Do Lexical and Contextual Coreference Resolution Systems Degrade Differently under Mention Noise? An Empirical Study on Scientific Software Mentions

Atilla Kaan Alkan[1], **Felix Grezes**[1], **Jennifer Lynn Bartlett**[1], **Anna Kelbert**[1], **Kelly Lockhart**[1], **Alberto Accomazzi**[1]

[1]Harvard-Smithsonian Center for Astrophysics, Cambridge, MA, USA
{atilla.alkan, felix.grezes, jennifer.bartlett, anna.kelbert, kelly.lockhart, alberto.accomazzi}@cfa.harvard.edu

Abstract

We present our participation in the SOMD 2026 shared task on cross-document software mention coreference resolution, where our systems ranked second across all three subtasks. We compare two fine-tuning-free approaches: Fuzzy Matching (FM), a lexical string-similarity method, and Context Aware Representations (CAR), which combines mention-level and document-level embeddings. Both achieve competitive performance across all subtasks (CoNLL F_1 of 0.94–0.96), with CAR consistently outperforming FM by 1 point on the official test set, consistent with the high surface regularity of software names, which reduces the need for complex semantic reasoning. A controlled noise-injection study reveals complementary failure modes: as boundary noise increases, CAR loses only 0.07 F_1 points from clean to fully corrupted input, compared to 0.20 for FM, whereas under mention substitution, FM degrades more gracefully (0.52 vs. 0.63). Our inference-time analysis shows that FM scales superlinearly with corpus size, whereas CAR scales approximately linearly, making CAR the more efficient choice at large scale. These findings suggest that system selection should be informed by both the noise profile of the upstream mention detector and the scale of the target corpus. We release our code to support future work on this underexplored task.

Keywords: cross-document coreference resolution, mention detection noise, scientific software mentions

1. Introduction

Coreference resolution is the task of identifying all mentions in a text, including proper nouns, definite descriptions, pronouns, and nominal expressions that refer to the same real-world entity (Jurafsky and Martin, 2026). It is a core component of natural language understanding pipelines, underpinning downstream tasks such as information extraction (Zelenko et al., 2004), question answering (Chai et al., 2022), and summarisation (Huang and Kurohashi, 2021). The automatic detection and disambiguation of software mentions in scientific literature has gained increasing attention as a means of improving research reproducibility and enabling large-scale meta-analyses of the use of methodologies across disciplines (Schindler et al., 2021, 2022; Du et al., 2021). In the scientific domain, and particularly for software mentions, coreference is predominantly nominal, with mentions taking the form of proper names, abbreviations, version-qualified strings, or URL references rather than pronouns or definite descriptions. This interest has been catalysed in part by shared evaluation campaigns: the SOMD 2025 (Upadhyaya et al., 2025) shared task, focused on detecting software mentions and their semantic relations in scientific text, attracted a range of systems submissions (Ojha et al., 2025; Rastogi and Tiwari, 2025; Mandic et al., 2025; Silva et al., 2025) and demonstrated both the feasibility and the remaining challenges of automated software mention extraction. Building on this foundation, SOMD 2026 extends the challenge to the cross-document setting, where the goal shifts from detecting individual mentions to resolving which mentions, across an entire collection of papers, refer to the same software entity. This is non-trivial: the same tool may appear under its full name, an acronym, a version-qualified string, or a URL, while conversely the same surface form may legitimately denote different tools in different disciplinary contexts. While coreference resolution has been studied extensively in both within-document and cross-document settings for scientific text (Chaimongkol et al., 2014; Luan et al., 2018; Brack et al., 2021; Forer and Hope, 2024), these efforts have focused primarily on general scientific concepts, biomedical entities, and argumentative discourse structures. Software mentions, with their particular mix of proper names, versioned identifiers, and abbreviations, constitute a distinct entity type that has received no dedicated coreference treatment. SOMD 2026 thus serves as the first standardised benchmark for this task.

In this paper, we present our participation in the SOMD 2026 shared task and address the following research questions:

- **RQ1** *Can a simple, unsupervised lexical base-*

Proceedings of the 3rd Int. Workshop on Natural Scientific Language Processing (NSLP 2026) @ LREC 2026, pages 97–107
12 May 2026.

line compete with a contextual embedding approach on cross-document software coreference? Given the high surface regularity of software names, we hypothesize that fuzzy string matching may constitute a strong baseline that is difficult to surpass without task-specific supervision.

- **RQ2** *How robust are these approaches to different types and levels of annotation noise?* Real-world annotations are imperfect, and understanding system behaviour under controlled degradation provides insight into practical deployment limits.
- **RQ3** *What is the precision–speed trade-off between the two approaches?* For large-scale literature mining pipelines, inference efficiency is a major concern alongside accuracy.

Our main contributions are: (i) two competitive unsupervised baselines for the SOMD 2026 shared task; (ii) as part of this work, a systematic noise injection study that, to our knowledge, is the first robustness analysis conducted in the context of software mention coreference resolution; and (iii) a characterisation of the inference-time trade-off between lexical and neural approaches on this task. We release our code[1] to support reproducibility and future work on this task.

The remainder of this paper proceeds as follows. Section 2 reviews related work and positions software mention coreference as a distinct problem. Section 3 describes the shared task, corpus, and training data analysis that informed our design choices. Section 4 presents our two systems. Section 5 describes the evaluation metrics and noise injection protocol. Section 6 reports results and robustness findings. Section 7 discusses broader implications and limitations. Section 8 concludes with a summary of key insights and directions for future research.

