CorpusClues: Scalable Unsupervised Similarity Search for Historical Texts Using MinHash-LSH

Paulien Lemay[1], Klaas Bentein[1], Els Lefever[2]
[1]Department of Greek Linguistics, [2]LT3 Language and Translation Technology Team, Ghent University
[1]Blandijnberg 2, 9000 Ghent, Belgium, [2]Groot-Brittanniëlaan 45, 9000 Ghent, Belgium
{paulien.lemay, klaas.bentein, els.lefever}@ugent.be

Abstract

CorpusClues is a prototype web-based platform for large-scale, unsupervised clustering of textual data, designed to address the specific challenges of historical corpora. It leverages the well-established computational techniques of MinHash and Locality-Sensitive Hashing (LSH) at the character level in order to detect structural similarities between texts even when exact patterns diverge. This approach makes CorpusClues robust to orthographic variation, such as historical spelling differences, while remaining fast and language-agnostic, capable of processing large and heterogeneous corpora without relying on language-specific models or preprocessing. Researchers can explore resulting clusters through interactive visualizations and exportable data, gaining access to patterns that would otherwise require the slow and uncertain process of manual collation. Evaluation against labeled gold standards shows that the system consistently produces high-quality clustering, accurately reconstructing relationships between texts despite substantial orthographic variation. By combining computational efficiency with user-friendly design, CorpusClues provides an accessible yet rigorous means of uncovering formulaicity and textual transmission at scale, opening new possibilities for the study of historical textual traditions.

Keywords: MinHash-LSH, unsupervised clustering, historical linguistics

1. Introduction

Tracing recurring patterns in texts is a central research topic in historical linguistics. In oral traditions, formulaic expressions are widely recognized as mnemonic and compositional tools (Parry, 1930). These sequences of words are repeatedly employed to support the structure of oral performance. Formulaicity is also evident in written traditions, persisting across genres and eras. Legal charters, letters, and epigrams often employ conventionalized sequences and recurrent phrases that structure narratives, reinforce genre conventions, and guide composition within specific textual forms (Beihammer, 2020; Giannikou et al., 2024; Nachtergaele, 2023). Beyond these structural roles, formulaic expressions also play a role in linguistic change. Like other aspects of language, formulaic language evolves over time, with some expressions emerging, shifting in frequency, or acquiring new functions in response to social and cultural developments (Buerki, 2020).

However, detecting such patterns remains a non-trivial task. Historical corpora are marked by orthographic variation, scribal conventions, dialectal diversity, and textual fragmentation (Deforche et al., 2024). Even short expressions can appear in multiple variant forms, making their underlying similarity difficult to trace (Swaelens et al., 2024). Manual collation of such material is labor intensive, error-prone, and infeasible for large or heterogeneous collections. As corpora grow in size and diversity, the lack of scalable similarity detection methods becomes an increasing challenge for systematically identifying recurring patterns.

CorpusClues[1] addresses this challenge by integrating unsupervised similarity detection algorithms into a user-friendly web application designed for linguists, historians, and philologists. Users simply upload a CSV file with an ID column and a text column, explore automatically detected similarities through an interactive interface, and export selected clusters for further analysis. They do not need to engage in labor-intensive manual annotation or write custom scripts. By automating similarity detection, CorpusClues enables scholars to focus on substantive research questions, such as studying manuscript traditions, tracing linguistic change, or analysing the use of formulaic expressions within literary contexts.

The system employs MinHash combined with Locality-Sensitive Hashing (LSH), a well-established approach for similarity detection (Broder, 1997; Manku et al., 2007; Zamora et al., 2016). Operating at the level of orthographic similarity, this method is computationally efficient, language-agnostic, and does not rely on large annotated datasets. Although it does not capture semantic relationships, MinHash-LSH offers an accessible, low-effort first step for exploratory similarity analysis in large corpora.

[1]https://lrec2026.myaddr.tools/

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 838–847
11–16 May 2026.

2. Related Work

2.1. Orthographic Similarity Search

Early efforts to quantify orthographic similarity primarily relied on character-based distance metrics, which assess the dissimilarity between two strings by counting the minimal operations needed to transform one into the other. Notable algorithms in this category include Levenshtein (Levenshtein, 1966), Damerau-Levenshtein (Damerau, 1964), Jaro (Jaro, 1995), and Jaro-Winkler (Winkler, 1990). These methods support unsupervised similarity assessment and have been widely applied in tasks such as spelling correction, record linkage, and approximate string matching (Navarro, 2001). A key limitation, however, is their computational cost, typically $\mathcal{O}(mn)$ for strings of lengths m and n, which grows rapidly for longer sequences[2]. While indexing strategies and algorithmic optimizations can mitigate some of this burden (Navarro, 2001), the fundamental challenge of scalability persists when comparing large volumes of strings.

Token-based similarity approaches address the scalability problem by representing texts as sets of discrete units, such as words or short sequences (n-grams), and assessing similarity based on the degree of overlap between these token sets. Jaccard Similarity (Jaccard, 1901) is a common measure in this framework, calculated as the size of the intersection of two token sets divided by the size of their union. Although straightforward, computing Jaccard similarity exhaustively for all document pairs in a large corpus requires a quadratic number of comparisons, which quickly becomes impractical.

One solution we can employ to handle this scalability issue is MinHash, a probabilistic method for estimating set similarity. By applying multiple hash functions to token sets, MinHash generates compact signature vectors that preserve the likelihood of overlap, allowing us to estimate similarity efficiently in a way that is inherently language-agnostic (Broder, 1997). When combined with Locality-Sensitive Hashing (LSH) (Gionis et al., 1999), these signatures can be grouped into buckets of likely similar items, further reducing the number of comparisons to only those candidate pairs that share a bucket.

2.2. Computational Similarity Approaches in Historical Linguistics

In historical and comparative linguistics, computational similarity methods are employed for a range of tasks.

