ChainAlign: Cross-Lingual Sentence Alignment via Sparse Co-linear Chaining

Owen N. Goldner
Amazon
vgoldner@amazon.com

Connor C. Gramazio
Amazon
cgramazi@amazon.com

Abstract

Cross-lingual sentence alignment establishes correspondence between a document and its translation, where correspondence is not always complete, one-to-one, or monotonic. Current embedding-based aligners score sentence pairs with a multilingual encoder, then trace a single monotonic path across the full alignment grid. This forced path cascades errors through non-parallel or reordered content. We present ChainAlign, which instead adapts *seed-chain-extend* from computational genomics: it seeds multi-resolution candidate anchors from multilingual embeddings, selects the maximum-weight non-crossing subset by co-linear chaining in $O(N \log N)$ via a Fenwick tree, and fills the residual gaps with local dynamic time warping. Because the chain imposes no coverage requirement, non-parallel content becomes unaligned gaps rather than cascading errors, and multi-chain extraction handles reordering that a monotonic path cannot represent. On three benchmarks across four language pairs, ChainAlign achieves the highest strict and lax F1, ahead of Bertalign, SentAlign, and Vecalign. Evaluation shows the sparse formulation outperforms exhaustive dynamic programming using the same edge weight function.

1 Introduction

We present ChainAlign, a cross-lingual sentence alignment system that adapts the seed-chain-extend paradigm from computational genomics to establish sentence-level correspondences between a document and its translation. Translation is not always one-to-one. Most sentences correspond one-to-one between source and target, but translators also split and merge them, and occasionally insert, delete, or reorder content (for example, adding cultural context absent from the source). A sentence aligner is therefore only as accurate as its ability to model correspondences that are not guaranteed to be complete, one-to-one, or monotonic. These departures, not the one-to-one majority, are where current aligners still struggle, and they grow more common as training data shifts from purpose-collected parliamentary transcripts toward diffuse, web-mined, and literary sources; alignment quality increasingly bounds the machine translation and multilingual models built on top of it.

Many sentence aligners used in machine translation share a two-stage design: a multilingual encoder scores candidate pairs, and a search routine selects a globally coherent set of correspondences. The encoders have largely converged on LaBSE-style embeddings (Feng et al., 2022), and modern methods differ mainly in the search step. Every current search routine, whether the dynamic time warping of Vecalign (Thompson and Koehn, 2019), the two-pass dynamic program of Bertalign (Liu and Zhu, 2023), or the shortest-path search of SentAlign (Steingrimsson et al., 2023), traces one monotonic path across the *entire* alignment grid at a cost quadratic in the sentence counts. This forced path is precisely what fails on non-parallel or reordered content, where a poor local choice in an ambiguous region propagates forward and displaces otherwise-confident matches. Computational genomics has long solved a structurally identical problem: establishing partial, rearranged correspondences between sequences at chromosome scale. Its answer, *seed-chain-extend* (Myers and Miller, 1995; Li, 2018), finds confident local matches, chains a co-linear subset of them, and aligns densely only within the residual gaps.

ChainAlign brings this paradigm to cross-lingual sentence alignment: multilingual embeddings supply seeds, an $O(N \log N)$ Fenwick-tree chain (Fenwick, 1994) selects the optimal non-crossing anchor set, and local dynamic time warping fills residual gaps. Unlike heuristic anchoring (Kraif, 2024), anchor selection here is an optimization problem with an exact solution: co-linear chaining reduces to a weighted longest-increasing-subsequence prob-

lem solvable in $O(N \log N)$, so the chain is a global optimum rather than the output of a filter.

Across three benchmarks and four language pairs, ChainAlign outperforms Bertalign, SentAlign, and Vecalign on strict and lax F1. Its advantage is sharpest off the clean case: as non-parallel content is injected into a document it stays accurate (losing under 2% strict F1 at 40% injection) because unmatched spans become gaps rather than forced errors, and its multi-chain variant recovers reordering that a single monotonic path cannot represent.

Our contributions are threefold:

1. Seed-chain-extend, the standard genome-alignment paradigm, transfers to cross-lingual sentence alignment as an $O(N \log N)$ Fenwick-tree chaining algorithm.

2. Sparse co-linear chaining outperforms exhaustive dynamic programming on sentence alignment: against a DTW baseline using the same edge weight function, it gains 1.9, 0.6, and 0.5 strict F1 points on Text+Berg, MAC, and OPUS.

3. Non-parallel spans become unaligned gaps, not cascading errors, and multiple chains capture reordering that a single monotonic path cannot.

The same sparsity enables $k > 2$ language multiple alignment (e.g., Fazili and Goswami (2026)) via k-dimensional chaining. We provide a sketch of the approach, leaving full-exploration to future work.

2 Related Work

Cross-lingual sentence alignment. Sentence alignment predates neural embeddings: early aligners such as Hunalign (Varga et al., 2005) relied on sentence-length statistics and bilingual lexicons. Embedding-based aligners, which now substantially outperform them, share a LaBSE-style encoder (Feng et al., 2022) and differ mainly in the search over its similarities. Vecalign (Thompson and Koehn, 2019) uses an approximate dynamic time warping (DTW) borrowed from time series (Salvador and Chan, 2007); Bertalign (Liu and Zhu, 2023) runs a two-pass dynamic program over LaBSE similarities; and SentAlign (Steingrimsson et al., 2023) casts alignment as Dijkstra shortest-path search. Closest to our approach, Kraif (2024) independently arrives at an anchor-plus-local-DP structure, but selects its anchors by heuristic filtering with no optimality guarantee. Token-level aligners such as SimAlign (Jalili Sabet et al., 2020) and fast_align (Dyer et al., 2013) address the related but distinct problem of word rather than sentence correspondence. Vecalign, Bertalign, and SentAlign each recover a single monotonic path that covers both documents in full, whether by exact search (SentAlign) or by a windowed or downsampled approximation of it (Vecalign, Bertalign). Full coverage presupposes a near-monotonic correspondence. This holds for genres typically translated literally (e.g., news) but weakens for genres translated interpretively (e.g., novels), where sentences are more likely to be reordered, split, or merged – a regime that monotonic-leaning benchmarks (Volk et al., 2010) have masked.

Cross-domain transfer of alignment techniques. Sequence alignment is a shared primitive across biology, natural language processing (NLP), speech, and time series (Wang and Jiang, 1994; Gale and Church, 1993; Sakoe and Chiba, 1978; Cuturi and Blondel, 2017), and importing a mature technique from one domain into another has repeatedly proven productive. Vecalign's use of DTW is itself such a transfer, from time series into sentence alignment (Thompson and Koehn, 2019). In embedding space, optimal transport has crossed into NLP for bilingual lexicon induction (Zhang et al., 2017) and cross-lingual alignment via Gromov–Wasserstein distance (Alvarez-Melis and Jaakkola, 2018; Peyré and Cuturi, 2019), while profile-HMM ideas from biology have informed speech forced alignment (McAuliffe et al., 2026). These transfers import not just an algorithm but the accumulated understanding of when it works. Our work continues this tradition, importing co-linear chaining from genomics (Myers and Miller, 1995).

