Reading Order Inference for Complex Document Layouts

Iddo Hakim, Sharva Gogawale, Omer Ventura, Gal Grudka, Daria Vasyutinsky-Shapira, Berat Kurar-Barakat, and Nachum Dershowitz

School of Computer Science and AI, Tel Aviv University, Ramat Aviv, Israel
{iddoh,sharvag,omerventura,galgrudka}@mail.tau.ac.il
{dariashap,berat,nachumd}@tauex.tau.ac.il

Abstract. Reading order inference remains a critical bottleneck in the digitization of complex historical manuscripts, where pages contain multiple spatially interleaved reading streams, the canonical example being the *Glossa Ordinaria* layout, in which a central text is surrounded by commentaries that wrap around it in non-rectangular, non-convex regions. We present a *training-free*, graph-based framework: each OCR text line becomes a node in a directed candidate-transition graph, edges are scored by a weighted additive ensemble of two lightweight language-model signals (causal language model conditional likelihood and BERT next-sentence prediction, NSP; a third sentence-embedding signal was evaluated but did not improve reading order), and the global reading order is recovered as a degree-constrained directed path cover. To avoid the cascading "edge-theft" failures of greedy edge selection, we propose a max-regret inference rule that prioritizes commitments with high opportunity cost. We evaluate on synthetic Glossa Ordinaria grid layouts, on *23 ALTO page geometries* (10 historical source pages plus mirrored and flipped variants), and on a *140-page multi-column English subset of OmniDocBench*, comparing our method against the canonical recursive XY-cut (PaddleOCR PP-StructureV3) and two LayoutReader variants (layout-only and text+layout) on identical inputs. On wrap-around Glossa layouts our method recovers 95% of ground-truth successor edges on average vs. XY-cut's 50%; on the OmniDocBench multi-column subset it reaches 88% macro edge accuracy versus XY-cut's 75% and LayoutReader's 25%. The LayoutReader baselines transfer poorly due to a word-level vs. line-level granularity mismatch. We additionally verify mirror-invariance under horizontal and vertical page reflections: Our method changes by less than 1 percentage point, classical XY-cut by 2 points, and LayoutReader-T by up to 8 points.

Keywords: Reading order inference · Document layout analysis · Non-Manhattan layouts · Training-free · Max-regret inference · Historical manuscripts · XY-cut · LayoutReader · Mirror invariance

Fig. 1: Two examples of non-Manhattan layouts. *Left:* a printed Hebrew Bible page, main text flanked by an Aramaic translation, two commentaries wrapping around them, Masorah parva as abbreviated notes in an internal margin, and Masorah magna spanning the top and bottom margins, all automatically (and imperfectly) line-segmented. *Right:* a page of a manuscript (Codex Bodmer 25) of the Greek Bible with two regions of catena commentary.

1 Introduction

Reading order inference reconstructs the intended sequential flow of text elements detected by OCR. Accurate order is a prerequisite for searchable, continuous text streams and downstream document understanding; yet, it remains fragile whenever a page contains multiple spatially interleaved narratives.

For simple single-column pages, ordering is recovered by a top-to-bottom scan. Visually complex documents often contain *multiple partially independent reading streams*: parallel columns, marginalia, interlinear glosses, side notes, and figure captions, where *geometry alone is underdetermined*: several plausible successors may be nearby in space but belong to different narrative tracks.

Historical manuscripts amplify these difficulties. A canonical extreme is the *Glossa Ordinaria* page, where a central text is surrounded by multiple commentaries that wrap around it in non-rectangular, sometimes non-convex regions. Lines from different streams may be spatially interleaved, and human readers rely primarily on *semantic continuity* to remain within a stream. Classical geometric heuristics, including recursive XY-cut and proximity-based clustering, implicitly assume that spatial adjacency implies sequential adjacency, which often fails on such multi-text layouts [7,8,6]. We treat the canonical recursive XY-cut as our primary geometric baseline.

A natural alternative is to learn reading order from annotated documents, and modern multimodal models can be effective in-domain. But reading-order supervision is typically collected at the word/token level (e.g. ReadingBank for LayoutReader [14]), which differs from the OCR text-line atomic unit produced by historical OCR engines. As Section 6 shows, off-the-shelf LayoutReader variants, both layout-only and text-plus-layout (LayoutReader-T), transfer poorly to line-level historical inputs without retraining, even when fed the exact same boxes and text as our method. High-quality reading-order supervision for rare historical layouts is also expensive to obtain. This motivates *training-free* meth-

ods that exploit general linguistic priors already present in pretrained language models.

Goal. Recover reading order in complex, non-Manhattan, multi-stream layouts without annotated reading-order datasets or layout-specific fine-tuning.

Key idea. Pretrained language models, even small ones, encode strong local coherence signals. We model a page as a directed graph of candidate "next-line" transitions; nodes are OCR lines and edge weights quantify semantic continuity. Given this scored graph, we search for a globally consistent set of successor relations under the constraint that each line has at most one predecessor and one successor. Throughout, we isolate layout inference from OCR noise by evaluating on known line boxes with controlled text; end-to-end robustness to OCR errors is out of scope (Section 9).