In cognate detection, for example, researchers identify word forms across languages that derive from a common ancestral word. Tools like the Automated Similarity Judgment Program (ASJP) (Wichmann et al., 2025) and LexStat (List, 2012) illustrate this approach. ASJP normalizes phonetic transcriptions and uses Levenshtein distance to compare sequences at the character level, enabling systematic comparisons of basic vocabulary across hundreds of languages. LexStat builds on ASJP by statistically evaluating the likelihood of specific sound correspondences, thereby also relying on consistent orthographic and phonetic representations.

Another case is the detection of variant spellings, especially in historical corpora where orthographic conventions depend on time and location. Methods such as weighted edit distance and deep-learning models have been applied to identify such variants (Deforche et al., 2024; Swaelens et al., 2024; Barteld, 2017).

A further example is the study of formulaic expressions and recurring sequences in Byzantine epigrams (Bentein and Giannikou, 2025; Giannikou et al., 2024), where n-gram–based approaches have been used to analyze how scribal habits are reflected in formulaic language over time.

However, these approaches are often limited in scalability or cross-linguistic applicability. Techniques such as MinHash-LSH, developed in large-scale text similarity research, address these issues by approximating similarity efficiently and in a language-independent manner. CorpusClues builds on this idea, making it possible to investigate phenomena such as the distribution of formulaic expressions or spelling variance at scale.

3. System Architecture

The proposed architecture implements two complementary clustering strategies: in-memory processing and streaming disk-based processing, unified under a common interface that automatically selects the appropriate method based on dataset size. Both strategies yield identical clustering outputs, differing only in their computational mode. For datasets exceeding 10 000 documents, the framework employs streaming processing to manage memory constraints, while smaller datasets use in-memory processing for faster execution.

3.1. Preprocessing

Input data are provided as a structured CSV file, where each row represents an individual text unit to be processed. Depending on the research design, these units may correspond to verses, sentences, or full documents. As shown in Figure 1, the system offers users the possibility to apply basic normal-

[2] $\mathcal{O}(mn)$, or *Big O notation*, describes how the computational cost of an algorithm grows as the input size increases.

ization operations such as diacritics removal, lowercasing, and punctuation removal. Providing these options ensures that routine text clean-up steps do not have to be implemented in custom scripts, thereby facilitating a more streamlined workflow. In addition, the system always applies Unicode normalization in the NFC (Normalization Form Canonical Composition) standard. NFC ensures that base characters and their diacritics are combined into a single Unicode code point, so that visually identical characters are treated consistently even if they were originally encoded differently. This prevents artificial discrepancies in similarity calculations and guarantees that text is processed according to a stable, canonical representation. Beyond these operations, the system refrains from imposing or offering further transformations, leaving users free to implement additional preprocessing procedures externally.

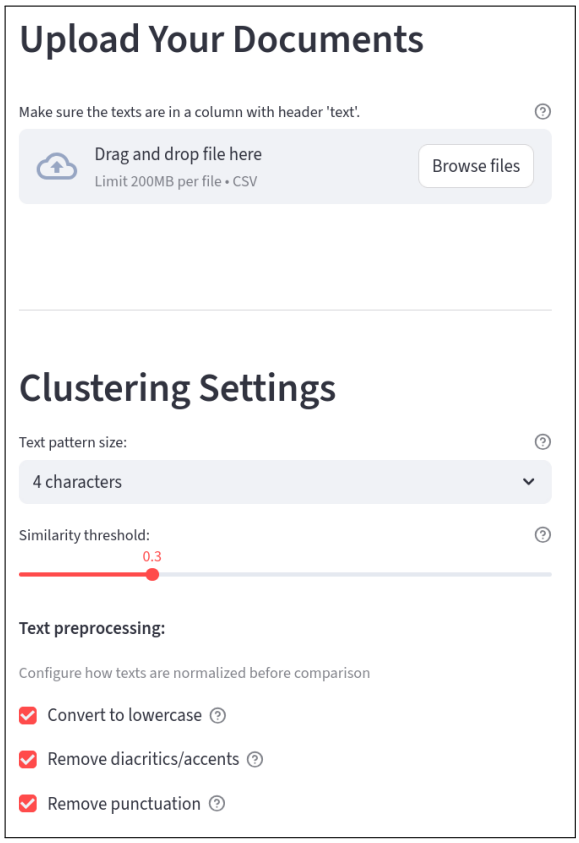

Figure 1: Upload page with preprocessing.

3.2. MinHash Generation

Following preprocessing, the system constructs character shingles, contiguous substrings of fixed length that capture local textual patterns by breaking documents into short overlapping sequences. The shingle size is user-configurable, allowing adaptation to different textual granularities. Both strategies use identical shingle generation logic but differ in their optimization approaches for processing these shingles. The memory-optimized strategy uses adaptive parallelized computation based on batch size to generate shingles rapidly across multiple documents simultaneously. The streaming strategy processes documents sequentially, generating shingles one document at a time to prioritize memory efficiency over computational speed.

Once shingles are generated, each document is represented by a MinHash signature of 64 values, which can be thought of as a compact fingerprint derived from its shingles. Documents with similar content will have similar signatures, allowing fast similarity comparisons without examining every shingle. The memory-optimized strategy maintains all MinHash signatures in memory for rapid cross-document similarity detection. The streaming strategy stores MinHash signatures in a SQLite database for datasets that exceed available memory, trading some performance for scalability

3.3. Pairwise Similarity Computation

Both strategies use LSH to organize documents into buckets, enabling documents from different batches to be grouped together based on signature similarity and limiting Jaccard computation to candidate pairs within shared buckets. The memory-optimized strategy keeps this information in memory, whereas the streaming strategy persists the LSH index to disk, performing LSH queries against this persistent index to identify candidate pairs. Only document pairs exceeding the predefined similarity threshold are considered connected.

3.4. Clustering

Clustering is performed using an optimized disjoint-set approach (Cormen et al., 2022), which merges all connected documents into clusters. These clusters are transitive, meaning that documents can be grouped together not only through direct similarity but also via indirect connections through other similar documents, as illustrated in Figure 2. Each document is then assigned a certainty score representing its cluster membership confidence. This score is calculated as the average Jaccard similarity between the document and all other members of its assigned cluster, providing a quantitative measure of how well the document fits within its cluster.

