Knowledge Graph Extraction from Agricultural Pest Documents: A BiomedBERT Pipeline with NER and Relation Extraction

PestCLEF at CLEF 2026

Hemeci Alin-Andrei[1]

[1]*University of Bucharest, Bucharest, Romania*

Abstract

This paper describes my system submitted to PestCLEF at LifeCLEF 2026 [2], a shared task requiring the automatic extraction of knowledge graphs from web documents about agricultural pests and plant diseases. Given a plain text document, the system must identify ecological relationships between entities such as pests, plants, diseases, vectors, locations, and dates, and output them as structured triplets. My approach is a two-stage pipeline: a Named Entity Recognition (NER) model that identifies entities and their types in the document, followed by a Relation Extraction (RE) model that classifies the relationship between each candidate pair of entities. Both models are fine-tuned from BiomedBERT[1], a biomedical transformer pre-trained on scientific full text, which provides familiarity with domain specific Latin nomenclature and scientific terminology. I demonstrate that fine-tuning the Relation Extraction model on entities detected by the Named Entity Recognition model is critical for consistency between training and inference, and that per class confidence thresholding improves precision. My final system achieves a private F1 score of 0.475 on the hidden test set, placing second in the competition under the team name *Hem*.

Keywords

knowledge graph extraction, named entity recognition, relation extraction, agricultural pest monitoring, BiomedBERT, PestCLEF 2026

1. Introduction

Agricultural epidemiology relies on monitoring the spread of pests and plant pathogens across regions and host species. Knowledge about these interactions is scattered across thousands of heterogeneous web documents written in varied vocabularies and styles. Automating the extraction of structured knowledge from such documents would enable large-scale research and early-warning systems for quarantine pests.

PestCLEF 2026 [**?**], part of the LifeCLEF lab at CLEF 2026 [2], formalises this challenge as a knowledge graph extraction task. Participants receive plain text web documents and must produce a set of typed relational triplets of the form (subject, predicate, object). The evaluation metric is the F1 score computed over all predicted triplets. The competition was hosted on Kaggle; I participated as a single-member team under the name *Hem* and placed second out of all participating teams.

This paper describes my system and the design decisions made throughout development. I discuss the data characteristics that shaped my modelling choices, the architectures of my two models, and an iterative development process that guided successive improvements from an initial private F1 of 0.138 to a final private score of 0.475. The private F1 refers to performance on the full hidden test set, which constitutes the official evaluation metric. The official submitted system corresponds to submission v13 in Table 3; submission v14 was identified retrospectively as achieving a higher private F1 but was not selected during the competition.

[1]https://huggingface.co/microsoft/BiomedNLP-BiomedBERT-base-uncased-abstract-fulltext

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

alin-andrei.hemeci@s.unibuc.ro (H. Alin-Andrei)

2. Task and Data Description

2.1. Task Definition

The task is document level knowledge graph extraction [1]. Each document describes one or more quarantine pests monitored by the French Epidemiomonitoring platform. The system must extract all relevant ecological facts as triplets of the form (subject entity, predicate, object entity).

2.2. Schema

The ontology defines seven node types and seven edge types. Node types include: Pest (pathogen organisms, typically referred to by their Latin binomial name), Plant (host crops), Disease (plant diseases caused by a pest), Vector (organisms that transport pests), Dissemination_pathway (physical means of pest dispersal), Location (geographical places of any scale), and Date (dates, periods, or seasons of observation).

The seven edge types are: Causes (Pest to Disease), Affects (Disease to Plant), Transmits (Vector to Disease or Pest), Found_on (Pest or Vector to Plant or Dissemination_pathway), Located_in (any entity to Location), Occurs_on (any entity to Date), and Dispersed_by (Pest or Disease to Dissemination_pathway).

2.3. Dataset

The EPOP corpus contains 247 web documents annotated by 30 domain experts following double blind annotation guidelines. The corpus is split into training (110 documents, 1374 relations), development (55 documents, 656 relations), and test (82 documents). Training and development documents include gold entity spans with character offsets, entity type labels, coreference groups, and knowledge graph edges. The test set contains only raw text.