2. Related Work

Software Mention Detection and Disambiguation The extraction of software mentions from scientific text has received growing attention, with corpora such as SoftCite (Du et al., 2021) and SoMeSci (Schindler et al., 2021) providing annotated mentions of software names, version numbers, URLs, and developer attributes. These efforts primarily focus on named entity recognition and attribute extraction rather than on coreference. Shared evaluation campaigns have further advanced the field: Grezes et al. (2022) organized the DEAL 2023 shared task on entity detection in astrophysics literature, which included software mentions as a target entity type. More recently, SOMD 2025 (Upadhyaya et al., 2025) targeted the detection of software mentions and their relational attributes as named entities in scholarly text, attracting a range of submissions exploring detection and relation extraction (Ojha et al., 2025; Rastogi and Tiwari, 2025; Mandic et al., 2025; Silva et al., 2025). Across these efforts, the focus has consistently remained on mention-level extraction; cross-document disambiguation and coreference resolution have received less dedicated treatment, a gap that SOMD 2026, to the best of our knowledge, is the first shared task to directly address.

Scientific Coreference Resolution Coreference resolution has been extensively studied in newswire and general-domain text (Lee et al., 2017; Joshi et al., 2020), but scientific text presents distinct challenges: dense technical terminology, heavy use of abbreviations, and domain-specific entity types are poorly covered by general-purpose systems. Within-document coreference for scientific text has been widely addressed in the biomedical domain (Zweigenbaum et al., 2012; Lu and Poesio, 2021), supported by annotated corpora such as MedStract (Li et al., 2014), Genia-MedCo (Li et al., 2014), and DrugNerAR (Segura-Bedmar et al., 2009). Coreference corpora have also been developed for broader scientific domains (Chaimongkol et al., 2014; Brack et al., 2021; Luan et al., 2018) and for astrophysics (Alkan et al., 2024). Cross-document scientific coreference has received comparatively less attention: Cattan et al. (2021b) introduced SciCo for cross-document coreference of scientific concepts, while recent work has explored LLM-based relational reasoning (Forer and Hope, 2024) and knowledge-graph-grounded entity linking (Dong et al., 2025) to improve cross-document resolution. Despite this growing body of work, software mentions, as a distinct entity type combining versioned identifiers and abbreviations, remain understudied.

Coreference Resolution Methods Early coreference research relied on unsupervised heuristics and rule-based approaches, drawing on linguistic constraints such as syntactic and semantic compatibility (Hobbs, 1978; Grosz and Sidner, 1986; Grosz et al., 1995; Haghighi and Klein, 2010), with recent work demonstrating that simple unsupervised rules remain competitive in certain settings (Stolfo et al., 2022). The field has since shifted toward supervised neural architectures for both within-document and cross-document resolution (Clark and Manning, 2016; Wiseman et al., 2015; Tourille et al., 2020; Gliosca and Amsili, 2019; Barhom et al., 2019; Cattan et al., 2021a), and more recently started to explore zero-shot approaches leveraging

[1] https://github.com/adsabs/SOMD-2026

pre-trained language models such as BERT (Devlin et al., 2019), SciBERT (Beltagy et al., 2019), and LLMs with prompting strategies (Blevins et al., 2023; Le et al., 2022; Le and Ritter, 2023). However, these methods are designed for mention types that differ substantially from those in the SOMD 2026 shared task. Existing systems typically target pronominal anaphora and nominal expressions with high lexical diversity, whereas the annotation scheme adopted by the SOMD 2026 organisers focuses exclusively on explicit software name mentions (a mention type characterised by a high degree of surface form similarity between corefering expressions). This distinction matters: the linguistic variation that motivates complex neural architectures is largely absent here, making heavyweight models both unnecessary and costly. While lighter alternatives such as FastCoref (Otmazgin et al., 2022) reduce computational overhead, they remain expensive for large-scale literature mining pipelines and are designed for general coreference rather than this specific mention type. The high surface regularity of software names and the need for scalable processing together motivate our choice of two lightweight, unsupervised approaches: fuzzy string matching, which directly exploits lexical similarity, and clustering over contextual embeddings, which captures semantic variation without fine-tuning.

3. Task and Corpus

3.1. Task Definition

SOMD 2026 frames software mention disambiguation as a cross-document coreference resolution problem: given a set of software mention spans with their surrounding sentences and metadata, the goal is to partition all mentions into clusters such that each cluster corresponds to a single underlying software entity. The three subtasks differ in the quality of the input mentions and the scale of the corpus:

- **Subtask 1** operates over gold-standard annotated mentions, providing an upper-bound evaluation of coreference resolution in isolation from mention detection errors;
- **Subtask 2** operates over automatically predicted mentions, reflecting real-world conditions where upstream mention detection is imperfect and introduces noise into the coreference input;
- **Subtask 3** operates over predicted mentions at a larger scale, explicitly targeting the computational efficiency challenge that arises as the volume of documents and the density of software name variants increase.