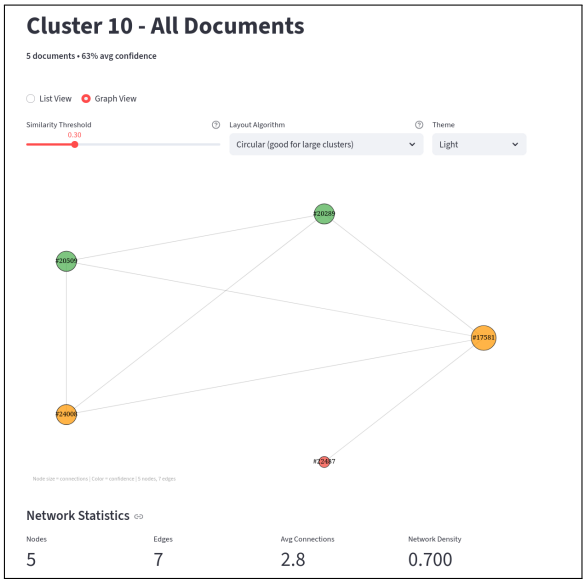

Figure 2: Cluster showing close, moderate, and weak relations. IDs denote uploaded texts.

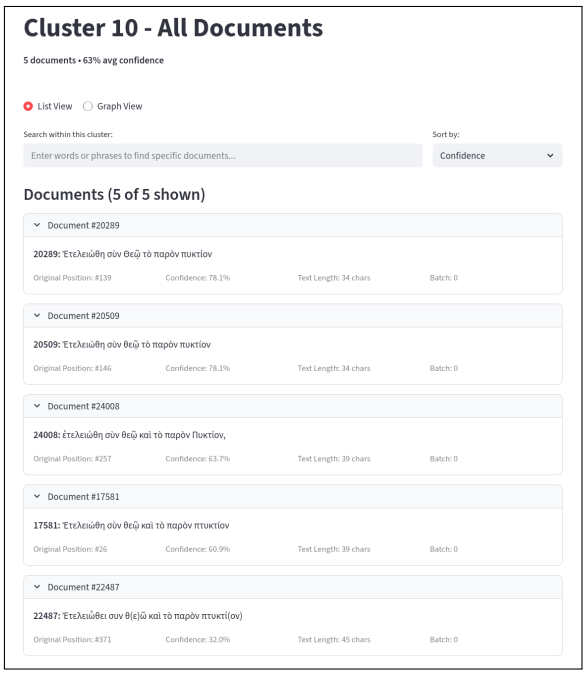

Figure 3: List view of the cluster shown in Figure 2.

4. Gold Standard Evaluation

4.1. Experiment Setup

To evaluate clustering accuracy, we conducted a test using the Database of Byzantine Book Epigrams (DBBE) for a gold-standard case study (Demoen et al., 2023). DBBE is a curated digitized collection of over 12 000 metrical texts from approximately 500 to 1500 AD, typically located at the beginning, end, or in the margins of manuscripts. These epigrams served paratextual functions, including structuring content, praising authors, acknowledging patrons and scribes, and guiding readers. The database is organized into Types, representing normalized epigrams, and Verse Variants, grouping similar individual verses based on orthographic similarity (Ricceri et al., 2023). For this evaluation, we used a 575-epigram gold-standard subset from DBBE (Deforche et al., 2024), originally selected for research on orthographic variation and small enough to enable detailed inspection of clustering results.

Evaluation was performed at the verse level to determine whether the application was capable to reconstruct the Verse Variant groups recorded in DBBE. Verse-level evaluation was chosen as it provides a simple, small unit for assessing algorithm performance. We experimented with different shingle sizes and a range of Jaccard thresholds, testing them against the available ground truth. Although such labels are not available in real-life scenarios, this setup allowed us to estimate the upper bound the application could potentially achieve.

For each configuration, we assessed the performance using the Adjusted Rand Index (ARI) (Hubert and Arabie, 1985) and V-measure (Rosenberg and Hirschberg, 2007). ARI captures pairwise agreement between predicted and true groupings, reflecting how consistently individual verses are assigned to the same clusters as in the ground truth groups. V-measure evaluates clustering structure in terms of homogeneity and completeness. These two measures are complementary: ARI is sensitive to the correctness of individual pairwise assignments, whereas V-measure captures the overall structural alignment of clusters with true groups.

In addition, we calculated two metrics that provide a more intuitive interpretation of clustering quality. Firstly, mean group purity examines how well each ground truth group stayed together. For each true group, we identified which predicted cluster received the most verses from that group, then calculated what percentage ended up in that dominant cluster, averaging across all groups. Values close to 1.0 are ideal. Secondly, the overclustering ratio counts how many predicted clusters the algorithm created compared to the number of ground truth groups. A ratio of 1.0 is ideal. Ratios below 1.0 indicate underclustering, while ratios above 2.0 indicate excessive fragmentation.

To assess the effect of preprocessing on clustering in the case of the DBBE dataset, we conducted experiments both on the original texts and on versions in which diacritics and punctuation were removed and text was lowercased. This demonstrates how researchers can modify preprocessing to influence clustering results, highlighting the flexibility afforded by the system's preprocessing design.

All experiments can be reproduced using the

scripts and guidelines available on https://github.com/PaulienLem/lrec-demo.

4.2. Quantitative Results Original Text

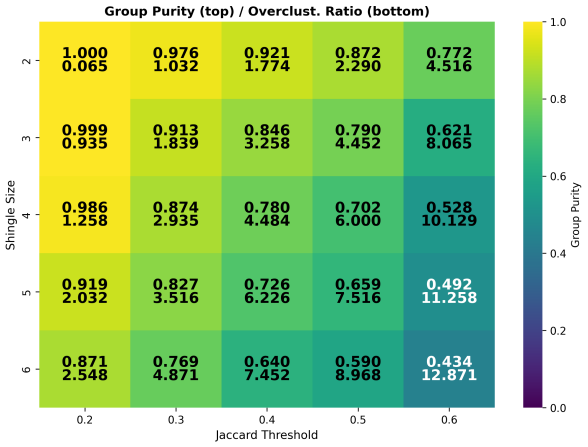

Figure 4: Group purity and overclustering with no preprocessing.