Contributions

1. **Graph formulation.** Reading order as a degree-constrained directed path cover over OCR lines, enabling multiple disjoint reading streams under one-predecessor / one-successor constraints.
2. **Training-free semantic scoring.** A weighted additive ensemble of causal LM conditional likelihood and BERT next-sentence prediction. We additionally evaluate sentence-embedding cosine similarity and find it does not improve reading order (Section 7).
3. **Regret-based inference.** A max-regret edge-selection algorithm that reduces greedy myopia by prioritizing decisions with high opportunity cost.
4. **Expanded evaluation across synthetic, historical, and public-benchmark layouts.** Synthetic "Glossa Ordinaria layout" pages [12] created by interlacing distinct Project Gutenberg books[1] into non-convex topologies; *23 ALTO page geometries* [5] from 10 historical source pages and their mirrored and flipped variants, a substantial expansion over prior small-scale studies; and a 140-page English multi-column subset of OmniDocBench [9] for direct comparison on a public benchmark.
5. **Multi-baseline comparison.** We benchmark against the canonical recursive XY-cut [7] (PaddleOCR PP-StructureV3), LayoutReader [14] (layout-only), and LayoutReader-T (text+layout), on identical OCR-line inputs, and verify mirror-invariance under horizontal and vertical page reflections.

2 Related Work

Reading order inference sits at the intersection of page layout analysis and document structure recovery. While many documents admit a near-linear scan, visually complex pages may contain multiple interleaved reading streams (marginalia, parallel commentaries) for which geometric adjacency is an unreliable proxy for sequential adjacency.

[1] `https://www.gutenberg.org`

2.1 Geometric Heuristics

Early reading-order approaches rely on spatial regularities: projection-profile and recursive partitioning (XY-cut) [7], nearest-neighbor clustering (Docstrum) [8], and efficiency-tuned XY-cut variants [6]. These work well on Manhattan layouts but degrade when streams are spatially interleaved and region boundaries are non-convex, as in many historical manuscripts. We use the canonical recursive XY-cut as implemented in PaddleOCR's PP-StructureV3 pipeline as our geometric baseline (Section 6), so comparisons reflect a standard, citable implementation rather than a re-implementation.

2.2 Learning-Based and Semantic Approaches

Recent methods learn reading order as sequence generation or as pairwise relation prediction over layout entities, with multimodal inputs and supervised datasets [15,14,13]. LayoutReader [14] is trained on ReadingBank, a corpus of *word-level* layout and reading order from rendered DOCX files; we evaluate both its layout-only and text-plus-layout (LayoutReader-T) configurations on our OCR-line and OmniDocBench paragraph inputs (Section 6), where the word-to-line and word-to-paragraph granularity mismatch becomes a concrete limitation. End-to-end multimodal parsers such as Dots.OCR and PaddleOCR-VL operate at line granularity but bundle detection, recognition, and reading order into a single pipeline and do not expose their reading-order module for evaluation on externally supplied boxes, so a clean apples-to-apples comparison is not currently possible. Such systems are typically accurate in-distribution but depend on labeled reading order and may require fine-tuning for rare historical layouts. In NLP, sentence ordering and discourse coherence use local continuity cues and global optimization [1,4], but document reading order differs in three ways: the units are OCR lines (not sentences), the problem is constrained by 2D candidate adjacency, and multi-stream outputs (sets of disjoint paths) are often required rather than a single permutation. Our approach is training-free, uses pretrained language models as local semantic oracles, and combines them with explicit graph inference under degree constraints.

2.3 Reading-Order Representation and Evaluation

Reading order may be non-linear or partially specified, and evaluation becomes non-trivial when segmentation differs across systems [2]. Public benchmarks such as OmniDocBench [9] cover diverse modern documents (academic papers, financial reports, multi-column layouts) and evaluate ordering over document components or paragraph-like regions rather than OCR lines. For our wrap-around, non-Manhattan target setting, region decomposition is itself unreliable: regions may merge or split incorrectly even when individual line boxes remain usable (a Sayre's-paradox-like situation, where correct region ordering needs correct regions, but finding them depends on the reading streams). We therefore evaluate directly over OCR lines and successor relations on our 23-page ALTO corpus,

and complement this with an evaluation on the multi-column English subset of OmniDocBench at its native paragraph granularity (Section 7.4). The graph formulation operates on (bbox, text) tuples regardless of unit size, so both granularities are handled with the same model. ALTO and similar archival formats can encode layout structure and optionally include ordering, but reading order is frequently missing or unreliable in complex manuscripts, motivating automatic inference.

3 Problem Formulation

We assume an OCR system produces a set of N text lines.

3.1 Candidate Transition Graph

We build a directed candidate graph $G = (V, E_{\text{cand}})$ with $V = \{1, \ldots, N\}$. An edge $(u, v) \in E_{\text{cand}}$ denotes that line v is a plausible successor of line u. Candidate generation controls both runtime and ambiguity; we use permissive "next-column" candidate sets to stress-test inter-column continuation decisions.

- *In artificial grid layouts*, each node belongs to a discrete column. For each node u, we connect it to *all* nodes in the next column to the right. This produces a dense candidate set in which geometry alone cannot determine which inter-column transition is the correct continuation.
- *In realistic layouts*, where nodes are derived from OCR text bounding boxes, we mimic the same stress test using bounding boxes: for each line u, we connect it to *all* lines below u (same-column successors) and to *all* lines whose left edge lies to the right of u's right boundary (next-column successors). This permissive set makes both same-column continuations and wrap-around "next-column" transitions candidates, so geometry alone cannot determine the correct successor in complex page geometries.

