BioNLP412: Knowledge Graph Extraction on Plant Pests from Web Documents

PestCLEF at CLEF 2026

Mihai Radu Radulescu[1], Tudor Gabriel Arabagiu[1]

[1]University of Bucharest, Faculty of Mathematics and Computer Science, Str. Academiei nr. 14, Bucharest, Romania

Abstract

Our paper explores several machine learning methods for extracting knowledge graphs from plant-health web documents. We work with the EPOP corpus, a collection of web pages annotated with typed entities (pests, plants, diseases and others) and the relations between them, and frame the task as document-level relation extraction. We first set a baseline using a classic logistic-regression model, and then explore several approaches to surpassing it, built around the ModernBERT transformer. We achieve a top result of 0.4266 macro-averaged F1 on the private test set using a two-stage neural pipeline—one stage detecting entity mentions, the other classifying the relations between them—whose random-seed variance we reduce by combining five independently seeded runs into a single agreement ensemble.

Keywords

PestCLEF, LifeCLEF, Knowledge graph extraction, Document-level relation extraction, Named entity recognition, Information extraction, Knowledge graph, Plant health, ModernBERT, Bio-Medical NLP

1. Introduction

Plant pests, and the crop diseases they spread, are a serious and growing threat to global food security and agricultural economies. Knowledge about pest outbreaks—the pests involved, the crops they affect, where they occur and how they are transmitted—is scattered across a continuous stream of news articles, technical reports and institutional notices, and is expressed in inconsistent vocabularies, which makes large-scale epidemiological monitoring difficult. This paper explores several machine learning approaches for automatically extracting this knowledge from plant-health web documents, framed as document-level knowledge graph extraction: given a document, we detect the relevant entities (pests, plants, diseases, vectors, locations and dates) and the typed relations that hold between them. We carried out this work as part of the PestCLEF 2026 shared task [1], held within the LifeCLEF evaluation lab [2] at the CLEF 2026 conference.

Our results show that, of several approaches considered, the best performing one is an ensemble of five independently seeded runs of a two-stage ModernBERT pipeline [3], achieving a macro-averaged F1 score of 0.4266 on the private test set. Our team's official ranking, however, was determined by a different, less strong submission, as we discuss in Section 4.

2. Dataset

The shared task is based on the EPOP corpus (Epidemiomonitoring of Plants), a collection of plant-health web documents released for PestCLEF 2026 [1] and described in detail by Yao et al. [4]. The corpus was collected and annotated by a team from INRAE and Paris-Saclay University. Its documents are real-world web pages —news articles, technical reports, blog posts and institutional notices—harvested by the French epidemiomonitoring platform, which tracks twenty quarantine plant pests of interest to Europe. Although the original pages were written in several languages, every text in the released corpus

CLEF 2026 Working Notes, 21 – 24 September 2026, Jena, Germany

mihai-radu.radulescu@s.unibuc.ro (M. R. Radulescu); tudor-gabriel.arabagiu@s.unibuc.ro (T. G. Arabagiu)

https://github.com/ramihai/PestCLEF2026 (M. R. Radulescu)

Entity type	Description
Pest	Pathogen organism, usually a microorganism named by its Latin binomial.
Plant	Host crop, given by scientific or vernacular name (citrus, grape, wheat).
Disease	A disease caused by a pest; names are frequently abbreviated.
Vector	Organism, mostly an insect, that carries a pest to new crops.
Dissemination_pathway	Means of pest transport, e.g. plant parts or human artefacts.
Location	Geographical place at any scale, from continent to municipality.
Date	Date, period or season of an observation.

Table 1
The seven entity types of the EPOP annotation schema.

Relation	Subject type(s)	Object type(s)
Affects	Disease	Plant
Causes	Pest	Disease
Dispersed_by	Disease, Pest	Dissemination_pathway
Found_on	Pest, Vector	Plant, Dissemination_pathway
Located_in	Disease, Pest, Plant, Vector	Location
Occurs_on	Disease, Pest, Plant, Vector	Date
Transmits	Vector	Disease, Pest

Table 2
The seven binary relations and the (subject type, object type) pairs admitted by the schema.

has been translated into English, so that participants can build language-independent pipelines. Each document is also provided with its original HTML layout (tags such as `h1`, `p` and `li`, with character offsets), which can be used to segment a document into semantic sections.

The corpus was annotated by thirty plant-health and NLP experts, following a double-blind procedure with automatic consistency checks. The annotation has two layers. The first consists of text-bound annotations placed directly on the running text: named entities with character offsets, entity normalizations, binary relations, and n-ary events. From this layer, a gold knowledge graph is automatically derived for each document. Coreference sets are also annotated, so that the several mentions of one real-world entity—for instance a scientific name, a vernacular name and an acronym—collapse into a single graph node. Two properties of this scheme are important for our work. First, every gold edge carries a list of accepted surface forms for its subject and its object (on average 2.92 forms per subject and 1.57 per object), so a predicted edge is correct as long as its arguments fall within this equivalence class. Second, the test documents are released without any text-bound annotation: the entities are not given at inference time and must be detected by the system itself.

The annotation schema defines seven entity types and seven binary relation types. The entity types and their descriptions are given in Table 1. The relation types, together with the (subject type, object type) pairs they admit, are given in Table 2. The schema is strict: a predicted edge whose subject or object type violates these constraints is counted as wrong even when its surface forms look plausible. This argument-type restriction is one of the main difficulties of the task, since semantically close types—most notably Pest and Disease—are easily confused.

