpoint had already covered, and deduplication then discarded their output (§3.2). Sending them only to the cards, SURF did not process would have cost cents rather than tens of euros, so this is an artifact of how the work unfolded rather than a cost the method requires.

[9] One caveat relates to the 32B comparison specifically. The institutional LLM proxy serves an AWQ 4-bit-quantized build (§5.3), so we are strictly comparing against a quantized model rather than a full-precision 32B. Quantization can cost a few points of accuracy, and a full-precision deployment might close part of the gap.

[10] GPT-4o labeled four street ranges (5,470 cards) and GPT-4o-mini the other four plus the random batch (6,444 cards), each card sent as two segment calls—on the order of 24,000 requests. The institutional proxy runs LiteLLM, which meters each request against its published price list; for these two models that list carries OpenAI's rates of $2.50/$10.00 per 1M input/output tokens for GPT-4o and $0.15/$0.60 for GPT-4o-mini. Prompt and completion volumes are estimated, so the total is accurate to roughly ±15%. The distribution is lopsided: GPT-4o accounts for 93% of the spend on 46% of the cards, and the entire GPT-4o-mini run costs under $5. The 893 cards that actually reached the Stage 1 pool would have cost about $0.58 to extract on their own.

Second, the resulting picture is best understood as a distillation-flavored bootstrap: frontier and mid-tier APIs seed a local model that then surpasses every one of its teachers after gold refinement. This is an encouraging result for cultural-heritage data processing, where a target collection is fixed and finite: a modest, locally deployable model fine-tuned on a few hundred in-domain examples can beat frontier APIs at a one-time bootstrap cost and without sending sensitive archival data to a third party at scale.

8.3 Segmented Extraction Benefits

We measured the two-segment (header crop + main-entries crop) approach against single-pass full-card extraction directly, running the same model (Qwen 2.5-32B) on the 140 held-out cards in both modes. The two-segment approach scored F_1 0.915 vs 0.888 for single-pass, a gain driven almost entirely by precision (0.971 vs 0.867): giving the model a zoomed, cropped view of just the main-entries table sharply reduces spurious rows and field bleed from surrounding regions, at a small recall cost (0.865 vs 0.910). The two modes also see genuinely different amounts of the card: single-pass additionally returns the reverse-side follow-up table, 445 rows across 88 of the 140 cards, which the two-segment crop excludes by construction. Those rows land in their own field rather than contaminating the main entries, so they do not explain the precision gap (single-pass emits 71 spurious main-entry rows against the two-segment run's 13); but they do mean the two modes are not scored over the same task. We therefore treat them separately and do not mix them in the main comparison; all Table 1 results use the two-segment pipeline. One caveat on reading the two together: this ablation predates the same-day baseline batch and uses a single earlier run of the 32B model, so its two-segment figures (F_1 0.915) differ from the three-run mean reported for the same model and mode in Table 1 (0.933 ± 0.002). The ablation is internally consistent—both modes come from that same earlier run—so the comparison it supports is unaffected, but the absolute values are not directly comparable across the two tables.

8.4 Limitations

1. Geographic bias: The current gold standard (700 cards) spans 216 archival boxes and 315 street names but is weighted towards a few; the largest box, inv. 197, alone contributes 196 cards. Because coverage is uneven, per-field accuracy may degrade on under-represented streets and hands; we have not measured this per neighborhood, and broader coverage plus per neighborhood reporting would rule it out. A mitigation is to expand the gold standard beyond 700 cards towards broader, more even street coverage. **2. Noisy Stage 1 data quality.** Stage 1 training uses automated extractions with unknown error rates. Some errors may be learned and require Stage 2 to correct. **3. Follow-up entries:** The current pipeline covers the header and main entries only. The follow-up table on the reverse of the card is a distinct extraction target rather than more of the same: it records co-residents (*inwonenden*) rather than heads of household, under a different column schema, and so it needs its own prompt, its own annotation effort, and its own evaluation. The omission is substantial: across

our 700 annotated cards, the follow-up tables hold 2,407 rows against 2,633 in the main tables, and 65% of cards carry at least one follow-up entry, so extending coverage would nearly double the number of person-records recoverable from the collection. **4. Evaluation set size:** The 140-card held-out test set is relatively small. A larger held-out set spanning more streets would strengthen the generalization assessment. **5. Empty and non-standard cards:** Both fine-tuned models invent table rows on empty (pre-printed, no-data) cards rather than emitting an empty entry list, which is the dominant source of their false positives (§7.3). Neither the base 7B nor the API baselines do this, identifying it as a fine-tuning artifact with a clear cause: the gold training mix contains virtually no empty cards, so the fine-tuned model learns that a housing-card image always implies a populated table. Adding a modest number of empty cards and other non-standard examples (index cards, snipped-corner cards) to Stage 2, or a lightweight pre-classification step that routes empty cards past the extractor, would remove this failure mode.