3.2 Degree Constraints and Path Cover

Reading order within each stream is a directed path: each line has at most one successor and at most one predecessor. We seek a subset $E \subseteq E_{\text{cand}}$ that maximizes the total score:

$$\max_{E \subseteq E_{\text{cand}}} \sum_{(u,v) \in E} S(u, v) \quad \text{s.t.} \quad \deg^+(u) \leq 1,\ \deg^-(v) \leq 1\ \forall u, v. \tag{1}$$

where $S(u, v)$ is the semantic continuity score for the edge. Under the degree constraints alone, a maximizing edge set decomposes into a collection of disjoint directed paths and (in general) directed cycles. Because cycles do not correspond to valid reading orders, our inference procedures explicitly avoid cycle creation and return an acyclic path cover, possibly with multiple disjoint streams.

4 Training-Free Semantic Edge Scoring

For each candidate edge (u, v) we compute a semantic continuity score $S(u, v)$ using pretrained models *without fine-tuning*. Scores are computed independently per edge and cached, decoupling expensive model inference from the downstream search algorithm.

4.1 Causal Language Model Conditional Likelihood

Let a causal language model (CLM) define token log-likelihood $\log P_{\mathrm{CLM}}(\cdot)$ under a fixed tokenizer. We instantiate the CLM with EleutherAI/pythia-410M and compute all CLM scores on a GPU in inference mode without fine-tuning. Given two line fragments t_u and t_v, we score the per-token continuation likelihood of t_v conditioned on t_u:

$$s_{\mathrm{clm}}(u, v) = \frac{1}{|t_v|} \log P_{\mathrm{CLM}}(t_v \mid t_u),$$

where $|t_v|$ denotes the number of tokens in t_v. In practice, we compute this by concatenating the token sequences $[t_u; t_v]$, running one forward pass, and summing log-probabilities only over the token positions belonging to t_v (i.e., masking out the prefix t_u) [10].

Context Truncation. Because causal models have a finite context window, we truncate t_u to the most recent L tokens (we use a fixed L across all experiments) so that $[t_u; t_v]$ fits within the model context.

Optional Frequency Normalization. Rare-token artifacts can inflate s_{clm} by over-rewarding unlikely but highly specific continuations. We therefore optionally normalize by the unconditional likelihood of t_v:

$$\widetilde{s}_{\mathrm{clm}}(u, v) = s_{\mathrm{clm}}(u, v) - \kappa \cdot \frac{1}{|t_v|} \log P_{\mathrm{CLM}}(t_v), \tag{2}$$

with $\kappa \in [0, 1]$ controlling the correction strength. When $\kappa = 0$, no normalization is applied. We use a different symbol (κ) here than for the ensemble blending weights below (α, β, γ, see Sec. 7) to make the two roles distinct.

4.2 Next Sentence Prediction

BERT's next sentence prediction (NSP) head produces $p_{\mathrm{nsp}}(u, v) \in (0, 1)$ indicating whether t_v is a plausible continuation of t_u [3]. NSP is the binary sentence-continuity objective on which BERT was pretrained, which makes it a standard, training-free probe for precisely the adjacency relation a reading-order edge encodes. We convert this to an additive score in log space:

$$s_{\mathrm{nsp}}(u, v) = \log\big(\max(\varepsilon, p_{\mathrm{nsp}}(u, v))\big),$$

where ε is a small constant to avoid $\log(0)$.

4.3 Sentence Embedding Similarity

We compute embeddings h_u, h_v using a sentence-transformer encoder [11] and define:

$$s_{\text{sim}}(u, v) = \cos(h_u, h_v).$$

Since cosine similarity can be negative, we use it directly as a bounded additive signal (no logarithm). We include this signal for completeness; the ablation in Section 7 shows it does not improve reading order, so all reported results set $w_{\text{sim}} = 0$.

4.4 Weighted Additive Ensemble

We combine all signals with a weighted additive ensemble:

$$S(u, v) = w_{\text{clm}} \cdot \widetilde{s}_{\text{clm}}(u, v) + w_{\text{nsp}} \cdot s_{\text{nsp}}(u, v) + w_{\text{sim}} \cdot s_{\text{sim}}(u, v) - \lambda \cdot d_{\text{Man}}(u, v),$$

where $w_{\text{clm}}, w_{\text{nsp}}, w_{\text{sim}} \geq 0$ and $\lambda \geq 0$. We use Manhattan distance between bounding-box centers, $d_{\text{Man}}(u, v) = |c_x(u) - c_x(v)| + |c_y(u) - c_y(v)|$, as an optional regularizer that discourages long-distance "teleportation" edges in very dense candidate graphs. Unless stated otherwise, we set $\lambda = 0$ and rely on candidate gating for locality.

Fixed-Parameter Transfer. The ensemble weights are selected once by a sweep on the synthetic grid task (Section 7) and then *transferred unchanged* to all ALTO and OmniDocBench experiments. The sweep selects $w_{\text{sim}} = 0$ (the embedding never helped and degraded results as its weight grew, Section 7), so the deployed scorer uses only $(w_{\text{clm}}, w_{\text{nsp}})$. We do not re-tune per dataset or per document: the Section 6 numbers are a fixed-parameter transfer test, not in-sample fits.

