Computational Humanities Research

www.cambridge.org/chr

ALMA: Automated alignment of ancient texts using linguistic and semantic analysis

Sophie Robert-Hayek[1], Andrew Hardwick[2], Davide D'Amico[2] and Frédérique Rey[3]

[1]Sens, Texte, Informatique et Histoire (STIH), Sorbonne Université, France; [2]Écritures, Université de Lorraine, France and [3]Humboldt-Universität zu Berlin, Germany

Research Article

Cite this article: Robert-Hayek Sophie, Andrew Hardwick, Davide D'Amico and Frédérique Rey. 2026. "ALMA: Automated alignment of ancient texts using linguistic and semantic analysis." *Computational Humanities Research*, **2**:e26, https://doi.org/10.1017/chr.2026.10045

Received: 25 October 2025 UTC
Revised: 26 June 2026 UTC
Accepted: 27 July 2026 UTC

Keywords:
ancient languages; automatic alignment; collation; computational philology

Corresponding author:
Sophie Robert-Hayek;
Email: sophie.robert@sorbonne-universite.fr

Abstract

Textual criticism, the task of reconstructing a text as close as possible to its original form based on multiple ancient copies, is a central concern of philologists studying ancient literature. After identifying and cataloging the available manuscripts, scholars align the texts to produce a list of variants, a process known as *collation*. Automating this task has been one of the earliest areas of interest for applying computational methods to the humanities. Most current approaches rely on sequence alignment algorithms, originally developed for bioinformatics. However, recent advances in monolingual and multilingual neural alignment, capable of modeling semantic, syntactic, and grammatical relationships, remain largely underutilized in this field. In this article, we introduce Alignment and Learning for Manuscript Analysis (ALMA). This new pipeline integrates lexical, grammatical, and semantic analysis to perform automatic textual alignment. We evaluate ALMA against three established alignment methods, each representing a different paradigm, on four manuscripts of the Gospel of John, two in Latin and two in Greek. ALMA achieves mean alignment scores of 94.12 percent (Latin) and 95.77 percent (Greek), significantly outperforming all baseline methods: alignment scores are improved over a deep-learning-only approach by 1.93 percent and 1.09 percent, over a grammar-based approach by 23.39 percent and 10.14 percent, and over a sequence alignment method by 38.57 percent and 13.39 percent, respectively, for Latin and Greek. Notably, in this study, ALMA is the only approach to achieve a median alignment score of 100 percent on both datasets. These results highlight ALMA's substantial potential for advancing computational approaches for textual collation and more generally for computational philology.

Plain language summary

A major bottleneck when working with ancient manuscripts containing the same text is the alignment task, which is necessary to highlight the divergences between different copies. Traditionally performed manually, this task is both very time-consuming and error-prone, and often limits the number of texts a scholar can work with when establishing a text.

In this article, we propose a new pipeline for word-level alignment, Alignment and Learning for Manuscript Analysis, which combines lexical, grammatical, and semantic analyses. We test our approach on Greek and Latin manuscripts of John's Gospel, one of the most widely transmitted texts in Western history.

These experiments demonstrate that our approach outperforms all baselines and improves alignment scores over a deep-learning-only approach by 1.93 percent and 1.09 percent for Latin and Greek, respectively, over a grammar-based approach by 23.39 percent and 10.14 percent for Latin and Greek, respectively, and over a sequence alignment method by 38.57 percent and 13.39 percent for Latin and Greek, respectively. Notably, our approach is the only one to achieve a median alignment score of 100 percent across both datasets.

Introduction

Philology, or the study of ancient texts, begins with the task of textual criticism, which aims to produce a text as close as possible to the original one (Maas 1958) and reconstruct the historical development of textual traditions. This endeavor necessarily begins with an initial phase of *recensio*, consisting of cataloging the available copies of the text under study, called *witnesses*, followed by a stage of *collatio*, consisting of the systematic recension of variant readings between the texts, to output a "collation." This task requires aligning the available witnesses, which is particularly time-consuming and error-prone when the texts under study have numerous

This article was awarded Open Data and Open Materials badges for transparent practices. See the Data availability statement for details.

© The Author(s), 2026. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (https://creativecommons.org/licenses/by/4.0), which permits unrestricted re-use, distribution and reproduction, provided the original article is properly cited.

witnesses and a complex transmission history that may span multiple languages. Ever since the advent of computers, researchers have sought to automate this task (Froger 1966, 1968), making automatic collations a prime exemplar of the first investigated possible applications of computational tools to problems that were once the exclusive domain of the humanities, leading to the creation of the cross-disciplinary field of "computational philology."

Until the recent significant advances in natural language processing (NLP), most existing algorithms for automatic collation relied on methods inspired by protein sequence alignment in bioinformatics, treating texts as sequences and aligning them as an optimization problem. Recently, the development of novel neural network architectures has opened new avenues for text comparison and alignment, which are used, for example, in machine translation. Yet, to date, computational philology has not fully explored the potential of these recent innovations to improve the quality of collation algorithms.

The present article aims to address this gap by proposing a new pipeline, called Alignment and Learning for Manuscript Analysis (ALMA), which combines lexical, grammatical, and semantic analysis to perform textual alignment. The relevance of this pipeline for textual alignment is then evaluated and compared with three of the most widely used and currently best-performing alignment algorithms by aligning two Latin and two Greek witnesses of the Gospel of John.

This article makes the following contributions to existing automatic approaches to collation, and more broadly, to computational philology:

- The proposal of a novel hybrid pipeline for text alignment, presenting mean alignment scores of 94.12 percent (Latin) and 95.77 percent (Greek).
- A systematic evaluation of existing alignment algorithms, taken from each paradigm of existing alignment algorithms, for the alignment of Latin and Greek manuscripts.
- A ground-truth dataset for the evaluation of alignment algorithms in Latin and Greek, composed of 4,880 and 5,078 tokens, respectively.
- An open-source implementation of the ALMA pipeline,[1] as well as the creation of a general-purpose Python library providing an easy-to-use API to perform collation using all of the algorithms presented in this article.

We have structured this article as follows. The "Problem statement, formalization, and tool positioning" section states formally the problem this study addresses. The "Automatic collation and current state of the art" section presents the current state of the art and describes the algorithms selected for comparing our proposed pipeline. "The ALMA pipeline" section presents the components of the ALMA pipeline, and the "Evaluation setup" section describes the chosen methodology for comparing our pipeline to the state of the art. An ablation study of our proposed pipeline, as well as an analysis of the impact of hyperparameters on our algorithm, is available in the "Hyperparameter tuning and pipeline optimization using ablation study" section. A comparison of our algorithm with the state of the art is presented in the "Comparison with the state of the art" section, and we provide an error analysis and discuss the limitations of our method in the "Error analysis and main methodological limitations" section. We conclude our article and list the research perspectives opened up by this study in the "Conclusion and perspectives" section.

Problem statement, formalization, and tool positioning

Problem statement

As stated in the "Introduction" section, the alignment process is the first required step for comparing two divergent witnesses, that is, two handwritten material artifacts that preserve a specific version of a text. These alignments then enable the detection of variant readings, such as substitutions, omissions, insertions, and transpositions, which are essential for understanding how the text was transmitted across centuries of copying.

The goal of this article is to automate this process by formalizing it as a *computational text alignment problem*. The core challenge is thus to determine the correct set of correspondences between the words in a reference text and those in a compared text, accounting for complex textual phenomena, such as synonymous substitutions, metathesis, and semantically equivalent but lexically distinct expressions.

Formalizing the alignment problem

Formally, within our computational framework, we consider the alignment problem as the task of aligning a reference witness $(r_i)_{1\leq i\leq n}$, an ordered sequence of n words, with a compared witness $(c_j)_{1\leq j\leq k}$ of k words.

This alignment is represented as a binary matrix $A \in \{0,1\}^{n\times k}$, where

$$a_{ij} = \begin{cases} 1, & \text{if word } r_i \text{ is aligned with word } c_j, \\ 0, & \text{otherwise.} \end{cases}$$

By allowing multiple 1s in a row or a column, this framework allows many-to-one and one-to-many alignments, frequent in the cases of glosses. An example of an alignment matrix for the same verse taken from two Old Latin manuscripts that will be used as our benchmarking test sets is shown in Figure 1. Several alignment challenges can be seen in this example: *lucet* is transposed in the compared witness, *eam*/*eum* is a gender variant, and *compraehenderunt*/*conpraehenderunt* is a spelling variant.

Integration within the editorial workflow

Automatic alignment tools, including ALMA, are designed to be used at a specific stage in the editorial workflow, as illustrated in Figure 2.

As mentioned in the introduction, the workflow begins with cataloging the witnesses available for the text under study (*recensio*) and producing transcriptions. These transcriptions can be done manually or using HTR/OCR pipelines, followed by a post-correction stage to ensure the quality of the pipeline's input: regardless of the chosen method, the transcription must constitute a faithful diplomatic representation of the manuscripts. ALMA is therefore downstream of this transcription phase: its goal is to accelerate the first step of the *collatio* phase, namely, the alignment of the different transcriptions. It is this systematic alignment that allows the expert to detect variants significant to understanding the text's history, which are then noted in the critical apparatus associated with the edition. ALMA thus necessarily takes high-quality transcriptions as input, which are then used to produce the critical edition itself.

[1] https://github.com/SphRbtHyk/alma.

https://doi.org/10.1017/chr.2026.10045 Published online by Cambridge University Press

Reference witness: r = (et, lux, in, tenebris, lucet, et, tenebrae, eam, non, compraehenderunt)
Compared witness c = (et, lux, lucet, in, tenebris, et, tenebrae, eum, non, conpraehenderunt)

Alignment matrix:

	et	lux	lucet	in	tenebris	et	tenebrae	eum	non	conpraehenderunt
et	1	0	0	0	0	0	0	0	0	0
lux	0	1	0	0	0	0	0	0	0	0
in	0	0	0	1	0	0	0	0	0	0
tenebris	0	0	0	0	1	0	0	0	0	0
lucet	0	0	1	0	0	0	0	0	0	0
et	0	0	0	0	0	1	0	0	0	0
tenebrae	0	0	0	0	0	0	1	0	0	0
eam	0	0	0	0	0	0	0	1	0	0
non	0	0	0	0	0	0	0	0	1	0
compraehenderunt	0	0	0	0	0	0	0	0	0	1

Figure 1. Example of an alignment matrix of John 1:5 in *Codex Corbeiensis Secundus* (reference witness) and *Codex Palatinus* (compared witness).

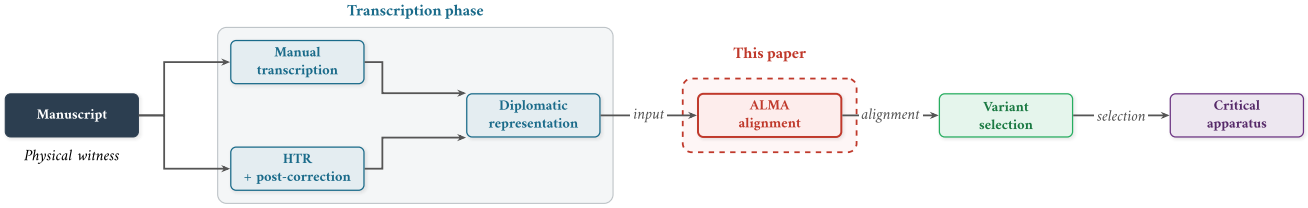


Figure 2. Integration of ALMA within the editorial workflow.

Automatic collation and current state of the art

Sequence-based collation algorithms

Dom Froger was one of the first scholars to understand the potential of computational approaches for philology, in his seminal book on computational philology research, *La critique des textes et son automatisation* (Froger 1968), where he describes a possible use of automatic tools for collation (Froger 1966; Nury and Spadini 2020). This algorithm aimed to simulate the philologist's manual approach: a main reference text is selected, and other texts are compared against it. The texts are then compared word by word until a difference is found. The algorithm then continues comparing words using a sliding window until a match between the texts is found.

Following this proposal, automatic collation approaches began to rely on sequence-analysis algorithms, inspired by protein sequence alignment, as early as the late 1980s (Robinson 1989). This class of algorithms aligns words using string similarity between their raw or lemmatized forms to minimize the total cost of sequence alignment. Among the existing algorithms, we can mention the Needleman–Wunsch algorithm (Needleman and Wunsch 1970), progressive multiple alignment (Spencer and Howe 2004), and, more recently, Dekker's algorithm (Dekker and Middell 2011). Dekker's algorithm, which will be compared to ALMA, identifies the smallest set of longest-matching tokens that accounts for all variants between two texts.[2] Tokens can then either be considered as exact matches if the tokens are the same, or similar matches if their string similarity, such as the Levenshtein distance (Levenshtein 1966), falls below a set threshold.

While these approaches are computationally very efficient and widely used, they present several drawbacks. First of all, this class of algorithms is fundamentally based on sequence proximity and string identity (or identity of pre-processed forms, such as lemmas), and its alignment strategy remains agnostic to syntactic function and semantic meaning. This makes them inherently ill-suited to languages with freer word order, such as Latin and Greek, where meaning is conveyed by inflectional morphology rather than linear order. Secondly, because they rely entirely on token-level comparisons, they cannot generalize to cross-language alignment and fail to realize the potential of automatic multilingual collation. Finally, they are very vulnerable to alignment ambiguity caused by high-frequency function words, such as conjunctions, which are omnipresent in Greek and Latin texts and often function as punctuation. When multiple tokens in both witnesses share identical words, the algorithm lacks the syntactic or semantic context to choose the correct correspondence. This often results in a "first-available" match error based on positional proximity, which then propagates, skewing the entire subsequent alignment: an incorrect early alignment of a common conjunction can force the misalignment of subsequent content words, as the algorithm becomes locked into a locally coherent but globally erroneous path.