Our participation covers all three subtasks. Subtasks 2 and 3 operate on automatically predicted mentions, so the noise level in the input is inherent to the upstream mention detection pipeline and varies in ways that are difficult to quantify directly. To complement the official evaluation and gain a more controlled understanding of how input quality affects each system, we conduct a noise-injection study on the gold-standard training data, systematically varying the noise level across two perturbation types (**RQ2**). The scale dimension of Subtask 3 similarly motivates our inference-time analysis (**RQ3**).

3.2. Corpus Statistics

The shared task datasets comprise scholarly documents from scientific disciplines, annotated with software mention spans and their coreference chains. Two distinct training sets are provided: Subtask 1 uses gold-standard annotations, while Subtasks 2 and 3 share a training set of automatically predicted mentions. Table 1 reports corpus statistics for both.

Statistic	**Subtask 1** *(gold)*	**Subtasks 2 & 3** *(predicted)*
Corpus		
Documents	973	967
Sentences with mentions	2,153	2,140
Mention instances	2,974	2,860
Unique surface forms	837	791
Coreference chains		
Total clusters	733	699
Avg chain length	4.06	4.09
Max chain length	367	366
Singleton rate	51.7%	52.5%
Cross-doc rate (all clusters)	20.6%	20.9%
Cross-doc rate (non-singletons)	42.7%	44.0%
Mention surface forms		
Avg surface forms / cluster	1.14	1.13
Avg intra-cluster lexical sim.	0.881	0.887

Table 1: Corpus statistics for the SOMD 2026 training sets. Subtask 1 uses gold-standard mentions; Subtasks 2 and 3 share a common training set of automatically predicted mentions. Cross-doc rate (non-singletons) excludes singleton clusters, which require no linking decision.

The two training sets are relatively similar across all statistics, suggesting that the automatic mention detector used for Subtasks 2 and 3 is of high quality: it recovers a comparable number of mentions (2,860 vs. 2,974), a similar coreference chain structure, and a nearly identical lexical similarity profile. The primary difference lies in the slightly lower mention count and number of unique surface forms, reflecting mentions that the detector failed to recover. This observation is consequential for our experimental design: since the real-world

noise introduced by the upstream detector in Subtasks 2 and 3 is inherently mild and unquantified, we complement the official evaluation with a controlled noise injection study that systematically explores a wider range of noise levels, allowing us to characterise how each system degrades as input quality decreases (**RQ2**).

The coreference structure of both training sets reveals several properties that are consequential for system design. The clusters exhibit a highly skewed length distribution, with maximum chain lengths of 367 and 366 and average lengths of 4.06 and 4.09 for Subtasks 1 and 2&3, respectively. This is consistent with a small set of high-frequency software names, such as MATLAB, dominating the corpus, whereas most tools appear infrequently. Notably, singleton rates of 51.7% and 52.5% indicate that over half of all mentions lack a coreferent counterpart, implying that any coreference system must be conservative in its linking decisions.

An analysis of the coreference chain structure reveals an important nuance regarding the cross-document nature of the task. The raw cross-document cluster rate is approximately 20% in both training sets, suggesting that the resolution problem is predominantly within-document. However, this figure is strongly influenced by the high singleton rate of around 52%: singletons trivially belong to a single document and require no linking decision. Among non-singleton chains, the cross-document rate rises to 42.7% and 44.0% for Subtasks 1 and 2&3, respectively, confirming that the task poses a genuine cross-document disambiguation challenge for the majority of chains that actually require coreference linking.

Most consequentially for our system design choices, both training sets exhibit a high degree of lexical regularity within coreference chains. The average number of distinct surface forms per cluster is 1.14 and 1.13, respectively, indicating that coreferring mentions are almost always near-identical strings rather than paraphrases or pronominal references. This is confirmed by average intra-cluster lexical similarities of 0.881 and 0.887, substantially higher than what would be expected in general coreference corpora where chains mix proper names, nominal descriptions, and pronouns. This property directly motivates our choice of lightweight unsupervised approaches and supports the hypothesis underlying **RQ1** that lexical similarity alone may constitute a strong signal for this task.

4. Systems

4.1. Fuzzy Matching

The fuzzy matching system clusters software mentions based on the lexical surface similarity. For each pair of mention strings m_i and m_j, we compute a similarity score $s(m_i, m_j) \in [0, 1]$ using the Ratcliff/Obershelp algorithm (Ratcliff and Obershelp, 1988), as implemented by SequenceMatcher in Python's difflib library. The Ratcliff/Obershelp algorithm computes similarity as twice the number of matching characters divided by the total number of characters in both strings, where matching characters are identified by recursively finding the longest common substring and applying the same procedure to the non-matching regions on either side. This makes it sensitive to the overall structure of the string rather than character-level edit distance, and well-suited to software names where shared substrings are a strong indicator of coreference (e.g., *"GraphPad Prism"* and *"GraphPad Prism 8"*). Two mentions are linked if $s(m_i, m_j) \geq \theta$, where the threshold θ is tuned on the training set. Clusters are then formed by applying transitive closure over all linked mention pairs, such that if m_i is linked to m_j and m_j is linked to m_k, all three are assigned to the same cluster regardless of the direct similarity between m_i and m_k. The fuzzy matching system operates solely on mention strings and is entirely agnostic to document context, mention type, and surrounding text. Its computational complexity is superlinear in the number of unique mention strings, which is manageable given the relatively small number of unique surface forms in the corpus (837 and 791 for Subtasks 1 and 2&3, respectively; see Table 1).