5 Inference Algorithms

Given candidate edges E_{cand} and scores $S(u, v)$, we infer a degree-constrained directed path cover: each node has at most one successor and at most one predecessor, yielding multiple disjoint reading streams. The maximum-weight degree-constrained acyclic path cover of Eq. (1) is a combinatorial problem we do not solve to optimality; the algorithms below are fast heuristics, and max-regret in particular carries no optimality guarantee, only reordering which decisions are committed first.

5.1 Greedy Selection Baselines

We consider two greedy constructions that repeatedly select high-scoring edges subject to feasibility: (i) *local greedy*, which processes nodes in a fixed scan order and assigns each node its best feasible successor, and (ii) *global greedy*, which repeatedly commits the highest-scoring feasible edge overall. While fast, greedy methods often fail by *edge theft*: an early incorrect edge consumes a node's in-degree, making the correct predecessor later unreachable and causing cascading errors.

Algorithm 1 Max-regret inference (cycle-avoiding path cover)

Input: candidate edges E_{cand}, scores $S(u, v)$
Output: edge set E (acyclic path cover)
$E \leftarrow \emptyset$; initialize all nodes as unassigned-in and unassigned-out
while there exists an unresolved node with at least one feasible outgoing candidate **do**
  for all unresolved nodes u **do**
    Let $\mathcal{C}(u) = \{v : (u, v) \in E_{\text{cand}}$ and (u, v) is feasible$\}$
    Sort $\mathcal{C}(u)$ in descending order of $S(u, v)$ to get $v_1, v_2, \ldots$
    $\text{Regret}(u) \leftarrow S(u, v_1) - S(u, v_2)$ {or 0 if $|\mathcal{C}(u)| < 2$}
  end for
  $u^\star \leftarrow \arg\max_u \text{Regret}(u)$ {tie-break by $S(u, v_1)$}
  Commit the best feasible edge $(u^\star, v)$ by scanning $v_1, v_2, \ldots$ until one does not create a cycle
  Add the committed edge to E and mark $u^\star$ assigned-out and v assigned-in
end while

return E

5.2 Max-regret Edge Selection

To reduce greedy myopia, we prioritize decisions with high opportunity cost. For each unresolved node u, let v_1 and v_2 denote the top two *feasible* outgoing candidates by score, where feasibility means: (a) u has no assigned successor, (b) v has no assigned predecessor, and (c) adding (u, v) would not create a directed cycle in the current partial solution. We define regret as:

$$\text{Regret}(u) = S(u, v_1) - S(u, v_2). \tag{3}$$

If a node has fewer than two feasible candidates, we set $\text{Regret}(u) = 0$ so that low-ambiguity nodes (e.g., boundary nodes with a single remaining option) do not dominate the schedule.

At each iteration, we select the node with maximum regret and commit its best feasible edge. If the best edge would form a cycle, we fall back to the next-best feasible candidate for that node.

Complexity. With N nodes and M candidate edges, each iteration selects one edge and updates feasibility. A straightforward implementation runs in $O(M \log d)$ time if per-node candidate lists are pre-sorted (d is average out-degree), with near-constant-time cycle checks using union-find-style successor tracing.

6 Experimental Setup

To isolate layout inference from OCR recognition noise, we evaluate on pages where the set of line bounding boxes is known and text content is controlled.

We use three complementary settings: (i) synthetic grid layouts for controlled ablations over topology and candidate density, (ii) authentic historical page geometries extracted from ALTO XML [5] that preserve realistic, irregular line placements and non-rectangular region boundaries, and (iii) a multi-column English subset of the OmniDocBench public benchmark [9] for an apples-to-apples comparison on a community-recognized dataset.

6.1 Synthetic Grid Complex Layouts

We instantiate layouts on an $R \times C$ grid partitioned into disjoint regions (text streams) with non-convex wrap-around topologies (L-, H-, Y-, and O-shapes). Nodes in each region are populated with consecutive lines from a distinct Project Gutenberg book, producing spatially interleaved but semantically independent streams. Ground-truth reading order inside each region is a fixed deterministic traversal (row-major or column-major, held constant per experiment).

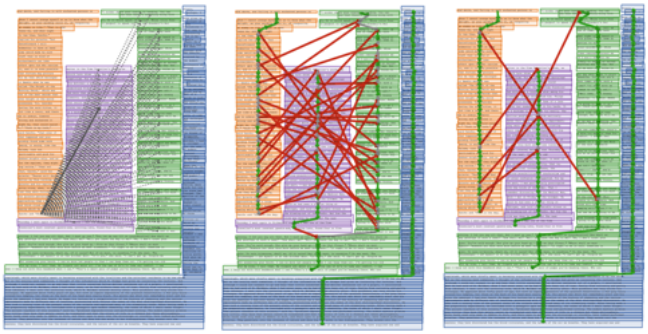
6.2 Historical Geometries from ALTO XML

We parse `TextLine` elements (and polygons when available) from ALTO XML. Our corpus comprises *23 page geometries*: 10 historical source pages, 2 horizontal mirrors, 10 vertical (top-bottom) flips, and one two-column ALTO as a Manhattan sanity check. The mirror and flip variants are produced by reflecting bounding-box coordinates and re-deriving each group's reading order under a column-major edge-aligned clustering rule, preserving the *semantic* reading order of the source page while breaking directional bias; they are used in Section 7.3 as a mirror-invariance test.