Syntax analysis-based collation algorithms

To address the limitations of sequence-based alignment models presented in the previous section, models that employ modern NLP tools to incorporate grammatical, syntactic, and semantic information during the alignment process have been developed. Among the most efficient approaches for monolingual alignment, Sultan, Bethard, and Sumner (2014) have proposed an algorithm that relies on a language model to identify lemmas, dependencies, named entities, and content words, and that aligns words with similar functions from the reference and the compared witnesses. The algorithm begins by aligning similar word sequences by lemma, then uses syntactic dependencies and nearby context to align the remaining words, using a similarity measure based on a paraphrase database (PPDB). Finally, stop-words are aligned

[2] A more advanced version of Dekker's algorithm adds a second phase of transposition detection to this first phase, but this has not yet been implemented in the standard Python library implementing Dekker's algorithm, CollateX (Dekker and Middell 2022). We therefore do not use it in our benchmarking.

https://doi.org/10.1017/chr.2026.10045 Published online by Cambridge University Press

using a predefined list and contextual cues. Although this aligner accounts for lemmas, allowing it to match inflected forms unlike Dekker's algorithm, it cannot detect semantically equivalent words not present in the PPDB and thus cannot function properly for languages without a well-annotated PPDB. Additionally, it restricts alignments to unique token pairs, thereby preventing the capture of one-to-many substitutions. In this study, due to its strong performance, the algorithm proposed by Sultan et al. has been selected for our benchmarking and is referred to as "POS."

Deep learning-based collation algorithms

Rapid advances in NLP, following the development of novel neural network architectures such as transformers (Vaswani et al. 2017), have led to the development of new alignment models. While these models were primarily intended for machine translation via multilingual alignment, they can be easily adapted to the task of monolingual alignment. Methods can be separated into supervised approaches (Nagata, Chousa, and Nishino 2020; Stengel-Eskin et al. 2019) that require training a model to perform the alignments, and unsupervised approaches, able to learn directly from the co-occurrence of terms between the source and the target sentence (Dou and Neubig 2021; Sabet et al. 2020; Zenkel, Wuebker, and DeNero 2020). When it comes to the study of these models for the alignment of ancient languages, the related literature is considerably sparser: other than the major studies performed by Palladino, Shamsian, and Yousef (2022) focusing on aligning Greek and Latin translations, little work has been done for the alignment of ancient languages, and no systematic benchmarking has been performed.

Of particular importance to us due to its performance, versatility across a wide range of languages, and ability to be fine-tuned on a given dataset is the *SimAlign* algorithm proposed by Sabet et al. (2020), which we selected for our comparison with ALMA. This unsupervised method compares sentences via cosine similarity computed between token embeddings. To match two tokens using this similarity matrix, the authors propose a new algorithm called *itermax*, which performs the matching iteratively, in the case of words where the similarity argmaxes do not correspond in the source and target sentence: the alignment between tokens is then performed with a discounting factor α, until a match is found. A pruning approach ensures that similarities are meaningful by using an entropy measure before matching, and a proximity criterion is used to minimize distortion. While this approach has proven to be effective, especially when the model is fine-tuned on the dataset to align, it suffers from two key limitations: (1) The method tends to overmatch words by relying on greedy iterative matching, which is ill-suited for texts containing insertions or paraphrased material in either the reference or the compared witness. This often results in false positive alignments, especially in contexts where textual variation or expansion is present; (2) The alignment process prioritizes semantic similarity without accounting for the syntactic function of the matched words. As a result, it may incorrectly align words from different grammatical categories (e.g., verbs with nouns) as long as their semantic content is related.

When comparing this algorithm to ALMA, we use the most effective version presented in the article for benchmarking: we retrieve the embeddings from the eighth layer of the neural embeddings and employ the *itermax* algorithm.[3] The two models tested by the original authors, namely, mBERT and XLM-RoBERTa, are fine-tuned on our domain-specific dataset, as detailed in the "Neural models training" section.

[3]The superior performance of the *itermax* algorithm compared to the *argmax* approach has been confirmed on our dataset in preliminary experiments.

Limits of the current state of the art

To overcome the current limitations of the state of the art, we introduce the ALMA pipeline, which integrates the core principles of existing text alignment paradigms. It moves beyond a single-method approach by jointly optimizing lemma alignments, grammatical coherence, and semantic similarity, thereby leveraging the strengths of each existing approach while mitigating their weaknesses. To systematically evaluate the performance of our new suggested algorithm, we compare it to three algorithms of the state of the art, each representing a dominant paradigm: Dekker's algorithm (sequence-based) as implemented in CollateX, Sultan et al.'s algorithm (grammar-based), and Sabet et al.'s algorithm SimAlign (deep learning-based).

The ALMA pipeline

General workflow

The ALMA pipeline implements a deterministic workflow (see Figure 3 for a schematic view of the pipeline and a step-by-step example of the alignment process is provided in the Appendix) to compute word-level alignments between two input witnesses. The pipeline is designed to integrate linguistic information hierarchically, progressing from lexical and semantic similarity to syntactic structure, thereby maximizing alignment quality while reducing the potential errors of each existing alignment approach. Each step operates only on tokens not matched in previous steps.

The general workflow is designed as follows and details regarding the inner workings of each component are provided in the "Detail of each component of the pipeline" section.

1. *Lemma-based matching:* The first component matches tokens based on a strict lemma identity to match identical lexical correspondences, similarly to sequence-based approaches. Word order is not taken into account at this step to provide resilience for languages with a freer word order.
2. *Semantic similarity matching:* To account for lexical variation such as synonymy, the algorithm then matches the different tokens based on their similarity in terms of embedding, as detailed in the "Component 2: Semantic matcher – Detecting synonyms using embeddings" section. We perform semantic matching before syntactic matching to establish simpler alignments, such as exact matches, transpositions, and synonymic substitutions, before moving on to more complex grammatical correspondences. Additionally, this approach is reasonable from a philological viewpoint: vocabulary and meaning often remain stable throughout textual transmission, whereas syntactic structures are more likely to diverge across witnesses. Additionally, because semantic matching precedes syntactic analysis, it helps minimize error propagation from potential syntactic parsing inaccuracies, a particularly significant consideration for resource-constrained languages.
3. *POS-based matching:* The next step matches tokens that share the same POS tag among tokens that have not yet been matched, to capture words with the same function in the reference and compared witnesses that were not considered as close by the semantic matcher.

https://doi.org/10.1017/chr.2026.10045 Published online by Cambridge University Press

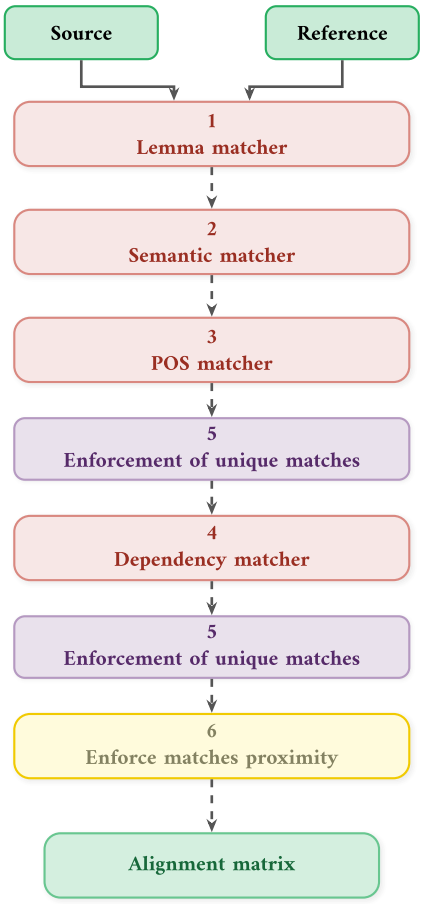


Figure 3. Overview of the multi-stage ALMA matching pipeline.

4. *Dependency syntactic matching:* This step incorporates deeper syntactic structure by matching tokens that share the same grammatical dependency relation, using universal dependency tags, to capture words that play an identical functional role in their respective syntactic trees.

Two additional filters, whose efficiency will be discussed in the ablation study of the "Hyperparameter tuning and pipeline optimization using ablation study" section, are applied to enhance the robustness of the pipeline:

- *Enforcing uniqueness of matches using similarity:* A uniqueness (injective) constraint is applied to the alignments proposed by the POS and Dependency matchers. This constraint is not applied after the Semantic matcher, allowing ALMA to capture genuine one-to-many phenomena, such as paraphrases, at that stage. The rationale for this selective injectivity is a calculated risk management strategy. While the POS and Dependency matchers effectively identify tokens that share a grammatical role, they are agnostic to meaning and thus highly prone to proposing multiple, spurious alignments for common function words or syntactically similar but semantically distinct content words (e.g., aligning every noun in the reference with every noun in the target). Enforcing semantic injectivity at these later stages serves as a safeguard against syntactic and grammatical overmatch, which would otherwise cascade through the pipeline. As discussed in the ablation study presented in the "Hyperparameter tuning and pipeline optimization using ablation study" section, this constraint improves precision without significantly reducing recall, since the majority of genuine one-to-many alignments are already captured at the semantic step.
- *Proximity-based pruning:* A final post-alignment pruning step enforces a proximity constraint by ensuring that matches from the reference and compared witnesses are performed within the same word neighborhood. While the initial matching pipeline operates without word-order constraints to handle free-word-order languages, a final proximity filter ensures topological plausibility by pruning long-distance alignments and ensuring contextually coherent alignments.

Detail of each component of the pipeline

Component 0: Parsing the witnesses

The initial step in the alignment process involves parsing both the reference and compared witnesses to collect three key pieces of information for the matching process: the lemmatized form of the considered words, the part of speech of each word, and the sentence syntax. We support two parsing libraries, SpaCy (Honnibal et al. 2020) and Stanza (Qi et al. 2020).

Component 1: Lemma matcher – Detecting common vocabulary

The second step of the pipeline consists of aligning together words presenting the same lemma. The algorithm iterates through the content of the lemmatized witnesses and marks as a match words that show precisely the same lemmatized form.

Component 2: Semantic matcher – Detecting synonyms using embeddings

The semantic matcher then matches together words considered synonyms within the same embedded space. From an embedding matrix of the two segments to be aligned, the cosine similarity between each pair of words is calculated to create a similarity matrix $S \in \mathbb{R}^{n \times k}$. For each vector embedding of word r_i, $\mathbf{r_i}$, in the reference witness and of word c_j, $\mathbf{c_j}$, in the compared witnesses, the coordinate element (i,j) of S corresponds to

$$S_{ij} = \cos(\mathbf{r}_i, \mathbf{c}_j) = \frac{\mathbf{r}_i \cdot \mathbf{c}_j}{\|\mathbf{r}_i\| \|\mathbf{c}_j\|}.$$

Each row of the S matrix is then normalized by transforming it into z-normalized scores:

$$z_{ij} = \frac{S_{ij} - \mu_i}{\sigma_i},$$

where μ_i and σ_i denote the mean and standard deviation of the similarities of row i.

This transformation allows us to assess the homogeneity of similarity scores and avoid treating words as synonyms when their similarity does not stand out from the rest of the sentence. To ensure this, we consider only tokens with a z-score above a predefined threshold as potential matches. For each unaligned token in the source sentence, the target tokens with similarity scores exceeding the threshold are added as alignments in the binary matrix.

Formally, for each reference word r_i that has not yet been aligned, the compared word c_{l^*} is considered as a match if $z_{ij^*} > \delta$, where δ is a fixed threshold, that corresponds to the standard deviation from the mean of the retained similarity. Because of this, if two words are selected as synonyms, it implies that their similarity is at least δ standard deviations above the mean μ_i. This allows us to capture only words that stand out in terms

https://doi.org/10.1017/chr.2026.10045 Published online by Cambridge University Press

of semantic proximity, instead of matching words using a simple *argmax* in terms of similarity, which might be due to the random fluctuation of a flat distribution of semantic proximity in the case of a compared and target verse presenting no synonym in common. The coordinate element (i,l^*) of the alignment matrix $A \in \{0,1\}^{n\times k}$ is then set to 1. The impact of the value of this score is evaluated in the "Hyperparameter tuning and pipeline optimization using ablation study" section.

The same procedure is then applied column by column to address the missing alignments from the perspective of the compared sentence.

Component 3: POS matcher – Matching based on grammatical category

This matching operation compares the grammatical categories of unaligned words between the reference and compared witnesses. Words that have not been aligned in previous lemma-based or semantic matching steps, and that share the same part-of-speech tag in both witnesses, are considered potential matches. This step captures words that serve identical grammatical functions but exhibit lexical variation or semantic divergence that the previous matchers did not capture. The resulting alignments are subsequently filtered to enforce uniqueness based on embedding similarity, ensuring that only the most semantically plausible match is retained when multiple candidates share the same POS tag.