Co-linear chaining. Co-linear chaining originates in computational genomics (Myers and Miller, 1995). An anchor is a local match with a position in each of the two sequences. Chaining selects the maximum-weight set of anchors whose positions strictly increase in both sequences. Such a set is co-linear and non-crossing: drawn as edges between the two documents, no two cross, so the selected anchors preserve order. Non-crossing is the chaining term for this order constraint; the DTW and DP literature calls the same property monotonicity, the difference being that chaining imposes it on a selected subset of anchors rather than on a complete path. Myers and Miller (1995) solve the k-sequence problem in subquadratic $O(N \log^k N)$

time by line sweep with orthogonal range search. For two sequences under a zero gap cost (no penalty on the distance between consecutive anchors) and strict non-overlap precedence—our setting—the optimal chain is computable exactly in $O(N \log N)$ time, dominated by sorting (Abouelhoda and Ohlebusch, 2005); we implement the anchored prefix-maximum query with a Fenwick tree (Fenwick, 1994). Abouelhoda and Ohlebusch (2005) give the best known bounds for $k > 2$ and treat the zero-gap-cost case as the base of their construction, while Mäkinen and Sahlin (2020) admit overlapping anchors. Practical genome aligners often trade this exactness for speed: minimap2 (Li, 2018), for instance, chains heuristically, bounding the predecessor search per anchor. At the sentence scale, however, the anchor set is small enough that ChainAlign retains the exact $O(N \log N)$ chain. Jain et al. (2022) further show that under a specific gap-and-overlap cost the optimal chaining cost equals an anchored edit distance (their Theorem 2), which places the chaining objective on a formal footing. Their formulation minimizes gap and overlap penalties with no per-anchor reward, so it is a different objective from ChainAlign's per-anchor weight maximization; we claim no edit-distance guarantee for our chain. Chaining alone leaves inter-anchor gaps unaligned; in genomics these are closed by local base-level DP, and the natural NLP analogue is a dense routine such as multi-resolution DTW (Salvador and Chan, 2007) applied only to the short residual spans.

Filling the gap with ChainAlign No prior sentence aligner uses exact co-linear chaining: Vecalign and Bertalign approximate the alignment with windowed or downsampled search, and SentAlign optimizes a full monotonic path rather than an anchor subset. Both ingredients already exist – multilingual embeddings supply high-quality anchors, and genomics supplies an exact, theoretically grounded chaining routine – yet no prior work has combined them. ChainAlign does: a sparse-then-dense structure that, because the chain imposes no coverage requirement, admits partial and non-monotonic correspondences that a single forced path cannot represent.

3 Method

ChainAlign's seed-chain-extend alignment spans three steps: (1) seeding forms candidates between similar sentence spans; (2) co-linear chaining selects the optimal non-crossing set of anchors; and (3) multi-resolution dynamic time warping extends the gaps between those anchors. We illustrate how ChainAlign differs from three other embedding-based aligners (Vecalign, Bertalign, and SentAlign) in Figure 1.

3.1 Seed: Alignment Anchor Candidates

Let N and M denote the number of source and target sentences respectively, assuming $N \geq M$ without loss of generality. A multilingual embedding model transforms sentences to the same vector space (see Figure 2a); we use LaBSE[1] (Feng et al., 2022); in principle, any contrastively-trained multilingual encoder can be substituted, though alignment quality may vary with the encoder's cross-lingual discriminability. Following Thompson and Koehn (2019), who embed concatenated sentence blocks for scoring k:l alignments, we precompute embeddings at multiple resolutions (single, pairs, triples, and quadruples of adjacent sentences). For a span of k consecutive sentences starting at position i, the multi-resolution embedding $s_i^{(k)}$ is obtained by concatenating the text of s_i through s_{i+k-1} (space-separated) and encoding the result as a single input to the encoder; we do not average individual sentence embeddings, as concatenation preserves inter-sentence coherence and produces more discriminative representations for merged spans. We use cosine similarity to score candidate alignments between sentence spans, creating a bipartite graph between source and target sentences where candidate edges are weighted by cosine similarity; multi-resolution edges may span multiple consecutive sentences on either side. We generate edges at resolutions (1,1), (2,1), (1,2), (3,1), (1,3), (2,2), (3,2), (2,3), (4,1), and (1,4), covering all (k, l) combinations where $k + l \leq 5$.

These edges create a dense graph with up to $10NM$ edges, but we employ three techniques to reduce edge count and create a sparse search space for chaining. First, for parallel corpora where we expect near-complete correspondence, we limit candidate edges to those satisfying $|i/N - j/M| < 0.15$, i.e., source and target positions must be within 15% of each other in fractional document coordinates (Sakoe and Chiba, 1978), preventing spurious matches between similar sentences repeated throughout the text. We chose to operationalize

[1] https://huggingface.co/sentence-transformers/LaBSE

Search strategies over the alignment grid: (a)-(c) search a restricted or chunked grid and resolve m:n in stages; (d) selects a maximal-weight chain over sparse multi-resolution candidates in one step

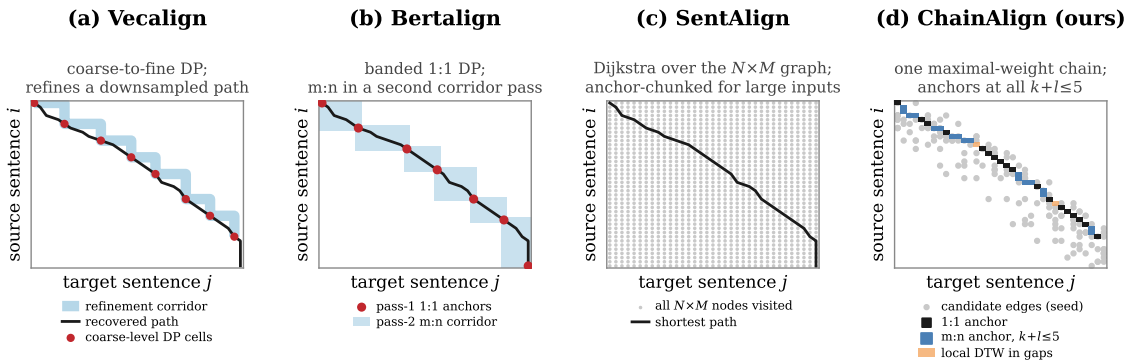

Figure 1: Search strategies over the alignment grid. All four recover a monotonic alignment but differ in what restricts the search and when m:n resolution is decided. Vecalign (a) approximates the path on a downsampled grid, then refines it in a corridor; Bertalign (b) runs a banded 1:1 DP and defers m:n to a second corridor pass; SentAlign (c) finds the optimal path over the $N \times M$ graph, chunked by 1:1 hard-delimiter anchors for large inputs. Each stages resolution and searches a dense, banded, or chunked grid. ChainAlign (d) instead seeds sparse candidates at all resolutions $k{+}l \leq 5$ at once and selects a single maximum-weight non-crossing chain, running dense DTW only in the residual gaps.

thresholds at 15% through trial and error, and implementations may elect to treat the threshold as an additional alignment hyperparameter.

Second, we filter all edges below a similarity threshold of $\tau = 0.3$. Lastly, we apply top-K pruning ($K = 10$) per source row for each multi-resolution edge set. For symmetry, one may additionally retain only edges that are top-K from both source and target perspectives (mutual top-K); in our experiments this produced identical results to unidirectional pruning. This leaves us with a sparse graph with at most $10KN = 100N$ edges, i.e., candidate anchors. The resulting sparse graph is illustrated in Figure 2b.