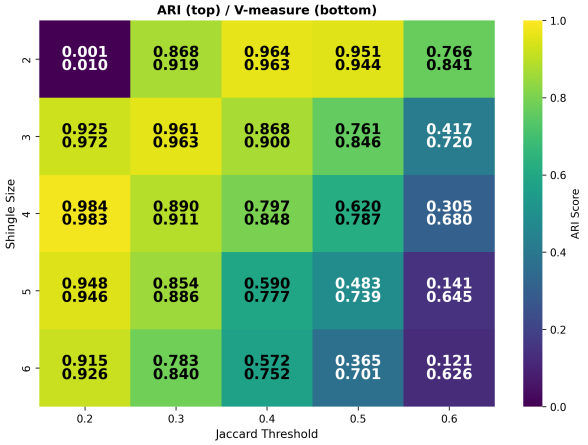

Figure 5: ARI & V-Measure with no preprocessing.

The output of the application on the original text, evaluated against the Verse Variant labels from DBBE, is summarized in Figures 4 and 5, which, respectively, show the clustering structure and the corresponding evaluation metrics.

At the smallest shingle size and the lowest similarity threshold, we noticed underclustering, with the algorithm creating few but large clusters and merging distinct groups together. This is to be expected, since the combination of short character sequences and permissive matching criteria means that texts need to share only a small fraction of their substrings to be considered similar. However, group purity is high, indicating that individual Verse Variant groups were kept intact within these large clusters. The corresponding ARI and V-measure scores are poor for the lowest configuration.

When we increased the shingle size while keeping the similarity threshold at 0.2, all four metrics converged toward their ideal values, indicating balanced clustering, where both the number of clusters and their composition align well with the ground truth. In practical terms, this configuration means that roughly 20% of all 3–4 character substrings must overlap for two verses to be considered similar. This optimal range suggests that, in this case, orthography-based comparison works best at a relatively fine-grained granularity.

For settings with a Jaccard threshold of up to 0.5 and a shingle size of up to 4, the ARI, V-measure, and purity remained reasonably high, indicating that most verses from each true group still clustered together, although some overclustering began to appear. Beyond these settings, all metrics declined noticeably, reflecting substantial fragmentation and reduced cluster quality.

The moderate divergence between metrics in mid-range settings underscores the importance of interpreting results in combination rather than relying on a single measure. It also highlights that evaluation must be guided by the research question, as different applications may place greater weight on structural stability or practical usability.

4.3. Quantitative Results Preprocessed Text

For the second experiment, we applied the application's option to remove accents and punctuation and to apply lowercasing. The effect was first examined in group purity and overclustering ratios (Figure 6). This showed that preprocessing slightly mitigated excessive fragmentation at higher thresholds, while balance was improved at the optimal settings. This enhancement likely arises because diacritic removal reduces minor spelling variations, allowing the algorithm to treat nearly identical verses as the same.

Other than that, we also compared ARI and V-measure between the original and the preprocessed text (Figure 7). Preprocessing led to a modest increase in both scores. The results became more robust overall, with a wider range of parameter settings yielding good performance. However, extreme values still produced suboptimal clustering, highlighting that careful tuning remains necessary.

While preprocessing slightly stabilized clustering in this scenario, its utility is context-dependent: in other corpora or analytical contexts, accents, diacritics and punctuation may encode meaningful distinctions. Consequently, the system retains preprocessing options as configurable, enabling researchers to adapt them according to the specific analytical objectives of their study.

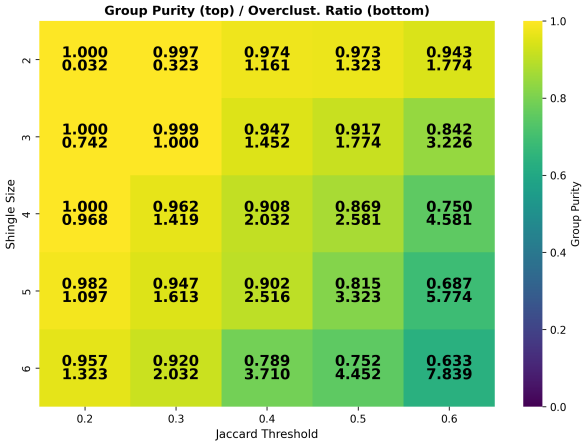

Figure 6: Group purity and overclustering with preprocessing.

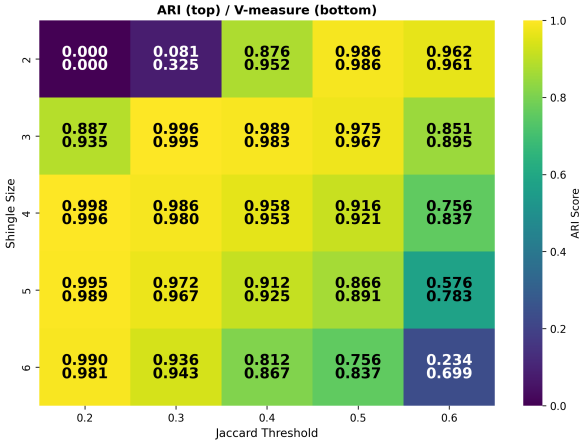

Figure 7: ARI & V-Measure with preprocessing.

4.4. Qualitative Results

To explore the effects of preprocessing on a qualitative level, we ran clustering with a shingle size of 4 and a Jaccard threshold of 0.2, a setting that performed well on both unprocessed and preprocessed texts. By keeping the parameters identical, any differences in the resulting clusters can be attributed solely to the preprocessing choices.

Examining the clustering results for the unprocessed texts in greater detail, we find that out of the thirty one ground truth groups, six were split across multiple predicted clusters, while one predicted cluster merged material from two distinct ground truth groups. To illustrate typical behavior, we focus on true group 14802, which contains variants of the verse καὶ οἱ στρατευόμενοι ἰδεῖν τὸ νίκος ("...and the soldiers to see the victory"), as displayed in Table 1. Most verses of this true group remain clustered together, with only two assigned to separate singleton clusters. Variants in word choice or orthography, such as τὸ κέρδος versus τὸ νῖκος or ἰδεῖν, ειδήν, and ηδὶν, do not disrupt clustering for the majority of verses. By contrast, combinations of both spelling differences and editorial markings trigger separate classification.