The corpus contains 247 documents, split into 110 for training, 55 for development and 82 for the hidden test set; the split was designed to preserve the distribution of entities and relations across the three subsets. Document length varies widely, from 73 to 29,722 characters, with a median of roughly 2,400 characters and about 4.5% of documents exceeding 8,000 characters (Figure 3). The training and development splits together contain 5,097 named entities, and their distribution is heavily skewed: Location, Pest and Plant alone account for more than 74% of all mentions, while Vector accounts for barely 2% (Figure 2). The same imbalance appears at the relation level. Across the two splits we count

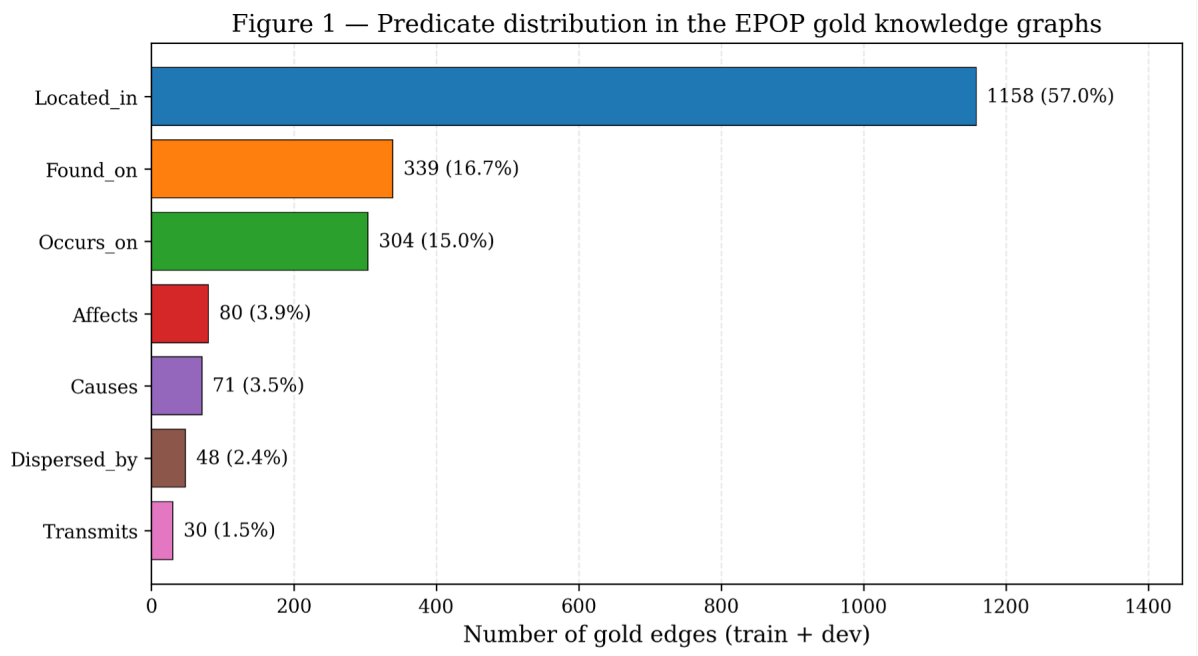

Figure 1: Distribution of predicates in the EPOP gold knowledge graphs (train and development splits). Located_in dominates with 57% of all edges, while Transmits is the rarest at 1.5%.

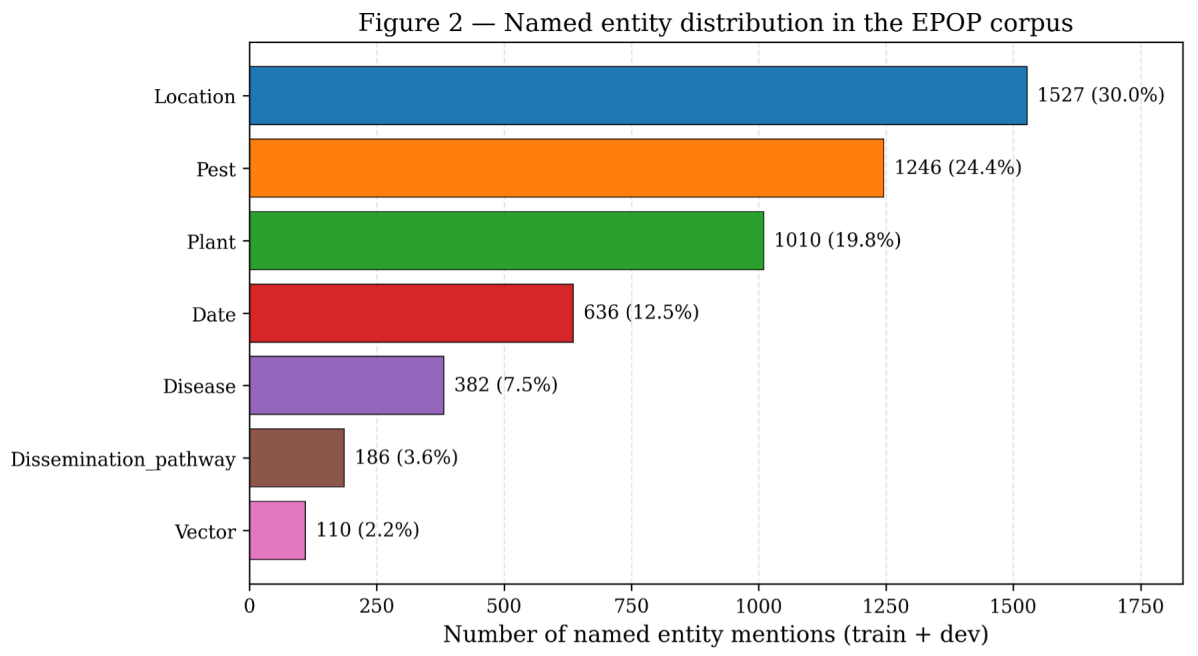

Figure 2: Distribution of named entity types in the EPOP corpus (train and development splits). Location, Pest and Plant together account for over 74% of all entity mentions.