8.5 Practical Deployment Considerations

We ran the full-archive extraction end to end. Using vLLM continuous batching on NVIDIA A100 (40 GB) GPUs, the pipeline processed the entire corpus of 473,593 cards, sharded 16 ways (one A100 per shard, 8 CPU cores each), in ∼2h25m wall-clock (∼39 A100-GPU-hours total).[11]

A key deployment finding concerns generation length. Housing cards vary widely in resident count, and the distribution is long-tailed: across the full 473,593-card run, the median card carries 3 main entries and the mean 4.1, but the 95th percentile is 11 and the densest cards reach 19. The gold set matches this shape (median 3, mean 3.8, maximum 17). Cards with ten or more entries are only 6.9% of the collection, so a generation budget calibrated on typical cards leaves the tail unservable while appearing to work. An early configuration capped generation at 512 new tokens, which silently truncated the main-entries table mid-record on many-resident cards: the model emitted valid JSON up to the cutoff, then an incomplete record, yielding a parse failure that unfairly penalized exactly the hardest cards. We traced this to a training/inference mismatch: the fine-tuning sequence budget (`MAX_LENGTH`) had itself been short enough to truncate the longest gold targets, so the model was partly trained to stop early. Retraining with a longer sequence budget (2816 tokens, covering an image+prompt block of ∼987 tokens plus answers up to ∼1829 tokens) and raising the inference cap to 2048 new tokens removed the truncation entirely: across the full 473K-card run, zero cards were truncated, and cards with up to 19 entries were extracted complete. For tabular historical documents with variable record counts, the generation-length budget is a parameter that must be calibrated jointly at training and inference time.

On cost, those 38.7 GPU-hours are what the scheduler billed us: at the `gpu_a100` partition's rate of 128 SBU per GPU-hour, 4,954 SBU, or €79 at SURF's published 2025 rate of €16 per 1,000 SBU, which is 2.2% of the project's compute grant, and a figure taken from the accounting records rather than estimated. The comparison is with an estimated $5,000–9,000 in GPT-4o list API charges at this volume

[11] 473,224 cards (99.92%) were extracted successfully; 365 cards (0.08%) produced unparseable output, and 4 had missing images.

(473,593 cards × 2 segments, at $2.50/1M input and $10/1M output tokens over typical per-card token counts): a difference of roughly two orders of magnitude, with no dependence on per-request rate limits or the availability of a specific API. Because the 7B model fits on a single 40 GB GPU, the 16-way sharding above is purely a throughput choice: the same run is reproducible on one such GPU at proportionally longer wall-clock time, without multi-GPU infrastructure.

9 Future Work

Four extensions follow directly from the limitations above. First, the printed per-entry male/female occupant counts (the M/V columns), currently excluded from the schema (§3.3), would enrich the household-composition signal of the downstream database if they can be extracted reliably; establishing whether the scans support this requires a dedicated annotation round. Second, the follow-up entries on the card reverse remain unaddressed. The layout is fairly similar to the main entries, so we assume limited additional training data is needed. The main difficulty relates to linking the follow-up entries to the main entries, as the strategy for linking has not been executed consistently by the administrators of the registration cards. Third, the cards contain internal cross-references (destination fields reading "see ⟨street⟩ ⟨number⟩") whose resolution is currently a downstream database concern; teaching the extractor to type them explicitly would remove a class of downstream misclassifications at the source. Fourth, on the deployment side, distilling the fine-tuned 7B into a smaller student would lower the hardware floor below a 40 GB card, and adding empty-card and non-standard-card examples to Stage 2 (§8.4) should close the one remaining systematic failure mode.

10 Conclusion

We presented a curriculum learning approach for extracting structured data from historical Dutch housing cards, demonstrating that a 7B-parameter vision-language model fine-tuned in two stages (noisy pre-training followed by gold-standard refinement) surpasses larger commercial models (GPT-4o and a 32B non-specialized model) on this task while enabling cost-effective local deployment.

Beyond the specific result, the pieces we contribute are meant to be reusable: a two-stage curriculum that pairs broad, noisy pre-training with gold-standard refinement; a segmented header/entry extraction strategy; a matched comparison against GPT-4o, GPT-4o-mini, and a 32B Qwen model; and a dataset of 700 human-validated annotations spanning 216 archival boxes and 315 streets — together yielding a single-GPU-deployable model that transcribed the full 473K-card archive for €79 in compute after a modest one-time bootstrap (§8.2).