3. Data Analysis

Before modelling, I conducted an exploratory analysis that directly shaped my design decisions.

Text length. Using the BiomedBERT tokeniser, 51.5% of documents exceed the standard 512 token BERT limit, with a maximum of 8134 tokens and a mean of 685 tokens. This means a standard BERT model will truncate substantial portions of many documents.

Predicate imbalance. The predicate distribution is highly skewed, as shown in Figure 1: Located_in accounts for 57.0% of all relations in the combined training and development set, while Transmits represents only 1.5%. This imbalance required class weighting during fine-tuning and motivated the per class confidence thresholds applied at inference time.

Entity type distribution. As shown in Figure 2, Location and Pest are the most frequent entity types, consistent with the dominance of Located_in as a predicate. Vector appears in only 2.2% of entity mentions and Dissemination_pathway in 3.6%, which complicates learning for the corresponding relation types Transmits and Dispersed_by.

Candidate pair imbalance. From the 110 training documents, only 1374 of 64968 possible entity pairs are positive examples, a negative to positive ratio of 46.3:1. Without correction, a classifier would achieve 97.9% accuracy by predicting no relation everywhere, but zero F1 on actual relations.

Type constrained relations. A key finding is that the schema is almost deterministic given entity types. As shown in Table 1, Causes occurs exclusively between Pest subjects and Disease objects, and Affects exclusively between Disease subjects and Plant objects. I exploit this structure as a hard filter to reduce the candidate pair space, eliminating type incompatible pairs before any model inference.

Coreference. The same biological concept frequently appears under multiple surface forms within a single document. For example, the pathogen *Tomato brown rugose fruit virus* appears as ToBRFV, Tomato brown rugose fruit virus, and Tomato Brown Rugose Fruit Virus in the same document. Without

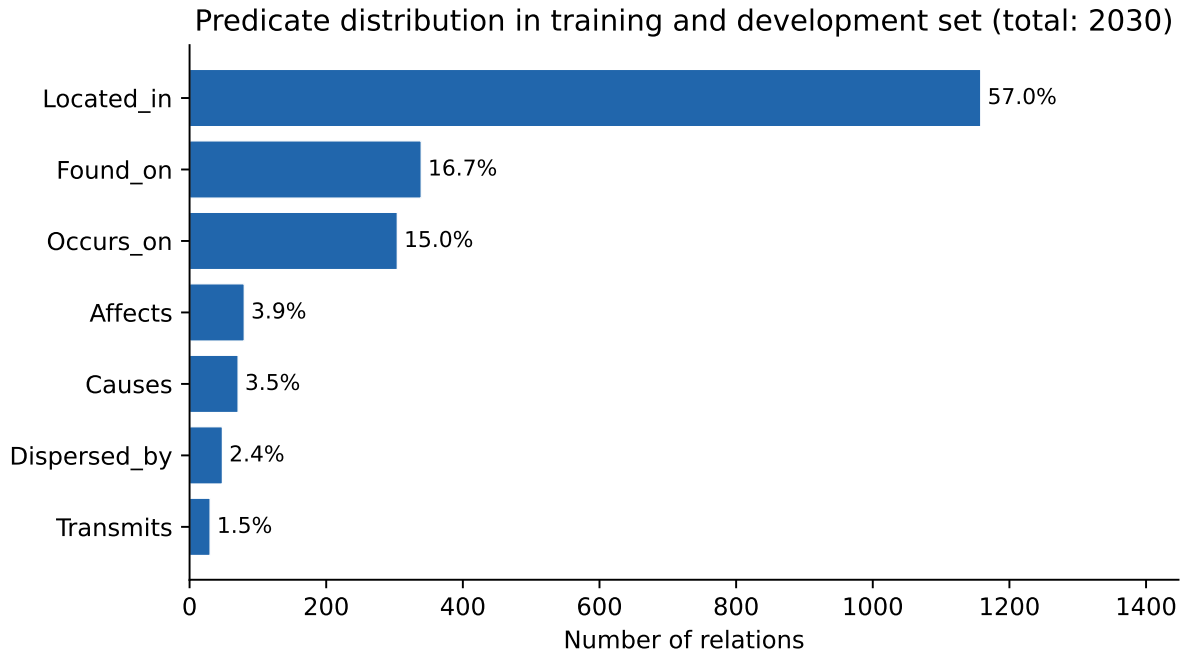

Figure 1: Predicate distribution in the combined training and development set.