2,030 gold edges (Figure 1): Located_in alone makes up 57% of them, whereas Transmits represents only 1.5%. A document carries about 12.5 gold edges on average (Figure 4), which is sparse next to the roughly 30 entities it contains—emitting an edge for every schema-valid entity pair therefore yields far more candidates than there are true edges, which motivates the candidate-pair pruning we describe in Section 3.

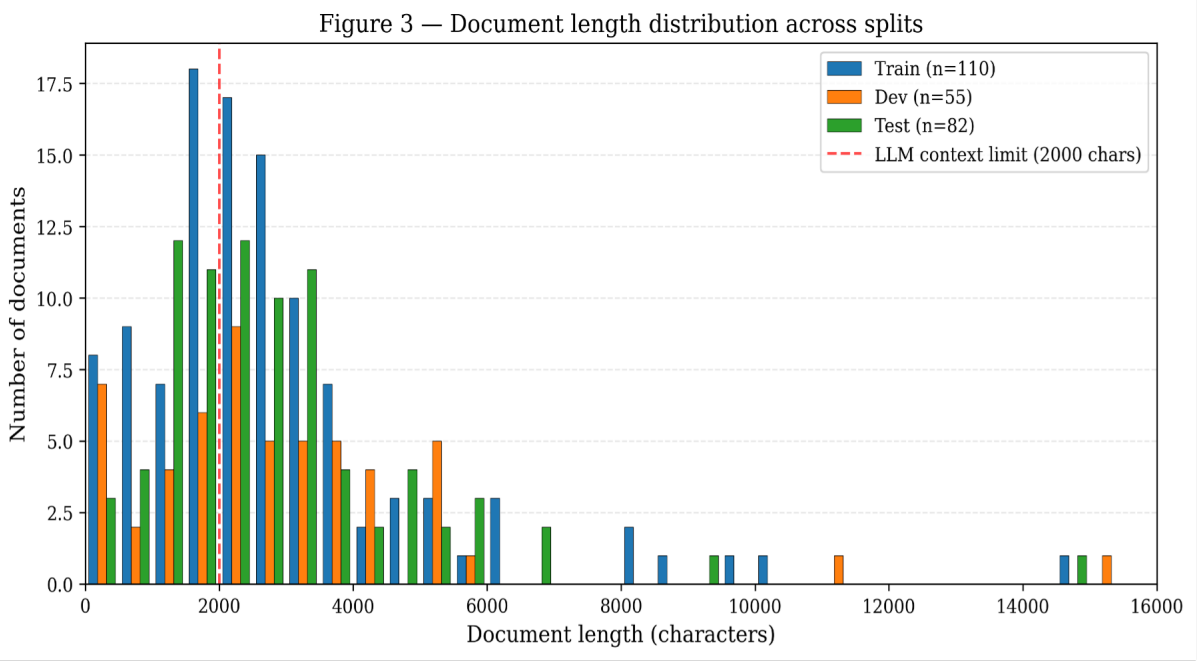

Figure 3: Distribution of document length, in characters, across the train, development and test splits.

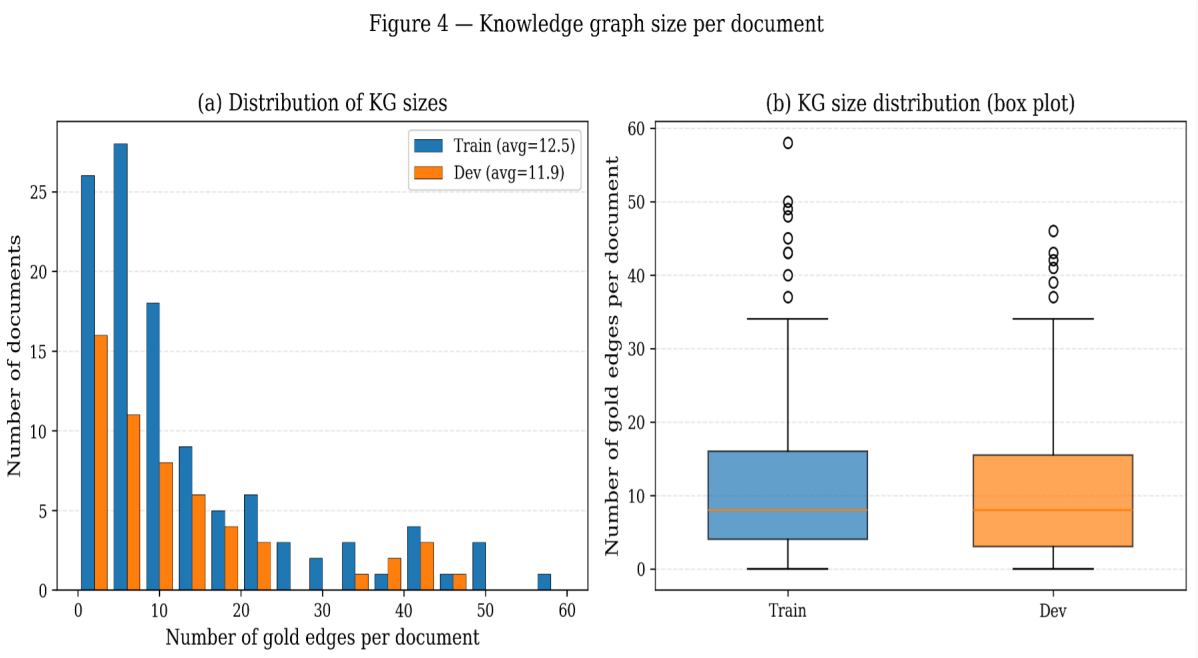

Figure 4: Distribution of knowledge graph size (gold edges per document) across the train and development splits.