3.2 Chain: Select Alignment Anchors

The seed phase produces far more candidate edges than are mutually compatible, with many overlapping or crossing each other. The chain phase selects the maximum-weight non-crossing subset: no two selected edges may overlap in source or target position, and their relative order must be preserved in both sequences, visualized in Figure 3. This problem is known in bioinformatics as co-linear chaining (Myers and Miller, 1995), and reduces to a weighted variant of Longest Increasing Subsequence on the target dimension after sorting edges by source position.

The naive solution to weighted LIS is an $O(E^2)$ dynamic programming solution:

```
for each edge e = (s_i^(k), t_j^(l), w)
  in source order:
for each preceding edge e' = (s_i'^(k'),
      t_j'^(l'), w'):
  if i'+k' <= i and j'+l' <= j:
    dp[e] = max(dp[e], dp[e'] + w)
```

Sorting edges by source position ensures each edge is processed after all edges beginning earlier. The inner loop then reduces to a prefix maximum query on target position: among all valid predecessors, find the one with highest chain score whose target span ends before the current edge's target span begins. A Fenwick tree (Fenwick, 1994) answers prefix maximum queries in $O(\log M)$, replacing the $O(E)$ scan.

Sorting alone does not fully enforce the source non-overlap constraint: an edge is a valid predecessor only if it *ends* before the current edge begins, not merely starts before. We enforce this via deferred commitment: each edge is inserted into the Fenwick tree only after the sweep passes its source endpoint, implemented as a priority queue keyed by source end position (see Algorithm 1).

Edge weight function. Each candidate edge $e = (s_i^{(k)}, t_j^{(l)})$ receives a weight combining semantic similarity with a length-ratio prior:

$$w(e) = \cos^2(e) \cdot \left[\log_2\!\left(1 + \frac{\min(c_s,\, c_t \cdot r)}{\max(c_s,\, c_t \cdot r)}\right)\right]^{\frac{1}{2}} \quad (1)$$

where $\cos(e) = \cos(s_i^{(k)}, t_j^{(l)})$ denotes the multi-resolution embedding pair's cosine similarity, $c_s =$

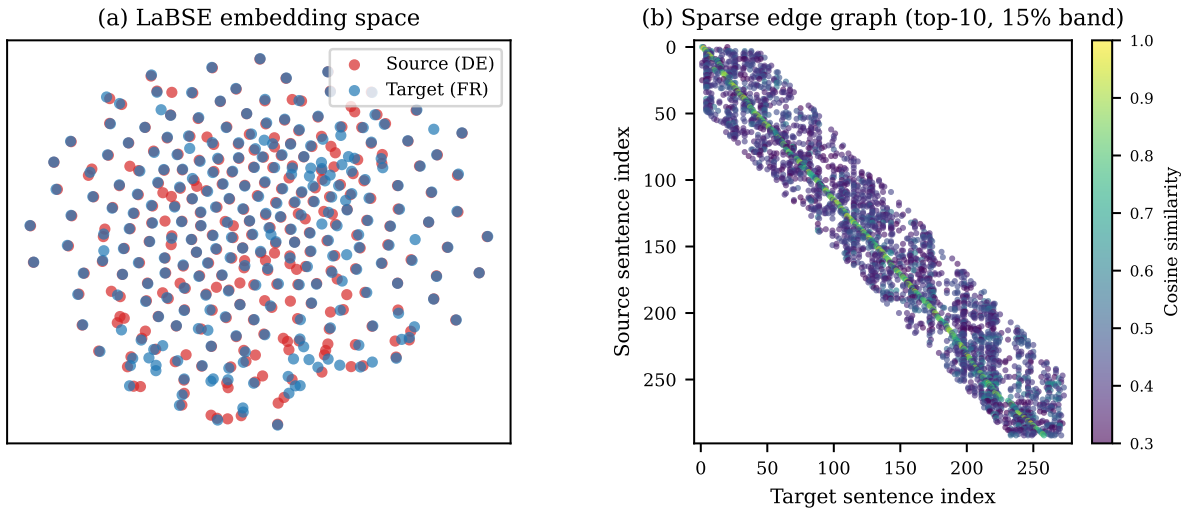

Figure 2: (a) LaBSE embeds source (DE) and target (FR) sentences into a shared space where translations cluster (shown: t-SNE). (b) The sparse 1:1 edge graph with threshold filters and top-K pruning. Edges cluster on the diagonal with $\sim$5% density. Multi-res. edges (not shown) augment the graph with merged-sentence candidates.

$\sum_{m=i}^{i+k-1} |s_m|$ and $c_t = \sum_{n=j}^{j+l-1} |t_n|$ are the character lengths of source and target spans, and $r = \sum |s| / \sum |t|$ is the corpus-level character ratio.

Squaring the cosine prevents chains of many weak edges from outscoring fewer strong ones: two edges at $\cos = 0.5$ outscore one at $\cos = 0.8$ under linear weighting ($0.5 + 0.5 > 0.8$), but squaring reverses this ($0.25 + 0.25 < 0.64$). The length-ratio term follows the principle that parallel segments have proportional character lengths (Gale and Church, 1993), adapted from Bertalign (Liu and Zhu, 2023). Without this term, the chain preferentially selects narrow spans on split translations. As additional target sentences are concatenated, embedding similarity decreases, biasing the chain toward 1:2 alignments where the gold is 1:3. The length term peaks when combined target character length matches the source, counteracting this dilution (see ablation in Section 4.4).

Because the chain maximizes total edge weight rather than enforcing a path from position 0 to N, the backtracked chain spans only positions where high-confidence edges exist; all other positions become gaps for the extend phase. On mostly-aligned parallel corpora, this achieves high coverage, leaving small local gaps for DTW so that errors in DTW cannot propagate beyond a single gap. Nonetheless, co-linear chaining naturally handles noisy input. Unlike DTW-based aligners that force a path from $(0, 0)$ to (N, M), the chain imposes no coverage requirement. Regions without high-similarity anchors are delegated to the extend phase, which aligns them if local evidence supports it or leaves them unpaired via skip transitions. This naturally handles one-sided insertions and deletions: inserted content creates gaps with no viable anchors, and the extend phase's skip cost allows those positions to go unmatched without cascading errors. Non-monotonic input can also be aligned by extracting multiple chains from the same edge set (Section 4.6).

Algorithm 1 Co-linear Chaining

Require: Edges $\mathcal{E} = \{e = (s_i^{(k)}, t_j^{(l)}, w_e)\}$
Ensure: Maximum-weight non-crossing chain $\mathcal{C}$

```
Sort E by i
Init Fenwick tree T over [1, M]; heap P ← ∅
for each edge e = (s_i^(k), t_j^(l), w_e) in sorted order do
    while P has entries e' = (s_i'^(k'), t_j'^(l'), w_e') with i' + k' ≤ i do
        T.UPDATE(j' + l', dp[e'], e')        ▷ Commit
    end while
    (ν, p) ← T.PREFIXMAX(j)
    dp[e] ← w_e + ν
    parent[e] ← p
    Push e onto P keyed by i + k
end for
Backtrack from arg max_e dp[e] via parent
```

3.3 Extend: Fill Gaps Between Anchors

Between consecutive anchors, a gap of g source and h target sentences remains unaligned. We resolve