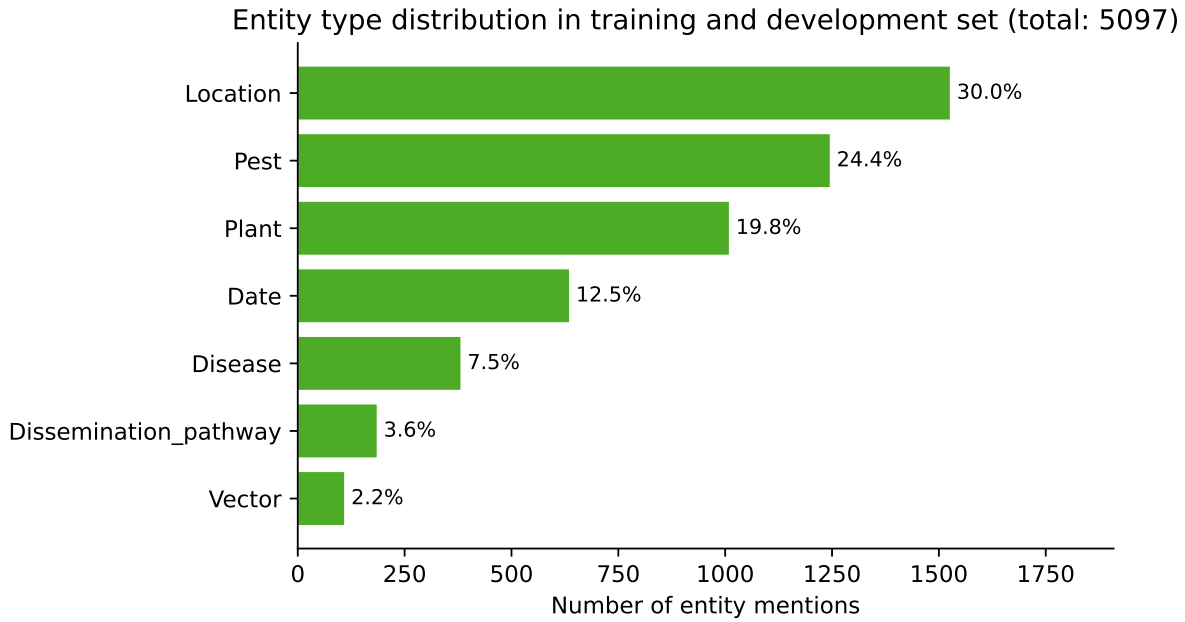

Figure 2: Entity type distribution in the combined training and development set.

coreference resolution, these would be treated as distinct entities, producing duplicate candidate pairs and inflating coverage statistics.

4. System Description

4.1. Overview

My system is a two-stage pipeline. First, a Named Entity Recognition model identifies entity spans and their types in the input text. Second, a Relation Extraction model classifies the relationship between each

Table 1
Valid entity type pairs per predicate, derived from corpus analysis. Only these combinations are used as candidate pairs for relation extraction.

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

candidate pair of detected entities. Both models share the same pre-trained encoder: BiomedBERT [3][1], a BERT base model pre-trained on PubMed abstracts and full text biomedical articles. I selected this model because the documents contain extensive Latin nomenclature, abbreviated species names, and technical terminology that general domain models represent poorly. Both models use a maximum sequence length of 512 tokens and a dropout rate of 0.1.

4.2. Named Entity Recognition

The NER model adds a token level linear classifier on top of BiomedBERT. For a sequence of T tokens padded or truncated to a maximum length of 512, the encoder produces a matrix $\mathbf{H} \in \mathbb{R}^{T \times 768}$, and the classifier produces label scores $\mathbf{Y} \in \mathbb{R}^{T \times 15}$, where 15 corresponds to the BIO label set: one O class plus two labels (B and I) for each of the seven entity types.