3. Method

3.1. Task and evaluation metric

The shared task is a document-level relation extraction problem: given a document, a system must output the edges of its knowledge graph, each edge being a triple (subject, predicate, object). Entity offsets and entity normalization are not required—only the surface forms of the two arguments are evaluated. Submissions are scored with the macro-averaged F1 score. For each document, a predicted edge is counted as correct when its predicate matches a gold predicate and its subject and object surface

Feature group	Description
Entity types	Subject type, object type and the ordered type pair.
Distance	Bucketed sentence gap, layout-block gap and character offsets between the mentions, with same-sentence, same-layout and textual-order flags.
Mention counts	Bucketed number of mentions of each entity and of their product.
Entity names	Casefolded canonical form of the subject and the object, and their first tokens.
Context words	Per-pair top-k tokens of a ± 100-character window and of the inter-mention span.
Relation triggers	One flag per relation, set when a hand-curated keyword of that relation occurs near the pair.

Table 3
Feature groups of the logistic-regression baseline. All features are binary presence indicators.

forms each fall within the lists of accepted forms of that gold edge; precision, recall and F1 are computed per document and then averaged over the test set. Because the gold edges admit several accepted surface forms, the metric tolerates minor lexical variation, but it remains sensitive to precision: a system that emits many more edges than the gold contains is penalised even when its recall is perfect.

3.2. Logistic-regression baseline

Since the shared task does not provide a baseline, we set our own with a logistic-regression model. Relation extraction is more structured than plain text classification, so the baseline does not operate on raw documents: it reuses the candidate-pair representation of the task. For every document we enumerate ordered pairs of canonical entities and keep only those whose entity types are admitted by some relation and whose mentions lie within three sentences or six layout blocks of one another. Each surviving pair is then classified by seven independent binary logistic-regression classifiers, one per relation, so that a pair may be assigned zero, one or several relations; schema-invalid and duplicate triples are removed afterwards. When the classifier for a relation is trained, only pairs whose entity types are compatible with that relation are used, so the model is never asked to learn constraints that the schema already encodes.

Each pair is described by a sparse, hand-engineered feature vector—in contrast to the contextual encoder of the neural pipeline, and, notably, without any TF-IDF or learned text representation. The feature groups are summarised in Table 3; they combine entity-type indicators, distance and layout buckets, mention-count buckets, casefolded entity names, per-pair bag-of-words tokens drawn from the immediate context, and a hand-curated keyword trigger for each relation. All feature values are binary presence indicators, and the dictionaries are vectorised with scikit-learn's `DictVectorizer` [5].

Each per-relation classifier is a `LogisticRegression` estimator with L2 regularisation ($C = 1.0$), the `liblinear` solver, and balanced class weights to compensate for the low positive rate of most relations. After fitting, its decision threshold is re-tuned by an F1-maximising grid search, using the same calibration routine as the neural pipeline. We evaluate the baseline in two settings: an oracle setting in which it is given the gold entities, which isolates the quality of relation classification, and an end-to-end setting in which entities are first detected by a simple string-matching lexicon built from the training annotations. The gap between the two settings (Section 4) measures the cost of mention detection and motivates the neural detector of our main system.

3.3. Two-stage neural pipeline

Our main system is a two-stage neural pipeline built on ModernBERT [3], a recent long-context bidirectional encoder. The two stages—mention detection and relation classification—are illustrated in Figure 5

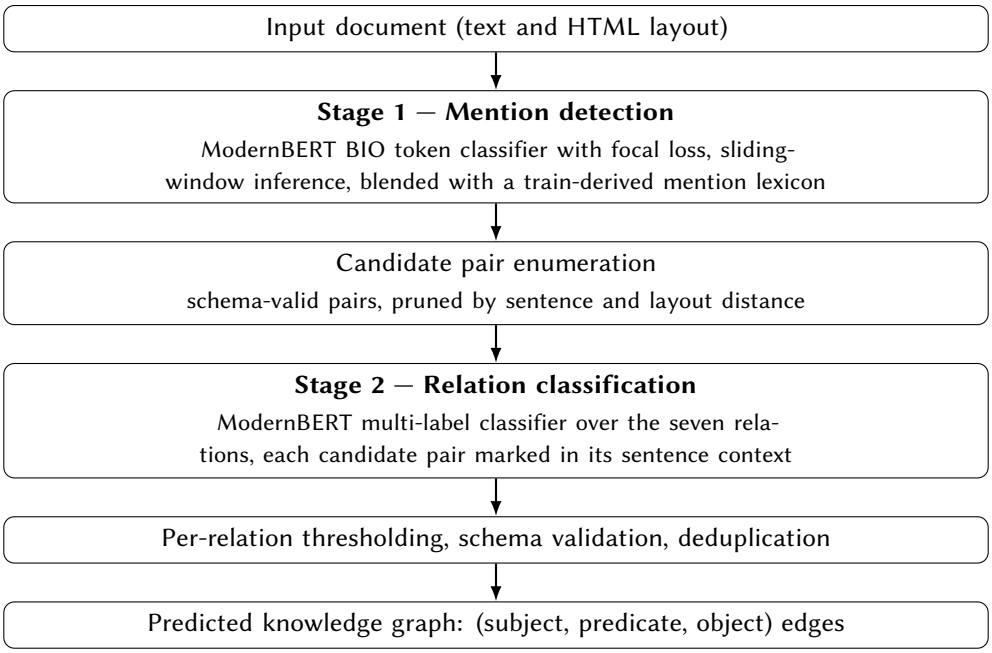

Figure 5: The two-stage extraction pipeline. Both stages use the same `ModernBERT-base` encoder.

and share the same `ModernBERT-base` encoder (149M parameters). The training hyperparameters are summarised in Table 4.