Each node is populated with consecutive Project Gutenberg words up to the node's box width; the text then continues into the next node in the region. RTL source layouts (e.g., Hebrew) are canonicalized to LTR by mirroring x-coordinates ($x' = W - x$). The synthetic grid generator and the line-level reading-order annotations for all 23 ALTO pages will be released publicly.

6.3 Metrics

We report *edge accuracy*: the fraction of non-terminal lines whose predicted successor matches the ground-truth successor. The last line of each stream is excluded since it has no successor. Errors are decomposed into *same-stream skips* (predicted successor stays in the correct stream but skips the immediate GT successor) and *cross-stream links* (predicted successor belongs to a different stream). Across the 23-page ALTO corpus, edge accuracy is computed over thousands of GT successor edges and tens of thousands of candidate transitions (e.g., Fig. 2 alone contributes 5,783 candidate edges). For OmniDocBench we additionally report the Normalized Edit Distance metric defined in the OmniDocBench paper [9] for direct comparability against its published leaderboard.



(a) (b) (c)

Fig. 2: Next-column stress test on an ALTO-based graph (5,783 candidate edges). (a) Candidate generator connects each line to all lines to the right of its right boundary. (b) Basic regret produces many spurious links. (c) Additive ensemble regret recovers the critical inter-stream continuation.

7 Results

We compare our full method, the *Weighted Additive Ensemble with Max-Regret inference*, against baselines spanning both the geometric and learning-based traditions. On synthetic grids we use a *Column Concatenation* baseline (nodes grouped by column, ordered top-to-bottom within columns, concatenated left-to-right). On ALTO pages we benchmark against three literature baselines: *XY-Cut* [7] as implemented in PaddleOCR PP-StructureV3 (`recursive_xy_cut`); *LayoutReader* (LR) [14], the layout-only LayoutLMv3 model (`hantian/layoutreader`); and *LayoutReader-T* (LR-T), the text+layout seq2seq variant from `nielsr/layoutreader-readingbank`, fed the same OCR-line boxes and the same line text that our pipeline uses. All four methods are evaluated on identical line-level inputs per page.

7.1 Synthetic Grid Results

On the 8×8 instance, performance rises from 50 to 59.2 ± 0.4 correctly recovered successor edges out of 61 (geometry-only $\rightarrow$ ours). On 16×16, it rises from 230 to 235.4 ± 8.2 out of 253 (Table 1). The smaller margin on the larger grid is expected: as the synthetic layout grows, a larger fraction of the page becomes trivially recoverable by column-wise ordering, leaving less headroom for semantic re-ranking.

7.2 ALTO Results

Table 2 reports edge-accuracy averages over the 11 pages in our main ALTO test corpus (10 historical source pages plus Sanity2Col), grouped by reading-order

Table 1: Synthetic grid: geometry-only baseline vs. ours, mean ± std over 5 seeds.

		Column Concatenation		Ours	
Dataset	GT edges	Correct	Acc. (%)	Correct	Acc. (%)
8×8	61	50	82.0	59.2 ± 0.4	97.0 ± 0.7
16×16	253	230	90.9	235.4 ± 8.2	93.0 ± 3.2

Table 2: ALTO 11-page corpus: per-regime average edge accuracy (%). Best per row in **bold**.

Regime	n	Ours	XY-Cut	LR	LR-T
Manhattan-ceiling	4	96.0	**100.0**	27.9	26.6
Mixed / near-Manhattan	3	95.3	**96.1**	15.7	18.3
Wrap-around / Glossa	4	**94.8**	49.7	22.3	14.1
All 11	11	**95.4**	80.5	22.5	19.8

regime. The 10 vertical-flip and 2 horizontal-mirror variants are reserved for the robustness analysis in Section 7.3.

A clear three-regime picture emerges. On *Manhattan-ceiling* pages (Asher, Pesach_v2, O_IE104261952, Sanity2Col), XY-Cut recovers 100% of GT edges; our method averages 96.0%, within 11 percentage points on each individual page. These pages confirm that the XY-Cut implementation is sound and expose the only regime in which our method is not the top performer: the semantic ensemble occasionally re-orders adjacent same-column lines when short line-level text fragments make local continuation probabilities ambiguous.

On *wrap-around / Glossa Ordinaria* pages (Lieberman, Y_IE20854727, O_IE102667218, Y_IE30700205), where the reading order does not respect simple whitespace partitions, the gap between our method and XY-Cut is large: 94.8% vs. 49.7% on average. On Lieberman we recover *100.0%* of GT edges vs. XY-Cut's *39.6%*; on Y_IE20854727, *98.9%* vs. *40.5%*. The failure mode of XY-Cut on these pages is the bridging-gloss mechanism analyzed in Section 8.

On *mixed / near-Manhattan* pages (Berakhot2B, L_IE104330339, Vatican Palatina), XY-Cut recovers 91 – 99% of GT edges and our method matches it within ±4 percentage points. Across all 11 pages, the LayoutReader variants are uncompetitive: LR averages 22.5% (range 0 – 55), LR-T 19.8% (range 6 – 34). The text channel does not close the gap; the granularity mismatch with ReadingBank's word-level supervision is structural (Section 8).