I use the Begin-Inside-Outside (BIO) tagging scheme, where B marks the first token of an entity span, I marks continuation tokens, and O marks tokens that belong to no entity. This scheme unambiguously encodes entity boundaries, allowing the model to distinguish a single multi-token entity from two adjacent entities. Note that the BIO scheme cannot represent discontinuous or nested entities; the EPOP corpus does not contain such cases according to the annotation guidelines, so this limitation does not affect the current system. However, if future versions of the corpus include discontinuous mentions, an alternative tagging scheme such as BIOES would be required.

Label alignment is performed using the offset mapping returned by the HuggingFace tokeniser, which maps each subword token to its character span in the original text. Special tokens ([CLS], [SEP], [PAD]) receive the label -100, which is ignored by PyTorch's cross-entropy loss. For subword tokens that are not the first token of a word, I assign the I tag of the enclosing entity if one exists, or O otherwise.

Class weights are computed as $w_c = \min(N/(C \cdot n_c),\ 20)$, where N is the total number of labelled tokens (excluding -100), $C = 15$ is the number of classes, and n_c is the frequency of class c. The cap of 20 prevents extreme weights for very rare classes such as the B-Vector label (which occurs only 14 times in the development set) from destabilising fine-tuning.

At inference time, entity spans are reconstructed from the predicted BIO sequence by grouping consecutive B and I tags of the same type. Duplicate entities with the same surface form and type within a document are removed before constructing candidate pairs.

4.3. Relation Extraction

The Relation Extraction model encodes a pair of entities together with the document text as a single sequence:

```
[CLS] <type1> entity1 [SEP] <type2> entity2 [SEP] text [SEP]
```

[1] https://huggingface.co/microsoft/BiomedNLP-BiomedBERT-base-uncased-abstract-fulltext

Type tags such as <Pest> and <Plant> are prepended to each entity surface form to provide explicit type information to the model. The [CLS] token representation is passed through a dropout layer and a linear classifier over eight classes: the seven relation types plus no_relation.

Candidate pair construction. For each document, I generate all ordered entity pairs (A, B) where A and B are distinct entities, and filter them using the valid type pairs shown in Table 1. This eliminates type incompatible pairs and reduces the candidate space from tens of thousands to a few hundred per document.

Coreference resolution. I apply the Union-Find algorithm to the `identity_coreferences` field in the training annotations. Each entity starts in its own group; for each coreference group in the annotations, all member entities are merged into a single group using union by root. The canonical form of each group is the shortest surface form among all members. This prevents duplicate pairs and ensures consistent matching against the gold knowledge graph. At inference time on the test set, no coreference annotations are available; I rely on surface-form deduplication only. The impact of coreference merging was not ablated separately, as the annotations were always used when available; its contribution is subsumed within the overall pipeline results.

Negative sampling. After filtering by valid type pairs, the remaining negative pairs (pairs with no relation in the gold knowledge graph) are randomly sampled at a ratio of 5:1 with respect to positive pairs. This reduces the effective class imbalance from 46:1 to approximately 4:1. Random sampling uses a fixed seed of 42 for reproducibility. Class weights further correct for the remaining imbalance during fine-tuning.

Evolution of entity sources. The source of entities used to construct RE fine-tuning examples evolved significantly across system iterations, and this evolution was the most impactful design decision I made.

Initially, I used gold annotated entities directly from the `text_bound_annotations` field provided in the training data, and at inference time on the test set I substituted a dictionary of all entity surface forms seen in training and development documents, using substring matching to identify entities in test documents. This approach produced noisy entities, missed entities whose surface forms did not appear in training data, and produced spurious matches such as treating the phrase “for the first time” as a Date entity.

In a second iteration, I fine-tuned a NER model and used its predictions to identify entities in test documents, but continued to fine-tune the RE model on gold annotated entities. This created a fine-tuning/inference distribution mismatch: the RE model learned to classify relations between clean, expert annotated entities, but received imperfect NER detected entities at inference time.

In the final system, I run the NER model on all training and development documents and use the detected entities to construct RE fine-tuning examples. This ensures that the RE model sees the same type of entity representations at fine-tuning time and at inference time, eliminating the distribution mismatch that had degraded earlier results.