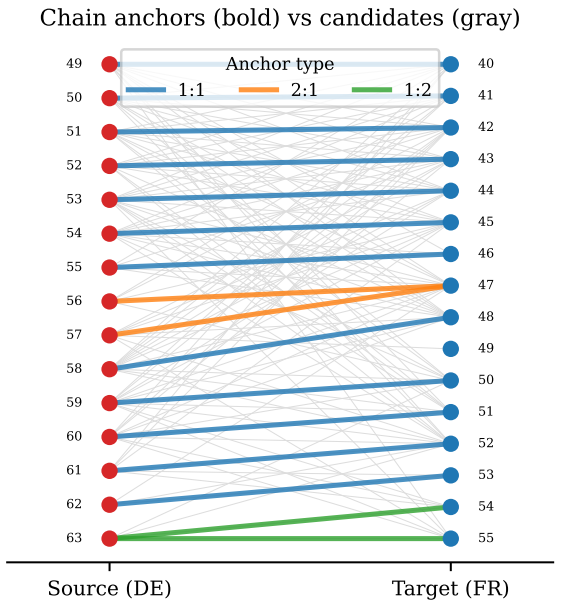

Figure 3: Chain anchors (bold, colored by resolution type) selected from candidate edges (gray). The chain selects the maximum-weight non-crossing subset, including multi-resolution anchors (2:1, 1:2) where merging is supported by embedding similarity.

these locally using multi-resolution DTW.

For each resolution (k, l) with $k, l \geq 1$ and $k + l \leq 5$ – the same set used in the seed phase – we define the match score:

$$d^{(k,l)}(i,j) = \cos(s_i^{(k)}, t_j^{(l)}) - \frac{1}{2}\left(\bar{C}_{i\cdot}^{(k,l)} + \bar{C}_{\cdot j}^{(k,l)}\right) \quad (2)$$

where $\bar{C}_{i\cdot}^{(k,l)}$ and $\bar{C}_{\cdot j}^{(k,l)}$ are the row and column means of the cosine similarity matrix $C^{(k,l)}$ for resolution (k, l). As opposed to using squared similarity, subtracting these means controls for sentences that are generically similar to many candidates, isolating pair-specific signal and mitigating the hubness effect, in which certain points in high-dimensional spaces appear disproportionately often as nearest neighbors (Radovanović et al., 2010; Artetxe and Schwenk, 2019).

The optimal alignment path through the gap is found via the recurrence:

$$F(i,j) = \max \begin{cases} \max\limits_{\substack{k,l\geq 1 \\ k+l\leq 5}} \left\{ F(i-k, j-l) + d^{(k,l)}(i-k, j-l) \right\} \\ F(i-1, j) + \delta \\ F(i, j-1) + \delta \end{cases} \quad (3)$$

with base case $F(0, 0) = 0$, and $\delta < 0$ a skip penalty allowing sentences to remain unaligned. We use $\delta = -0.5$ throughout; this value assumes the input is predominantly parallel, penalizing skips heavily enough that the aligner prefers low-confidence matches over gaps. When the input is known to contain substantial non-parallel content, $\delta = -0.1$ permits more gaps, which is used only in the insertion robustness experiment (Section 4.5), where non-parallel material is injected by design.

The alignment is recovered by backtracking from $F(g, h)$. Gaps between anchors are small; at this scale, unbanded DTW is both exact and negligible in runtime. Together with chain anchors, filled gaps complete the alignment.

Full pseudocode for all three phases and a complete hyperparameter listing appear in Appendix A.

3.4 Algorithmic Complexity

Recall that N is the source sentence count, M is the target sentence count, K is the top-K pruning constant, and E is the candidate edges post-pruning. We now introduce a new variable L to refer to $\max(N, M)$ to aid complexity definition.

With top-K pruning across 10 resolutions, $E \leq 10KL = O(L)$. The seed phase computes 10 cosine similarity matrices via BLAS-optimized matrix multiplication in $O(NM)$. The chain phase is $O(L \log L)$ total. It: 1) sorts E edges in $O(E \log E)$; 2) performs E Fenwick prefix-max queries in $O(\log M)$ each; and 3) executes E heap operations in $O(\log E)$. The extend phase runs a DTW within each inter-anchor gap, at a cost equal to the gap's area – the product of its source and target spans. Because residual gaps are short each DTW is negligible[2], and the $O(L)$ gaps contribute $O(L)$ in total. Overall runtime is $O(NM + L \log L)$, dominated by the matrix multiplications in the seed phase. In practice, embedding inference is the bottleneck, and is shared across all embedding-based aligners.

Top-K pruning is what bounds the candidate set at $E \leq 10KL$; the threshold τ and the diagonal band prune it further without changing this asymptotic size. The pruning is lossless: disabling all three leaves strict F1 within 0.004 on our benchmarks, so it removes redundant candidates rather than correct anchors. Sparsity is thus a tractability device, not an accuracy trade-off. Without it the seed phase retains the full $O(NM)$ graph, which does not fit in memory at book length.

[2]Our evaluations (Sec. 4) typically produced 95% anchor coverage and 1-2 sentence gaps.

4 Evaluations

We evaluate ChainAlign against three embedding-based sentence aligners: Bertalign, SentAlign, and Vecalign, using three benchmark corpora. Bertalign and SentAlign represent the current state of the art for embedding-based alignment. We include Vecalign under both LaBSE and the LASER embeddings it was published with, as it is the only baseline whose published configuration uses a different encoder. We do not re-evaluate earlier length- and lexicon-based aligners such as Hunalign (Varga et al., 2005), which prior work reports as substantially weaker on these benchmarks.

4.1 Datasets

We compare ChainAlign against Bertalign, SentAlign, and Vecalign on the following datasets:

1. Text+Berg[3] (Volk et al., 2010): Seven articles from the Swiss Alpine Club, manually aligned between French and German.

2. MAC[4]: 24 chapters sampled from six Chinese literary novels, manually aligned to their English translations.

3. OPUS Books[5]: Copyright-free books aligned at paragraph-level by András Farkas, then automatically aligned at sentence level. Sentence-level mappings are manually reviewed for a portion of books; we evaluate on seven of these (EN-DE and EN-ES). As these gold alignments originate from an automatic aligner, they exhibit a bias toward 1:1 mappings and occasional boundary errors; we use them for system comparison (where all systems face the same gold) but exclude them from our ablation study.

Sentence and token counts, gold pair counts, and the distribution of alignment types for each corpus appear in Appendix B.

4.2 Evaluation Scoring

We report precision, recall, and F1 under two matching criteria. Under **strict** matching a predicted pair is correct only if it exactly matches a gold pair; under **lax** matching it is correct if it overlaps a gold pair on both sides. For example, two 1:1 pairs where the gold has one 2:2 count as two false positives under strict F1 but two true positives under lax F1.

We run Bertalign[6], SentAlign[7], Vecalign[8], and ChainAlign with the same scoring script to ensure fair comparison. We exclude null alignments (1:0, 0:1) from the recall computation, consistent with Bertalign's evaluation.

We set $\tau = 0.3$ (seed similarity threshold) to maximize aggregate strict F1 across Text+Berg and MAC. Re-selecting it under two protocols that do not see the evaluation data – on a disjoint half of each corpus's documents, and on OPUS alone – recovers the reported figures to within 0.002 strict F1, and performance is flat across $\tau \in [0.20, 0.30]$. Pruning parameters ($K = 10$, band $= 15\%$) were selected for computational efficiency. No per-dataset tuning is applied.

4.3 Main Results

Table 1 shows the strict and lax F1 scores of ChainAlign, Bertalign, SentAlign, and Vecalign (under both LASER and LaBSE) on all three datasets, averaging the seven OPUS books (full OPUS results in Table 7).

