Hybrid Retrieval-Guided Knowledge Graph Extraction with Precision Repair for PestCLEF 2026

PestCLEF at CLEF 2026

Seine A. Shintani[1,2,3,*]

[1]Department of Biomedical Science, College of Life and Health Sciences, Chubu University, 1200 Matsumoto-cho, Kasugai, Aichi, Japan
[2]Center for AI, Mathematical and Data Sciences, Chubu University, 1200 Matsumoto-cho, Kasugai, Aichi, Japan
[3]Institute of Advanced Research, Nagoya University, Nagoya, Aichi, Japan

Abstract

This working note describes the system submitted by team SAS3 to PestCLEF 2026, a LifeCLEF task on document-level knowledge graph extraction from web documents about plant pests and crop diseases. The final approach combined schema-constrained candidate generation, train/development-set graph reconstruction, alias gazetteers, near-duplicate retrieval, candidate-level verification, and conservative precision repair based on small validation submissions. A standalone learned extractor was not competitive, but a high-confidence add-on strategy over a defined reference backbone produced consistent gains. The two final selected submissions reached a public F-score of 0.45603 and a final private F-score of 0.50952, ranking the team first on the private leaderboard. Both selected runs included a high-value relation in document 102433 linking Asian citrus psyllid to Candidatus Liberibacter asiaticus and CLas; the two runs differed by one exact-location hedge edge in document 100506.

Keywords

Knowledge graph extraction, relation extraction, plant pests, LifeCLEF, PestCLEF, retrieval-guided extraction

1. Introduction

PestCLEF 2026 is a LifeCLEF challenge on knowledge graph extraction from web documents about plant pests and crop diseases. We cite both the LifeCLEF 2026 overview paper and the PestCLEF 2026 competition overview paper as requested by the organizers [1, 2]. The task is framed as document-level relation extraction: systems predict a graph of relations mentioned in each document, while exact offsets for predicted entities are not required [3]. Relations cover ecological interactions and events involving host plants, pests, diseases, vectors, dissemination pathways, dates, and locations.

The main difficulty is that the target graph is document-level rather than sentence-only. A pest, a disease, a host plant, and several locations may be mentioned in different sections of the same article. The expected output may also preserve multiple surface aliases for the same biological entity. At the same time, over-predicting edges can substantially reduce precision, especially in documents with few or no gold relations.

Our approach was hybrid and iterative. We first built an automatic candidate-generation pipeline from the training and development annotations and the EPOP document text. We then used small validation submissions to identify fragile document clusters and calibrate exact surface forms. This note focuses on the final selected runs, the system components used to build them, and the final private leaderboard result.

CLEF 2026 Working Notes, 21 – 24 September 2026, Jena, Germany
*Corresponding author.
s-shintani@fsc.chubu.ac.jp (S. A. Shintani)
https://sites.google.com/view/shintani-lab/ (S. A. Shintani)
0000-0002-1084-2549 (S. A. Shintani)

Table 1
Development-set observations used to guide precision repair.

Item	Value
Development documents	55
Empty gold-KG documents	5
Mean gold edges per development document	11.93
Median gold edges per development document	8
Main high-risk predicate	`Located_in`
Main recall risk	Alias and exact-span mismatch

2. Task and data

The task uses the EPOP corpus, described by the PestCLEF task page as 247 web documents focusing on monitored pests [3]. The corpus is released as a plant-health web-document resource [4]. The annotation dataset is described as a training and development dataset for information extraction in plant epidemiomonitoring and contains named entities, normalizations, and binary and ternary relationships referring to character positions in the documents [5]. In our local files, the split contained 110 training documents, 55 development documents, and 82 test documents. The test annotations were not available during the competition. The Kaggle page provided the competition interface, the data download route, and the public/private leaderboard mechanism [6].

The submission format is a CSV file with one row per test document and a JSON list of triples for the predicted knowledge graph. Each triple contains `predicate`, `subject`, and `object` fields. The schema used in our code constrained each predicate by coarse entity types: `Affects(Disease, Plant)`, `Causes(Pest, Disease)`, `Dispersed_by(Disease or Pest, Dissemination_pathway)`, `Found_on(Pest or Vector, Plant or Dissemination_pathway)`, `Located_in(Object, Location)`, `Occurs_on(Object, Date)`, and `Transmits(Vector, Disease or Pest)`.