This work demonstrates that specialized fine-tuning on modest hardware can produce models competitive with, or superior to, large commercial alternatives for focused historical document understanding tasks. Our approach enables cultural-heritage institutions to unlock structured data from historical archives without ongoing API costs and, because the model runs locally, without transmitting records containing personal data to third-party services at scale. The resulting transcriptions remain personal

data, and their onward use and publication are governed by the same privacy regime as the source records.

Acknowledgements. Acknowledgements have been removed for anonymous review.

Declarations

- **Funding:** Removed for anonymous review.
- **Conflict of interest/Competing interests:** The authors declare no competing interests.
- **Ethics approval and consent to participate:** The processing of personal data was conducted under the access conditions of the Amsterdam City Archives and institutional data-protection review; no human-subjects experimentation was performed. Reference number removed for anonymous review.
- **Consent for publication:** Not applicable.
- **Data availability:** The corrected dataset transcribes personal data of individuals, some of whom are still living, and the source cards are made available by the Amsterdam City Archives on a view-only basis for scientific purposes; the annotations will therefore be made available for scientific use under access conditions agreed with the archive and the institutional data steward.
- **Materials availability:** Not applicable.
- **Code availability:** Training scripts and model checkpoints will be released upon publication.
- **Author contribution:** Removed for anonymous review.
- **Use of large language models:** Vision-language models are the object of study and are documented in §§3–5. Coding agents (Claude) were used to test code.

Appendix A Extraction Prompts

Both prompts are used identically across all models compared (§5). Header and main-entries segments are prompted separately (§4.1).

Header prompt:

```
Extract HEADER ONLY from Dutch housing card (1954-1989). All fields
are TYPED (printed).
Return JSON: {"street":"name","house_number":"123",
"house_appendix":"I hoog","codenummer":"1234","buurtletter":"AB"}
Rules:
- Transcribe exactly as printed
- house_number: the bare number, plus a trailing unit letter if
  present ("34c", "123A")
- house_appendix: floor/position qualifiers AFTER the number
  (e.g. "I hoog", "huis", "II en III hoog", "hoek Floraweg");
  empty string "" if none. Roman numerals (I/II/III) and words like
  huis/hoog/achter/voor go HERE, not in house_number.
```

```
- codenummer: 4-digit STRING, preserve leading zeros ("0383", "0050")
- Use null for missing codenummer, empty string for missing
  buurtletter/house_appendix
Examples:
{"street":"Prinsengracht","house_number":"123A","house_appendix":"",
"codenummer":"1452","buurtletter":"AB"}
{"street":"Govert Flinckstraat","house_number":"67",
"house_appendix":"II huis","codenummer":"0341","buurtletter":"K2"}
{"street":"Rozengracht","house_number":"6","house_appendix":"I hoog",
"codenummer":"0383","buurtletter":"H"}
Return ONLY JSON. No explanations.
```

Main entries prompt (with critical continuation rules):

```
Extract MAIN ENTRIES ONLY from Dutch housing card table (1954-1989).
Return JSON array.
Entry format:
{"datum":"250566","gezinshoofd":"name","year_of_birth":"21",
"datum_vertrek":"101270","waarheen":"city","remarks":"notes"}
TABLE STRUCTURE:
- year_of_birth column has header "jaar" (rotated 90 degrees)
- Most entries START with a TYPED date (DDMMYY format)
- FIRST entry may have NO arrival date (datum: null) - resident was
  already living there
CRITICAL CONTINUATION RULES:
- Names (gezinshoofd) may span TWO lines - combine them into one field
- Lines with BOTH no arrival date AND no departure date are possibly
  continuations (except first entry)
- Lines with NO arrival date but WITH departure date are VALID entries
  (people who joined later)
- Only treat a line as continuation if it has neither datum nor
  datum_vertrek
- year_of_birth ("jaar"): 2-digit STRING with leading zeros ("04", "21")
- Dates: 6-digit STRING DDMMYY or null for missing
- Use null for missing dates, empty string "" for missing text fields
Example with name overflow:
If you see:
  250566  Johannes van der Berg-    21  101270  Rotterdam  Notes
          Smit
Combine as: "gezinshoofd": "Johannes van der Berg-Smit"
Return ONLY JSON array. Extract ALL typed dates!
```

References

[1] Lafia, S., Bleckley, D.A., Alexander, J.T.: Digitizing and parsing semi-structured

historical administrative documents from the G.I. Bill mortgage guarantee program. J. Doc. **79**(7), 225–239 (2023) https://doi.org/10.1108/JD-03-2023-0055

[2] OpenAI: GPT-4o system card. Preprint at https://arxiv.org/abs/2410.21276 (2024)