ChainAlign achieves the highest strict and lax F1 on all three benchmarks. We outperform Bertalign by 0.5 points on Text+Berg, 0.4 on MAC, and 1.0 point on OPUS (significance in Table 1). The gap to SentAlign is larger: 0.9 points on Text+Berg, 5.7 on MAC, and 1.4 on OPUS, and larger still to Vecalign under either encoder. Lax F1 is near-perfect across all systems ($\geq$0.96), indicating that errors are predominantly boundary disagreements rather than wrong-content matches.

4.4 Ablation Study

Each component of our algorithm contributes positively to both Text+Berg and MAC (Table 2). The base configuration uses $(1,1)$, $(2,1)$, and $(1,2)$ edges in the seed phase with raw cosine similarity as the weight; DTW in gaps uses the full $k + l \leq 5$ set throughout. Extending the seed to all $k + l \leq 5$ resolutions provides the largest single gain on both corpora: since the chain achieves high anchor coverage and the extend phase cannot modify anchor boundaries, alignment quality depends on the chain having access to correct-width edges. The final two rows show how square cosine weighting and

[3] https://github.com/bfsujason/bertalign/tree/main/text+berg

[4] https://github.com/bfsujason/mac

[5] https://opus.nlpl.eu/legacy/Books.php

[6] https://github.com/bfsujason/bertalign

[7] https://github.com/steinst/SentAlign

[8] https://github.com/thompsonb/vecalign

System	Strict F1			Lax F1		
	text+berg	MAC	OPUS	text+berg	MAC	OPUS
Vecalign (LASER)	.904	.853	.851	.990	.984	.976
Vecalign (LaBSE)	.863	.879	.863	.989	.996	.979
SentAlign	.932	.846	.867	.989	.973	.962
Bertalign	.936†	.899†	.871	.989	.993	.975
ChainAlign (ours)	**.941**	**.903**	**.881**	**.992**	**.998**	.979

Table 1: Strict and lax F1 on human-reviewed benchmarks. All systems use LaBSE except Vecalign (LASER). OPUS is the mean over 7 books (Table 7). Strict and lax F1 are defined in Section 4.2. ChainAlign is highest in every column, though on OPUS lax its margin falls below the reported precision. † marks the two comparisons that are not significant under a document-level paired bootstrap (Koehn, 2004), 10,000 resamples (p=0.15, p=0.12); the other ten reach $p < 0.01$.

	text+berg	MAC
Base (k+$l \leq 3$, raw sim)	.915	.807
+ extended resolutions (k+$l \leq 5$)	.930	.874
+ cos^2	.940	.888
+ length prior (full model)	**.941**	**.903**
DTW, same edge weight function	.922	.897

Table 2: Cumulative ablation (strict F1) on human-annotated benchmarks. Each row adds one component to the previous configuration. The final row replaces co-linear chaining with exhaustive DTW over the full grid while holding the edge weight function of Eq. 1 fixed; on OPUS the corresponding figures are .881 and .876.

% ins.	Vec	Sent	Bert	Chain	−LP
0%	.874	.941	.949	.950	.948
10%	.857	.937	.898	.948	.944
20%	.854	.937	.856	.942	.943
30%	.854	.937	.818	.932	.941
40%	.812	.934	.782	.933	.941
50%	.812	.934	.742	.930	.941
60%	.812	.934	.709	.920	.941
Degr.	−6.2	−0.7	−24.0	−3.0	−0.8

Table 3: Strict F1 on Text+Berg with unrelated source content inserted at the midpoint. Vec is Vecalign (LaBSE); −LP disables the length-ratio prior. Degradation is the change from 0% to 60%. First-quartile results: Table 8.

the length-ratio prior further improve boundary selection by rewarding high-confidence, length-consistent matches.

4.5 Insertion Robustness

To evaluate robustness to non-parallel content, we inserted sentences from the seventh Text+Berg document (excluded from evaluation) into each remaining source document, at rates up to 60% and at two positions: the midpoint and the first quartile. Throughout we set $\delta = -0.1$ and disable the diagonal band, since one-sided insertion breaks the near-diagonal correspondence the band assumes; in our implementation the two are controlled together. The baselines are run at their default skip and null-alignment settings, which already permit unmatched content.

ChainAlign and SentAlign lose under 3 points at 60% insertion while Bertalign loses 24, roughly 4 points per 10% inserted (Table 3). Forced coverage alone does not explain this: Vecalign also aligns the full grid, yet degrades 6.2 points. The difference is plausibly how the skip cost is set, since Vecalign calibrates it from the observed cost distribution at alignment time whereas Bertalign's is fixed. This yields three tiers rather than a binary: full coverage with a fixed skip cost collapses, full coverage with a calibrated one degrades moderately, and no coverage requirement degrades least. Disabling the length-ratio prior reduces ChainAlign's degradation to 0.8 points, comparable to SentAlign's 0.7, because the prior assumes a corpus-level character ratio that one-sided insertion shifts, penalizing otherwise-valid alignments in the uncontaminated regions.

4.6 Non-Monotonic Alignment

While our benchmark datasets are monotonic (except for a handful of mappings in Text+Berg), real-world, noisier data has cases where mappings can cross: source sentence 1 maps to target sentence 2, while source sentence 2 maps to target sentence 1. ChainAlign can be extended to extract multiple non-overlapping chains during the chain phase.

We replace the Fenwick LIS with a gap-bounded dynamic programming algorithm following minimap2 (Li, 2018): for each edge processed in source order, we search backwards through at most h predecessors, linking it to the highest-scoring predecessor whose source and target gaps are both within a threshold G. Edges separated by gaps exceeding G cannot belong to the same chain, causing the DP to naturally segment the input into monotonic blocks. After the forward pass, we extract chains via multi-backtrack: starting from the highest-scoring unused edge, we trace parent pointers to recover a complete chain, mark its positions as used, and repeat until no edges remain. Each chain is internally monotonic; chains may cross each other globally. Extend runs independently within each chain, filling gaps between consecutive anchors with local DTW. Pairs produced by extend that conflict with positions claimed by other chains are discarded.

On the Text+Berg documents concatenated with shuffled target ordering[9] (991×1011 sentences), single-chain alignment achieves 0.721 strict F1 while multi-chain extraction (G=3, h=200, minimum chain length 1, no banding) recovers 0.927, within 1.4 points of aligning each document independently (0.941).

4.7 Runtime

Figure 4 shows algorithm-only wall-clock time (excluding embedding inference, which is shared across all systems) on an AMD EPYC 7R13 (8 vCPU, 16 GB RAM). All timings are the median of five runs after four warmup iterations. SentAlign uses its default subdivision threshold; it is infeasible beyond 5K sentences due to memory requirements.

ChainAlign and Bertalign exhibit similar scaling, both dominated by embedding similarity computation. ChainAlign is approximately 1.4× slower due to pre-computing 10 full similarity matrices (one per (k, l) combination) for sparse edge extraction, whereas Bertalign computes multi-resolution similarities lazily within a narrow corridor defined by its first-pass anchors. Vecalign scales more gently than either, since its coarse-to-fine DP never forms the full $O(NM)$ similarity matrix, but its larger constant factors leave it slower than both at every size we test: 0.23 s against 0.04 s at 200 sentences, narrowing to 14.7 s against 14.1 s at 14K.