Stage 1: mention detection. The first stage detects typed entity mentions as a token-classification problem, tagging each token with one of 15 BIO labels (an outside label plus a begin and an inside label for each of the seven entity types). Because documents are frequently longer than the encoder input we use here, they are processed with a sliding window of 1024 tokens and a stride of 256, and the per-window predictions are merged into document-level spans. The token classifier is trained with focal loss [6] ($\gamma = 2.0$) with capped class weighting, which down-weights the many easy outside tokens and concentrates the gradient on the rarer entity boundaries. The neural predictions are then blended with a lexicon of entity surface forms collected from the training annotations: when the lexicon and the classifier cover the same span, the higher-confidence label is kept. Finally, per-type confidence thresholds tuned on the development set, together with a small set of hand-curated denylists, remove low-quality spans.

Stage 2: relation classification. The second stage decides, for each candidate pair of detected mentions, which relations hold between them. Candidate pairs are enumerated within each document and pruned to those that satisfy the schema and whose mentions lie within a short sentence- or layout-block distance of each other; this pruning is important because, as noted in Section 2, the number of schema-valid pairs greatly exceeds the number of true edges. Each surviving pair is encoded by marking its subject and object mentions with special tokens inside a window of ± 2 sentences of context, and a multi-label head with one sigmoid output per relation predicts which of the seven relations apply. The relation classifier is trained with a binary cross-entropy loss in which each relation is weighted by its inverse frequency, and it is trained on a mixture of gold and predicted mentions so that it is exposed at training time to the noisy spans it will encounter at inference. Per-relation decision thresholds are then calibrated on the development set, and a final pass discards schema-violating triples and removes duplicates.

Training and threshold variants. Both stages are fine-tuned for three epochs with AdamW, using the hyperparameters in Table 4. We refer to a single training run of this pipeline, with its development-

Hyperparameter	Value
Shared	
Encoder	`ModernBERT-base` (149M)
Optimizer	AdamW, weight decay 0.01
Schedule	linear, warmup ratio 0.06
Learning rate	2×10^{-5}
Epochs	3 (early stopping, patience 2)
Batch size	4
Stage 1 — mention detection	
Max sequence length	1024
Sliding-window stride	256
Loss	focal loss, $\gamma = 2.0$
Class-weight cap	$12.0\times$
Lexicon confidence	0.55
Stage 2 — relation classification	
Max sequence length	768
Context window	± 2 sentences
Loss	BCE, inverse-frequency weighted
Candidate pruning	≤ 3 sent. / ≤ 6 layout blocks
Gold/predicted mix	50 / 50

Table 4
Training hyperparameters of the two-stage pipeline. The base and threshold-tuned configurations differ only in the per-relation decision thresholds. Complete reproducibility settings are given in Appendix A.

calibrated thresholds, as the *base configuration* (`v14` in our code release). We additionally produce a *threshold-tuned configuration* (`v18`), identical in every respect except that the per-relation decision thresholds are re-selected by a wider grid search on the development set; this trades precision for recall on some relations, most visibly `Causes`. These two configurations, together with the five-seed ensemble described next, are the building blocks of our submissions; their commit hashes and configuration file paths are given in Appendix A.

3.4. Seed ensembling and prediction filtering

A single training run of the pipeline is sensitive to random initialisation: in our experiments the base configuration varied noticeably between seeds, and a run that scored well on the public leaderboard did not necessarily generalise to the private test set. Our two best submissions are therefore both ensembles, built to reduce this variance.

Leading model: five-seed agreement ensemble. We train the base configuration five times, with different random seeds, obtaining five independent pipelines. At inference, each pipeline scores its own candidate pairs, and we aggregate the results by averaging, for every distinct (document, subject, object) key, the per-relation sigmoid scores of the seeds that produced that key. Because mention detection is itself stochastic, the five runs do not propose the same candidate pairs: as Table 5 shows, almost half of all candidate pairs are proposed by a single seed, and only 31% by all five. Such low-agreement pairs are largely stochastic false positives, so we keep only the pairs proposed by at least four of the five seeds—retaining roughly a third of the candidates—and apply the base configuration's decision thresholds directly, without any further calibration on the development set. We found that re-calibrating the thresholds at this stage overfitted the development set and hurt the private-leaderboard score. This submission is our best on the official (private) test set.

Intersection of the two configurations. Our second submission is a consensus filter across the two operating points of the same trained pipeline: we keep only the triples that appear in both the

Seeds proposing the pair	Pairs	Share
1	3,062	47%
2	921	14%
3	337	5%
4	208	3%
5	2,044	31%
Total	6,572	100%

Table 5
Number of test candidate pairs proposed by exactly k of the five seeds. The five-seed agreement ensemble retains only pairs with $k \geq 4$.

System	Public	Private
Logistic regression (baseline)	0.4034	0.2602
Mistral 7B (LLM)	0.2447	0.1783
Base configuration (v14)	**0.4430**	0.3472
Threshold-tuned configuration (v18)	0.2969	0.3708
Intersection (v14 ∩ v18)	0.4411	0.3850
Five-seed agreement ensemble	0.4405	**0.4266**

Table 6
Macro-averaged F1 on the public and private PestCLEF 2026 Kaggle leaderboards; the private leaderboard determines the final ranking.

base configuration's and the threshold-tuned configuration's submissions. No additional training is involved—v14 and v18 differ only in the per-relation decision thresholds applied to the same relation-classifier outputs—so the intersection is best understood not as an ensemble of two models, but as a high-precision prediction set built from two thresholdings of one model. It obtained our best score on the public test set, although it generalised less well to the private test set than the five-seed ensemble (Section 4).