Although the official task uses an F-score metric [3], our internal development emphasized document-level F1 behavior and per-document error analysis. This made the final system conservative: a small number of false positives in a low-edge document could remove the benefit of many plausible but unsupported additions elsewhere.

3. Development-set audit

Before constructing final submissions, we audited the development annotations. We clustered entity mentions through identity coreference links and shared type/normalization values, then projected annotated relations to concept-level graph edges. This reconstruction reproduced the development gold knowledge graphs, which gave us confidence that graph construction could be deterministic once concept clusters were known.

The development audit highlighted three practical risks. First, `Located_in` dominated the graph, but locations appeared at several levels of granularity, including country, region, province, city, and locality. Second, alias-rich pest and disease clusters meant that the same biological entity could be represented by several surface strings in the expected graph. Third, several development documents had empty graphs, so a system that created edges whenever entities were mentioned would have poor precision.

4. Automatic extraction pipeline

The first stage built an alias gazetteer from train and development concept clusters. For each concept, we collected surface strings and normalizations, then matched them in EPOP documents using boundary-aware regular expressions. Match features included count, title occurrence, lead-paragraph occurrence,

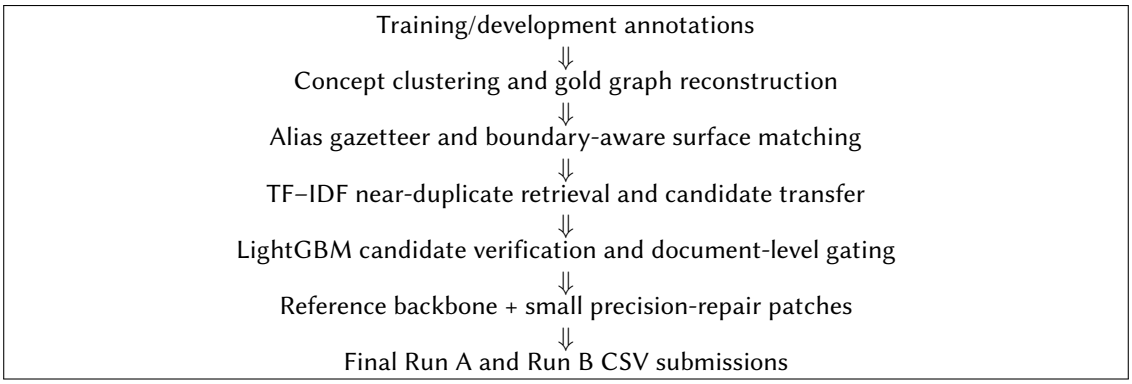

Figure 1: Block diagram of the final run-construction pipeline. The learned components were used mainly to propose and rank candidate triples; the final selected runs were conservative add-ons over a reference backbone.

and local sentence proximity between candidate subject and object strings.

Figure 1 summarizes the pipeline. The system first reconstructed graph supervision from train/development annotations, then generated schema-compatible candidates in test documents, and finally used candidate verification and precision repair to construct the two selected runs.

The second stage retrieved near-duplicate or topically similar labeled documents for each test document. We used TF-IDF representations over words and character n-grams, following the standard term-weighting idea of weighting terms by corpus frequency and document specificity [7]. The vectorization and related machine-learning utilities were implemented with scikit-learn [8]. Candidate edges were transferred from the most similar labeled documents only when compatible surface evidence existed in the target document. This was useful for repeated news families, including Xylella/Puglia reports and HLB/greening reports.

The third stage scored candidates using binary classifiers trained on candidate rows from the labeled train/development documents. We used gradient-boosted decision-tree verification, in particular LightGBM, because it is efficient for tabular features and widely used for high-performance gradient boosting [9]. A separate document-level gate estimated whether a document should contain any graph edges. Candidate features included predicate priors, source-document similarity, retrieval support count, mention proximity, entity type compatibility, and whether the evidence appeared in the title or lead paragraphs.

A five-fold out-of-fold diagnostic was used to check whether candidates could be separated from negatives. The large retraining run produced 1174 out-of-fold candidate rows, an out-of-fold candidate ROC-AUC of 0.7509, and a best diagnostic candidate-level F1 of 0.8227 at a threshold of 0.28. These diagnostics were useful for ranking candidates but were not sufficient for final submission selection by themselves. The main implementation settings are summarized in Table 2.