[9]Source: documents 001–007 in their natural order; target: the same documents concatenated in the order 003, 001, 005, 002, 007, 004, 006.

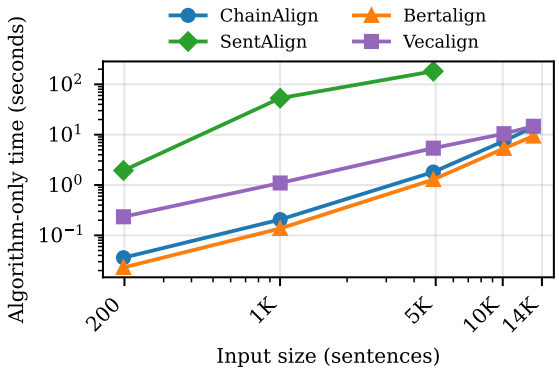

Figure 4: Algorithm-only runtime (log scale) versus input size, the geometric mean of source and target sentence counts. ChainAlign and Bertalign scale comparably, both dominated by $O(NM)$ similarity computation. Vecalign scales near-linearly but with higher constant factors, remaining slower than both across the tested range. SentAlign is 100–250× slower and infeasible beyond 5K sentences.

5 Discussion

ChainAlign achieves the highest strict and lax F1 on all three benchmarks; the improvement over Bertalign is significant on OPUS Books and within sampling noise on Text+Berg and MAC, while the improvements over SentAlign and Vecalign are significant on all three. Our central contribution is the application of co-linear chaining to sentence alignment: both problems seek a maximum-weight non-crossing subset of candidate matches between two sequences, and the genomics literature offers efficient and exact algorithms that cross-lingual sentence aligners have not previously exploited.

In computational biology, seed-chain-extend typically serves as a heuristic approximation to exhaustive dynamic programming over the full alignment space. Our results show that for sentence alignment, the sparse formulation outperforms exhaustive DP. To control for the scoring function, we evaluated a DTW baseline using the same $\cos^2$ weighting and length-ratio penalty as our edge weight function; it remains 1.9, 0.6, and 0.5 points behind on Text+Berg, MAC, and OPUS, respectively (Table 2). We hypothesize that the advantage lies in DTW's forced-path property: to produce a complete alignment from $(0, 0)$ to (N, M), DTW must commit to transitions in regions of ambiguous similarity, and poor local choices propagate forward, displacing subsequent high-confidence matches.

The chain phase selects the globally optimal anchor set; DTW then operates in small, isolated gaps where errors cannot propagate.

Beyond aggregate F1, error profiling on Text+Berg and MAC reveals that ChainAlign predominantly over-splits, whereas Bertalign over-merges. The over-splitting reflects a structural property of the chain phase: we maximize the sum of edge weights, so two moderate-confidence 1:1 edges can collectively outscore a single higher-confidence 2:1 edge. The $\cos^2$ weighting and length-ratio prior both reduce this bias but do not eliminate it entirely. Bertalign's forced-path DP exhibits the opposite tendency: when boundary evidence between adjacent sentence pairs is weak, the merged $n{:}m$ embedding scores higher than the individual 1:1 embeddings, and the DP selects the merge. These complementary error profiles suggest that a hybrid approach could reduce both failure modes simultaneously.

A natural remedy for the over-splitting is a gap cost on the chain, penalizing uneven advancement between successive anchors as in genomic chaining (Li, 2018). We tested the separable cost $\lambda\,(g_s + g_t)$, where g_s and g_t are the source and target sentences left unaligned between an anchor and its predecessor; it folds into the Fenwick prefix-max and reduces to our objective at $\lambda = 0$. Split false positives fall monotonically as λ increases, from 36 to 18 (4.19% to 2.10% of predicted pairs), but strict F1 moves by at most 0.004, and the best λ is inconsistent across benchmarks: Text+Berg peaks near 0.005 and MAC near 0.05, and each corpus's optimum degrades the other. The chain trades split errors for merge errors rather than eliminating them. We suspect the limited benefit is intrinsic: minimap2's anchors are near-uniform-weight k-mer matches, so its gap cost supplies the signal that separates true chains from spurious ones, whereas here the squared cosine already carries it; and its penalty approximates a stationary indel prior in which uneven advancement is error, whereas between sentences it reflects translation style. The coupling that expresses that penalty, $|g_s - \rho\, g_t|$, also cannot be answered by a prefix-max query, and would forfeit the $O(E \log E)$ chain and its generalization to $k > 2$ languages.

The insertion robustness and non-monotonic results both reflect the same underlying property: sparse anchor-based alignment makes no coverage guarantee, so non-parallel content creates gaps rather than cascading errors. SentAlign shares this property and similarly handles insertion well, but cannot recover non-monotonic structure. Bertalign's forced path degrades under both conditions. Multi-chain extraction extends this robustness to reordered documents, recovering 0.927 strict F1 on shuffled input versus 0.721 for single-chain alignment, within 1.4 points of aligning each document independently.

Runtime scaling confirms that the sparse formulation incurs no practical overhead: ChainAlign and Bertalign are both dominated by embedding similarity computation and scale comparably, and Vecalign, despite its near-linear coarse-to-fine search, is slower than both across the tested range; SentAlign's Dijkstra-based search is two orders of magnitude slower still.

6 Open Research Areas

Co-linear chaining's sparsity also makes simultaneous alignment of $k > 2$ languages tractable: intersection pruning across all pairwise language combinations keeps the edge count $O(L)$, and co-linear chaining generalizes to k dimensions via nested Fenwick trees with chain step $O(L \log^{k-1} L)$. In preliminary experiments on 3- and 4-way alignment, this single-pass, pivot-free approach matches or outperforms pivot-based methods, with no pivot-induced inconsistencies. Given that multiway parallel corpora yield substantially stronger cross-lingual representations than bilingual data (Fazili and Goswami, 2026), high-quality multiway alignment is increasingly important. We leave a full investigation to future work.

7 Conclusion

We presented ChainAlign, which applies co-linear chaining from computational genomics to cross-lingual sentence alignment. The chain selects a maximum-weight non-crossing anchor set, and multi-resolution DTW fills the residual gaps. On three benchmarks across four language pairs, ChainAlign outperforms Bertalign, SentAlign, and Vecalign on both strict and lax F1. Because the chain imposes no coverage requirement, non-parallel content creates gaps rather than cascading errors, and multi-chain extraction recovers reordering that a single monotonic path cannot represent.

8 Limitations

We evaluate with a single embedding model (LaBSE). While the algorithm is model-agnostic,

alignment quality depends on the encoder's cross-lingual discriminability, and we do not empirically verify performance with alternative encoders. Our threshold $\tau = 0.3$ is robust across all benchmarks tested but may require adjustment for encoders with different similarity distributions. Finally, our evaluation focuses on European and Chinese language pairs; performance on low-resource or typologically distant pairs (e.g., agglutinative languages) remains untested.

References

Mohamed Ibrahim Abouelhoda and Enno Ohlebusch. 2005. Chaining algorithms for multiple genome comparison. *Journal of Discrete Algorithms*, 3(2–4):321–341.

David Alvarez-Melis and Tommi Jaakkola. 2018. Gromov-Wasserstein alignment of word embedding spaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1881–1890, Brussels, Belgium. Association for Computational Linguistics.

Mikel Artetxe and Holger Schwenk. 2019. Margin-based parallel corpus mining with multilingual sentence embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3197–3203, Florence, Italy. Association for Computational Linguistics.