3.5. LLM-based extraction

For comparison, we also report a system developed in parallel by a co-author for the same shared task, which approaches the problem with a generative large language model rather than a fine-tuned encoder. It uses Mistral 7B Instruct [7], loaded in 4-bit quantization [8] so that it fits on a single 16 GB GPU. The model is prompted in a chat format with an instruction that lists the seven predicates and their argument-type constraints, three few-shot examples drawn from the training set, and the first 2,000 characters of the target document; as Figure 3 shows, this truncates a non-trivial fraction of the corpus, so the LLM's reported score should be read as a lower bound rather than a like-for-like comparison with the neural pipeline. Its output is parsed as a JSON array of edges, and triples with an invalid predicate or a schema-violating argument pair are discarded. Decoding is greedy. Unlike our two-stage pipeline, this system performs entity detection and relation extraction jointly, in a single generation step.

4. Results

Table 6 reports the macro-averaged F1 score of every system on the PestCLEF 2026 Kaggle leaderboard, and Figure 6 visualises the same scores. The leaderboard is split into a public part, computed during the evaluation phase on a small slice of the test set (about 18% of the test documents), and a private part, computed on the remaining majority and used for the final ranking. As the table and figure show, the two parts diverge substantially for several systems, and this divergence is the recurring theme of this section.

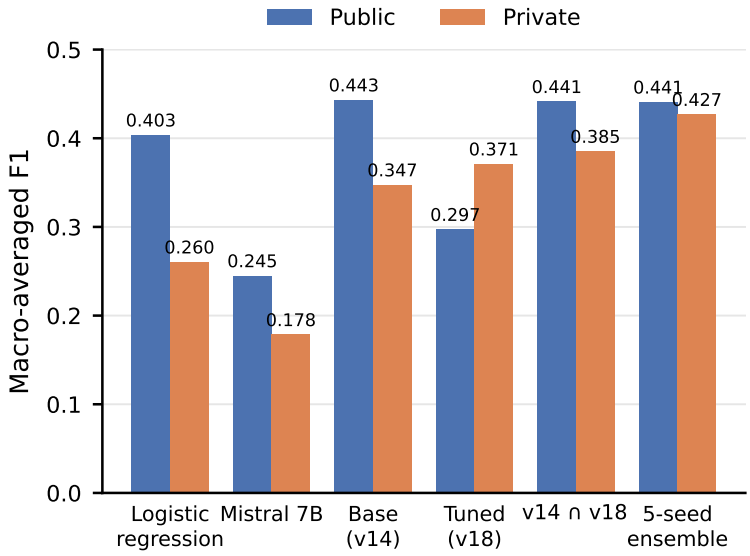

Figure 6: Macro-averaged F1 of each system on the public and private PestCLEF 2026 Kaggle leaderboards. Several systems score similarly on the public slice, but only the five-seed ensemble retains a comparable score on the private leaderboard that decides the final ranking.

The logistic-regression baseline reaches 0.4034 on the public leaderboard but only 0.2602 on the private one. Evaluated on the development set with gold entities it obtains an F1 of 0.370, which falls to 0.212 when entities are instead detected by the string-matching lexicon;[1] this gap shows that, for the baseline, mention detection rather than relation classification is the dominant source of error, and it is the reason our main system replaces the lexicon with a neural mention detector. The LLM-based comparison system, Mistral 7B with an abstract three-shot prompt, scores 0.2447 on the public leaderboard and 0.1783 on the private one—the lowest of all the systems we consider, on both. Even the logistic-regression baseline outperforms it, and the two-stage pipeline does so by a wide margin.

A single training run of the two-stage pipeline—the base configuration—scores 0.4430 on the public leaderboard, the highest public score of any system we built, but only 0.3472 on the private one. This drop of almost 0.10 is the clearest symptom of the leaderboard divergence: a configuration can look strong on the small public slice for reasons that do not carry over to the larger private set—here, the random initialisation of a single run. The threshold-tuned configuration behaves in the opposite way: re-selecting the per-relation decision thresholds on the development set lowers the public score to 0.2969 but raises the private score to 0.3708.

Our two best submissions are both ensembles. The intersection of the base and threshold-tuned configurations keeps only the edges predicted by both, and obtains our highest public score, 0.4411; on the private leaderboard, however, it reaches only 0.3850. The five-seed agreement ensemble matches it almost exactly on the public leaderboard (0.4405) but reaches 0.4266 on the private one—an improvement of 0.080 over the single-seed base configuration, and our best result by a clear margin. On the private leaderboard, this score of 0.4266 exceeds the 4th-place team's score of 0.4038. However, the five-seed ensemble was not one of the submissions we selected for the official ranking: applying the same public-leaderboard heuristic that we criticise above, we selected the base configuration and the intersection ensemble instead, whose best private score of 0.3850 placed us 5th in the final ranking. Our official rank is therefore itself a small instance of the phenomenon this section analyses: the public leaderboard, used as the selection signal, cost us one position by leaving our strongest system out of our final

[1] Development-set figures are macro-averaged over the seven relation types, whereas the official metric is macro-averaged over documents; the two are not directly comparable, so we report development scores only as a relative, within-system diagnostic.