16	καὶ	οἱ	στρατευόμενοι	[ἰδεῖν	τὸ	νῖκος]
	kai	hoi	strateuomenoi	[idein	to	nikos]
16	καὶ	οἱ	στρατευόμενοι	ἰδεῖν	τὸ	κέρδος
	kai	hoi	strateuomenoi	idein	to	kerdos
16	και	η	στρατεβόμενοι	ηδὶν	τον	ηκος
	kai	e	stratebomenoi	edin	ton	ekos
16	καὶ	οἱ	στρατευόμενοι	ἰδεῖν	τό	κέρδος,
	kai	hoi	strateuomenoi	idein	to	kerdos,
16	καί	οἱ	στρατευόμενοι	ειδήν	το	νήκος |
	kai	hoi	strateuomenoi	eiden	to	nikos |
16	κ(αὶ)	οἱ	στρατεβομ(εν)οι	ἰδεῖν	τω	νικος·
	k(ai)	hoi	stratebom(en)oi	idein	to	nikos;
16	κὰι	οἱ	στρατεβόμ(εν)οι	ἰδεῖν	τὸν	οἶκον·
	kai	hoi	stratebóm(en)oi	idein	ton	oikon;
16	κ(αὶ)	οἱ	στρατεύομενοι	ἠδειν	(...)	κος
	k(ai)	hoi	strateuomenoi	edein	(...)	kos
16	καὶ	οἱ	στρατεβό|μενοι |	εἰδεῖν	τὸ	νίκος
	kai	hoi	stratebó|menoi |	eidein	to	nikos
16	καὶ	οἱ	στρατευόμενοι	ἰδεῖν	τὸ	νίκος.
	kai	hoi	strateuomenoi	idein	to	nikos.
16	καὶ	οἱ	στρατευόμενοι	ἰδεῖν	τὸ	νίκος,
	kai	hoi	strateuomenoi	idein	to	nikos,
16	καὶ	οἱ	στρατευόμεν(οι) |	ἰδεῖν	τὸ	νῖκος
	kai	hoi	strateuomen(oi) |	idein	to	nikos
16	και	η	στρατεβόμενοι	ηδὶν	τον	ηκος
	kai	e	stratebomenoi	edin	ton	ekos
16	καί	η	στρατεβομενοι	ηδοῦν	τοῦς	ηκοῦς·
	kai	e	stratebomenoi	edoun	tous	ekous;
28	**και**	**οι**	**στρ<α>ττεβόμ(εν)οι**	**ιδιν**	**το**	**ν<ῖκος>,**
	kai	hoi	str<a>ttebóm(en)oi	hidin	to	n<ikos>,
29	**(καὶ)**	**ἡ |**	**στρατευόμ(εν)οι**	**ἠδὶν**	**το**	**νύκος**
	(kai)	e |	strateuóm(en)oi	edin	to	nukos

Table 1: True group 14802, spread across predicted clusters 16, 28 and 29

Turning to the preprocessed text, we observe that the only deviation from the ground truth arises from a large cluster that merges material from two distinct true groups, as shown in Table 2. The two merged groups can be represented by the following verses:

- Δόξα τῷ Θεῷ τῷ διδόντι ἀρχὴν καὶ τέλος. ("Glory to God who gives beginning and end.")
- Δόξα τῷ Θεῷ πάντων ἕνεκεν. ("Glory to God for the sake of all things.")

Although these verses differ substantially in their remaining vocabulary and structure, both begin with the identical phrase δόξα τῷ Θεῷ. Considering their word sets provides an intuitive way to understand the mechanism used by the application. The word-level intersection and union for these verses are:

$$\text{Intersection} = \{\Delta\acute{o}\xi\alpha, \tau\tilde{\omega}, \Theta\epsilon\tilde{\omega}\}$$

$$\text{Union} = \left\{ \begin{matrix} \Delta\acute{o}\xi\alpha, \tau\tilde{\omega}, \Theta\epsilon\tilde{\omega}, \tau\tilde{\omega}, \delta\iota\delta\acute{o}\nu\tau\iota, \\ \grave{\alpha}\rho\chi\grave{\eta}\nu, \kappa\alpha\grave{\iota}, \tau\acute{\epsilon}\lambda\circ\varsigma, \pi\acute{\alpha}\nu\tau\omega\nu, \ddot{\epsilon}\nu\epsilon\kappa\epsilon\nu \end{matrix} \right\}$$

$$J = \frac{|\text{Intersection}|}{|\text{Union}|} = \frac{3}{10} = 0.3$$

The recurring phrase Δόξα τῷ Θεῷ thus constitutes the entire intersection between the two verses. Despite their otherwise distinct vocabulary, this

shared sequence alone yields a Jaccard score of 0.3, sufficient to push the similarity beyond the clustering threshold and merge the two ground truth groups.