5. Precision repair and validation strategy

5.1. Reference, backbone, and patch terminology

We use three terms in the rest of the paper. The *initial reference* denotes the first non-empty high-precision submission used as an anchor in our iterative experiments. A *backbone* denotes the best validated submission available at a given stage before adding a new patch. Thus, the backbone was not an organizer-provided baseline; it was the cumulative submission produced by earlier extraction, cleanup, and validated add-on stages. A *patch* denotes a small set of manually inspected triples added to, or removed from, that backbone. For the final selected runs, the immediate pre-final backbone contained 929 triples. Run A added three triples (Table 4), and Run B added those three plus one Polignano hedge triple, yielding 932 and 933 triples respectively (Table 6).

The first large revision showed that replacing the entire graph with the learned extractor was harmful.

Table 2
Implementation details for the candidate-generation and verification pipeline. Model scores were used to prioritize candidate triples; the final selected submissions were conservative add-ons rather than a blind global-threshold output.

Component	Main settings and notes
Alias matching	Boundary-aware regular-expression matching using aliases from clustered train/development mentions and normalizations.
Document retrieval	Word and character TF–IDF similarity; candidate transfer was restricted to similar labeled documents with compatible target-document surfaces.
Candidate generation	Schema-constrained triples requiring subject/object surface evidence or strong alias evidence in the target document.
Verifier	LightGBM binary classifiers with 120 estimators, learning rate 0.05, 15 leaves, balanced class weights, and subsample/column sample 0.9 in the final ranking run.
Diagnostics	Five-fold out-of-fold candidate scoring used to rank candidates and identify high-risk predicates.
Final repair	Small document-level patches; each diagnostic submission usually changed one document or one edge group.

Pure learned submissions scored around 0.32 public F-score, whereas high-confidence add-ons over the backbone scored substantially higher. We therefore treated the learned model as a candidate source rather than the final predictor.

5.2. Local validation versus public-feedback repair

We separated three signals. First, train/development data were used for deterministic graph reconstruction, alias collection, retrieval, and candidate-verifier diagnostics. Second, small public-leaderboard submissions were used only as a coarse validation signal for a limited number of interpretable patches. Third, the final selected runs were chosen to keep the public score high while maintaining a conservative private-set hedge. This use of public feedback is not a reusable validation protocol; it is a competition-specific precision-repair stage and is discussed again as a limitation in Section 9.

Subsequent work used small validation submissions to test limited triple groups. Each submission changed as little as possible, ideally one document and one interpretable edge group. This allowed us to identify which document families were sensitive and which exact surface forms were accepted. Broad additions were avoided after repeated failures in `Located_in` and `Occurs_on` fan-out.

Several clusters drove most of the improvement. In document 105854, adding `Located_in(greening, regions of Limeira)` and `Located_in(greening, regions of Porto Ferreira)` produced a large gain, while HLB aliases, SP/MG aliases, Triangulo Mineiro, and date expansions were neutral or harmful. In Xylella/Puglia duplicate-family documents, exact locality surface forms were important: Monopoli helped, while several broader or related localities did not. In document 102433, the final major gain came from `Transmits` edges between Asian citrus psyllid and the CLas bacterium, rather than from L. capsica or generic psyllid/pepper-plant relations. Figure 2 and Table 3 should be read as a compact trace of these major diagnostic stages, not as a complete log of all submissions.

6. Final selected runs

The final two selected submissions both reached a public F-score of 0.45603 and a private F-score of 0.50952. The private leaderboard was calculated on approximately 82% of the test data after the competition ended, and these selected runs ranked first among teams on the Kaggle leaderboard [6].

Run A is the cleaner version, adding only the 100506 Monopoli exact-location span on top of the 102433 transmission backbone (Table 4). Run B adds the public-neutral Polignano exact-location span as a private-set hedge (Table 5). The two final runs differ by one edge only. This was intentional: the