Relation	Support	LR	Base (v14)	5-seed
`Located_in`	377	**0.239**	0.205	0.184
`Found_on`	106	0.298	**0.339**	0.326
`Occurs_on`	94	**0.195**	0.121	0.156
`Affects`	30	**0.340**	0.295	0.312
`Causes`	27	0.289	0.265	**0.328**
`Dispersed_by`	16	**0.125**	0.067	0.091
`Transmits`	5	0.000	0.000	0.000
Macro-F1		**0.212**	0.184	0.200

Table 7
Per-relation F1 on the 55-document development set (end-to-end predictions, each system using its own mention-detection component). *Support* is the number of gold instances of each relation. The Kaggle grader exposes only aggregate scores for the test set, so per-relation numbers are reported on the development set throughout. Best per row in bold.

submissions. Our team appears on the official leaderboard[2] as *BioNLP412*; the base configuration and the intersection ensemble were our two selected final submissions. The contrast between the two ensembles is instructive: the intersection attains precision by requiring two threshold settings to agree, which happens to suit the small public slice, whereas the five-seed ensemble also averages out the variance of the mention-detection stage, and it is this that generalises to the private test set.

So far the discussion has been at the aggregate level. Table 7 reports F1 per relation on the development set (the Kaggle grader exposes only aggregate scores for the test set). Two observations stand out. First, `Transmits`, the rarest relation with only 5 gold instances on dev and 25 in the training set, is completely unrecovered by both neural systems—a data-scarcity ceiling rather than a modeling limitation. Second, the five-seed ensemble improves over the base configuration on four of the seven relations (`Occurs_on`, `Affects`, `Causes`, and `Dispersed_by`), which are precisely the minority-support classes where seed averaging is expected to reduce variance.

5. Limitations and Future Work

Our development process was constrained by the evaluation setup. Model selection relied on a public leaderboard computed on roughly 18% of the test set and on a development set of only 55 documents. Neither proved a reliable guide: as Section 4 shows, the public leaderboard favoured configurations that did not generalise to the private test set, and re-calibrating decision thresholds on the small development set tended to overfit it. Our five-seed ensemble mitigates this instability but does not remove it, and a single training run of the pipeline remains noticeably sensitive to its random seed.

The pipeline itself has several structural limitations. Because it is staged, an entity that Stage 1 misses or mistypes cannot be recovered by Stage 2; the gap between the baseline's gold-entity and lexicon-entity scores (Section 4) is a direct measure of how costly this is. The relation classifier sees only a window of ± 2 sentences around a candidate pair, and candidate pairs are pruned by textual distance, so relations whose arguments are stated far apart in a document are systematically missed. As Table 7 shows, the rarest relation, `Transmits`, is not recovered at all by our neural systems, despite inverse-frequency loss weighting. Finally, all training was carried out on a single Apple M4 Pro using its MPS backend, which restricted us to the base size of ModernBERT, to three training epochs, and to a narrow hyperparameter search.

The private-leaderboard gain of 0.080 that our five-seed ensemble buys over the single-seed base configuration also costs roughly a $5\times$ training and inference budget, a trade-off appropriate for a leaderboard submission but unlikely to justify itself in a production deployment. A cheaper substitute—smaller ensembles (two or three seeds), or knowledge distillation of the ensemble into a single student

[2] https://www.kaggle.com/competitions/pest-clef-2026/leaderboard

model—would be a natural next step.

Several directions could address these limitations. The most immediate is text preprocessing, which the present system omits entirely: the pipeline was trained and run on the document text essentially as distributed, with no text normalisation or cleaning, in contrast to standard practice on noisy web and user-generated text. The EPOP documents are web pages that have been harvested and machine-translated into English, and they carry the artefacts one would expect—boilerplate and navigation fragments, irregular casing and whitespace, and inconsistent renderings of scientific names and abbreviations. A dedicated preprocessing stage—removing non-content blocks using the provided HTML layout, normalising whitespace and casing, and standardising entity surface forms—is therefore a natural and probably impactful next step. Surface-form standardisation is especially promising: because the evaluation matches each predicted argument against a list of accepted forms, cleaner and more canonical predictions could raise the score without any change to the models themselves.

A second direction is stronger handling of class imbalance in the relation classifier. We already use focal loss [6] for mention detection but only inverse-frequency weighting for relation classification; extending focal loss to the second stage, or adopting a robustness-oriented method such as Adversarial Weight Perturbation [9], could improve recall on the rare relations.

A third direction concerns the staged design itself. A joint or generative model that produces relation triples directly from text—in the spirit of seq2seq relation extractors such as REBEL [10]—would remove the error propagation between our two stages, at the cost of a more complex training setup. Independently of the model, a lightweight post-processing step that canonicalises predicted arguments using the coreference sets available in the training data could recover relations that are currently lost only to surface-form mismatch.

Finally, given more computational resources, larger encoders such as ModernBERT-large, longer training and a broader hyperparameter search are all worth revisiting, as is a larger or learned ensemble in place of the five-seed agreement vote used here.

The full code, configuration files, and per-seed artefacts of our submissions are released at https://github.com/ramihai/PestCLEF2026 under the MIT license. The commit hashes and configuration paths of the base configuration, the threshold-tuned configuration, and the five-seed ensemble are given in Appendix A.

Acknowledgments

Thanks to Ana-Sabina Uban, associate professor at the University of Bucharest, for her suggestion on approaching the shared task as a semester project in our Bio-NLP course, and to the PestCLEF 2026 organisers for their support and for providing the dataset and evaluation infrastructure.

Declaration on Generative AI

During the preparation of this work, the authors used Claude Opus 4.7 in order to: Grammar and spelling check, rephrasing for clarity improvement and unification of style. After using these tools/services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] 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.