δόξα	**τῶ**	**θ(ε)ῶ**	**παντων**	**|**	**ἕνεκεν·**		
doxa	***tō***	***th(e)ō***	***pantōn***	**|**	***heneken·***		
δόξα	**σοι**	**κ(ύρι)ε**	**πάντων**	**ἕνεκεν**	**+**		
doxa	***soi***	***k(yri)e***	***pantōn***	***heneken***	**+**		
+	**δοξα**	**τώ**	**θ(ε)ω**	**ἡμ(ῶν)**	**παντ(ων)**	**ἕνεκεν**	
+	***doxa***	***tō***	***th(e)ō***	***hēm(ōn)***	***pant(ōn)***	***heneken***	
χ(ριστ)ῶ	τῶ	δόντι	ἀρχὴν	καὶ	τέλος,	δόξα	
ch(rist)ō	*tō*	*donti*	*archēn*	*kai*	*telos,*	*doxa*	
+	δόξα	τῶ	θ(ε)ῶ	τῶ	δόντι	τὸ	τέλος·
+	*doxa*	*tō*	*th(e)ō*	*tō*	*donti*	*to*	*telos·*
δόξα	τὸ	δόντι	ἀρχὴν	(καὶ)	τέλο(ς):–		
doxa	*to*	*donti*	*archēn*	*(kai)*	*telo(s)*		
+	τῶ	τὸ	τέλος	δεδωκότι	θεῷ	δόξα	
+	*to*	*to*	*telos*	*dedōkoti*	*theo*	*doxa*	
Δόξα	τῶ	θ(ε)ῶ	τῶ	δόντι	ἀρχὴν	καὶ	τέλος.
Doxa	*to*	*th(e)o*	*to*	*donti*	*archēn*	*kai*	*telos.*
Δόξα	τῶ	θ(ε)ῶ	τῶ	δόντι	τὸ	τέλος:·	
Doxa	*to*	*th(e)o*	*to*	*donti*	*to*	*telos:·*	
Δόξα	τῷ	δόντι	τὴν	ἀρχὴν	καὶ	τὸ	τέλος
Doxa	*tō*	*donti*	*tēn*	*archēn*	*kai*	*to*	*telos*
δόξα	τῶ	δόντι	ἀρχὴν	κ(αὶ)	τέλος·		
doxa	*to*	*donti*	*archēn*	*k(ai)*	*telos·*		
χάριν,	δόξα	τῶ	θεῶ	τῶ	δόντι	τέλος	
charin,	*doxa*	*to*	*theo*	*to*	*donti*	*telos*	
Δόξα	τῶ	θ(ε)ῶ	τῶ	δόντι	ἀρχ(ὴν)	καὶ	τέλος:
Doxa	*to*	*th(e)o*	*to*	*donti*	*arch(ēn)*	*kai*	*telos:*
+	δόξα	τῶ	θ(ε)ῶ	τῶ	δόντι	μοι	τὸ	τέλος.
+	*doxa*	*to*	*th(e)o*	*to*	*donti*	*moi*	*to*	*telos.*
δόξα	τῶ	θ(ε)ῶ	τῶ	δόντι	ἀρχὴν	κ(αὶ)	τέλος
doxa	*to*	*th(e)o*	*to*	*donti*	*archēn*	*k(ai)*	*telos*
Τῶ	τὸ	τέλος	δεδωκότι	θεῶ	δόξα.		
To	*to*	*telos*	*dedōkoti*	*theo*	*doxa.*		
δόξα	τῶ	λόγω	τῶ	δόντι	τέλος·	ἀμήν	
doxa	*to*	*logō*	*to*	*donti*	*telos·*	*amēn*	
Δόξα	τῶ	θ(ε)ῶ	τῶ	δἴδόντι	τὸ	τέλος:·	
Doxa	*to*	*th(e)o*	*to*	*didonti*	*to*	*telos:·*	
τῶ	θ(ε)ῶ	δόξα.	τῶ	δόντι	ἀρχὴν	(καὶ)	τέλος
to	*th(e)o*	*doxa.*	*to*	*donti*	*archēn*	*(kai)*	*telos*
Δόξα	Θ(ε)ῶ	τῶ	δόντι	τῇ	βίβλω,	τέλος	+.
Doxa	*Th(e)o*	*to*	*donti*	*tē*	*biblō,*	*telos*	+.
+	δόξα	τῶ	θεῶ	τῶ	δόντι	τέλος:-	
+	*doxa*	*to*	*theo*	*to*	*donti*	*telos:-*	
+	δόξα	τῶ	θεῶ	τῶ	δόντι	τέλος:-	
+	*doxa*	*to*	*theo*	*to*	*donti*	*telos:-*	
δόξα	τῷ	θ(ε)ῷ	τῷ	δόντι	ἀρχὴν	καὶ	τέλος	ἀμὴν:·
doxa	*tō*	*th(e)ō*	*tō*	*donti*	*archēn*	*kai*	*telos*	*amēn:·*
Δόξα	τῷ	Θεῷ	τῷ	δόντι	ἀρχὴν	καὶ	τέλος.	
Doxa	*tō*	*Theo*	*tō*	*donti*	*archēn*	*kai*	*telos.*	
Δόξα	τῶ	θ(ε)ῶ	τῶ	δόντι	τέλος	γράφος		
Doxa	*to*	*th(e)o*	*to*	*donti*	*telos*	*graphos*		
Δόξα	τῷ	Θεῷ	τῷ	δόντι	τὸ	τέλος		
Doxa	*tō*	*Theo*	*tō*	*donti*	*to*	*telos*		
+	θ(ε)ῶ	τω	δεδωκότι	τέλος,	δόξα:-			
+	*th(e)o*	*to*	*dedōkoti*	*telos,*	*doxa:-*			
+	δόξα	τῷ	θ(ε)ῷ	τῷ	δόντι	τέλος,	ἀμήν:+	
+	*doxa*	*tō*	*th(e)ō*	*tō*	*donti*	*telos,*	*amēn:+*	
+	δόξα	Χ(ριστ)ῶ	τῶ	Θ(ε)ῶ	τῶ	δόντι	ἀρχὴν	καὶ	τέλος
+	*doxa*	*Ch(rist)ō*	*to*	*Th(e)o*	*to*	*donti*	*archēn*	*kai*	*telos*
δόξα	τῶ	θεῶ	τῶ	διδόντι	τὸ	τέλος			
doxa	*to*	*theo*	*to*	*didonti*	*to*	*telos*			
δόξα	Χ(ριστ)ῷ	τῷ	Θ(ε)ῶ	τῶ	δόντι	ἀρχὴν	καὶ	τέλος	
doxa	*Ch(rist)ō*	*tō*	*Th(e)o*	*to*	*donti*	*archēn*	*kai*	*telos*	
δόξα	τω	θ[ε]ῶ	δόντι	καὶ	ἀρχὴν	(καὶ)	τέλος˙		
doxa	*to*	*th[e]o*	*donti*	*kai*	*archēn*	*(kai)*	*telos˙*		
δόξα	τῶ	δόντι	ἀρχ(ὴν)	κ(αὶ)	τέλο(ς)·				
doxa	*to*	*donti*	*arch(ēn)*	*k(ai)*	*telo(s)·*				
+	δόξα	τῶ	δόντ(ι)	ἀρχὴν	(καὶ)	τέλο(ς)·			
+	*doxa*	*to*	*dont(i)*	*archēn*	*(kai)*	*telo(s)·*			
δόξα	τῶ	δόντι	ἀρχὴν	κ(αὶ)	ὁ	τέλο(ς)·			
doxa	*to*	*donti*	*archēn*	*k(ai)*	*ho*	*telo(s)·*			

Table 2: Predicted cluster 11, containing ground truth clusters 21690 and 19675.