4.2. Context Aware Representations

The context-aware representations system encodes each software mention using all-MiniLM-L6-v2 (Wang et al., 2020), a lightweight sentence embedding model from the Sentence Transformers library. The model comprises 6 transformer layers and 22M parameters and has been trained to produce semantically meaningful sentence-level representations via knowledge distillation from larger models. Its compact size makes it well-suited to large-scale mention encoding without requiring GPU acceleration. Rather than encoding the mention in its sentential context alone, we separately encode two complementary signals and combine them into a single representation. First, the normalised mention string is encoded independently, producing a mention-level representation $\mathbf{e}_m \in \mathbb{R}^d$ that captures the surface form of the software name. Second, a document-level representation $\mathbf{e}_d \in \mathbb{R}^d$ is constructed by aggregating up to ten unique mention-bearing sentences from the same document into a single string, which is then encoded with all-MiniLM-L6-v2. This document context captures the broader thematic and disciplinary setting in which the mention appears, providing a complementary signal for cases where the same surface

form refers to different software in different contexts. Both representations are independently normalised to unit length and combined as a weighted sum:

$$\mathbf{e} = \alpha \cdot \mathbf{e}_m + (1 - \alpha) \cdot \mathbf{e}_d \quad (1)$$

where $\alpha = 0.6$, giving slightly more weight to the mention-level signal. This design reflects the intuition confirmed by our corpus analysis (Section 3.2): software names are highly surface-regular, making the mention string the primary coreference signal, while document context provides a disambiguating signal for ambiguous cases. The combined representations are clustered using agglomerative clustering with cosine distance and average linkage. Since each mention is encoded independently by the sentence transformer without reference to other mentions, the encoding step scales linearly with corpus size; the subsequent agglomerative clustering step runs in $O(n^2 \log n)$ via SKLEARN's precomputed distance matrix approach, avoiding the naive $O(n^3)$ complexity of a stored-matrix implementation.

4.3. Threshold Tuning

The fuzzy matching system requires a single threshold hyperparameter θ, defining the minimum Ratcliff/Obershelp similarity score above which two mentions are linked. We perform a grid search over $\theta \in [0, 1]$ on the training set, selecting the value that maximises CoNLL F_1. Since no development set is provided by the shared task, we use the full training set for this purpose. The selected threshold is $\theta = 0.83$ for Subtask 1 and $\theta = 0.84$ for Subtasks 2 and 3. The context-aware system uses a fixed distance threshold of $\delta = 0.4$ for the agglomerative clustering step, which was set empirically without formal tuning. For the noise-injection experiments, the threshold θ is re-tuned at each noise level using the same grid-search procedure, and the best achievable performance is reported for each noise condition. This provides an optimistic upper bound on the robustness of the fuzzy matching method, assuming that the system has access to a clean validation signal at each noise level. The implications of this choice are discussed further in Section 7.

5. Experimental Setup

5.1. Evaluation Metrics

All official test-set scores are computed by the shared task organisers using the standard coreference resolution metrics: MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), and CEAFe (Luo, 2005). The primary metric is CoNLL F_1 (Pradhan et al., 2014), defined as the unweighted average of the three F_1 scores. In addition to coreference performance, we report the average inference time for each system in order to characterise the precision–speed trade-off between the two approaches (**RQ3**).

5.2. Noise Injection Protocol

As discussed in Section 3.1, the noise level introduced by the automatic mention detector in Subtasks 2 and 3 is inherent to the upstream pipeline and cannot be directly quantified. To complement the official evaluation, we conduct a controlled internal robustness analysis by injecting noise directly into the training set mentions and evaluating both systems on the resulting perturbed data. This diagnostic study is independent of the official test set and is not intended to be directly compared with the results in Table 3; its purpose is to assess how each system degrades as input quality decreases under controlled, quantifiable noise conditions (**RQ2**). The two noise types we consider are motivated by realistic failure modes of mention detection systems. **Boundary modification** simulates span boundary errors, which are among the most common annotation and detection mistakes in span-level tasks: a mention span is randomly extended or truncated, producing a slightly incorrect but plausible mention. **Mention substitution** simulates a context mismatch error, where a mention detection system correctly identifies a span as a software mention but associates it with the wrong software name: the mention string is replaced with a different software name sampled from the training set, preserving the syntactic structure of the sentence. Table 2 illustrates each noise type on a concrete example. Each perturbation type is applied at different intensity levels: 0%, 25%, 50%, 75%, and 100% of all mentions affected, and both systems are evaluated after each perturbation.

6. Results

6.1. Main Results on the Test Set

Table 3 reports the official test-set results for all participating SOMD 2026 systems alongside our two submissions.

The results reveal several findings. First, all systems, including our two unsupervised, fine-tuning-free baselines, achieve very high CoNLL F_1 scores across all subtasks, ranging from 0.94 to 0.96. This confirms the hypothesis underlying **RQ1**: the high lexical regularity of software mention chains (avg. intra-cluster similarity of 0.881; see Table 1) implies that even a simple surface-matching approach constitutes a strong baseline for this task. Second, our context-aware representations system consistently outperforms the fuzzy matching approach,

Noise Type	Example
Original	We used MATLAB for statistical analysis and visualization.
Boundary Modification	We used MATLAB for statistical analysis and visualization. We used the MATLAB for statistical analysis and visualization.
Mention Substitution	We used Python for statistical analysis and visualization. We used NumPy for statistical analysis and visualization.