[2] 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.
[3] B. Warner, A. Chaffin, B. Clavié, O. Weller, O. Hallström, S. Taghadouini, A. Gallagher, R. Biswas, F. Ladhak, T. Aarsen, N. Cooper, G. Adams, J. Howard, I. Poli, Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference, arXiv preprint arXiv:2412.13663 (2024). URL: https://arxiv.org/abs/2412.13663.
[4] X. Yao, C. Nédellec, J. Xia, R. Bossy, Consistency–accuracy correlation in hard-prompted LLMs for entity and relation extraction: Empirical findings from plant-health data, Genomics & Informatics 24 (2026) 3. doi:10.1186/s44342-025-00063-2.
[5] 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, É. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.
[6] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, arXiv preprint arXiv:1708.02002 (2017).
[7] A. Q. Jiang, A. Sablayrolles, A. Mensch, et al., Mistral 7B, arXiv preprint arXiv:2310.06825 (2023).
[8] T. Dettmers, A. Pagnoni, A. Holtzman, L. Zettlemoyer, QLoRA: Efficient finetuning of quantized LLMs, arXiv preprint arXiv:2305.14314 (2023).
[9] D. Wu, S.-T. Xia, Y. Wang, Adversarial weight perturbation helps robust generalization, arXiv preprint arXiv:2004.05884 (2020).
[10] P.-L. Huguet Cabot, R. Navigli, REBEL: Relation extraction by end-to-end language generation, in: Findings of the Association for Computational Linguistics: EMNLP 2021, Association for Computational Linguistics, Punta Cana, Dominican Republic, 2021, pp. 2370–2381. doi:10.18653/v1/2021.findings-emnlp.204.

A. Reproducibility details for the two-stage pipeline

This appendix records the technical settings needed to reproduce the two-stage ModernBERT pipeline of Section 3. Configuration files corresponding to the base configuration, the threshold-tuned configuration, and each of the five ensemble seeds are shipped in the code release; commit hashes are given at the bottom of the table.

Setting	Value
Encoder and training	
Encoder checkpoint	`answerdotai/ModernBERT-base` (149M parameters)
Frozen parameters	None
Framework versions	PyTorch 2.11.0, Transformers 5.5.0, Python 3.14.0
Device	Apple M4 Pro (MPS backend, float32)
Optimiser	AdamW, weight decay 0.01
LR schedule	Linear with warmup ratio 0.06
Epochs	3 for the base and threshold-tuned configurations; 8 for the ensemble seeds; early-stopping patience 2 in all cases, which halts training at 3–5 epochs in practice
Training time (mention)	$\sim$3 minutes per epoch
Training time (relation)	$\sim$65–70 minutes per epoch
Ensemble seeds	13, 42, 1337, 7, 2025
Stage 1 — Mention detection	
BIO labels	15 total (`O` plus `B-`/`I-` for each of the seven entity types)
Focal loss	$\gamma = 2.0$; sub-word continuations are masked with `ignore_index` $= -100$
Class-weight cap	Rare-class weight clamped to $12\times$ the mean; the `O` class capped at $1\times$
Sliding window	`max_length` $= 1024$ tokens, stride 256, padded to max length
Lexicon	Built from training-set surface forms; minimum alias length 3 characters; type conflicts resolved by majority vote; matched at inference by an Aho–Corasick automaton with word-boundary check
Lexicon confidence	Fixed floor of 0.55, raised to $\tau + 0.05$ when a per-type threshold τ is higher
Blending rule	For each $(\text{type}, \text{start}, \text{end})$ triple, the highest-confidence source wins
Denylists	Hand-curated per-type stop lists (32 entries total, on Date, Location, Vector, Dissemination_pathway)
Threshold tuning	Data-adaptive grid: unique dev-set span confidences, sub-sampled to 25 points; objective $0.75\times$ candidate recall $+0.25\times$ mention F1
Stage 2 — Relation classification	
Marker tokens	28 special tokens of the form <SUBJ_Type>, </SUBJ_Type>, <OBJ_Type>, </OBJ_Type>, added to the tokenizer and mean-resized in the embedding matrix
Context window	Sentences covering both mentions, extended by a 2-sentence radius on each side
Sentence segmentation	Regex `(?<=[.!?])\s+`; no abbreviation handling
Layout blocks	Read from the dataset's `layout` field; layout distance is the minimum index difference over all mention pairs
Candidate pruning	Schema-valid pairs with sentence gap ≤ 3 and layout gap ≤ 6; tightened further for `(*, Date)` and `(*, Location)` pairs
Candidate enumeration	All within-document ordered pairs; self-pairs excluded
Training data mix	Gold examples in full, plus seeded-random-sampled predicted pairs (up to 50%) from a first pass of the mention detector on the training set
Loss weighting	`BCEWithLogitsLoss` with $\text{pos\_weight}_r = (1 - p_r)/p_r$, where p_r is the positive rate of relation r
Threshold tuning	30 quantiles from the 60th to 99.5th percentile of the score distribution, plus five boundary anchors, per relation; F1-maximising within $[0.05, 0.95]$, narrowed for rare relations
Ensemble	
Aggregation key	(doc_id, subject_type, subject, object_type, object) from raw canonical forms; no case folding, punctuation stripping, or diacritic normalisation
Score combination	Arithmetic mean of per-seed sigmoid outputs, over the seeds that produced the key
Seed-count filter	Keys proposed by fewer than 4 of the 5 seeds are discarded
Thresholding	Per-relation defaults of the base configuration, applied inclusively; no development-set recalibration
Deduplication	Exact string match on $(\text{subject}, \text{predicate}, \text{object})$; first occurrence kept
Version identifiers	