Marco Cuturi and Mathieu Blondel. 2017. Soft-DTW: A differentiable loss function for time-series. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *PMLR*, pages 894–903.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Barah Fazili and Koustava Goswami. 2026. Enhancing multilingual embeddings via multi-way parallel text alignment. *Preprint*, arXiv:2602.21543.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. Language-agnostic BERT sentence embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891, Dublin, Ireland. Association for Computational Linguistics.

Peter M. Fenwick. 1994. A new data structure for cumulative frequency tables. *Software: Practice and Experience*, 24(3):327–336.

William A. Gale and Kenneth W. Church. 1993. A program for aligning sentences in bilingual corpora. *Computational Linguistics*, 19(1):75–102.

Chirag Jain, Daniel Gibney, and Sharma V. Thankachan. 2022. Algorithms for colinear chaining with overlaps and gap costs. *Journal of Computational Biology*, 29(11):1237–1251.

Masoud Jalili Sabet, Philipp Dufter, François Yvon, and Hinrich Schütze. 2020. SimAlign: High quality word alignments without parallel training data using static and contextualized embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1627–1643, Online. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Olivier Kraif. 2024. Adaptative bilingual aligning using multilingual sentence embedding. *Preprint*, arXiv:2403.11921.

Heng Li. 2018. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18):3094–3100.

Lei Liu and Min Zhu. 2023. Bertalign: Improved word embedding-based sentence alignment for chinese–english parallel corpora of literary texts. *Digital Scholarship in the Humanities*, 38(2):621–634.

Veli Mäkinen and Kristoffer Sahlin. 2020. Chaining with overlaps revisited. In *31st Annual Symposium on Combinatorial Pattern Matching (CPM)*, pages 25:1–25:12, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Michael McAuliffe, Kaylynn Gunter, Michael Wagner, and Morgan Sonderegger. 2026. Montreal forced aligner and the state of speech-to-text alignment in 2026. *Preprint*, arXiv:2606.18466.

Eugene W. Myers and Webb Miller. 1995. Chaining multiple-alignment fragments in sub-quadratic time. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 38–47. ACM.

Gabriel Peyré and Marco Cuturi. 2019. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5–6):355–607.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. 2010. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11:2487–2531.

Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49.

Stan Salvador and Philip Chan. 2007. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580.

Steinthor Steingrimsson, Hrafn Loftsson, and Andy Way. 2023. SentAlign: Accurate and scalable sentence alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 256–263, Singapore. Association for Computational Linguistics.

Brian Thompson and Philipp Koehn. 2019. Vecalign: Improved sentence alignment in linear time and space. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1342–1348, Hong Kong, China. Association for Computational Linguistics.

Dániel Varga, László Németh, Péter Halácsy, András Kornai, Viktor Trón, and Viktor Nagy. 2005. Parallel corpora for medium density languages. In *Proceedings of RANLP 2005*, pages 590–596, Borovets, Bulgaria.

Martin Volk, Noah Bubenhofer, Adrian Althaus, Maya Bangerter, Lenz Furrer, and Beni Ruef. 2010. Challenges in building a multilingual alpine heritage corpus. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Lusheng Wang and Tao Jiang. 1994. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1(4):337–348.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Earth mover's distance minimization for unsupervised bilingual lexicon induction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1934–1945, Copenhagen, Denmark. Association for Computational Linguistics.

A Full Algorithm and Hyperparameters

Algorithms 2–4 provide complete pseudocode for all three phases of ChainAlign. Table 4 lists all hyperparameters, which are fixed across all experiments with no per-dataset tuning. Given precomputed multi-resolution embeddings, the algorithms fully specify the alignment procedure.

Parameter	Value
Similarity threshold τ	0.3
Band fraction b (parallel text only)	0.15
Top-K per row per resolution	10
Cosine exponent	2
Length-ratio prior exponent	0.5
Skip cost δ (parallel text)	-0.5
Skip cost δ (non-parallel)	-0.1
Max resolution $k + l$	5
Embedding model	LaBSE

Table 4: Hyperparameters (fixed across all experiments). The band and the skip cost are controlled together: the $\delta = -0.1$ configuration, used where input is not assumed parallel, also disables banding, since displaced correspondences would otherwise fall outside the window.

B Corpus Statistics

Table 5 reports the size and alignment-type composition of each benchmark.

Gold alignment types are distributed as shown in Table 6. This is consistent with the extended resolutions and the length-ratio prior producing their largest gains on MAC (Table 2), since both target precisely the split-boundary decision. Per-document sentence counts range from 36 to 293 for Text+Berg (mean 142), 100 to 335 for MAC (mean 200), and 686 to 4,668 for OPUS (mean 2,067).

C OPUS Per-Book Results

Table 7 shows the strict and lax F1 scores for all five systems on each OPUS book individually.

D Insertion Robustness at Both Positions

Table 8 reports the insertion experiment of Section 4.5 at both insertion positions.

Algorithm 2 ChainAlign: Seed Phase

Require: Source embeddings $\mathcal{S}^{(k)}$, target embeddings $\mathcal{T}^{(l)}$, source segments $s_{1..N}$, target segments $t_{1..M}$, threshold τ, band fraction b, top-K
Ensure: Candidate edge set $\mathcal{E}$

Compute corpus-level character ratio $r \leftarrow \sum |s_i| / \sum |t_j|$
$\mathcal{E} \leftarrow \emptyset$
for each resolution (k, l) with $k \geq 1,\ l \geq 1,\ k + l \leq 5$ **do**
  $N_k \leftarrow N - k + 1$; $M_l \leftarrow M - l + 1$
  $C \leftarrow \cos(\mathcal{S}^{(k)},\ \mathcal{T}^{(l)})$ ▷ $N_k \times M_l$ cosine similarity matrix
  for each source row $i \in [0, N_k)$ **do**
    Restrict to diagonal band: $\{j : |i/N_k - j/M_l| < b\}$
    Retain top-K cells in row i with $C_{ij} \geq \tau$
  end for
  for each retained cell (i, j) **do**
    $c_s \leftarrow \sum_{m=i}^{i+k-1} |s_m|$; $c_t \leftarrow \sum_{n=j}^{j+l-1} |t_n|$
    $\ell(i, j) \leftarrow \left[\log_2\!\left(1 + \frac{\min(c_s,\, c_t \cdot r)}{\max(c_s,\, c_t \cdot r)}\right)\right]^{1/2}$
    $w \leftarrow C_{ij}^2 \cdot \ell(i, j)$
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(i,\ i{+}k,\ j,\ j{+}l,\ w)\}$
  end for
end for
return $\mathcal{E}$

Corpus	Pair	Docs	Sentences		Tokens		Gold pairs		Non-1:1
			src	tgt	src	tgt	scored	null	
Text+Berg	DE–FR	7	991	1,011	19,151	21,316	916	58	21.0%
MAC	ZH–EN	24	4,799	6,573	138,465*	103,690	4,345	0	39.5%
OPUS	DE–EN, EN–ES	7	14,467	17,242	478,882	469,926	13,765	0	23.9%

Table 5: Corpus statistics. Tokens are whitespace-delimited except *, which counts Chinese characters. "Scored" gold pairs are those entering precision and recall; "null" pairs (1:0 and 0:1) are excluded from recall following Bertalign's evaluation, and occur only in Text+Berg. "Non-1:1" is the share of scored gold pairs whose source or target side spans more than one sentence.