Component 4: Dependency matcher – Matching based on syntactic function

The final matching operation incorporates deeper syntactic structure by comparing the grammatical dependency relations of unaligned words. Words that remain unaligned after the previous three steps, and that share the same syntactic function (as defined by their universal dependency relation labels) in their respective dependency trees, are considered matches. This component aims to capture words that play identical functional roles, such as subjects, objects, or modifiers, even when they differ in both lexical form and grammatical category. As with the POS-based matches, these alignments undergo a uniqueness filter based on semantic similarity to select the most contextually appropriate correspondence when multiple dependency-based matches are possible.

Component 5: Enforcing unique matches using the semantic similarity matrix

After the POS and dependency matchers, we enforce matching uniqueness to avoid matches of words with similar grammatical functions but with different semantics.

To ensure the uniqueness of alignments between the two witnesses, an injective alignment constraint is applied to the binary alignment matrix $A \in \{0,1\}^{n\times k}$, as alignment outputted at each step may contain multiple alignments per row or column. We select in each row i (if $\sum_j A_{ij} > 1$, i.e., there is more than one match for word i in the reference segment) the column j^* that maximizes the similarity:

$$j^* = \arg\max_j S_{ij},$$

all other alignments A_{ij} for $j \neq j^*$ are set to 0.

A symmetric operation is then applied to the columns: for any column j such that $\sum_i A_{ij} > 1$, only the row i^* satisfying

$$i^* = \arg\max_i S_{ij}$$

is retained. Because of this transformation, the grammatical matching, based on POS or the syntactic tree, always selects the option with the highest similarity in terms of embedding and thus of context.

Component 6: Enforcing local proximity between word matches

Finally, the last step of the alignment process involves enforcing local proximity between word matches. This ensures that aligned words are present in the same window in both the reference and the compared witnesses, thereby avoiding matches that are too far apart in the segment. While the SimAlign paper relies on the distortion correction proposed in the IBM 2 alignment model (Brown et al. 1993), we adopt a more straightforward approach by applying a window-based constraint, measuring proximity along the diagonal of the alignment matrix.

For each match pair at coordinates (i,j), we compute a relative distortion measure:

$$\delta_{ij} = \frac{i}{n} - \frac{j}{k},$$

that is, the difference between the normalized relative positions in each sentence, where segments can have very different lengths. If the absolute distortion $|\delta_{ij}|$ exceeds a fixed threshold $\tau \in [0,1]$ corresponding to the percentage of the lengths of the aligned segments, the alignment is removed, that is, A_{ij} is set to 0. Different values of the proximity window are tested in the evaluation section. This operation retains only alignments located at a reasonable proportional distance from the main diagonal of the matrix, which helps the correspondences to be constrained to follow an approximate linear order.

Optimization of the pipeline

All the pipeline elements presented above are systematically evaluated using an ablation protocol to optimize the pipeline architecture, select the most effective one, and determine the best-performing hyperparameters. The results of these experiments and our final choice of architecture are reported in the "Hyperparameter tuning and pipeline optimization using ablation study" section.

Implementation and reproducibility

All the code used in this article is available as a Python package on GitHub.[4] We use the Python port of CollateX and the authors' implementations for the POS algorithm and SimAlign. ALMA is implemented alongside each algorithm and accessed via a straightforward wrapper API. The scripts for training the models, obtaining sample ground-truth data from the manuscript transcriptions, and benchmarking are also available, enabling all results presented in this article to be reproduced.

For the end-user, ALMA can be used in two complementary modes depending on the user's technical background. For computational researchers who require programmatic access or batch processing, ALMA exposes a Python API that is fully documented. For users with limited programming experience, ALMA is distributed with a web application that can be deployed locally from the source code. The interface allows the user to paste or upload the two witnesses to be aligned, select the desired algorithm and configure its hyperparameters, and visualize the resulting alignment as a list of matches and non-matches.

[4] https://github.com/SphRbtHyk/alma.

https://doi.org/10.1017/chr.2026.10045 Published online by Cambridge University Press

Extending to other languages

Although our experiments have only been conducted on Greek and Latin so far, our model can be immediately extended to languages that have a SpaCy or Stanza parser, and its implementation, both via the Python API and via the web interface, allows for this. Users can also use their own trained model, whether with the SpaCy API or the Stanza API.

The main problem lies in the desire to use ALMA for languages that still lack lemmatizers and taggers. Our implementation anticipates this case by allowing complete disabling of the tagger and, therefore, of linguistic analysis. While this necessarily reduces performance, we show in the ablation study (see the "Hyperparameter tuning and pipeline optimization using ablation study" section) that this inability to use a tagger does not result in an extreme drop in performance, and ALMA remains a very good candidate for automatic alignment. This dependency on neural models as a current limitation of our work is discussed in the "Dependence on neural model" section.

Evaluation setup

Selected codices

To demonstrate ALMA's effectiveness on alignment tasks representative of philological work, we evaluate the algorithm on two monolingual alignment tasks, each drawn from a pair of manuscripts of the Gospel of John. This text from the New Testament was selected as relevant to our study for three main reasons: (1) the New Testament is one of the most copied texts in Western history, with nearly 6,000 witnesses in Greek and 10,000 witnesses in Latin. This large number of witnesses has prevented an exhaustive study, as such an analysis is beyond the capabilities of manual-only methods. High-performing automatic alignments thus have a strong potential for the discovery of relationships between manuscripts that have not yet been explored; (2) this text has been transmitted through several well-identified textual families, which has made it possible to choose, for this study, manuscripts from different textual traditions. This ensures that our selected texts present variants relevant to evaluating the performance of our algorithm; (3) high-quality transcriptions and manuscript scans are available thanks to numerous transcription projects, enabling us to use error-free transcriptions for our dataset. It must additionally be noted that we are faithful to the existing transcriptions and do not supply missing words or letters due to damage.

Latin manuscripts

For the Latin corpora, we selected two codices containing two different Latin versions of the Gospel of John, derived from the *Vetus Latina* tradition. These Latin texts, which circulated freely until the standardization of the Vulgate by Jerome of Stridon, show great freedom in the text they transmit, as they come from divergent textual families and possibly from different translations (Burton 2000; Houghton 2016). The first codex selected, *Codex Corbeiensis Secundus*, presents a type of text known as "European." The manuscript, long held in the library of Corbie Abbey and now preserved at the *Bibliothèque nationale de France* (Paris, France, under ID 17225), is written on vellum in uncial script. Its copying date is estimated to be between the 4th and 5th centuries AD (Buchanan 1907). The second codex is the *Codex Palatinus*, belonging to the so-called "African" textual tradition, due to its strong affinity with quotations from the African Church Fathers. This codex, probably copied in Italy and currently preserved at Buonconsiglio Castle (Trento, Italy, under ID 1589) (Gryson 1999), is also written on vellum, previously dyed purple. Because these two manuscripts come from different textual traditions, they exhibit numerous divergences and present several challenges for the alignment algorithms. We rely on the transcription by Burton et al. in their electronic synoptic edition of the Old Latin manuscripts (Burton et al. 2015).

Greek manuscripts

For alignment in Greek, we select two Greek witnesses to the Gospel of John, both from different textual traditions. The first selected witness is *Codex Vaticanus*, one of the oldest complete codices of the biblical text in Greek. Dating from the 4th century and copied on vellum in uncial script, its origin is debated between Italy and Alexandria. It has been attested as belonging to the Vatican Library since the 15th century, under the ID Vat. gr. 1209. It is considered one of the most representative witnesses to the so-called Alexandrian text.

The second codex chosen for our study, the *Codex Bezae*, presents some of the most divergent texts in the Gospels and the Acts of the Apostles. Of disputed origin, possibly from Gaul, it was preserved for centuries at the monastery of Saint-Irénée in Lyon, then resurfaced during the Reformation, possibly at the Council of Trent and in the critical apparatus of the New Testament text established by Stephanus in his Editio Regia. Recovered by Theodore Beza after the sack that destroyed the monastery where it was kept, it was then sent to Cambridge and corresponds to what is now called the "Western" type of text.

All transcriptions used are provided by the Institute for New Testament Textual Research (2024).

Building the alignment dataset

Annotation protocol and guidelines

All alignments have been performed by domain experts using the UGARIT tagging platform (Yousef et al. 2022). The annotation was initially done by the first annotator directly on the platform, and 40 percent of the annotated verses in each language were randomly selected and commented on by the second annotator to evaluate inter-annotator agreement and discuss any potential annotation conflicts. We established the following annotation rules for the different possibilities of observed matches and how to resolve them:

- *Sure match*: A match is considered sure when it is possible to find an unambiguous correspondence between the two texts, particularly in the case of an exact lexical match, grammatical or morphological variants, or clear synonymic substitutions.
- *Omissions and insertions*: Tokens that are present in only one of the two witnesses and have no correspondence in the other are left unaligned when the annotator determines that there is no strict semantic equivalent in the other witness. Notably, structural omissions, such as an absent subject, are treated as omissions.
- *Conjunctive discourse markers*: When a discourse marker (e.g., *δέ*, *καί*, and *autem*) is found in only one witness, and there is no structural counterpart in the other, the marker is treated as an insertion and left unaligned. When two witnesses use distinct conjunctions occupying the same syntactic position, alignment is governed by semantic equivalence: functionally

https://doi.org/10.1017/chr.2026.10045 Published online by Cambridge University Press

Table 1. Characteristics of the monolingual gold-standard alignment.

	Latin	Greek
Compared codices	Palatinus/ Corbeiensis Secundus	Bezae/ Vaticanus
Total tokens	4,880	5,078
Tokens with matches (%)	87.9	89.9
Tokens without matches (%)	12.1	10.1
Non-identity matches (%)	25.1	10.6

synonymous markers are aligned as sure matches, whereas markers with distinct discursive functions (coordination vs. opposition, for example) are left as unaligned and counted as an insertion/omission.

- *One-to-many alignments*: When a word in one witness is semantically rendered by several words in the other witness, the word is associated with the several other words. A typical example encountered when annotating the dataset is when one of the parallel words is negated, such as *non sciebat* and *nesciebat*.
- *Repetitions*: One difficulty encountered concerns the alignment of words that are repeated in one witness but not in the other: for example, if a conjunction is repeated several times, it is difficult to determine which conjunction each alignment refers to. In these cases, we apply a "nearest counterpart" rule: the target token nearest in the linear sequence to the source token is selected for alignment.

Inter-annotator agreement was measured using Cohen's $\varkappa$ computed at the token-pair level, and we computed $\varkappa_{\text{Latin}} = 0.86$ and $\varkappa_{\text{Greek}} = 0.94$. This corresponds to almost perfect agreement on the Landis–Koch scale (Landis and Koch 1977), showing a strong reliability regarding our tagging choices. Most of our inter-annotator disagreements were linked to the possible alignment of ambiguous discourse markers. Additionally, we observe that $\varkappa$ is lower for Latin ($\varkappa = 0.86$) than for Greek ($\varkappa = 0.94$), consistent with the higher proportion of non-identity matches in the Latin dataset, as described in Table 1. All remaining disagreements were resolved by adjudication before the gold standard was finalized.

Data sampling and characteristics

For 19 chapters of the Gospel of John in these four codices, we randomly select 15 verses per chapter, resulting in a total of 285 verses for each language. This sampling yielded 4,880 Latin and 5,078 Greek tokens for alignment. A summary of the data used for benchmarking is provided in Table 1. The two datasets provide distinct challenges. The Latin data exhibit significant scribal variation, with over 25 percent of matched tokens being synonyms, paraphrases, or grammatical variants. The Greek dataset is more stable, but, being written without diacritics, it requires training a custom parsing pipeline (detailed in the "Neural models training" section), which is a potential source of error during alignment.

For further interpretation of the alignment algorithms' performance, it is important to consider that these are alignments of the same textual tradition, albeit with some scribal modifications. The high proportion of tokens with an annotated counterpart (>87%) is characteristic of the domain and provides a realistic benchmark.

Data availability

All datasets are publicly visible on the UGARIT tagging platform. They are also available in the GitHub repository accompanying this article in the *CSV* and *XML* formats, to facilitate future work on automatic alignment algorithms for Greek and Latin, and provide a dataset for further benchmarking of alignment algorithms for these languages.

Evaluation score

We evaluate alignment performance using the standard alignment error rate (AER) metric, designed for evaluating the quality of alignment algorithms (Och and Ney 2000), which considers precision and recall against a gold standard containing both "Sure" (S) and "Possible" (P) alignments. However, in our context of aligning manuscript variants within the same language, the distinction between "Sure" and "Possible" alignments is unnecessary, as the manual alignment is monolingual and the variations are due to scribal changes on the same textual tradition, with considerably less ambiguity than when comparing translations. Consequently, for our gold-standard alignments, we define the set of sure alignments S to be identical to the set of possible alignments P. Under the condition that $S = P$, the AER simplifies to

$$\text{AER}(A,S) = 1 - \frac{2 \times |A \cap S|}{|A| + |S|}$$

with S as the set of all gold-standard alignments and A as the set of alignments proposed by an algorithm. We report the *alignment score* (AS), defined as $(1 - AER) \times 100$, to return a performance that equals 100 percent in the case of perfect matching, alongside precision and recall to indicate the algorithm's tendency to propose incorrect links (affecting Precision) and its tendency to miss valid ones (affecting Recall). We thus define

$$\text{Precision} = \frac{|A \cap S|}{|A|}, \quad \text{Recall} = \frac{|A \cap S|}{|S|}.$$

All scores are calculated per verse, and we report the mean over verses within each dataset.