Table 2: Illustration of noise injection methods applied to a software mention. The original mention is shown in blue, while noisy mentions are shown in red. Boundary modification extends or truncates mention spans; mention substitution replaces the mention string with another software name sampled from the training set. Each noise type simulates a different failure mode of upstream mention detection. All methods are tested at noise rates of 0%, 10%, 25%, 50%, 75%, and 100%.

	Subtask 1				Subtask 2				Subtask 3			
System	CoNLL	B^3	$CEAF_e$	MUC	CoNLL	B^3	$CEAF_e$	MUC	CoNLL	B^3	$CEAF_e$	MUC
System A†	**0.98**	**0.99**	**0.96**	**0.99**	**0.98**	**0.99**	**0.95**	**0.99**	**0.96**	**0.97**	**0.92**	**0.99**
System B†	0.92	0.94	0.86	0.97	0.92	0.93	0.85	0.98	‡	‡	‡	‡
Fuzzy Matching	0.95	0.95	0.91	0.98	0.95	0.94	0.91	**0.99**	0.94	0.94	0.90	**0.99**
Context Aware	0.96	0.96	0.93	0.98	0.96	0.96	0.93	**0.99**	‡	‡	‡	‡

Table 3: Official test-set results for SOMD 2026. CoNLL is the average of MUC, B^3, and CEAFe F_1. **Bold** = best overall. Underline = best among our systems. † = other participants. ‡ = no submission on Codabench.

achieving a CoNLL F_1 of 0.96 on both Subtasks 1 and 2 compared to 0.95 for FM. While the absolute gap is modest (one point), it is consistent across all coreference metrics. The improvement is most visible on $CEAF_e$ (0.93 vs. 0.91), which penalises both over- and under-clustering and is therefore a more sensitive indicator of clustering quality than MUC, which is known to favour recall. This suggests that the document-level contextual representations in CAR provide a small but consistent benefit over purely lexical matching, likely by resolving ambiguous cases in which the same surface form can refer to different software across different disciplinary contexts. Third, both our systems outperform System B† on all subtasks by a margin of 3–4 CoNLL F_1 points, despite requiring no training or fine-tuning. This further underscores the strength of unsupervised lexical and lightweight embedding approaches for this task, and suggests that the added complexity of supervised systems may not yet be warranted given the current scale and lexical regularity of the corpus. Fourth, the performance of all systems is remarkably stable across Subtasks 1, 2, and 3. The transition from gold-standard mentions (Subtask 1) to automatically predicted mentions (Subtask 2) produces no measurable degradation for any system, which is consistent with our corpus analysis showing that the two training sets are nearly identical in their statistical properties (Table 1). This suggests that the upstream mention detector used by the shared task organisers is of high quality.

6.2. Robustness to Mention Noise

We evaluate robustness by injecting controlled noise into the training sets of Subtasks 1 and 2. Table 4 reports degradation curves for both noise types. As noted in Section 4.3, FM threshold θ is retuned at each noise level, so FM scores represent an optimistic upper bound.

Boundary Modification FM loses 0.20 CoNLL F_1 points over the full noise range (0.70 $\rightarrow$ 0.50) while CAR loses only 0.07 (0.82 $\rightarrow$ 0.75), a three-fold robustness advantage. FM degrades approximately linearly, while CAR's curve is nearly flat. This asymmetry reflects how each system processes mention strings: FM relies on exact character-level matching, so adding or removing one token directly degrades similarity scores. CAR encodes mentions as dense vectors, so minor boundary changes leave the semantic content largely intact (e.g., *"MATLAB for"* encodes similarly to *"MATLAB"*). This robustness is a property of the embedding representation itself, not evidence that document context provides a compensatory signal, as the mention substitution results confirm. Practically, CAR at 100% boundary noise (0.75) still outperforms FM at 0%

Noise Type	System	Injected Noise Rate ∅	25%	50%	75%	100%	Δ
Subtask 1 (gold mentions)							
Boundary Modification	Fuzzy Matching	0.70	0.65	0.60	0.54	0.50	0.20
	Context Aware	0.82	0.80	0.79	0.77	0.75	**0.07**
Mention Substitution	Fuzzy Matching	0.70	0.49	0.34	0.23	0.18	**0.52**
	Context Aware	0.82	0.57	0.39	0.25	0.19	0.63
Subtask 2 (predicted mentions)							
Boundary Modification	Fuzzy Matching	0.70	0.66	0.62	0.57	0.51	0.19
	Context Aware	0.82	0.80	0.80	0.78	0.76	**0.06**
Mention Substitution	Fuzzy Matching	0.70	0.50	0.34	0.23	0.18	**0.52**
	Context Aware	0.82	0.59	0.39	0.26	0.20	0.62

Table 4: CoNLL F_1 under all noise types at increasing noise rates on the training set. Δ = F_1 at 0% − F_1 at 100% (lower = more robust). Baseline scores differ from official test-set results (Table 3) as noise experiments are conducted on the training set. Underline = best F_1; **bold** Δ = most robust system per noise type.

noise (0.70).