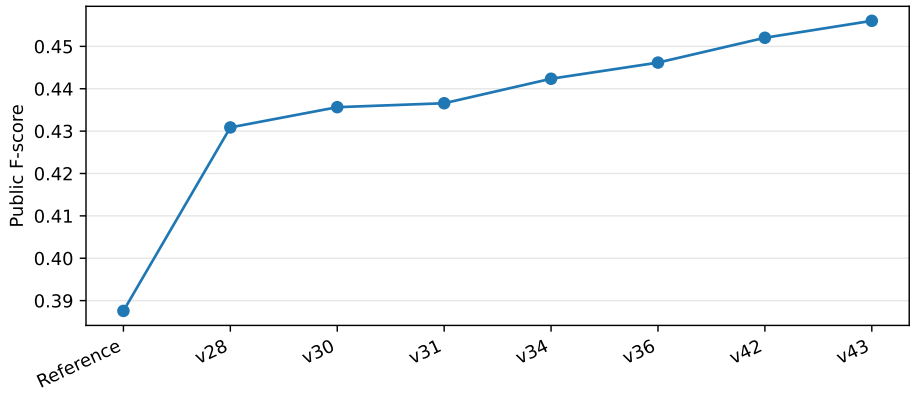

Figure 2: Public score progression after major refinement stages. Scores are leaderboard observations, not cross-validation estimates.

Table 3
Public score progression of the selected backbone.

Stage	Public F-score	Main change
Initial reference (anchor)	0.38757	First high-precision anchor submission
HLB precision repair	0.43087	Deleted a false-positive Porto location
Balanced add-on stage	0.43565	High-confidence additions over the then-current backbone
Host-recall refinement	0.43658	Small `Found_on` improvement
Xylella/Puglia refinement	0.44236	Exact locality/host-span repair
HLB exact-span transfer	0.44617	Added Limeira and Porto Ferreira region spans
CLas transmission relation	0.45203	Added Asian citrus psyllid to CLas relations
Final selected runs	0.45603	Added exact Monopoli location to the 102433 transmission backbone

Table 4
Final run A additions relative to the selected backbone.

Document	Predicate	Subject	Object
102433	`Transmits`	Asian citrus psyllid	Candidatus Liberibacter asiaticus
102433	`Transmits`	Asian citrus psyllid	CLas
100506	`Located_in`	Xylella	countryside of Monopoli

Table 5
Additional hedge edge in final Run B. Run B equals Run A plus this single public-neutral location edge.

Document	Predicate	Subject	Object
100506	`Located_in`	Xylella	countryside of Polignano

clean run reduces false-positive risk, while the hedge run keeps a plausible exact-span location that did not hurt public performance. Both runs keep the core 102433 `Transmits` relation that produced the largest late-stage public improvement.

7. Final private leaderboard and post-hoc observations

The final private leaderboard placed the team first with a private F-score of 0.50952. The second-place team scored 0.45962, and the third-place team scored 0.43992. The final selected submissions were not the highest-scoring exploratory submissions in every post-hoc view, but they were the official selected runs before the deadline and therefore determined the team result.

Table 6
Final selected submission statistics.

Run	Public F	Private F	Rows	Triples	Empty graphs
Run A: Monopoli only	0.45603	0.50952	82	932	8
Run B: Monopoli + Polignano	0.45603	0.50952	82	933	8

The final private-score reveal also showed that the public and private splits were not always aligned. Some exploratory runs had weaker public scores but strong private scores. They were not selected before the deadline and therefore did not determine the official selected-run result. This observation is nevertheless important: future systems should explicitly balance public gain, conservative precision repair, and private-diversity hedging.

8. Reproducibility

Code availability. The source code and reproducibility materials used for candidate generation, selected-run construction, and submission validation are available at https://github.com/cSAS3/pestclef2026-sas3. The archived release used for this working note is available at https://github.com/cSAS3/pestclef2026-sas3/releases/tag/v1.0.0. The repository contains source code, final selected run CSV files, patch specifications, validation scripts, and validation reports. It intentionally does not redistribute the official PestCLEF/EPOP raw data, train/development/test JSON files, or EPOP document texts; users should obtain the official task data through the official CLEF/LifeCLEF/PestCLEF data channels.

A supplementary archive named `PestCLEF2026_SAS3_Reproducibility_Package.zip` was also prepared for the organizers during revision. The archive and the public repository are intended for inspection and contain: (i) a manifest and README, (ii) final-run patch specifications for Run A and Run B, (iii) a validation script for submitted CSV files, (iv) a construction script showing how the final patches are applied to a backbone CSV, (v) implementation-setting summaries for the retrieval and LightGBM candidate-ranking stages, and (vi) the final run-validation report. The raw PestCLEF corpus, EPOP documents, and released train/development/test JSON files are not redistributed; they must be obtained through the official CLEF/LifeCLEF/PestCLEF data channels and used under the competition data-use terms.