7.3 Mirror-Invariance Robustness

Reading-order methods should be *geometry-equivariant*: mirroring or flipping a page should not change which line follows which under the (mirrored) reading order. Methods trained on left-to-right English may carry a directional prior

that violates this. We evaluate this on two horizontal mirrors (Y_IE20854727, Y_IE30700205) and 10 vertical flips of the source pages; ground-truth reading order is re-derived under a column-major rule with edge-aligned clustering so the *semantic* order is preserved. The worst-case $|\Delta\text{Acc.}|$ across mirrors is 0.6 percentage points for our method, 2.1 percentage points for XY-Cut, 8.7 percentage points for LR, and 7.5 percentage points for LR-T. LR-T *is not* mirror-invariant, consistent with its LTR-English training distribution and a concrete instance of the directional-prior limitation we discuss in Section 8.

7.4 Public-Benchmark Evaluation: OmniDocBench

We further evaluate on a 140-page English multi-column subset of OmniDocBench [9] (92 double-column, 48 three-column; sources: 38 academic papers, 36 exam papers, 32 newspapers, 17 magazines, 15 books, 2 textbooks; 2,085 ground-truth successor edges). OmniDocBench annotates reading order at the paragraph-block level rather than the OCR-line level that is the primary target of this paper, so we evaluate at its native granularity (each annotated block is one node in our graph). The graph formulation operates on (bbox, text) tuples regardless of unit size, so no methodological change is required; the same ensemble weights from the synthetic-grid sweep are transferred unchanged.

Table 3 reports per-source edge accuracy alongside the Normalized Edit Distance metric used by the OmniDocBench paper itself [9]. Two observations consistent with the ALTO picture: (i) our method wins by a wide margin on the subsets where geometric partition struggles, exam papers (+54.7 percentage points over XY-Cut) and academic literature (+13 percentage points), which are the multi-column academic-style pages identified as challenging in prior reviews. (ii) XY-Cut wins on strongly rectilinear layouts (newspapers, three-column books) where its projection profile finds clean gutters. LayoutReader underperforms uniformly, even on subsets close to its ReadingBank training distribution (academic literature: 37.2%), confirming the granularity-mismatch argument of Section 8: word-level supervision does not transfer to coarser units, whether OCR lines or paragraph blocks.

The OmniDocBench leaderboard is populated entirely by end-to-end image-input systems[2]; ours is the first reported result for the training-free (bbox + text) regime, with NED 0.24 on the multi-column subset and 0.10 on exam papers. We do not claim parity with the image-input leaderboard (e.g., MinerU at 0.08): those systems solve a different, full-image task, whereas our method uses only boxes and text.

7.5 Ablations

We sweep (α, β, γ) over 120 configurations on the 16×16 synthetic grid with 5 seeds and adopt $\alpha = 1.0, \beta = 0.2$, transferred unchanged to all ALTO and

[2] MinerU, Mathpix, Marker; GOT-OCR, Nougat; GPT-4o, Qwen2-VL-72B, InternVL2-76B [9]. Best reported English NED 0.08 (MinerU), 0.12 (Qwen2-VL-72B), 0.13 (GPT-4o).

Table 3: OmniDocBench English multi-column subset (140 pages, 2,085 GT successor edges). Edge accuracy (%, higher is better) and Normalized Edit Distance (NED, lower is better) per data source. Edge accuracy is over the 140 pages all methods processed; NED is over the full 142-page subset. Best per row in **bold**.

		Edge accuracy (%)			NED	
Source	n	Ours	XY-Cut	LR	Ours	XY-Cut
academic_lit.	38	**93.4**	80.4	37.2	**0.14**	0.27
exam_paper	36	**96.9**	42.2	10.9	**0.10**	0.62
newspaper	32	74.2	**96.2**	8.9	0.51	**0.07**
magazine	17	**89.1**	82.6	26.0	0.33	**0.30**
book	15	82.0	**85.8**	53.7	0.19	**0.18**
textbook	2	85.7	**100.0**	36.9	0.13	**0.00**
Macro	140	**88.0**	75.3	24.6	**0.24**	0.30

Table 4: Greedy vs. max-regret inference on the 16×16 grid using additive ensemble scores ($\alpha = 1.0$, $\beta = 0.2$), averaged over 5 seeds.

Method	Correct	Acc. (%)	Cross-stream	Same-stream
Column Concatenation baseline	230	90.9	—	—
Greedy inference	141.6 ± 7.8	56.0	27.8	85.6
Max-Regret inference	235.4 ± 8.2	93.0	4.4	9.2

OmniDocBench experiments; the additive ensemble with max-regret recovers 235.4 ± 8.2 of 253 GT edges (93.0%). Performance is much more sensitive to α and β than to γ in the tested range. The swept optimum in fact sets $\gamma = 0$: the sentence-embedding term gives no improvement, and when we force it active with a real similarity matrix its effect is at best neutral and degrades accuracy monotonically as γ grows (across the ALTO layouts edge accuracy never rises and falls steadily at larger γ). The two language-model signals therefore carry the method, the causal-LM term (α) dominant and NSP (β) a consistent further gain, so the deployed scorer omits the embedding. With the additive scores fixed at these weights, greedy edge selection recovers only 141.6 ± 7.8 of 253 GT edges (56.0%), below the geometry-only baseline (90.9%) and far below max-regret (93.0%). Replacing greedy with max-regret gains +37.0 percentage points on average. The error decomposition in Table 4 shows that greedy produces 27.8 cross-stream and 85.6 same-stream errors per run; max-regret reduces both to 4.4 and 9.2 respectively.