Compared algorithms and hyperparameters testing

We compare the three algorithms from each category of existing alignment algorithms described in the "Automatic collation and current state of the art" section to ALMA: the Dekker algorithm as implemented in CollateX (sequence-based), the POS algorithm (grammar-based), and SimAlign (deep learning-based). We then apply all the algorithms to the Latin and Greek ground-truth data and measure the evaluation scores for each algorithm, as defined in the previous section. The statistical significance of the performance difference is evaluated using Welch's t-test (Welch 1947), considering each verse as an individual and evaluating the variation in AS depending on the considered algorithms. We use a 95 percent confidence level, treating p-values below 0.05 as significant. When commenting on the significance of the results, we specify the value of the p-value itself. This complies with recent criticism of only reporting statistical significance rather than the value of the p-value itself, as this value is informative about the probability of obtaining a test statistic at least as extreme as the one observed under H_0 (Andrade 2019).

The hyperparameters tested for each algorithm are available in Table 2.

https://doi.org/10.1017/chr.2026.10045 Published online by Cambridge University Press

Table 2. Summary of tested hyperparameters for each algorithm.

Algorithm	Hyperparameters
Dekker	Use Levenshtein distance ∈ {*True*, *False*}
POS	No hyperparameters
SimAlign	Model ∈ {bert-cased, xlm-roberta, bert-cased-fine-tuned, xlm-roberta fine-tuned, laberta, laberta fine-tuned, greberta, greberta fine-tuned }
ALMA	Model ∈ {bert-cased, xlm-roberta, bert-cased-fine-tuned, xlm-roberta fine-tuned, laberta, laberta fine-tuned, greberta, greberta fine-tuned}
	Z-score threshold ∈ {1, 2, 3}
	Distortion window ∈ {10%, 30%, 70%}

Table 3. Size of the fine-tuning corpus and elapsed training time for each model in Latin and Greek.

Language	Total words in training set	Models	Elapsed time (s)
Latin	29,447	LaBERTa	1,495
		mBERT	413
		XLM-RoBERTa	1,988
Greek	163,415	GreBERTa	497
		mBERT	91
		XLM-RoBERTa	556

Neural models training

Grammatical parsing

When measuring the pipeline's performance on the Latin dataset, we used the LatinCy (Burns 2023) spaCy library, which provides a lemmatizer and a POS and dependency tagger. The Greek dataset presents an additional difficulty: the codices considered for our study lack accentuation (diacritics), as they were written in the uncial script, that is, majuscule.

As all currently available parsing models (such as GreCy [Myerston and López 2023], OdyCy [Kostkan et al. 2023], or more recently large language models [Riemenschneider and Frank 2023]) are trained on Ancient Greek from printed editions, they have been trained to tag only words with accentuation, and experience a severe drop in performance whenever applied to text without diacritics. We thus trained our own parsing model by fine-tuning the Stanza Ancient Greek model on the PROIEL dataset (Haug and Jøhndal 2008; Qi et al. 2020) after removing its diacritics. The performance of this model trained on the PROIEL dataset is 97.16 percent for UPOS tagging and 95.67 percent for lemmatization on the evaluation dataset. This newly trained model is also available on the article's companion GitHub repository.

Computing word embeddings

To build the similarity matrix, we trained several models and extracted contextual embeddings from them, which we then used for both the SimAlign model and ALMA. We used several different types of models for the embedding: (1) two non-specialized pretrained multilingual models, *Bert Multilingual Cased* (Devlin et al. 2019) and *XLM Roberta* (Conneau et al. 2020) in their base version, as in the original paper proposing the SimAlign algorithm; (2) two specialized models for Ancient Greek and Latin, taken from the work of Riemenschneider and Frank (2023): LaBERTa for the Latin embeddings and GreBERTa for the Greek; and (3) each of these models, but fine-tuned with the witnesses from each considered tradition: for our Old Latin alignments, we used data from seven complete witnesses of the Gospel of John (Gryson 1999), and for our alignment in Greek, four major codices containing the Gospel text.

The total number of tokens for each trained model, as well as the complete elapsed time for the training, is available in Table 3.

The models were trained using the masked language modeling objective, an unsupervised approach initially used to train the BERT model (Devlin et al. 2019). This involves randomly masking a proportion of tokens and training the model to predict the masked tokens using all tokens in the sequence. By learning the relationship between the different tokens in the sentence, the model can obtain their representations and thus their semantic proximity in the newly generated space.

In this study, we performed training by randomly selecting 15 percent of the input tokens to be predicted from their bidirectional context. We then use a partial data corruption procedure, replacing 80 percent of the selected tokens with the mask token, replacing another 10 percent with a random token drawn from the entire vocabulary, and leaving the remaining 10 percent unchanged. We then used the trained model to retrieve the embedding of each token to be collated, specifically from the eighth transformer hidden layer, as demonstrated to be the most effective in the SimAlign paper. Since the embeddings are performed at the token level, rather than directly on the word, we choose, like the authors of SimAlign, to average the representations of the subtokens corresponding to each word.

Hyperparameter tuning and pipeline optimization using ablation study

Impact and selection of hyperparameters

Impact of model tuning

Table 4 reports ALMA's performance across embedding models, showing that fine-tuning consistently affects the quality of the alignments. In both cases, the specialized models LaBERTa and GreBERTa provided the best out-of-the-box AS score compared to the more generic multilingual models XLM-RoBERTa and mBERT, suggesting that domain-specific pretraining is a key factor for alignment quality. However, the differences between the different selected embedding models are not statistically significant ($p > 0.3$ in all considered cases), indicating that even non-specialized models perform very efficiently and that our solution need not be restricted to languages with existing specialized models.

Fine-tuning systematically improved the performance of all the considered embedding models. For Latin, fine-tuning the LaBERTa model improved AS by 1.13 points relative to the non-fine-tuned version. For Greek, fine-tuning GreBERTa provided more modest gains of 0.41 AS points. In all cases, fine-tuning models on the data improved their performance, regardless of the model considered. Given the demonstrated importance of fine-tuning, ALMA provides a simple command-line interface for model training that requires only the parallel texts as input. Incorporating fine-tuning into the pipeline thus becomes straightforward for the users, including those coming from a humanities background.

https://doi.org/10.1017/chr.2026.10045 Published online by Cambridge University Press

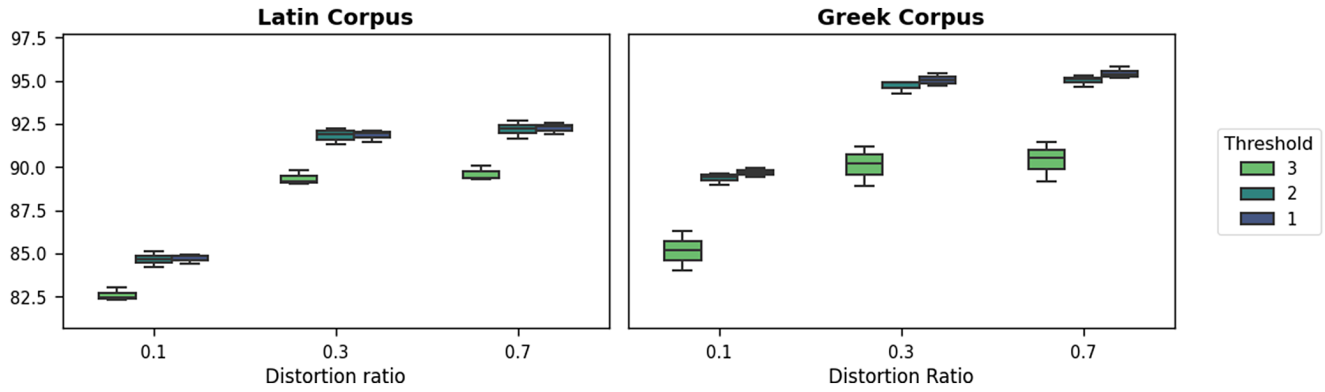


Figure 4. Impact of hyperparameters on AS. The impact of hyperparameters has been measured on the fine-tuned model.

Table 4. Impact of model fine-tuning in Latin and in Greek, ranked by AS score within each language.

Language	Base model	Tuned	AS
Latin	*LaBERTa*	*True*	89.17
		False	88.04
	XLM-RoBERTa	True	88.78
		False	87.48
	mBERT	True	88.41
		False	87.92
Greek	*GreBERTa*	*True*	92.08
		False	91.67
	mBERT	True	91.85
		False	91.26
	XLM-RoBERTa	True	91.05
		False	89.12

Note: The results are averaged over all other possible hyperparameter combinations. Best results are indicated in red.

Z-score threshold and distortion filter

The impact of the threshold used to determine whether two words are synonyms and the distortion filter indicating the percentage of distance allowed between matched words in the reference and the compared sentences are shown in Figure 4.

The negative impact of an excessively low distortion parameter is evident, as it significantly reduces the pipeline's AS performance on average by -6.98 points in Latin (p-value < 0.001) and -5.33 in Greek (p-value < 0.001) for each tested threshold when using a strict distortion filter of 10 percent. Using a window of 70 percent leads to a slight performance increase of 0.31 (p-value = 0.52) for the Latin and of 0.32 for the Greek (p-value = 0.73), compared to a window of 30 percent.

Regarding the impact of the semantic threshold, we find that, as expected, a threshold that is too high prevents matching because it treats no words as synonyms. Choosing the high value of 3 thus significantly reduces, on average, the AS by 2.54 points (Latin, p-value = 0.01) and 4.86 points (Greek, p-value < 0.001) across all configurations for each language. The advantage of using threshold 1 over 2 is slight, with a non-significant improvement of 0.03 (p-value = 0.98) for the Latin and of 0.33 (p-value = 0.74) for the Greek.

Globally, as indicated by the boxplot distribution, the impact of the two parameters is very similar across languages. There is also little interaction between the hyperparameters, with the threshold having a similar effect across distortion ratios. The best combination of threshold 1 and a distortion ratio of 70 percent yields significant improvements of 3.87 points for Latin (p-value < 0.001) and 4.27 points for Greek (p-value < 0.001) compared to all other averaged configurations. We can thus conclude that, aside from extreme values, the hyperparameters, including the selected models, have a moderate impact on the pipeline's performance when selected within a reasonable range. *We retain the best-performing semantic similarity threshold of 1, a distortion filter of 0.7, and the specialized model for each language for the remainder of the ablation study and the article.*

Design of the ablation study

To evaluate the importance of each of the components shown in Figure 3, select the best-performing architecture, and better understand the internal workings of the matching pipeline, we evaluate the performance of each component by asking the following questions: **(RQ1)** What is the performance of the pipeline when removing the components one by one? **(RQ2)** What is the performance of the pipeline when removing the pruning filters? and **(RQ3)** What is the performance of each component alone?

The performance is evaluated using the AS, and we set the hyperparameters to the best-performing values from the previous section ($\tau = 0.7$ for the distortion filter, 1 for the z-score, and GreBERTa and LaBERTa for Greek and Latin, respectively). The summary of the tested experimental variants of the pipeline is available in Table 5, and the results of the ablation tests are available in Table 6.

Contribution of each component (RQ1)

Removing the lemma-matching component, the first in the pipeline, results in a non-significant gain in AS for Latin (+0.49 points, p-value = 0.62) and a non-significant loss for Greek (-0.72, p-value = 0.79). This asymmetry confirms that lemma-matching plays a slightly different role across languages: for Latin, where the embedding model has been pre-trained on richer data, lemma filtering introduces marginal noise; for Greek, it provides a small but non-significant benefit by pre-filtering exact matches. The most dramatic performance drop (-2.70 in AS, p-value < 0.001 for Latin, -8.21, p-value < 0.001 for Greek) occurs when we

https://doi.org/10.1017/chr.2026.10045 Published online by Cambridge University Press

Table 5. Experimental protocol for the evaluation of the components of the ALMA pipeline.

Model variant	Description
Standard ALMA	**Full pipeline presented in Figure 3.**
RQ1: Ablation of each component	
ALMA -Lemma	The lemma-based matching step is removed.
ALMA -Semantic	The semantic similarity matching step is removed.
ALMA -POS	The part-of-speech matching step is removed.
ALMA -Dependency	The dependency relation matching step is removed.
ALMA - (POS+Dependency)	Both syntactic matching steps (POS and Dependency) are removed.
RQ2: Impact of removal of pruning filters	
ALMA -Uniqueness	The uniqueness constraint applied after the POS and Dependency matching steps is omitted.
ALMA -Proximity	The final proximity-based pruning step is omitted.
RQ3: Performance of stand-alone components	
Lemma only	Only the lemma-based matching step is active.
Semantic only	Only the semantic similarity matching step is active.
POS only	Only the part-of-speech matching step is active.
Dependency only	Only the dependency relation matching step is active.

Table 6. Results of the ablation study.