5. Performance Evaluation

To evaluate the scalability of the proposed clustering architectures, we designed a benchmark framework to systematically compare the in-memory and streaming implementations. The experiments utilized sentence-level data from the Duke Databank of Documentary Papyri[3], as segmented and made available through the Duke NLP project (Keersmaekers and Depauw, 2022)[4]. We selected only sentences containing no textual gaps to ensure that similarity measurements reflect genuine orthographic relationships and to prevent gap markers from generating artificial character shingles that would inflate false-positive LSH matches and distort benchmark results. This filtering produced a corpus of 113 596 sentences suitable for controlled performance evaluation.

Starting from 1 000 sentences, we successively doubled the dataset size at each step, continuing until the full dataset was reached. This incremental approach allowed us to observe how processing time and memory requirements scaled as the number of input sentences increased, highlighting performance characteristics at different collection sizes.

Experiments were performed on a system equipped with 14 CPU cores and 30.8GB of total memory, with the benchmark restricted to 60% of memory to leave adequate resources for other processes. Dedicated monitoring threads tracked memory usage, runtime, throughput, and efficiency, enabling systematic assessment of performance as dataset size grew. Parallel processing and aggressive memory management were applied consistently to ensure that memory from previous tests was fully released before each new run.

Metric	**Optimized**	**Streaming**
Peak Memory (MB)	2699.2	637.7
MB/doc	0.023761	0.005613
Processing Time (s)	1174.38	202.80
Time Complexity	$O(n^{1.65})$	$O(n^{1.08})$
Memory Complexity	$O(n^{1.25})$	$O(n^{0.96})$

Table 3: Scalability comparison of the in-memory and streaming methods on 113 596 sentences from the Duke NLP corpus.

The evaluation highlights distinct performance patterns between the two clustering approaches. The memory-optimized method performs strongly on smaller datasets, where parallel processing enables high throughput when the number of documents is low. Yet, as corpus size increases, its memory requirements grow superlinearly, restricting its use to smaller collections (Figure 8).

By contrast, the streaming approach exhibits a notable warm-up phase: for datasets below roughly 5 000 documents, per-document memory costs are relatively high due to fixed infrastructural overhead

[3] https://github.com/papyri/idp.data

[4] https://github.com/alekkeersmaekers/duke-nlp/

such as database initialization, index construction, and connection management. Beyond this point, per-document costs stabilize at consistently low levels and processing time remains low (Figure 9). The measured memory complexity exponent stays slightly below 1.0 (Table 3), suggesting that memory usage grows somewhat more slowly than dataset size once initialization costs are distributed over sufficient documents.

Taken together, these results indicate that method selection should be guided by corpus scale. For small collections, the memory-optimized approach is preferable, as its benefits from parallelization outweigh memory overhead and the streaming method's initial costs remain high. For larger-scale research the streaming approach becomes essential, offering superior memory efficiency and sustainable scalability that should, in principle, support collections of millions of documents on commodity hardware. It should be noted, however, that these projections extend beyond the empirically validated range. Further evaluation at larger scales will be required to confirm whether the observed sublinear scaling persists and to identify any potential bottlenecks that might arise at higher document volumes. For the live demo setup of CorpusClues, we limited input to a maximum of 60 000 rows and a 200MB upload due to server capacity constraints.

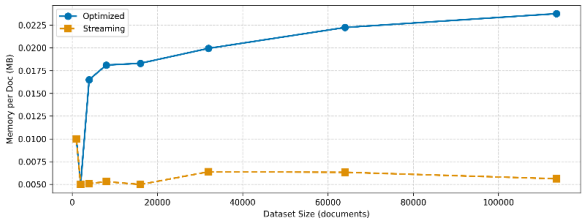

Figure 8: Evolution of memory use per document on dataset size increase.

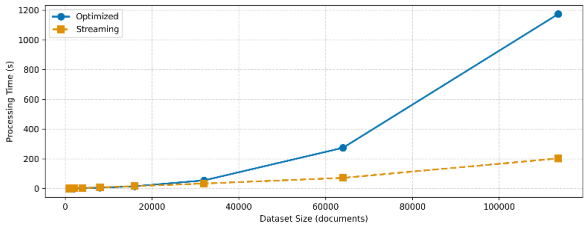

Figure 9: Evolution of processing time on dataset size increase.

6. Conclusion & Future Work

CorpusClues provides a scalable and flexible platform for unsupervised detection of textual similarity. Using character-level MinHash and LSH, it uncovers orthographic patterns while preserving user control over text preprocessing. Evaluation on Byzantine book epigrams demonstrates accurate reconstruction of similarity clusters despite substantial variation, whereas performance benchmarks indicate that, using a streaming-based approach, scaling to much larger datasets is achievable. By combining computational efficiency with linguistic insight, CorpusClues enables interactive exploration of formulaicity and textual transmission across large, heterogeneous corpora.

Future work could explore the empirical validation of performance across both longer textual units and diverse languages. Although the character-level approach is theoretically language- and script-agnostic, systematic evaluation on datasets in languages such as Arabic or Chinese would provide quantitative evidence of its generalizability and reveal any language-specific preprocessing requirements.

Next to validation, computational efficiency remains a key focus. Integration with distributed computing frameworks would enable parallel processing across multiple nodes, supporting fast analysis of millions of texts.