Runtime. On a single NVIDIA A40, per-page inference averages 93.5 s (median 88 s; 10.5 – 290 s) and peaks at 6.8 GB. Cost is dominated by per-edge semantic scoring, not the path search, and scales with candidate-edge density, so dense pages with many short, adjacent lines are the most expensive.

8 Analysis

This section summarizes dominant failure mechanisms in dense multi-stream layouts and explains why max-regret inference improves robustness and why the baselines we compare against fail in structurally specific ways.

8.1 Greedy vs. Max-Regret

Constrained greedy enforces the same degree constraints as our main solver but commits edges by absolute score (the globally best feasible edge at each step). In dense candidate graphs this is vulnerable to *premature commitment*: an early high-scoring but incorrect edge (u, v) claims target v, renders v's true predecessor infeasible, and displaces other nodes onto weaker alternatives, triggering cascades of cross-stream links rather than isolated local mistakes. Max-regret retains the same feasibility constraints but reorders which decisions to commit first, prioritizing high-opportunity-cost sources (nodes whose best and second-best outgoing candidates differ greatly in score, Eq. (3)) because delaying these "must-take" commitments would let a competing node claim their best target.

8.2 Baseline Failure Modes

XY-cut needs a whitespace gutter that does not exist. Recursive XY-cut [7] alternates horizontal and vertical projection-profile splits, choosing a cut wherever there is a contiguous run of background larger than a configured gap threshold. This requires a clean whitespace gutter fully separating the cells to be cut. On Glossa Ordinaria and similar bridging-gloss pages, one or more boxes (a wide gloss line, a marginal commentary that wraps across the gutter, or a full-width banner) intersect every candidate vertical cut, so the projection profile never reaches background between would-be columns. The recursion then either fails to split or splits on the wrong axis, after which every box in the misclassified region receives an incorrect successor in the terminal scan order. Our method bypasses both failure modes because it never decomposes the page into rectangles; a bridging row or shared whitespace band is just one more candidate node.

LayoutReader is trained at the wrong granularity. LayoutReader [14] and the LR-T text+layout variant are trained on ReadingBank, which provides reading-order supervision over *words*. Our inputs are OCR text *lines* or paragraph blocks, each spanning many words and several pixels of vertical extent, so box geometry, input sequence length, and box-to-token alignment all differ from training. We observe a consistent 6–34% edge-accuracy band across all ALTO pages with LR-T no better than LR despite being fed text. The mismatch is structural, not semantic; the text channel cannot make a word-level seq2seq emit correct line or paragraph successors. The same mechanism explains the mirror-invariance failure in Section 7.3: a model whose training distribution is LTR-English at word granularity has no opportunity to learn reading order as a function of *relative* layout.

8.3 Semantic Failure Modes

Three failure modes recur. *Context-insensitive continuation*: fragments score high simply because they are broadly probable, not because they truly follow t_u; the marginal subtraction in Eq. (2) discounts globally likely fragments and emphasizes context-specific lift. *Cross-stream hallucination*: generic cues (quotation marks, dialogue verbs) appear across multiple streams; combining NSP with CLM provides a complementary signal that reduces these merges at high candidate density. *Semantic teleportation*: permissive candidate graphs allow semantically similar but spatially distant lines to link; candidate gating bounds this in practice.

9 Limitations

Computational cost and score overlap. Our approach is training-free but not cost-free: scoring candidate edges requires running pretrained models over many line pairs, with cost scaling in the number of candidates per node. Candidate gating and score caching make this tractable in practice, but very dense candidate graphs (weak geometric constraints or all-to-all candidates) remain expensive. Performance is also bounded by *score overlap*: short or semantically generic line fragments may produce near-identical scores for correct and incorrect successors, making some errors irreducible without additional signals such as typography, indentation, or learned layout priors. This is most acute for very short segments (marginalia, page numbers, running heads) and degraded OCR, where every textual score weakens at once; fusing the semantic scores with layout and visual cues, so the ensemble can fall back on geometry where language is uninformative, is a natural extension.

Scope of evaluation. We deliberately isolate layout inference from OCR noise by using clean line boxes and controlled English text. End-to-end behaviour under realistic OCR error rates therefore remains uncharacterized; a CER/WER sweep on our populated ALTO pages is the most direct next-step robustness study. The semantic stack (CLM + NSP) is also validated only on English: right-to-left scripts, top-to-bottom and vertical writing systems, and historical language distributions with poor tokenizer coverage are out of scope, though the graph formulation and inference rule are unchanged under such substitutions. Our public-benchmark evaluation covers the multi-column English subset of OmniDocBench (Section 7.4) at its native paragraph granularity; extending this to the full OmniDocBench corpus, to non-English subsets, and to line-level evaluations on other public benchmarks is straightforward future work. Broader collections of annotated manuscripts and complex business documents, such as the Tobacco-800 collection, would further stress-test robustness on diverse real-world layouts. Our ensemble weights $(w_{\text{clm}}, w_{\text{nsp}})$ are likewise selected once on the synthetic grid and transferred unchanged; per-layout-family tuning could improve individual regimes at the cost of the cleanest transfer story.