Model variant	AS-Latin	AS-Greek
Standard ALMA	94.12	95.77
RQ1: Ablation of each component		
ALMA -Lemma	94.61	95.05
ALMA -Semantic	91.42***	87.56***
ALMA -POS	94.14	95.29
ALMA -Dependency	92.57	95.34
ALMA -(POS+Dependency)	92.55	95.31
RQ 2: Pruning filter ablation		
ALMA -Uniqueness	82.55***	85.93***
ALMA -Proximity	94.03	95.36
RQ 3: Individual component analysis		
Lemma only	68.19***	73.58***
Semantic only	86.71***	89.61***
POS only	46.04***	30.49***
Dependency only	66.57***	75.45***

Note: Hyperparameters are set using the value determined in the previous section. Best performances are indicated in red, and asterisks indicate the significance of the *p-value*, with the following ruleset: *** (*p-value* < 0.001), ** (*p-value* < 0.01), and * (*p-value* < 0.05).

remove the semantic component, highlighting its crucial role in ALMA's performance. This is particularly evident in Greek, where the linguistic annotation pipeline is less effective for POS and syntactic dependency analysis. As for the POS/Dependency component, removing both syntactic matchers costs 1.57 points of AS for Latin but only 0.46 points for Greek. This contrast reflects the quality of the underlying morpho-syntactic parser: the Latin in our witnesses has characteristics similar to those on which the tagging model was trained, so grammatical matching provides genuine signal, whereas for Greek the syntactic annotation pipeline is hampered by the lack of diacritics. Neither loss reaches statistical significance. We thus see a trade-off in ALMA's architecture: combining semantic and grammatical information, when the embedding and linguistic annotation pipelines perform well, increases the pipeline's matching capacity. However, when these pipelines are of lower quality, they negatively affect performance, and the particularities of the considered corpus, especially the ratio of differing tokens, should be taken into account when deciding which component to activate.

Contribution of the pruning filters (RQ2)

When we consider the impact of filtering on performance, we see the strong usefulness of the uniqueness filter, which is missing in SimAlign, with a very significant improvement of $+11.57$ (p-value < 0.001) for Latin, and an improvement of $+9.84$ (p-value < 0.001) for Greek: this is the component with the most positive impact on performance. Despite its simplicity, the proposed filter is highly effective in ensuring high-quality matching and mitigating the risk of overmatching arising from analysis of semantic proximity and grammatical role alone. The proximity filter, as observed during hyperparameter evaluation, exhibits small and non-significant gains in both languages: removing the filter yields a marginal loss of -0.09 (p-value = 0.89) for Latin and a slightly larger but not statistically significant loss of -0.41 for Greek (p-value = 0.56).

Performance of each component (RQ3)

Each component considered individually performs significantly worse than the pipeline as a whole, demonstrating the effectiveness of ALMA's well-rounded approach, which leverages the strengths of existing approaches. As expected, the approach using only the semantic component, which is very similar to SimAlign, yields the pipeline closest to ALMA's performance, with decreases of

https://doi.org/10.1017/chr.2026.10045 Published online by Cambridge University Press

Table 7. Mean performance scores on Latin.

	Model	Hyperparameters	AS	Precision	Recall
ALMA	LaBERTa-tuned	Thresh=1, Ratio=0.7	94.12	94.27	94.03
SimAlign	XLM-RoBERTa-tuned		92.34***	92.60	92.17
	XLM-RoBERTa		91.24***	91.21	91.38
	LaBERTa-tuned		90.65***	90.94	90.42
	mBERT		89.57***	89.18	90.07
	LaBERTa		89.26***	89.50	89.07
	mBERT-tuned		88.80***	87.70	90.05
POS			76.28***	72.14	81.19
CollateX		LevDist = False	67.92***	62.48	74.70
		LevDist = True	67.92***	62.48	74.70

Note: The best performance is displayed in red. Asterisks indicate the statistical significance (for AS only) of the improvement in ALMA's performance relative to the considered algorithms. Only the best hyperparameterization is retained for ALMA.

-7.41 (p-value = 0.00) for Latin and -6.16 (p-value = 0.00) for Greek. This result is very consistent with SimAlign's good performance in the state of the art and the strong potential of neural approaches for alignment. It also strengthens our claim that good performance is achievable when working with languages that currently still lack lemmatizers and taggers. Using POS alone for matching yields the lowest performance, with -48.08 (p-value = 0.00) for Latin and -65.28 (p-value = 0.00) for Greek, which is understandable given the potential ambiguity of matching based solely on parts of speech. Using dependencies alone, which is slightly more accurate for comparing the reference sentence and the compared sentence, yields a smaller performance loss, with -27.55 (p-value = 0.00) for Latin and -20.32 (p-value = 0.00) for Greek. For exact matching using the lemma, as in string-matching approaches such as CollateX, performance is much lower than that of the entire pipeline, with losses of -25.93 (p-value = 0.00) for Latin and -22.19 (p-value = 0.00) for Greek. We can therefore see the importance of combining the different approaches available in the state of the art, as each component of the pipeline interacts complementarily and performs much less efficiently on its own.

Final optimized pipeline

Because removing no component produced a statistically significant improvement, we retained the complete pipeline, as the marginal gain from removing the lemma-matching component in Latin (+0.49) does not justify altering the full pipeline. All the results presented in the "Comparison with the state of the art" section are computed using the full pipeline of Figure 3. In practice, we recommend that ALMA users first verify the advertised performance of their linguistic annotator to avoid performance degradation. Regarding the grammatical components, given their negligible impact on the Greek corpus, we retained them for our final evaluation. Similarly, given the negligible impact of the resulting performance drop, we have chosen to keep the proximity filter for the remainder of the article, using a window of 70 percent. Our final selected optimized workflow is thus the same as shown in Figure 3.

Comparison with the state of the art

Global alignment scores

The values of the scores selected to assess the performance of the different algorithms in Latin are presented in Table 7 and in Greek in Table 8. The asterisks in the table indicate the statistical significance of the performance improvement when using ALMA rather than the other algorithms tested.

These two tables show that ALMA significantly outperforms all tested approaches, including the best-performing version of SimAlign. ALMA improves AS by 1.78 (p-value = 0.001) and 1.03 (p-value = 0.03) in Latin and Greek, respectively. We thus see relative gains of 1.93 percent (Latin) and 1.09 percent (Greek) for ALMA compared to SimAlign, using fine-tuned XLM-RoBERTa for both Greek and Latin. The recall–precision trade-off is balanced across all the algorithms considered. Alignment with Greek manuscripts yields slightly higher scores for all the algorithms considered, due to lower token variation in the Greek manuscript dataset. This is particularly striking for CollateX's string-similarity algorithm, which shows a performance increase of more than 24 percent for Greek compared to Latin. Despite this improvement in Greek, CollateX's algorithm, although widely used, performs significantly worse than neural methods. Compared with CollateX, SimAlign (in its best hyperparameterization) achieves 36 percent and 12 percent improvements in AS for Latin and Greek, respectively, whereas ALMA achieves 39 percent and 13 percent improvements. Whether or not the Levenshtein distance is used for strict matching does not affect the results, because morphological or synonymic differences are too significant to be captured by a character-string comparison. The same observation holds for the POS algorithm, which underperforms compared to neural approaches. These results highlight the strength of neural approaches compared to traditional ones for alignment, as they effectively handle differences in word order, morphological variation, and synonymic substitutions encountered in scribal editing. This superior performance is verified even for models that have not been fine-tuned and have been used off the shelf, including non-standard Greek without diacritics. For ALMA, performance increases in all cases when fine-tuning is applied. The best models for SimAlign are not the specialized language

https://doi.org/10.1017/chr.2026.10045 Published online by Cambridge University Press

Table 8. Mean performance scores on Greek (sorted by AS per model class).

	Model	Hyperparameters	AS	Precision	Recall
ALMA	GreBERTa-tuned	Thresh=1, Ratio=0.7	95.77	95.69	95.91
SimAlign	XLM-RoBERTa-tuned		94.74*	94.67	94.88
	GreBERTa		94.62*	94.58	94.73
	GreBERTa-tuned		94.56*	94.46	94.74
	mBERT		93.71***	93.47	94.02
	mBERT-tuned		91.78***	90.48	93.27
	XLM-RoBERTa		91.24**	91.21	91.37
POS			86.95***	84.18	90.11
CollateX	LevDist = False		84.46***	81.15	88.29
	LevDist = True		84.46***	81.15	88.29

Note: Asterisks indicate the statistical significance of the improvement in ALMA's performance relative to the considered algorithms. Best performance is displayed in red. Only the best parameterization is retained for ALMA.

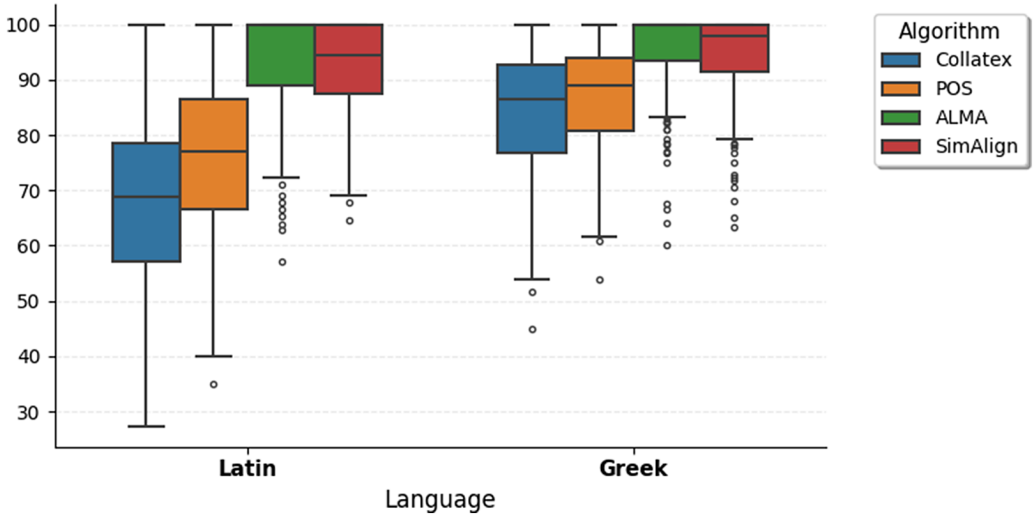


Figure 5. Statistical distribution of the AS at the verse level for the best-performing hyperparameterization of each algorithm.

models, but rather the XLM-RoBERTa model for both languages, specifically tuned on the considered corpus. SimAlign thus exhibits the opposite of ALMA, whose performance is greatly improved by using pre-trained models specific to the aligned languages. ALMA's architecture thus seems better suited to exploiting the semantic knowledge encoded by language-specific pretraining, and the integration of linguistic components appears to reinforce the signal provided by embeddings.

Statistical distribution of AS

The statistical distribution of alignment quality is shown in Figure 5.

This figure shows that median performance leads to the same ranking of algorithms as average comparisons. The lower performance observed for the Latin witnesses is consistent with their greater degree of token variation, with differences between Latin manuscripts leading to a general decline in performance compared to Greek. In particular, we note that ALMA achieves a median performance of 100 percent on both datasets, whereas SimAlign achieves 94.33 percent (Latin) and 97.82 percent (Greek) in its best configurations. This value indicates that ALMA achieves perfect alignment on more than half of the verses considered, unlike its competitors, and improves by 6.01 percent (Latin) and 2.23 percent (Greek) in median over the state of the art in terms of relative improvement. Regarding the variance across the algorithms, as measured by the interquartile ranges, ALMA shows the smallest interquartile range in both datasets. The Latin dataset, with the greatest differences between tokens, yields a higher interquartile range, especially for Dekker's algorithm, whose performance drops approximately to 25 percent for some alignments.

Case studies: SimAlign and ALMA

To illustrate the performance differences between SimAlign and ALMA, we analyze three alignments grouped into two case studies in which ALMA is more efficient. Error analysis of ALMA is provided in the "Error analysis and main methodological limitations" section to highlight some of the limits of our approach and possible research directions for improvement.

Case 1 – Uniqueness filter

One significant difference between SimAlign and ALMA is the application of the uniqueness filter for matches based on word

https://doi.org/10.1017/chr.2026.10045 Published online by Cambridge University Press

Table 9. Token alignments detected by ALMA and SimAlign on case 1 (Latin).

Algorithm	Matches	Unmatches
ALMA	ille ↔ ille, fuit ↔ erat, lucerna ↔ lucerna, ardens ↔ ardens, et ↔ et, lucens ↔ lucens, uos ↔ uos, autem ↔ autem, uoluistis ↔ uoluistis, ad ↔ ad, oram ↔ horam, exultari ↔ exultare, in ↔ in, lumine ↔ luce, eius ↔ eius	- ↔ lucet
SimAlign	ille ↔ ille, fuit ↔ erat, lucerna ↔ lucerna, ardens ↔ ardens, et ↔ et, **lucens ↔ lucet**, lucens ↔ lucens, uos ↔ uos, autem ↔ autem, uoluistis ↔ uoluistis, exultari ↔ exultare, ad ↔ ad, oram ↔ horam, in ↔ in, lumine ↔ lucet, lumine ↔ luce, eius ↔ eius	(none)

similarity available in ALMA. An example of the efficiency of this filter is on the alignment of the readings of John 5:35 in respectively manuscripts Palatinus and Corbeiensis Secundus:

1. *ille fuit lucerna ardens et lucens uos autem uoluistis exultari ad oram in lumine eius.*
[He was a lamp burning and shining, but you wished to be exulted for an hour in his light.]
2. *ille erat lucerna ardens et lucet lucens uos autem uoluistis ad horam exultare in luce eius.*
[He was a lamp burning and he shines shining, but you wished for an hour to exult in his light.]