4.4. Confidence Thresholding

To improve precision, I apply per class confidence thresholds at inference time. After computing softmax probabilities over the eight output classes, a relation is accepted only if the probability of the predicted class exceeds the class-specific threshold shown in Table 2. Predictions below the threshold are discarded regardless of their rank.

5. Experimental Setup

Hardware. All models were fine-tuned on Kaggle cloud infrastructure using NVIDIA T4 GPU. NER fine-tuning (40 epochs, 165 documents, batch size 8) completed in approximately 90 minutes. RE fine-tuning (20 epochs, approximately 11000 examples, batch size 16) completed in approximately 3 hours.

Table 2
Per class confidence thresholds applied at inference time. Thresholds were selected manually through empirical experimentation on the development set: frequent predicates that are prone to false positives receive stricter thresholds, while rare predicates receive more permissive thresholds to avoid missing true relations. The values shown represent the combination that achieved the best development set F1.

Predicate	Threshold
Located_in	0.70
Found_on	0.65
Occurs_on	0.65
Affects	0.55
Causes	0.55
Dispersed_by	0.45
Transmits	0.45

Table 3
System configurations and F1 scores across submission iterations. All versions use BiomedBERT as the base encoder. NER and RE epoch numbers refer to the checkpoint selected from the respective fine-tuning run.

Version	Description	Public F1	Private F1
Baseline	RE only, entity dictionary via substring matching, no NER	0.110	0.138
v2	NER added (10 epochs, train split only), RE on gold entities	0.194	0.192
v3	End to end: RE fine-tuned on NER entities, train and dev combined	0.342	0.425
v5	NER 30 epochs, RE 6 epochs, global threshold 0.65	0.360	0.409
v7	NER 30 epochs, RE with local context window (300 chars)	0.329	0.406
v8	NER 40 epochs (ep.39), RE 20 epochs (ep.12), threshold 0.65	0.379	0.459
v13 (selected)	NER 40 epochs (ep.39), RE epoch 12, per class thresholds	0.380	0.459
v14 (best)	NER 40 epochs (ep.30), RE epoch 12, per class thresholds	0.344	**0.475**

Hyperparameters. Both models were optimised with AdamW with learning rate 2×10^{-5} and weight decay 0.01, and gradient clipping at maximum norm 1.0. Dropout rate was set to 0.1. Maximum sequence length was 512 tokens for both models. Random seed was fixed at 42 for negative sampling.

NER fine-tuning strategy. I fine-tune on all 165 available documents (training and development combined) for 40 epochs with batch size 8, saving model weights every 5 epochs. Class weights are capped at 20. I observed that the model achieved its best generalisation at epoch 30 based on subsequent test leaderboard performance; the epoch 39 checkpoint appeared to have overfit to training specific patterns.

RE fine-tuning strategy. I construct the RE fine-tuning dataset by running the NER model (epoch 30 checkpoint) on all 165 documents and building candidate pairs as described above, yielding approximately 11000 examples. I fine-tune for 20 epochs with batch size 16, saving checkpoints every 2 epochs. Class weights are capped at 10. Based on ablation experiments using the development set, the epoch 12 checkpoint produced the best balance between underfitting and overfitting; the epoch 13 checkpoint showed slightly lower train loss but higher development loss, indicating the onset of overfitting.

Code availability. The code for this system is not yet publicly available as it is still being cleaned and documented, but can be provided upon request.

6. Results

Table 3 presents the progression of my system across key configurations. Public scores are computed on a subset of the test set visible during the competition; private scores reflect the full hidden test set and constitute the official evaluation. Figure 3 illustrates the progression of scores across iterations. The official final submission was v13; submission v14 was identified retrospectively as achieving a higher

private F1 of 0.475.