Mention Substitution The robustness ranking reverses: FM ($\Delta = 0.52$) is now more robust than CAR ($\Delta = 0.63$), despite starting from a lower clean baseline. Both systems converge to near-identical performance at 100% noise (FM: 0.18, CAR: 0.19). FM degrades approximately linearly, whereas CAR collapses more sharply at low noise levels: at 25% noise, CAR drops by 0.25 points, whereas FM drops by 0.21. This early collapse occurs because CAR's document context is constructed from mention-bearing sentences (Section 4.2): substituting mention strings corrupts both the mention-level and document-level embeddings simultaneously, so neither component can compensate for the other. This resolves the apparent tension with the boundary modification results: CAR's robustness advantage there reflects the tolerance of dense embeddings to minor perturbations, not an independent contextual signal. When mention content is substantively replaced, this advantage disappears.

Consistency and Summary Degradation patterns are consistent across Subtasks 1 and 2 (with differences of at most 0.01 across all conditions), confirming that the findings generalise to the predicted-mention setting. Overall, the two systems exhibit complementary robustness profiles: CAR is more robust to boundary noise while FM degrades more gracefully under mention substitution, and neither system is resilient to severe mention content corruption (**RQ2**).

6.3. Inference Time and Precision–Speed Trade-off

We report in Table 5 the inference time and precision–speed trade-off for both systems on the official test sets of Subtasks 1 and 2, measured on a CPU-only machine (Intel x86_64, 14 physical cores, 33.3 GB RAM) over 5 runs.

Subtask	System	Time (s)	CoNLL F_1	F1/sec
Subtask 1	FM	0.60 ± 0.00	0.95	1.584
	CAR	4.45 ± 0.69	0.96	0.215
Subtask 2	FM	47.06 ± 0.41	0.95	0.020
	CAR	45.39 ± 5.14	0.96	0.021

Table 5: Inference time and precision–speed trade-off on official test sets of Subtasks 1 and 2. Efficiency = CoNLL F_1 / inference time (s). Measured on an Intel x86_64 CPU (14 cores, 33.3 GB RAM) over 5 runs.

At small scale (Subtask 1, n = 743 mentions), FM is 7.4× faster than CAR (0.60s vs. 4.45s) at near-identical accuracy (0.95 vs. 0.96), making it the more efficient choice for small corpora. At large scale (Subtask 2, $n \approx$ 21,500 mentions), the picture reverses: the input size increases 29× relative to Subtask 1, but FM's inference time increases 78× (0.60s → 47.06s) while CAR's increases only 10× (4.45s → 45.39s), and both systems converge to nearly identical inference times (≈46s) with CAR still holding its marginal accuracy advantage. This result is counter-intuitive: despite being the simpler and lighter system, FM scales less favourably than CAR at a large scale. The reason is that FM's simplicity relies on pairwise string comparison, whose

cost grows faster than linearly with the number of mentions, whereas CAR's neural encoding (though more expensive per mention) processes each mention independently and therefore scales approximately linearly. FM's inference time is perfectly stable across runs ($\pm$0.00s), reflecting its deterministic procedure, whereas CAR exhibits higher variance ($\pm$0.69s on Subtask 1, $\pm$5.14s on Subtask 2), attributable to variability in the neural encoding step on the CPU; GPU acceleration would reduce both.

7. Discussion

Accuracy vs. scale (RQ3) At small scale (Subtask 1, $n = 743$), FM is 7.4$\times$ faster than CAR at near-identical accuracy, making it the more practical choice. At large scale (Subtask 2, $n \approx 21{,}500$), this advantage disappears: FM's inference time increases 78$\times$ while CAR's increases only 10$\times$, with both systems converging to similar inference times ($\approx$46s). This is consistent with FM's superlinear scaling in pairwise similarity computation, compared with CAR's approximately linear per-mention encoding. The practical recommendation is scale-dependent: FM for small corpora, CAR for large-scale pipelines.

Robustness, lexical regularity, and system selection (RQ1 & RQ2) A unifying finding across RQ1 and RQ2 is that the high lexical regularity of software mention chains, confirmed by an average intra-cluster similarity of 0.881 (Table 1), is the dominant factor shaping system behaviour. It explains why FM is competitive with CAR on clean data, why CAR's robustness advantage under boundary noise reflects embedding tolerance rather than genuine contextual signal, and why both systems collapse equally under mention substitution. Above a certain noise threshold, improving the upstream detector is more impactful than the choice of coreference method. Future work would benefit from document-level representations genuinely decoupled from mention string content. For instance, encoding non-mention sentences or document signals such as titles and abstracts.

Limitations Three limitations should be noted. First, FM's threshold θ was re-tuned at each noise level, so robustness results represent an optimistic upper bound. Second, CAR's document context is constructed from mention-bearing sentences, making it structurally dependent on mention string content. Third, inference times were measured on CPU only; GPU acceleration would reduce CAR's inference time and potentially widen its efficiency advantage at scale. Table 6 summarises these recommendations. The key insight is that neither system is universally superior: the right choice depends jointly on corpus scale and the error profile of the upstream mention detector.

Deployment condition	Recommended
Small corpus, high detector quality	FM
Large corpus ($n \gg$ 1k mentions)	CAR
Boundary errors dominate	CAR
Mention identity errors dominate	Improve detector

Table 6: Practical system selection guide based on deployment context.