This alignment presents three significant difficulties: (1) the variation in tense between the perfect (*fuit*) and the imperfect (*erat*); (2) the transformation between the infinitive passive "to be exulted" (*exultari*) and the infinitive active "to exult" (*exultare*); and (3) the reinforcement of *lucens* by the addition of *lucet*. The results given by ALMA and SimAlign are shown in Table 9.

We can thus see the ability of both algorithms to handle the first two difficulties, making them superior to approaches based on sequence similarity: the algorithms were able to handle the grammatical transformations from perfect/imperfect and between active/passive. However, SimAlign, which is based solely on semantic proximity, overmatches *lucens* with *lucet*, which come from the same semantic field. ALMA avoids this pitfall by applying a uniqueness filter, which selects the word with the most semantic correspondence: even though *lucet* and *lucens* have the same lemma, the exact match will always return a more substantial similarity in terms of embedding and will therefore be chosen by ALMA.

Another case study showing similar behavior is the alignment of two manuscript readings of the verse of John 1:13, respectively, in codices Bezae and Vaticanus:

1. οὐκ ἐξ ἐμάτων οὐδὲ ἐκ θελήματος σάρκος οὐδὲ θελήματος ἀνδρός ἀλλ ἐκ θεοῦ ἐγεννήθησαν[5]
Not from blood, nor from the will of the flesh, nor from the will of man, but from God they were born.
2. οἱ οὐκ ἐξ αἱμάτων οὐδὲ ἐκ θελήματος σάρκος ἀλλ ἐκ θεοῦ ἐγενήθησαν
Those that were not born from blood, nor from the will of the flesh, but from God.

These two verses present the following difficulties: (1) the addition of the subject οἱ (they) in the Vaticanus; (2) the addition of οὐδὲ θελήματος ἀνδρός (nor from the will of man) in the Bezae, near to οὐδὲ ἐκ θελήματος σάρκος (nor of the will of the flesh), presenting the challenge for the algorithms of not overmatching by matching together the two identical οὐδὲ and θελήματος; (3) the spelling variant between ἐγεννήθησαν and ἐγενήθησαν, presenting a single *ν* in the Vaticanus; and (4) the spelling variant between ἐμάτων and αἱμάτων. The results of the two matches are in Table 10.

While both algorithms successfully handled the addition of the subject οι, the orthographic variation of ἐγεννήθησαν and of αἱμάτων, the SimAlign algorithm performs an overmatch by considering the doublet οὐδὲ θελήματος as a match with the other οὐδὲ θελήματος, as well as matching αἱμάτων with θελήματος. ALMA performs the alignment without error: the additional expression οὐδὲ θελήματος has a weaker semantic proximity to the other οὐδὲ θελήματος and is therefore filtered by ALMA's uniqueness filter, while SimAlign once again performs an overmatch.

Case 2 – POS matcher

Another feature of ALMA compared to SimAlign is the grammatical filter, which enables matching tokens with the same POS and grammatical functions, even if they have different lemmas, provided their semantic similarity exceeds the predefined threshold. SimAlign lacks this feature and relies solely on contextual proximity to determine matches. An example of such a phenomenon can be found in the alignment of the two verses from John 9:5 (Palatinus *vs.* Corbeiensis Secundus):

1. *dum sum in seculo lux sum seculi*
While I am in the world, I am the light of the world.
2. *cum in hoc mundo sum lux sum huius mundi*
While I am in this world, I am the light of this world.

This alignment presents three difficulties: (1) the variant between *dum* and *cum*, which must be matched together; (2) the use of synonyms for *seculum* and *mundus*; and (3) the addition of the demonstratives *hoc* and *huius*, which should be left unmatched as additions.[6] The results of the alignments are available in Table 11.

We observe an overmatch in SimAlign when mapping demonstratives to their referents, as *seculo*↔*hoc* and *seculi*↔*huius* match together. At the same time, ALMA presents the expected perfect match, thanks to part-of-speech matching: Since *hoc* and *huius* have no common lemma between the target sentence and the source sentence, and have no token with the Part-of-Speech DET in the source sentence, they remain unaligned, which is the expected behavior.

[5]The diacritics were added by us for easier reading of the paper and, as mentioned above, are absent from the original transcriptions.

[6]Note that in the case of translation, for which SimAlign is designed, it may make sense to match words and their demonstratives together, but in the case of collation, which is our field of application, the addition of a demonstrative must be considered an addition, and therefore a non-match.

https://doi.org/10.1017/chr.2026.10045 Published online by Cambridge University Press

Table 10. Token alignments detected by ALMA and SimAlign on case 1 (Greek).

Algorithm	Matches	Unmatches
ALMA	ουκ ↔ ουκ, εξ ↔ εξ, εματων ↔ αιματων, ουδε ↔ ουδε, εκ ↔ εκ, θεληματος ↔ θεληματος, σαρκος ↔ σαρκος, αλλ ↔ αλλ, εκ ↔ εκ, θεου ↔ θεου, εγεννηθησαν ↔ εγενηθησαν	θεληματος ↔ -, ανδρος ↔ -, ουδε ↔ -, - ↔ οι
SimAlign	ουκ ↔ ουκ, εξ ↔ εξ, εματων ↔ αιματων, ουδε ↔ ουδε, εκ ↔ εκ, θεληματος ↔ θεληματος, σαρκος ↔ σαρκος, ουδε ↔ ουδε, θεληματος ↔ αιματων, θεληματος ↔ θεληματος, αλλ ↔ αλλ, εκ ↔ εκ, θεου ↔ θεου, εγεννηθησαν ↔ εγενηθησαν	ανδρος ↔ -, - ↔ οι

Table 11. Token alignments detected by ALMA and SimAlign on case 2.

Algorithm	Matches	Unmatches
ALMA	dum ↔ cum, sum ↔ sum, in ↔ in, seculo ↔ mundo, lux ↔ lux, sum ↔ sum, seculi ↔ mundi	- ↔ hoc, - ↔ huius
SimAlign	dum ↔ cum, sum ↔ sum, in ↔ in, **seculo ↔ hoc**, seculo ↔ mundo, lux ↔ lux, sum ↔ sum, **seculi ↔ huius**, seculi ↔ mundi	(none)

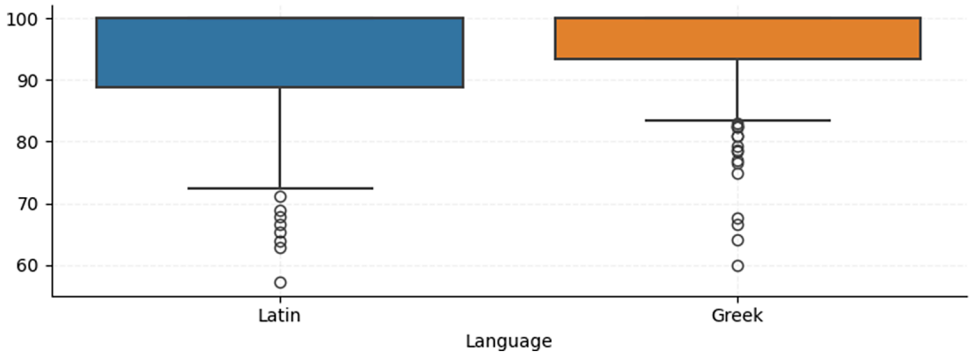


Figure 6. Statistical distribution of AS for ALMA with the best-performing configuration.

Error analysis and main methodological limitations

Error analysis

Quantitative analysis of errors

Figure 6 shows the statistical distribution of errors made by ALMA for all aligned verses. We can see the strong results of 100 percent median success, with an interquartile range of $[Q_1{=}88.88, Q_3{=}100]$ for Latin and $[Q_1{=}93.33, Q3{=}100]$ for Greek, showing ALMA's good overall performance on the datasets considered, with more than half of the verses perfectly aligned in both languages considered in this study. The performance gap between Greek and Latin can be explained, as with the median scores, by the greater freedom in token choice in the Latin corpus compared to the Greek one.

Despite these good performances, this figure highlights several verses with degraded performance, as we observe 84 at or below the first-quartile threshold of $Q_1 = 88.88$ for Latin, and 75 at or below the quartile $Q_1 = 93.33$ for Greek. The statistical characteristics of these lower-performing subsets are presented in Table 12, and practical cases from among them are qualitatively analyzed in the "Qualitative analysis" section.

The study of the characteristics of outliers shows that the lower-scoring subset contains a greater average number of differing tokens. This pattern suggests an association between token variation and lower alignment performance: outliers thus have a higher value of different tokens between the reference verse and the compared verse than in the data considered as a whole. Figure 7 illustrates this trend, showing a decrease in AS as the number of different tokens between the reference verse and the compared verse increases. This trend is particularly evident for Latin, where we see a very sharp decline in AS for verses where the number of different tokens between the reference and the target exceeds 9.

However, as shown in Table 12, the difference in length between the reference and the target has little impact on the AS, indicating that ALMA can effectively capture additions that lead to length differences thanks to its uniqueness filter. The adverse effects of a high rate of differing tokens are therefore not related to additions, but rather to word substitutions.

Qualitative analysis

We conclude this analysis of errors by studying two case studies, presenting typical examples of errors observed in ALMA.

Case 1 – Syntactic variants: One of ALMA's common errors involves not aligning several words together, due to an overly strict uniqueness pruning filter that does not account for changes in syntactic structure. This is the case, for example, when aligning John 15:18 in Latin (Corbeiensis Secundus *vs.* Palatinus):

1. *si seculum uos odis scitote quoniam me primo odiit*
If the world hates you, know it hated me first [scribal error, odis translated as odit].
2. *si mundus uos odit scitote quia me priorem odio habuit*
If the world hates you, know it had hate for me first.

https://doi.org/10.1017/chr.2026.10045 Published online by Cambridge University Press

Table 12. Characteristics of lower-quartile subset (at or below Q_1) for Latin and Greek and of the complete datasets.

	Lower-scoring subset (Lat)	Full dataset (Lat)	Lower-scoring subset (Gr)	Full dataset (Gr)
Total number	84	285	75	285
Average AS	81.27	94.12	85.29	95.77
Reference verse length (characters)	93.19	92.31	98.56	95.93
Difference in verses length (characters)	−1.74	−2.92	1.47	1.15
Number of differing tokens	5.23	3.79	2.29	1.75

Table 13. Token alignments detected by ALMA.

Algorithm	Matches	Unmatches
ALMA	si ↔ si, seculum ↔ mundus, uos ↔ uos, odis ↔ odit, scitote ↔ scitote, quoniam ↔ quia, me ↔ me, primo ↔ priorem, odiit ↔ habuit	– ↔ *odio*

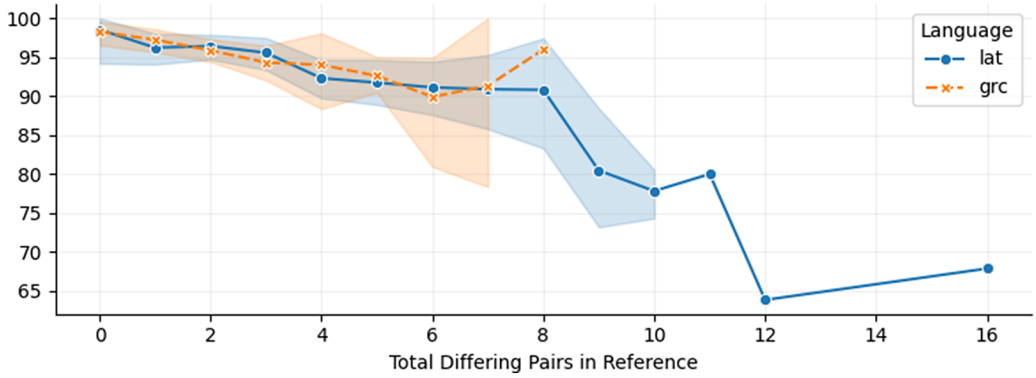


Figure 7. Evolution of the AS for ALMA with the best-performing configuration as a function of differing token count.

The challenge of this alignment is to (1) detect the synonymic change between *seculum* and *mundus* and (2) detect the syntactic structure change between a simple verbal construction (*to hate me*) and a periphrastic expression (*to have hate for me*), which requires matching the overall semantic unit rather than a word-for-word alignment. The result of the matching by ALMA is presented in Table 13.

We see that ALMA correctly detects the synonymous change between *seculum* and *mundus*, but fails to detect the variation in syntactic construction between *odiit* and *odio habuit*, as ALMA is unable to detect the grammatical differences between the two constructions. The model's token-level embedding approach struggles to recognize these syntactically distinct yet semantically equivalent structures, and the uniqueness filter prunes them too strictly. SimAlign, however, with its more lenient semantic-similarity requirements due to its more iterative matching using the *itermax* algorithm, produces the correct matching.

This phenomenon, which is common in the Latin dataset (another example can be the construction of *potuit cognoscere* [could have known]/*sciuit* [knew]), explains several of the errors encountered during the evaluation process, and further work on ALMA will aim to improve this aspect of the algorithm.