Corpus	1:1	2:1	1:2	3:1	1:3	1:4	2:2	other
Text+Berg	79.0	9.6	7.3	1.2	0.9	0.2	1.4	0.3
MAC	60.5	4.3	21.4	0.3	6.7	2.0	1.9	2.9
OPUS	76.1	3.4	19.4	0.2	0.0	0.0	0.9	0.0

Table 6: Distribution of gold alignment types (% of scored pairs). "other" collects all remaining types, each below 1%. MAC is the most merge-heavy corpus and is dominated by source-side splits (1:2, 1:3, 1:4 total 30.1%).

Algorithm 3 ChainAlign: Chain Phase

Require: Edges $\mathcal{E} = \{e = (i, i{+}k, j, j{+}l, w_e)\}$
Ensure: Maximum-weight non-crossing chain $\mathcal{C}$

Sort $\mathcal{E}$ by (i, j) ▷ Source start, then target start
Init Fenwick tree T over $[1, |\text{unique tgt\_ends}|]$; heap $P \leftarrow \emptyset$
for each edge $e = (i, i{+}k, j, j{+}l, w_e)$ in sorted order **do**
  while P has entries e' with $i' + k' \leq i$ **do**
    T.UPDATE(rank($j'{+}l'$), dp[e'], e') ▷ Commit e'
  end while
  $(\nu, p) \leftarrow T$.PREFIXMAX(rank_le(j)) ▷ Best predecessor with tgt_end $\leq j$
  dp[e] $\leftarrow w_e + \nu$; parent[e] $\leftarrow p$
  Push e onto P keyed by $i + k$
end for
Backtrack from $\arg\max_e$ dp[e] via parent pointers
return chain $\mathcal{C}$ in source order

	Strict F1					Lax F1				
Book	Vec-LS	Vec-LB	Sent	Bert	Chain	Vec-LS	Vec-LB	Sent	Bert	Chain
DE–EN Alice	.850	.860	.851	.860	**.869**	.969	.968	.956	.969	.968
DE–EN Verwandlung	.937	.935	.940	.956	**.962**	.998	.998	.989	.998	**.999**
DE–EN Prozess	.947	.954	.953	.957	**.959**	**.991**	.990	.989	.990	.990
EN–ES Alice	.820	.832	.837	.830	**.844**	.969	.974	.965	.964	**.975**
EN–ES Sense & Sensibility	.900	.914	.927	.930	**.937**	.984	.985	.976	.983	**.986**
EN–ES Robinson Crusoe	.562	.608	.618	.616	**.633**	.927	**.942**	.871	.930	.941
EN–ES Verwandlung	.940	.939	.946	.948	**.962**	.997	.997	.985	.994	**.998**
Mean	.851	.863	.867	.871	**.881**	.976	.979	.962	.975	.979

Table 7: Per-book results on OPUS human-reviewed alignments. Vec-LS and Vec-LB are Vecalign with LASER and LaBSE embeddings respectively; all other systems use LaBSE. Bold marks the best value in a column group where it is unique at the reported precision.

	Midpoint (q=0.5)					**First quartile** (q=0.25)				
% ins.	Vec	Sent	Bert	Chain	Chain−LP	Vec	Sent	Bert	Chain	Chain−LP
0%	.874	.941	.949	.950	.948	.874	.941	.949	.950	.948
10%	.857	.937	.898	.948	.944	.863	.941	.904	.951	.948
20%	.854	.937	.856	.942	.943	.852	.941	.858	.950	.948
30%	.854	.937	.818	.932	.941	.856	.941	.814	.940	.948
40%	.812	.934	.782	.933	.941	.856	.941	.773	.940	.948
50%	.812	.934	.742	.930	.941	.855	.941	.736	.939	.948
60%	.812	.934	.709	.920	.941	.852	.941	.710	.929	.948
Degradation	−6.2	−0.7	−24.0	−3.0	−0.8	−2.2	−0.0	−23.9	−2.1	−0.0

Table 8: Insertion robustness at both insertion positions (strict F1 on Text+Berg). Insertion displaces true correspondences by at most $d_{\max} = f \max(q, 1{-}q)/(1{+}f)$, reaching 0.187 at q=0.5 and 0.281 at q=0.25 in the 60% condition. Banding is disabled throughout (Section 4.5), so displacement is not bounded by a search window, and no system shows a discontinuity as it grows. The asymmetric position produces larger displacement at a given insertion rate yet widens the separation between systems rather than narrowing it.

Algorithm 4 ChainAlign: Extend Phase

Require: Chain $\mathcal{C}$, source embeddings $\mathcal{S}^{(k)}$, target embeddings $\mathcal{T}^{(l)}$, N, M, skip cost δ
Ensure: Gap-filled alignment pairs $\mathcal{P}$

Identify both-sided gap regions between consecutive anchors:
for consecutive anchors $e_a, e_b \in \mathcal{C}$ **do**
  if e_b.src_start $>$ e_a.src_end **and** e_b.tgt_start $>$ e_a.tgt_end **then**
    gaps $\leftarrow$ gaps $\cup$ (e_a.src_end, e_b.src_start, e_a.tgt_end, e_b.tgt_start)
  end if
end for
Add leading gap $(0, \mathcal{C}[0].\text{src\_start}, 0, \mathcal{C}[0].\text{tgt\_start})$ if both-sided
Add trailing gap $(\mathcal{C}[-1].\text{src\_end}, N, \mathcal{C}[-1].\text{tgt\_end}, M)$ if both-sided
for each gap (s_0, s_1, t_0, t_1) **do**
  $g \leftarrow s_1 - s_0$; $h \leftarrow t_1 - t_0$
  for each resolution (k, l) with $k{+}l \leq 5$ **do**
    $C^{(k,l)} \leftarrow \cos(\mathcal{S}^{(k)}[s_0 : s_1{-}k{+}1],\ \mathcal{T}^{(l)}[t_0 : t_1{-}l{+}1])$ ▷ Slice from full-document embeddings
    $d_{ij}^{(k,l)} \leftarrow C_{ij}^{(k,l)} - \frac{1}{2}(\bar{C}_{i\cdot}^{(k,l)} + \bar{C}_{\cdot j}^{(k,l)})$ ▷ Margin scoring
  end for
  Init $F[0..g][0..h] \leftarrow -\infty$; $F[0][0] \leftarrow 0$
  for $i \leftarrow 0$ **to** g; $j \leftarrow 0$ **to** h; $(i, j) \neq (0, 0)$ **do**
    for each transition (k, l) with $k \geq 1,\ l \geq 1,\ k{+}l \leq 5$ **do**
      if $i{-}k \geq 0$ **and** $j{-}l \geq 0$ **then**
        $F[i][j] \leftarrow \max(F[i][j],\ F[i{-}k][j{-}l] + d^{(k,l)}[i{-}k][j{-}l])$
      end if
    end for
    $F[i][j] \leftarrow \max(F[i][j],\ F[i{-}1][j] + \delta)$ ▷ Skip source
    $F[i][j] \leftarrow \max(F[i][j],\ F[i][j{-}1] + \delta)$ ▷ Skip target
  end for
  Backtrack from $F[g][h]$; append match transitions to $\mathcal{P}$ with absolute indices
end for
Interleave $\mathcal{C}$ anchors and $\mathcal{P}$ gap pairs in source order
return merged alignment