In addition to that, future work could focus on extending the capabilities of CorpusClues to support a wider range of analyses. One way to achieve this is by providing deeper insights into textual clusters. For instance, trend analysis and interactive exploration could uncover patterns and relationships that are not immediately apparent from basic cluster groupings. Another important direction is to expand the set of clustering algorithms. This could involve adding additional orthographic similarity methods or, even more ambitiously, incorporating semantic similarity algorithms to capture meaning-based relationships and enable the study of semantic change across historical corpora.

Finally, future work could also involve evaluating the system's effectiveness for scholarly workflows. Conducting structured user studies with historical linguists and philologists would allow assessment of the interface's intuitiveness, accessibility, and alignment with existing research practices. Based on such qualitative feedback, subsequent iterations of CorpusClues could focus on improving user-friendliness, complementing the scalable architecture and analytical capabilities to create a robust, user-validated platform for large-scale textual similarity analysis across diverse historical corpora.

7. Code and Data Availability

The complete source code for CorpusClues, including the web application, clustering algorithms, and all benchmark scripts, is publicly available at https://github.com/PaulienLem/

lrec-demo. The quality benchmark dataset (subset from the Database of Byzantine Book Epigrams) is included in the repository and enables full reproduction of the results discussed in Section 4. The performance benchmark dataset is not directly included due to size restrictions, but can be built by running the included script.

8. Acknowledgements

This work is part of the Annophis project (https://www.annophis.ugent.be/), supported by the Research Foundation Flanders (FWO) under grant I005524N.

9. Bibliographical References

Fabian Barteld. 2017. Detecting spelling variants in non-standard texts. In *Conference of the European Chapter of the Association for Computational Linguistics*.

Alexander Beihammer. 2020. *Epistolography and Diplomatics*, chapter 7. Brill, Leiden, The Netherlands.

Klaas Bentein and Kyriaki Giannikou. 2025. Formulaic language (medieval greek). In Georgios Giannakis, Emilio Crespo, Panagiotis Filos, Brian Joseph, and Theodoros Markopoulos, editors, *Encyclopedia of Greek Language and Linguistics Online*. Brill.

Andrei Z Broder. 1997. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, pages 21–29. IEEE.

Andreas Buerki. 2020. *Formulaic Language and Linguistic Change: A Data-Led Approach*. Cambridge University Press.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2022. *Data Structures for Disjoint Sets*, chapter 19. MIT Press.

Fred J Damerau. 1964. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176.

Maxime Deforche, Ilse De Vos, Antoon Bronselaer, and Guy De Tré. 2024. A hierarchical orthographic similarity measure for interconnected texts represented by graphs. *Applied Sciences*, 14(4):30.

Kristoffel Demoen, Gilbert Bentein, Klaas Bentein, Floris Bernard, Julián Bértola, Julie Boeten, Mathijs Clement, Cristina Cocola, Eline Daveloose, Sien De Groot, Pieterjan De Potter, Ilse De Vos, Krystina Kubina, Hanne Lauwers, Paulien Lemay, Renaat Meesters, Marjolein Morbé, Delphine Nachtergaele, Marthe Nemegeer, Nina Vanhoutte, et al. 2023. *Database of Byzantine Book Epigrams (1.0.0)*. Zenodo.

Kyriaki Giannikou, Colin Swaelens, Ilse De Vos, Els Lefever, and Klaas Bentein. 2024. Decoding byzantine book epigrams: an exploration of machine-assisted extraction of formulaic material. In *Workshop on Data-driven Approaches to Ancient Languages*, pages 22–32. Language & Translation Technology Team.

Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 1999. Similarity search in high dimensions via hashing. In *Vldb*, volume 99, pages 518–529.

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification*, 2(1):193–218.

Paul Jaccard. 1901. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579.

Matthew A Jaro. 1995. Probabilistic linkage of large public health data files. *Statistics in medicine*, 14(5-7):491–498.

Alek Keersmaekers and Mark Depauw. 2022. Bringing together linguistics and social history in automated text analysis of greek papyri. *Classics@*, 20:1–14.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

Johann-Mattis List. 2012. LexStat: Automatic detection of cognates in multilingual wordlists. In *Proceedings of the EACL 2012 Joint Workshop of LINGVIS & UNCLH*, pages 117–125, Avignon, France. Association for Computational Linguistics.

Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. 2007. Detecting near-duplicates for web crawling. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 141–150, New York, NY, USA. Association for Computing Machinery.

D. Nachtergaele. 2023. *The Formulaic Language of the Greek Private Papyrus Letters*. Trismegistos Online Publications, Leuven. TOP SS6: Revised version (2016) of a 2015 PhD. See also

the connected database TM Letters TOP SS6 (TM subscribers only).

Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88.

Milman Parry. 1930. Studies in the epic technique of oral verse-making. i. homer and homeric style. *Harvard Studies in Classical Philology*, 41:73–147.

Rachele Ricceri, Klaas Bentein, Floris Bernard, Antoon Bronselaer, Els De Paermentier, Pieterjan De Potter, Guy De Tré, Ilse De Vos, Maxime Deforche, Kristoffel Demoen, Els Lefever, Anne-Sophie Rouckhout, and Colin Swaelens. 2023. The database of byzantine book epigrams project: principles, challenges, opportunities. *Journal of Data Mining and Digital Humanities*, page 41.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 410–420.

Colin Swaelens, Maxime Deforche, Guy De Tré, Ilse De Vos, and Els Lefever. 2024. How relevant is part-of-speech information to compute similarity between greek verses in a graph database? In *Proceedings of the First Workshop on Data-driven Approaches to Ancient Languages (DAAL 2024)*, pages 33–43. Language & Translation Technology Team (LT3).

Søren Wichmann, Eric W. Holman, Cecil H. Brown, Matthew S. Dryer, and Qibin Ran. 2025. The asjp database (version 21) [data set]. https://asjp.clld.org/. Accessed September 2025.

William E Winkler. 1990. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. https://eric.ed.gov/?id=ED325505.

Juan Zamora, Marcelo Mendoza, and Héctor Allende. 2016. Hashing-based clustering in high dimensional data. *Expert Systems with Applications*, 62:202–211.