Baseline-comparison gap. End-to-end document parsers such as Dots.OCR and PaddleOCR-VL operate at line granularity but bundle detection, recognition, and reading order in a single pipeline, and do not expose their reading-order module for evaluation on externally supplied OCR boxes. A clean apples-to-apples comparison on identical inputs is therefore not currently possible; doing so would require either reimplementing their reading-order head or running both systems on their own full-pipeline outputs, with the attendant segmentation differences and partial-credit issues.

10 Conclusion

We presented a training-free framework for reading order inference in complex, non-Manhattan layouts. OCR lines become nodes in a candidate transition graph; edges are scored by a weighted additive ensemble of CLM likelihood and NSP.

On top of these local scores, a max-regret inference strategy outperforms greedy selection by prioritizing high-opportunity-cost decisions and reducing catastrophic "edge theft" failures under one-predecessor / one-successor constraints.

We evaluate across synthetic grids, 23 ALTO historical geometries, and a 140-page OmniDocBench multi-column subset [9], against the canonical recursive XY-cut (PaddleOCR PP-StructureV3) and two LayoutReader variants. Our method dominates on layouts where geometric partition is unreliable (Glossa, exam papers, academic), reaching 88.0% accuracy (NED 0.24) on OmniDocBench versus XY-cut's 75.3% (0.30) and LayoutReader's 24.6%, while structurally rectilinear layouts (newspapers, clean Manhattan) remain in XY-cut's strength regime. Our pipeline is also mirror-invariant to within ± 0.6 percentage points under horizontal and vertical reflections, whereas LayoutReader-T is not.

Future work will focus on integrating lightweight visual cues, such as indentation and font styles, improving robustness against OCR noise, expanding support for multilingual historical texts, and extending OmniDocBench to the full corpus and other public benchmarks.

Acknowledgments. We thank the reviewers for their suggestions. This research was funded in part by the European Union as part of "MiDRASH" [`https://www.midrash.eu`] (ERC project no. 101071829, with principal investigators: Daniel Stökl Ben Ezra, EPHE-PSL; Nachum Dershowitz, Tel Aviv University; Judith Olszowy-Schlanger, EPHE-PSL; and Avi Shmidman, Bar-Ilan University). 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

1. Barzilay, R., Lapata, M.: Modeling local coherence: An entity-based approach. Computational Linguistics **34**(1), 1–34 (2008). https://doi.org/10.1162/coli.2008.34.1.1
2. Clausner, C., Pletschacher, S., Antonacopoulos, A.: The significance of reading order in document recognition and its evaluation. In: Proc. 12th Int. Conf. on Document Analysis and Recognition (ICDAR). pp. 688–692 (2013). https://doi.org/10.1109/ICDAR.2013.141
3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proc. NAACL-HLT. pp. 4171–4186. ACL (2019). https://doi.org/10.18653/v1/N19-1423
4. Li, J., Hovy, E.: A model of coherence based on distributed sentence representation. In: Proc. EMNLP. pp. 2039–2048. ACL (2014). https://doi.org/10.3115/v1/D14-1218
5. Library of Congress: ALTO: Technical metadata for layout and text objects. https://www.loc.gov/standards/alto/ (2022)
6. Meunier, J.L.: Optimized XY-cut for determining a page reading order. In: Proc. 8th Int. Conf. on Document Analysis and Recognition (ICDAR). pp. 347–351 (2005). https://doi.org/10.1109/ICDAR.2005.182
7. Nagy, G., Seth, S.C.: Hierarchical representation of optically scanned documents. In: Proc. 7th Int. Conf. on Pattern Recognition (ICPR). vol. 1, pp. 347–349 (1984)
8. O'Gorman, L.: The document spectrum for page layout analysis. IEEE TPAMI **15**(11), 1162–1173 (1993). https://doi.org/10.1109/34.244677
9. Ouyang, L., Qu, Y., Zhou, H., Zhu, J., et al.: OmniDocBench: Benchmarking diverse pdf document parsing with comprehensive annotations. In: Proc. CVPR (2025)
10. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. Tech. rep., OpenAI (2019)
11. Reimers, N., Gurevych, I.: Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In: Proc. EMNLP-IJCNLP. pp. 3982–3992. ACL (2019). https://doi.org/10.18653/v1/D19-1410
12. Rozenberg, M., Munk, M., Kainan, A.: A Talmud page as a metaphor of a scientific text. Int. J. Qualitative Methods **5**(4), 30–44 (2006). https://doi.org/10.1177/160940690600500403
13. Wang, R., Fujii, Y., Bissacco, A.: Text reading order in uncontrolled conditions by sparse graph segmentation. In: Proc. Int. Conf. on Document Analysis and Recognition (ICDAR). pp. 3–21. Springer (2023). https://doi.org/10.1007/978-3-031-41731-3_1
14. Wang, Z., Xu, Y., Cui, L., Shang, J., Wei, F.: LayoutReader: Pre-training of text and layout for reading order detection. In: Proc. EMNLP. pp. 4735–4744. ACL (2021). https://doi.org/10.18653/v1/2021.emnlp-main.389
15. Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., Zhou, M.: LayoutLM: Pre-training of text and layout for document image understanding. In: Proc. 26th ACM SIGKDD. pp. 1192–1200 (2020). https://doi.org/10.1145/3394486.3403172