Case 2 – Complex synonymy substitutions: Another problematic case, again involving the non-detection of synonymous multi-token groups, is found in the alignment in the Greek text of John 4:33 in Codices Bezae and Vaticanus:

> **(1)** ἔλεγον δὲ ἐν ἑαυτοῖς οἱ *μαθηταί μή τις ἤνεγκε αὐτῷ φαγεῖν*
> The disciples were asking among themselves: has someone brought him to eat?
> **(2)** ἔλεγον οὖν οἱ *μαθηταὶ πρὸς ἀλλήλους μή τις ἤνεγκεν αὐτῷ φαγεῖν*
> The disciples were asking between themselves: has someone brought him to eat?

The difficulty in matching this phrase is the synonym used to render the expression "among themselves," with Bezae employing the construction ἐν ἑαυτοῖς and Vaticanus the construction *πρὸς ἀλλήλους*, which should be matched together. The results of ALMA are displayed in Table 14.

We see from these results that ALMA fails to align "ἐν" with "*πρός*" because the model relies primarily on direct lexical distributional similarities, without sufficient ability to recognize semantic equivalence between distinct prepositions in contextually equivalent constructions. Even though they share the same POS, the similarity filter prunes them too aggressively, leading to a missed match: the chosen approach, therefore, fails to detect the same discursive function of two different prepositions, which, depending on the context, may have synonymous or antonymous

https://doi.org/10.1017/chr.2026.10045 Published online by Cambridge University Press

Table 14. Token alignments detected by ALMA.

Algorithm	Matches	Unmatches
ALMA	ελεγον ↔ ελεγον, δε ↔ ουν, οι ↔ οι, μαθηται ↔ μαθηται, εαυτοις ↔ αλληλους, μη ↔ μη, τις ↔ τις, ηνεγκε ↔ ηνεγκεν, αυτω ↔ αυτω, φαγειν ↔ φαγειν	εν ↔ –, – ↔ προς

meanings ("ἓν" and "πρός"). A possible approach to mitigating this issue is to train the embedding models on aligned parallel corpora.

Current methodological limits of our approach

Dependence on neural model

One of the most substantial limitations of our existing approach is its dependence on the quality of neural models used for both morphological parsing and embedding. In our use case, the parsing models were either already available, as with Latin, or easily fine-tunable, as with Greek without diacritics. The same can be said about the embedding method, which, in our case, was already trained on the target languages and could be easily fine-tuned for our particular manuscripts. In languages with fewer out-of-the-box language models, such as Syriac or Coptic, ALMA requires training an embedding model from scratch, which is time-consuming and resource-intensive, as well as a morphological parser, which requires tagged data. In this case, string-matching-based algorithms, such as CollateX, possess a decisive advantage, as they require no prior training phase, and are for now the most efficient solution for automatic collation in languages without any NLP toolkit.

Handling of multi-word alignment

Although ALMA is capable of handling multi-word alignment during the semantic matching stage in the same way as SimAlign, this is restricted during the rest of the matching process by the match uniqueness component, which is necessary to avoid overmatching tokens with the same grammar. While our ablation study in the "Hyperparameter tuning and pipeline optimization using ablation study" section effectively demonstrated that this filter increased performance in general, our qualitative case study in the "Qualitative analysis" section has, however, highlighted cases where the poor ability to detect substitutions of equivalent phrases with different syntax led to performance loss. Further work to improve ALMA's resilience in these cases will explore the possibility of including constituency analysis using context-free grammars to identify when multiple tokens in the reference and the target correspond to semantically equivalent phrases.

Resource requirements

Another limitation of the ALMA algorithm that we can highlight is its higher computational resource requirements, both because of the fine-tuning phase on the input dataset, and when executing the algorithm itself, due to the six different layers that the input texts must pass through before obtaining an alignment, some layers being particularly resource-intensive, such as parsing all the words available in the input verses. ALMA's strength in combining the advantages of existing alignment algorithms, however, automatically entails additional costs in terms of resources and execution time.

Table 15. Execution time per algorithm with the best hyperparameters per algorithm class for aligning 10 tokens.

Algorithm	Average time (ms)
Dekker	0.4
POS	65.9
SimAlign	82.3
ALMA	215.7

Table 15 compares the average time of each algorithm for one alignment of 10 Latin tokens. The computations were averaged over 100 trial runs.

We can therefore see in this table the substantial increase in execution time when using ALMA, which combines several approaches, rather than a single approach: ALMA results in approximately 539 times the execution time of CollateX's implementation of Dekker's algorithm. However, this significant increase in execution time must be put into perspective in relation to its application. With 215.7 ms of alignment time required, it takes 1 minute 45 seconds and 1 minute 49 seconds, respectively, to align all the Latin and Greek tokens we considered in this study. We thus consider this a reasonable waiting time, given how long the alignment process can take when performed by hand and how faster algorithms like CollateX lead to poorer performance.

Even when putting in perspective the higher computational costs with the more modest gains compared to the SimAlign approach, as the absolute improvement over the best SimAlign configuration remains moderate (+1.78 AS for Latin and +1.03 for Greek), the additional time resource requirements of ALMA are justified for use-cases where alignment quality is paramount, as in editorial work, as it reaches a median AS of 100 percent on both datasets. For such precise and time-consuming tasks as witness alignment, even marginal gains in performance can save the philologist days of post-correction work. ALMA is thus best understood as a tool targeting fine-grained philological accuracy rather than large-scale, speed-critical alignment that can tolerate a noisier output.

Evaluation dataset

Another limitation of this study is the dataset selected for evaluation. While this dataset is highly relevant to philology and the study of ancient texts, as the New Testament is the ancient text that has been most widely transmitted over the centuries, with nearly 6,000 Greek manuscripts and 10,000 Latin manuscripts, it is also a relatively stable text because of its religious significance for the scribes who copied it: we see in Table 1 that for Latin, 25 percent of tokens are non-identity matches, and for Greek, 10 percent. Additionally, the versification of the biblical text eases the alignment task by providing built-in pre-alignment. Texts with-

https://doi.org/10.1017/chr.2026.10045 Published online by Cambridge University Press

out verses would benefit from a rough verse-level alignment phase (such as the approach developed in Bizzoni and Reboul 2015, or more recently using a *large language model* in Reboul 2024) before the more computationally expensive, fine-grained ALMA alignment.

It is therefore among the planned follow-ups of our work to evaluate ALMA on a "messier" tradition, such as secular works aligned at the chapter level, which have a higher number of changed tokens and require an extra pre-alignment step. This follow-up study on other traditions will enable us to analyze the AS as a function of varying token rates.

Conclusion and perspectives

In conclusion, this article presents the new ALMA pipeline (Alignment and Learning for Manuscript Analysis), which enables alignment, the first step in manuscript collation. ALMA achieves mean alignment scores of 94.12 percent (Latin) and 95.77 percent (Greek), significantly outperforming all baseline methods: alignment scores are improved over a deep learning-only approach by 1.93 percent and 1.09 percent, over a grammar-based approach by 23.39 percent and 10.14 percent, and over a sequence alignment method by 38.57 percent and 13.39 percent, respectively, for Latin and Greek. It is notably the only tested algorithm that achieves a median alignment score of 100 percent on both datasets. These results highlight ALMA's substantial potential for advancing computational approaches for textual collation and more generally for computational philology. Even more generally, we hope this work has demonstrated the importance of leveraging recent advances in neural architectures to improve current approaches to computational philology.

Our study has also opened several new research perspectives. Our first axis of research will be to perform the same experiment on multilingual data by matching on an aligned corpus of Greek and Latin New Testament manuscripts, as one of ALMA's major strength is its lack of adherence to a specific language. We will also test the impact on performance of using a language without pre-existing language models, such as the New Testament in its Syriac version, and assess the effect of low-resource availability on the SimAlign and ALMA algorithms. Finally, we will examine how ALMA behaves in relation to more fluid textual traditions than the religious text of the Gospel, to quantify the impact of differing token rates on the performance of each of the algorithms studied.

Data availability statement. All manually tagged data used to perform the alignment experiments is fully open source and available on the article's companion GitHub repository.

Author contributions. Conceptualization: S.R.-H. and D.D.; Data curation: S.R.-H. and D.D.; Formal analysis: S.R.-H.; Funding acquisition: S.R.-H. and F.R.; Investigation: S.R.-H.; Methodology: S.R.-H. and A.H.; Project administration: F.R.; Resources: S.R.-H.; Software: S.R.-H. and A.H.; Supervision: S.R.-H. and F.R.; Validation: S.R.-H.; Visualization: S.R.-H.; Writing – original draft: S.R.-H. and A.H.; Writing – review and editing: S.R.-H. and D.D.

Funding statement. This research was funded by the ANR SHERBET (ANR-23-CE38-0006) Project.

Competing interests. The authors declare none.

Ethical standard. The authors affirm that this research did not involve human participants.

References

Andrade, Chittaranjan. 2019. "The P Value and Statistical Significance: Misunderstandings, Explanations, Challenges, and Alternatives." *Indian Journal of Psychological Medicine* 41, no. 3: 210–15.

Bizzoni, Yuri, and Marianne Reboul. 2015. "Contextual Distribution for Textual Alignment." *World Academy of Science, Engineering and Technology International Journal of Cognitive and Language Sciences*, 2068–2072. Paris. https://hal.science/hal-03167990.

Brown, Peter F., Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. "The Mathematics of Statistical Machine Translation: Parameter Estimation." *Computational Linguistics* 19, no. 2: 263–311.

Buchanan, Edgar Simmons. 1907. *The Four Gospels from the Codex Corbeiensis (ff²), Together with Fragments of the Catholic Epistles, of the Acts and of the Apocalypse from the Fleury Palimpsest (H)*. Oxford: Clarendon Press. Codex Lat. 17225 and Lat. 6400 G, Bibliothèque nationale de France.

Burns, Patrick J. 2023 "LatinCy: Synthetic trained pipelines for Latin NLP." Preprint.

Burton, Philip. 2000. *The Old Latin Gospels: A Study of their Texts and Language*. Oxford: Clarendon Press.

Burton, Philip H., Jon Balserak, Hugh A. G. Houghton, and David C. Parker. 2015. *Vetus Latina Iohannes: The Old Latin Manuscripts of John's Gospel: The Verbum Project*, 2.0 edition. Birmingham.

Conneau, Alexis, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. "Unsupervised Cross-Lingual Representation Learning at Scale." In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, edited by **Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault**, 8440–51. Association for Computational Linguistics.

Dekker, Ronald, and Gregor Middell. 2011. "Computer-Supported Collation with CollateX: Managing Textual Variance in an Environment with Varying Requirements." In *Supporting Digital Humanities 2011: Answering the Unaskable*, edited by **Bente Maegaard**. Copenhagen: University of Copenhagen.

Dekker, Ronald, and Gregor Middell. 2022. "CollateX—Software for Collating Textual Sources."

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding." In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dou, Zi-Yi, and Graham Neubig. 2021. "Word Alignment by Fine-Tuning Embeddings on Parallel Corpora." In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, edited by **P. Merlo, J. Tiedemann, and R. Tsarfaty**, 2112–28. Association for Computational Linguistics

Froger, Jacques. 1966. "La Collation des Manuscrits à la Machine électronique." *Bulletin d'information de l'Institut de Recherche et d'Histoire des Textes* 13 (1964–1965): 135–71.

Froger, Jacques. 1968. *La Critique des Textes et Son Automatisation*, volume 7 of Initiation Aux nouveautés de la Science. Paris: Dunod.

Gryson, Roger. 1999. *Altlateinische Handschriften = Manuscrits Vieux Latins: Répertoire Descriptif. MSS 1-275*, Number 1/2A in Vetus Latina: Die Reste der Altlateinischen Bibel. Freiburg: Herder.

Haug, Dag T. T., and Marius L. Jøhndal. 2008. "Creating a Parallel Treebank of the Old Indo-European Bible Translations." In *Proceedings of the Second Workshop on Language Technology for Cultural Heritage Data (LaTeCH 2008)*, edited by **C. Sporleder and K. Ribarov**, 27–34. Marrakech, Morocco: European Language Resources Association (ELRA).

Honnibal, Matthew, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. "spaCy: Industrial-Strength Natural Language Processing in Python." Online.

Houghton, Hugh A. G. 2016. *The Latin New Testament: A Guide to its Early History, Texts, and Manuscripts*. Oxford: Oxford University Press.

Institute for New Testament Textual Research. 2024. "New Testament Virtual Manuscript Room (NTVMR)." Transcriptions accessed via the NTVMR,

maintained by the INTF, University of Münster. https://ntvmr.uni-muenster.de/.

Kostkan, Jan, Márton Kardos, Jacob Palle Bliddal Mortensen, and Kristoffer Laigaard Nielbo. 2023. "OdyCy—A General-Purpose NLP Pipeline for Ancient Greek." In *Proceedings of the 7th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, edited by **Stefania Degaetano-Ortlieb, Anna Kazantseva, Nils Reiter, and Stan Szpakowicz**, 128–34. Dubrovnik: Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.latechclfl-1.14.

Landis J., Richard, and Gary G. Koch. 1977. "The Measurement of Observer Agreement for Categorical Data." *Biometrics* 33, no. 1: 159–74.

Levenshtein, Vladimir I. 1966. "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals." *Soviet Physics Doklady* 10, no. 8: 707–10.

Maas, Paul. 1958. *Textual Criticism*. Oxford: Oxford University Press.