8. Conclusion

We presented two unsupervised, fine-tuning-free systems for cross-document software mention coreference resolution at SOMD 2026, ranking second across all three subtasks. Our results answer three research questions. **RQ1**: The high lexical regularity of software mention chains makes unsupervised lexical methods strong baselines, competitive with contextual approaches without any task-specific supervision. **RQ2**: The two systems exhibit complementary robustness profiles: CAR is more robust to boundary noise, while FM degrades more gracefully under mention substitution, and neither is resilient to severe mention content corruption, pointing to upstream mention detection as the primary bottleneck for future progress. **RQ3**: FM is more efficient at a small scale while CAR scales more favourably to large corpora, making scale a decisive factor in system selection. Taken together, these findings provide concrete guidance for practitioners (Table 6): system selection should be informed by the upstream detector's noise profile and the scale of the target corpus, not by accuracy alone. We release our code[2] and hope that the noise-injection protocol introduced here serves as a reusable, robustness-evaluation framework for this underexplored task.

9. Bibliographical References

Atilla Kaan Alkan, Felix Grezes, Cyril Grouin, Fabian Schussler, and Pierre Zweigenbaum. 2024. Enriching a time-domain astrophysics corpus with named entity, coreference and astrophysical relationship annotations. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 6177–6188, Torino, Italia. ELRA and ICCL.

[2] https://github.com/adsabs/SOMD-2026

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains.

Shany Barhom, Vered Shwartz, Alon Eirew, Michael Bugert, Nils Reimers, and Ido Dagan. 2019. Revisiting joint modeling of cross-document entity and event coreference resolution. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4179–4189, Florence, Italy. Association for Computational Linguistics.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Terra Blevins, Hila Gonen, and Luke Zettlemoyer. 2023. Prompting language models for linguistic structure. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6649–6663, Toronto, Canada. Association for Computational Linguistics.

Arthur Brack, Daniel Uwe Müller, Anett Hoppe, and Ralph Ewerth. 2021. Coreference Resolution in Research Papers from Multiple Domains. *arXiv e-prints*, page arXiv:2101.00884.

Arie Cattan, Alon Eirew, Gabriel Stanovsky, Mandar Joshi, and Ido Dagan. 2021a. Cross-document coreference resolution over predicted mentions. *CoRR*, abs/2106.01210.

Arie Cattan, Sophie Johnson, Daniel S. Weld, Ido Dagan, Iz Beltagy, Doug Downey, and Tom Hope. 2021b. Scico: Hierarchical cross-document coreference for scientific concepts. *ArXiv*, abs/2104.08809.

Haixia Chai, Nafise Sadat Moosavi, Iryna Gurevych, and Michael Strube. 2022. Evaluating coreference resolvers on community-based question answering: From rule-based to state of the art. In *Proceedings of the Fifth Workshop on Computational Models of Reference, Anaphora and Coreference*, pages 61–73, Gyeongju, Republic of Korea. Association for Computational Linguistics.

Panot Chaimongkol, Akiko Aizawa, and Yuka Tateisi. 2014. Corpus for coreference resolution on scientific papers. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3187–3190, Reykjavik, Iceland. European Language Resources Association (ELRA).

Kevin Clark and Christopher D. Manning. 2016. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 643–653, Berlin, Germany. Association for Computational Linguistics.

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.

Zhang Dong, Mingbang Wang, Songhang deng, Le Dai, Jiyuan Li, Xingzu Liu, and Ruilin Nong. 2025. Cross-Document Contextual Coreference Resolution in Knowledge Graphs. *arXiv e-prints*, page arXiv:2504.05767.

Caifan Du, Johanna Cohoon, Patrice Lopez, and James Howison. 2021. Softcite dataset: A dataset of software mentions in biomedical and economic research publications. *J. Assoc. Inf. Sci. Technol.*, 72(7):870–884.

Lior Forer and Tom Hope. 2024. Inferring scientific cross-document coreference and hierarchy with definition-augmented relational reasoning. *ArXiv*, abs/2409.15113.

Quentin Gliosca and Pascal Amsili. 2019. Résolution des coréférences neuronale : une approche basée sur les têtes (neural coreference resolution : a head-based approach). In *Actes de la Conférence sur le Traitement Automatique des Langues Naturelles (TALN) PFIA 2019. Volume II : Articles courts*, pages 409–416, Toulouse, France. ATALA.

Felix Grezes, Sergi Blanco-Cuaresma, Thomas Allen, and Tirthankar Ghosal. 2022. Overview of the first shared task on detecting entities in the astrophysics literature (DEAL). In *Proceedings of the First Workshop on Information Extraction from Scientific Publications*, pages 1–7, Online. Association for Computational Linguistics.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Barbara J. Grosz and Candace L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 385–393, Los Angeles, California. Association for Computational Linguistics.

Jerry R. Hobbs. 1978. Resolving pronoun references. *Lingua*, 44(4):311–338.

Yin Jou Huang and Sadao Kurohashi. 2021. Extractive summarization considering discourse and coreference relations based on heterogeneous graph. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3046–3052, Online. Association for Computational Linguistics.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Daniel Jurafsky and James H. Martin. 2026. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models*, 3rd edition. Online manuscript released January 6, 2026.

Nghia T. Le, Fan Bai, and Alan Ritter. 2022. Few-shot anaphora resolution in scientific protocols via mixtures of in-context experts. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2693–2706, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Nghia T. Le and Alan Ritter. 2023. Are large language models robust coreference resolvers?

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Lishuang Li, Liuke Jin, Zhenchao Jiang, Jing Zhang, and Degen Huang. 2014. Coreference resolution in biomedical texts. In *2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 12–14.