[3] Bai, J., Bai, S., Yang, S., Wang, S., Tan, S., Wang, P., Lin, J., Zhou, C., Zhou, J.: Qwen-VL: A versatile vision-language model for understanding, localization, text reading, and beyond. Preprint at https://arxiv.org/abs/2308.12966 (2023)

[4] Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. In: Advances in Neural Information Processing Systems 36 (NeurIPS) (2023)

[5] Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: Proceedings of the 26th Annual International Conference on Machine Learning (ICML), pp. 41–48 (2009). https://doi.org/10.1145/1553374.1553380

[6] Soviany, P., Ionescu, R.T., Rota, P., Sebe, N.: Curriculum learning: A survey. Int. J. Comput. Vis. **130**(6), 1526–1565 (2022) https://doi.org/10.1007/s11263-022-01611-x

[7] Xu, B., Zhang, L., Mao, Z., Wang, Q., Xie, H., Zhang, Y.: Curriculum learning for natural language understanding. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 6095–6104 (2020). https://doi.org/10.18653/v1/2020.acl-main.542

[8] Bluche, T., Louradour, J., Messina, R.: Scan, attend and read: End-to-end handwritten paragraph recognition with MDLSTM attention. In: 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), pp. 1050–1055. IEEE, Piscataway, NJ (2017)

[9] Ares Oliveira, S., Seguin, B., Kaplan, F.: dhSegment: A generic deep-learning approach for document segmentation. In: 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 7–12. IEEE, Piscataway, NJ (2018). https://doi.org/10.1109/ICFHR-2018.2018.00011

[10] Kim, G., Hong, T., Yim, M., Nam, J., Park, J., Yim, J., Hwang, W., Yun, S., Han, D., Park, S.: OCR-free document understanding transformer. In: Computer Vision – ECCV 2022, pp. 498–517. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-19815-1_29

[11] Muehlberger, G., *et al.*: Transforming scholarship in the archives through handwritten text recognition: Transkribus as a case study. J. Doc. **75**(5), 954–976 (2019) https://doi.org/10.1108/JD-07-2018-0114

[12] Nockels, J., Gooding, P., Ames, S., Terras, M.: Understanding the application of handwritten text recognition technology in heritage contexts: A systematic

review of Transkribus in published research. Arch. Sci. **22**(3), 367–392 (2022) https://doi.org/10.1007/s10502-022-09397-0

[13] Nockels, J., Gooding, P., Terras, M.: The implications of handwritten text recognition for accessing the past at scale. J. Doc. **80**(7), 148–167 (2024) https://doi.org/10.1108/JD-09-2023-0183

[14] Ali, D., Milleville, K., Verstockt, S., Weghe, N., Chambers, S., Birkholz, J.M.: Computer vision and machine learning approaches for metadata enrichment to improve searchability of historical newspaper collections. J. Doc. **80**(5), 1031–1056 (2024) https://doi.org/10.1108/JD-01-2022-0029

[15] Ehrmann, M., Romanello, M., Najem-Meyer, S., Doucet, A., Clematide, S.: Overview of HIPE-2022: Named entity recognition and linking in multilingual historical documents. In: Experimental IR Meets Multilinguality, Multimodality, and Interaction (CLEF 2022). Lecture Notes in Computer Science, vol. 13390, pp. 423–446. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-13643-6_26

[16] González-Gallardo, C.-E., Boros, E., Girdhar, N., Hamdi, A., Moreno, J.G., Doucet, A.: Yes but... Can ChatGPT identify entities in historical documents? In: 2023 ACM/IEEE Joint Conference on Digital Libraries (JCDL), pp. 184–189. IEEE, Piscataway, NJ (2023). https://doi.org/10.1109/JCDL57899.2023.00034

[17] Kaplan, F., Di Lenardo, I.: Big data of the past. Front. Digit. Humanit. **4**, 12 (2017) https://doi.org/10.3389/fdigh.2017.00012

[18] Wang, P., Bai, S., Tan, S., Wang, S., Fan, Z., Bai, J., Chen, K., Liu, X., Wang, J., Ge, W., et al.: Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. Preprint at https://arxiv.org/abs/2409.12191 (2024)

[19] Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: LoRA: Low-rank adaptation of large language models. In: International Conference on Learning Representations (ICLR) (2022)

[20] Bai, S., Chen, K., Liu, X., Wang, J., Ge, W., Song, S., Dang, K., et al.: Qwen2.5-VL technical report. Preprint at https://arxiv.org/abs/2502.13923 (2025)

[21] Mathew, M., Karatzas, D., Jawahar, C.V.: DocVQA: A dataset for VQA on document images. In: IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), pp. 2200–2209 (2021)