Myerston, Jacobo, and Jose López. 2023 "greCy: Ancient Greek Spacy Models for Natural Language Processing in Python."

Nagata, Masaaki, Katsuki Chousa, and Masaaki Nishino. 2020. "A Supervised Word Alignment Method Based on Cross-Language Span Prediction Using Multilingual BERT." In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, edited by **B. Webber, T. Cohn, Y. He, and Y. Liu**, 555–65. Association for Computational Linguistics.

Needleman, Saul B., and Christian D. Wunsch. 1970. "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins." *Journal of Molecular Biology* 48, no. 3: 443–53.

Nury, Elisa, and Elena Spadini. 2020. "From Giant Despair to a New Heaven: The Early Years of Automatic Collation." *Information Technology* 62, no. 2: 61–73.

Och, Franz J., and Hermann Ney. 2000. "Improved Statistical Alignment Models." In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, 440–7. Hong Kong: Association for Computational Linguistics.

Palladino, Chiara, Farnoosh Shamsian, and Tariq Yousef. 2022. "Using Parallel Corpora to Evaluate Translations of Ancient Greek Literary Texts: An Application of Text Alignment for Digital Philology Research." *Journal of Computational Literary Studies* 1, no. 1.

Qi, Peng, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. "Stanza: A Python Natural Language Processing Toolkit for Many Human Languages." In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, edited by **A. Celikyilmaz and T.-H. Wen**, 101–8. Association for Computational Linguistics.

Reboul, Marianne. 2024. "Aligning Greek and Latin Texts to French Translations Using Large Language Models: An Experiment on Classical Philosophical Texts." *Paper presented at "Horizons de la Philologie numérique"*. Naples. https://hal.science/hal-04719660.

Riemenschneider, Frederick, and Anette Frank. 2023. "Exploring Large Language Models for Classical Philology." In **Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki**, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 15181–99, Toronto: Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.acl-long.846.

Robinson, Peter M. W. 1989. "The Collation and Textual Criticism of Icelandic Manuscripts (1): Collation." *Literary and Linguistic Computing* 4, no. 2: 99–105.

Sabet, Masoud Jalili, 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*, edited by **T. Cohn, Y. He, and Y. Liu**, 1627–43. Association for Computational Linguistics.

Spencer, Matthew, and Christopher J. Howe. 2004. "Collating Texts Using Progressive Multiple Alignment." *Computers and the Humanities* 38: 253–70.

Stengel-Eskin, Elias, Tzu-ray Su, Matt Post, and Benjamin Van Durme. 2019. "A Discriminative Neural Model for Cross-Lingual Word Alignment." In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th Int. Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, edited by **K. Inui, J. Jiang, V. Ng, and X. Wan**, 910–20. Hong Kong: Association for Computational Linguistics.

Sultan, Md Arafat, Steven Bethard, and Tamara Sumner. 2014. "Back to Basics for Monolingual Alignment: Exploiting Word Similarity and Contextual Evidence." *Transactions of the Association for Computational Linguistics* 2: 219–30.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. "Attention Is All You Need." In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017)*, edited by **I. Guyon, U. von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett**, 6000–10. Long Beach, CA: Curran Associates, Inc.

Welch, Benjamin L. 1947. "The Generalisation of Student's Problems When Several Different Population Variances Are Involved." *Biometrika* 34, nos. 1–2: 28–35.

Yousef, Tariq, Chiara Palladino, Farnoosh Shamsian, and Maryam Foradi. 2022. "Translation Alignment with Ugarit." *Information* 13, no. 2: 65.

Zenkel, Thomas, Joern Wuebker, and John DeNero. 2020. "End-to-End Neural Word Alignment Outperforms GIZA++." In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, edited by **D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault**, 1605–17. Association for Computational Linguistics.

Appendix

Step by step matching of John 9:5 for two Old Latin codices (*Codex Corbeiensis secundus* and *Codex Rehdigeranus*), for illustrative purposes.

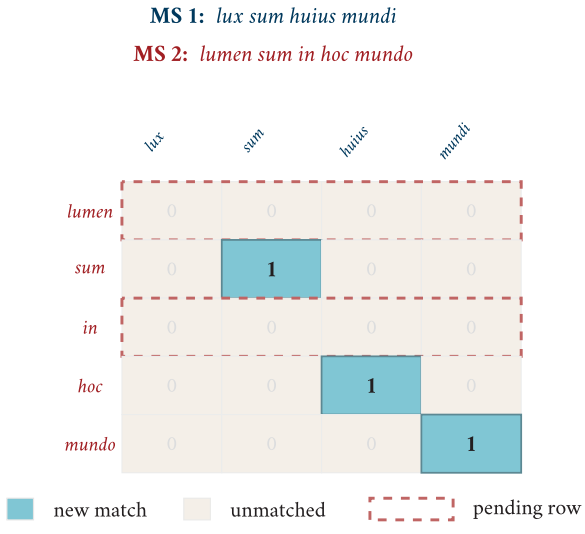


Figure A1. Step 1 – Lemma matching. Three pairs resolved via shared lemma: *sum*↔*sum*, *hoc*↔*huius* (both forms of *hic*), and *mundo*↔*mundi* (both forms of *mundus*). *Lumen* and *in* remain unmatched (dashed).

https://doi.org/10.1017/chr.2026.10045 Published online by Cambridge University Press

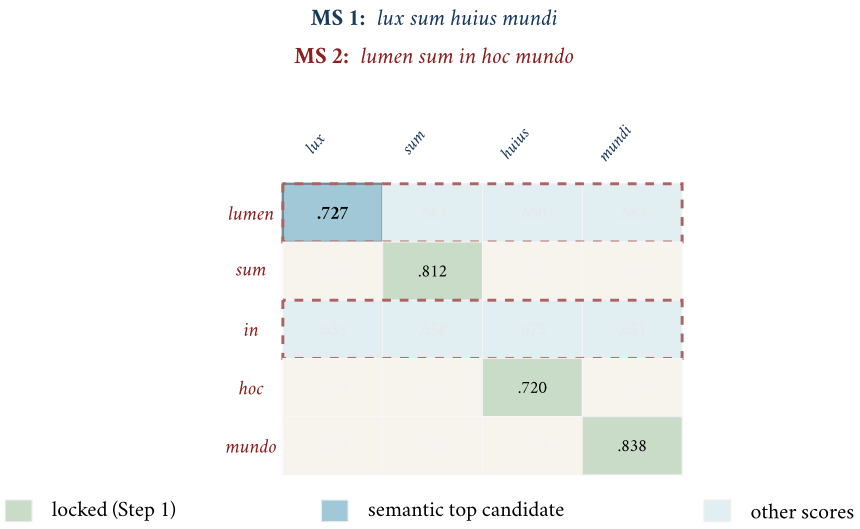


Figure A2. Step 2 – Semantic similarity scores. For *lumen* (row 1), the highest score is 0.727 against *lux* (col. 1, teal), committing the pair *lumen*↔*lux*. For *in* (row 3), no score crosses the disambiguation threshold; *in* remains unmatched. Locked rows (Step 1 matches) are grayed and not acted upon.

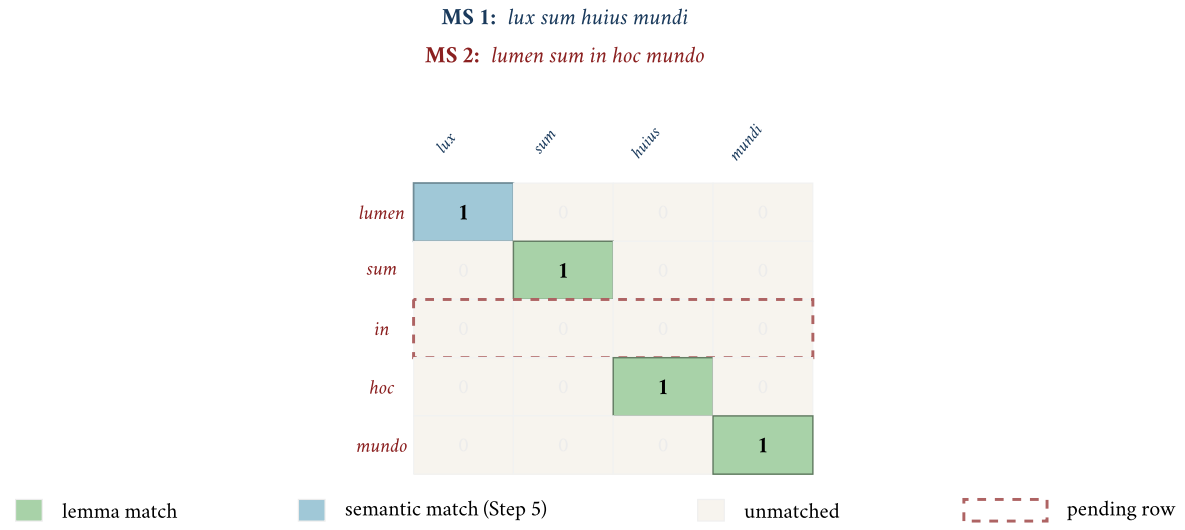


Figure A3. Step 3 – POS matching. No ambiguity to resolv. No new pairs are found. The matrix is unchanged from Step 2. *In* (row 3) remains unmatched.

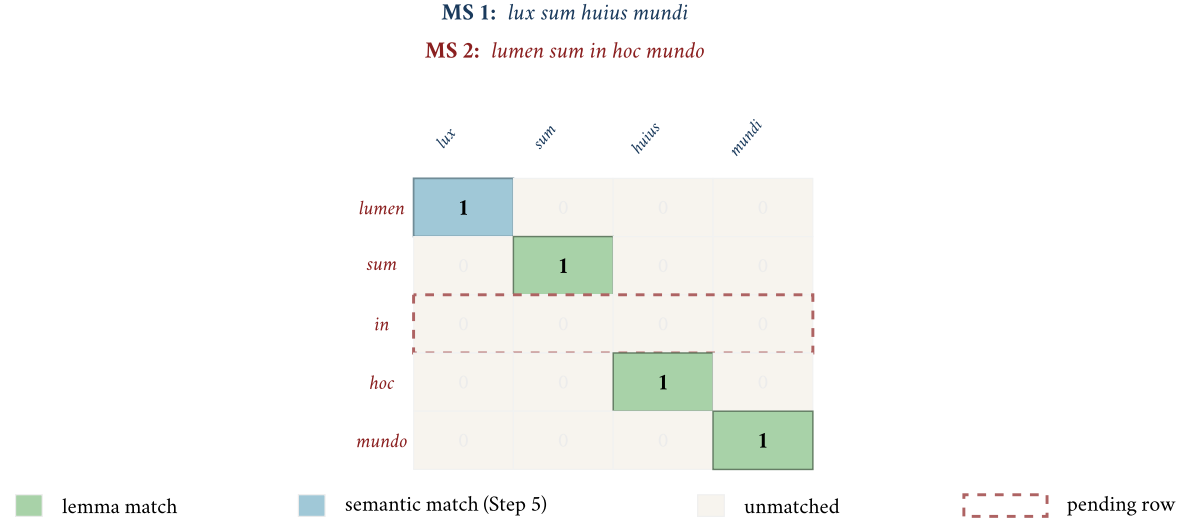


Figure A4. Step 4 – Dependency matching. Syntactic dependency relations find no additional candidates. The matrix is unchanged from Steps 1 and 2. *In* (row 3) remains unmatched.

https://doi.org/10.1017/chr.2026.10045 Published online by Cambridge University Press

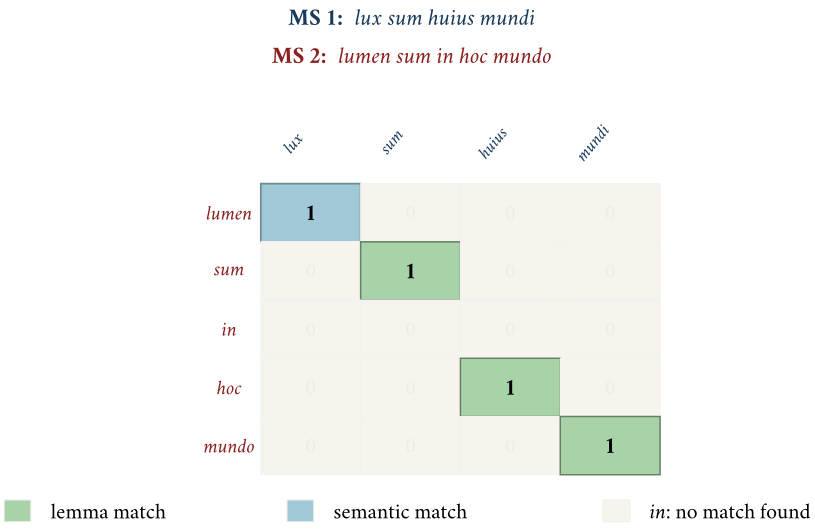


Figure A5. Final matrix $A \in \{0,1\}^{5\times4}$. Four of five token pairs are aligned: *lumen*↔*lux* (semantic), *sum*↔*sum*, *hoc*↔*huius*, and *mundo*↔*mundi* (all lemma). The preposition *in* has no counterpart in manuscript 1 and remains definitively unmatched, reflecting a genuine lexical addition in manuscript 2.

https://doi.org/10.1017/chr.2026.10045 Published online by Cambridge University Press