Pengcheng Lu and Massimo Poesio. 2021. Coreference resolution for the biomedical domain: A survey. In *Proceedings of the Fourth Workshop on Computational Models of Reference, Anaphora and Coreference*, pages 12–23, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 25–32, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Stasa Mandic, Georg Niess, and Roman Kern. 2025. From in-distribution to out-of-distribution: Joint loss for improving generalization in software mention and relation extraction. In *Proceedings of the Fifth Workshop on Scholarly Document Processing (SDP 2025)*, pages 146–153, Vienna, Austria. Association for Computational Linguistics.

Vaghawan Ojha, Projan Shakya, Kristina Ghimire, Kashish Bataju, Ashwini Mandal, Sadikshya Gyawali, Manish Dahal, Manish Awale, Shital Adhikari, and Sanjay Rijal. 2025. SOMD 2025: Fine-tuning ModernBERT for in- and out-of-distribution NER and relation extraction of software mentions in scientific texts. In *Proceedings of the Fifth Workshop on Scholarly Document Processing (SDP 2025)*, pages 154–163, Vienna, Austria. Association for Computational Linguistics.

Shon Otmazgin, Arie Cattan, and Yoav Goldberg. 2022. F-coref: Fast, accurate and easy to use coreference resolution. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 48–56, Taipei, Taiwan. Association for Computational Linguistics.

Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. 2014. Scoring coreference partitions of predicted mentions: A reference implementation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 30–35, Baltimore, Maryland. Association for Computational Linguistics.

Pranshu Rastogi and Rajneesh Tiwari. 2025. Extracting software mentions and relations using transformers and LLM-generated synthetic data at SOMD 2025. In *Proceedings of the Fifth Workshop on Scholarly Document Processing (SDP 2025)*, pages 173–181, Vienna, Austria. Association for Computational Linguistics.

John W. Ratcliff and John A. Obershelp. 1988. JUL88: Pattern Matching: The Gestalt Approach. *Dr. Dobb's Journal*, 13:46–71.

D. Schindler, F. Bensmann, S. Dietze, and F. Krüger. 2022. The role of software in science: a knowledge graph-based analysis of software mentions in PubMed Central. *PeerJ Computer Science*, 8:e835.

David Schindler, Felix Bensmann, Stefan Dietze, and Frank Krüger. 2021. Somesci- a 5 star open data gold standard knowledge graph of software mentions in scientific articles. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 4574–4583, New York, NY, USA. Association for Computing Machinery.

Isabel Segura-Bedmar, Mario Crespo, Cesar de Pablo, and Paloma Martínez. 2009. Drugnerar: linguistic rule-based anaphora resolver for drug-drug interaction extraction in pharmacological documents. In *Proceedings of the Third International Workshop on Data and Text Mining in Bioinformatics*, DTMBIO '09, page 19–26, New York, NY, USA. Association for Computing Machinery.

Gabriel Silva, Mário Rodriges, António Teixeira, and Marlene Amorim. 2025. Inductive learning on heterogeneous graphs enhanced by LLMs for software mention detection. In *Proceedings of the Fifth Workshop on Scholarly Document Processing (SDP 2025)*, pages 164–172, Vienna, Austria. Association for Computational Linguistics.

Alessandro Stolfo, Chris Tanner, Vikram Gupta, and Mrinmaya Sachan. 2022. A simple unsupervised approach for coreference resolution using rule-based weak supervision. In *Proceedings of the 11th Joint Conference on Lexical and Computational Semantics*, pages 79–88, Seattle, Washington. Association for Computational Linguistics.

Julien Tourille, Olivier Ferret, Aurélie Névéol, and Xavier Tannier. 2020. Modèle neuronal pour la résolution de la coréférence dans les dossiers médicaux électroniques (neural approach for coreference resolution in electronic health records). In *Actes de la 6e conférence conjointe Journées d'Études sur la Parole (JEP, 33e édition), Traitement Automatique des Langues Naturelles (TALN, 27e édition), Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition). Volume 2 : Traitement Automatique des Langues Naturelles*, pages 351–360, Nancy, France. ATALA et AFCP.

Sharmila Upadhyaya, Wolfgang Otto, Frank Krüger, and Stefan Dietze. 2025. SOMD2025: A challenging shared tasks for software related information extraction. In *Proceedings of the Fifth Workshop on Scholarly Document Processing (SDP 2025)*, pages 137–145, Vienna, Austria. Association for Computational Linguistics.

Marc B. Vilain, John D. Burger, John S. Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Message Understanding Conference*.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Sam Wiseman, Alexander M. Rush, Stuart M. Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Annual Meeting of the Association for Computational Linguistics*.

Dmitry Zelenko, Chinatsu Aone, and Jason Tibbetts. 2004. Coreference resolution for information extraction. In *Proceedings of the Conference on Reference Resolution and Its Applications*, pages 24–31, Barcelona, Spain. Association for Computational Linguistics.

Pierre Zweigenbaum, Guillaume Wisniewski, Marco Dinarelli, Cyril Grouin, and Sophie Rosset. 2012. Résolution des coréférences dans des comptes rendus cliniques. Une expérimentation issue du défi i2b2/VA 2011. In *Actes de la conférence RFIA 2012*, pages 978–2–9539515–2–3, Lyon, France. Session "Articles".