The immediate pre-final backbone is defined as the best validated submission before the document 102433 and 100506 patches were applied. It contained 929 triples. The supplementary construction script takes such a backbone CSV as input and applies the explicit patch specification in Table 4 and Table 5 to reproduce the two final selected runs. This does not reproduce all exploratory leaderboard submissions, but it makes the final selected run construction traceable.

The final validation checks were: 82 rows in each selected submission; columns `doc_id` and `knowledge_graph`; valid JSON in every `knowledge_graph` field; each predicted triple is a JSON object with string fields `predicate`, `subject`, and `object`; and no exact duplicate triples within any document. The two final selected submissions had 932 and 933 triples respectively (Table 6).

9. Limitations

The main limitation is public-split overfitting and reduced methodological generality of the final precision-repair stage. The public leaderboard covered only a fraction of the test set, and the private-score reveal showed that the public split was not always predictive of the private split. We mitigated this by avoiding broad high-recall edits and by selecting two closely related final runs with identical public scores, but a stronger private-robust selection strategy would have been preferable.

A second limitation is that the final stage involved manual error analysis and public-score feedback. This helped identify exact surface forms, but it is less scalable than a fully automatic system. In a

deployment setting, we would replace this stage with a stronger verifier or a calibrated generative model constrained by the schema.

Finally, the system did not fully exploit long-context neural relation extraction. The candidate model used surface matching, retrieval, and gradient-boosted verification. This was efficient and interpretable but likely missed cross-sentence relations requiring deeper semantic interpretation.

10. Conclusion

Our PestCLEF 2026 system combined retrieval-guided candidate extraction, schema-constrained verification, and conservative precision repair. The final selected runs reached a public F-score of 0.45603 and a private F-score of 0.50952, ranking first on the final private leaderboard. The key technical lesson was that broad recall was less effective than exact document-level surface selection, especially for `Located_in` and `Transmits` relations in sensitive document clusters. The private-score reveal also showed that private robustness can diverge sharply from public optimization, motivating future selection strategies that explicitly balance public gain, precision repair, and private-diversity hedging.

Acknowledgments

The author thanks the CLEF, LifeCLEF, and PestCLEF organizers for preparing the task and evaluation infrastructure.

Declaration on Generative AI

During the preparation of this work, the author used OpenAI ChatGPT for text proofreading assistance, Japanese-English translation support, document organization, and assistance in drafting auxiliary scripts and reports. This declaration follows the CEUR-WS policy on AI-assisting tools [10].

References

[1] L. Picek, L. Adam, S. Kahl, R. Bossy, L. Chrobak, H. Goëau, K. Papafitsoros, H. Klinck, W.-P. Vellinga, R. Planqué, T. Denton, K. Barnard, C. Nédellec, L. Deléger, M. Courtin, G. Martellucci, I. Moummad, F. Vinatier, P. Bonnet, A. Joly, Overview of LifeCLEF 2026: Ai challenges for biodiversity understanding and ecosystem management, in: International Conference of the Cross-Language Evaluation Forum for European Languages (CLEF), Springer, 2026.

[2] R. Bossy, M. Courtin, L. Deléger, X. Yao, C. Nédellec, Overview of PestCLEF 2026: Knowledge graph extraction from plant health literature, in: Working Notes of CLEF 2026 - Conference and Labs of the Evaluation Forum, 2026.

[3] ImageCLEF/LifeCLEF, Pestclef2026: Knowledge graph extraction on plant pests from web documents, 2026. URL: https://www.imageclef.org/PestCLEF2026, accessed 7 May 2026.

[4] Plateforme ESV, Epop corpus: corpus of documents in plant health, 2025. URL: https://doi.org/10.57745/YKSEPY. doi:10.57745/YKSEPY.

[5] MaIAGE, Plateforme ESV, Training and development dataset for information extraction in plant epidemiomonitoring, 2025. URL: https://doi.org/10.57745/ZDNOGF. doi:10.57745/ZDNOGF.

[6] Kaggle, Pestclef @ lifeclef 2026 competition page, 2026. URL: https://www.kaggle.com/competitions/pest-clef-2026, accessed 7 May 2026.

[7] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, Information Processing and Management 24 (1988) 513–523. doi:10.1016/0306-4573(88)90021-0.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay,

Scikit-learn: Machine learning in python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[9] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, in: Advances in Neural Information Processing Systems 30, 2017.

[10] CEUR-WS, Policy on ai-assisting tools, 2024. URL: https://ceur-ws.org/GenAI/Policy.html, accessed 7 May 2026.