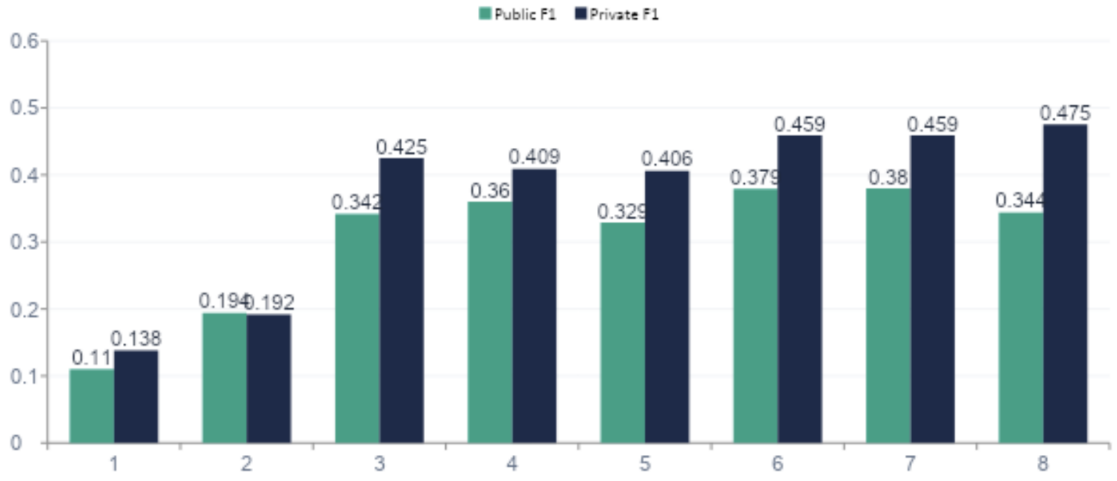

Figure 3: Public and private F1 scores across submission iterations.

The most significant single improvement came from transitioning to an end to end consistent pipeline (v3), where the RE model is fine-tuned on entities produced by the NER model rather than gold annotations. This eliminated the fine-tuning/inference distribution mismatch and raised the private F1 from 0.192 to 0.425.

Increasing NER fine-tuning from 10 to 40 epochs and fine-tuning both models on the full training and development set consistently improved performance. The optimal RE checkpoint at epoch 12 was identified through ablation on the development set; epoch 13 showed marginally lower train loss (0.0527 vs 0.0532) but higher development loss, indicating overfit.

The best private score of 0.475 was achieved with the NER checkpoint at epoch 30 from the 40 epoch run rather than epoch 39. I attribute this to the NER fine-tuning dynamics: train loss was stable and decreasing up to epoch 30, then exhibited oscillations between epochs 31 and 35 before recovering, suggesting that epoch 30 represents a more stable generalisation point. The epoch 39 checkpoint achieved lower train loss (0.0067 vs 0.0081) but apparently overfit to fine-tuning specific patterns not present in the test distribution.

6.1. NER Per-class Results

Table 4 presents per-class NER F1 scores obtained on the development set using a model fine-tuned on the training split only (110 documents). The final submitted model was fine-tuned on the combined training and development set and does not have a separate evaluation set available.

7. Error Analysis

Document truncation. With 51.5% of documents exceeding 512 tokens, relations mentioned in the latter half of long documents are systematically missed. The RE model receives only approximately 490 tokens of text after allocating approximately 20 tokens for the two entity strings and special tokens. This is the most significant known limitation of the current system.

Rare entity types. The B-Vector and B-Dissemination_pathway labels have very few fine-tuning examples, with 14 and 30 B tag occurrences respectively in the development set, leading to the lower F1 scores shown in Table 4. These errors propagate downstream to the RE model, which consequently struggles with Transmits and Dispersed_by relations.

Precision versus recall. Both models exhibit high recall but lower precision. On the development set, the NER model achieves macro recall of 0.88 but macro precision of 0.60. The RE model similarly

Table 4
Per-class NER F1 scores on the development set (model fine-tuned on training split only, 30 epochs). The final submitted model was fine-tuned on the combined training and development set and does not have a separate evaluation set available.

Label	Precision	Recall	F1
O	0.99	0.97	0.98
B-Pest	0.83	0.97	0.90
I-Pest	0.92	0.97	0.95
B-Plant	0.88	0.94	0.91
I-Plant	0.88	0.96	0.92
B-Disease	0.91	0.94	0.93
I-Disease	0.91	0.96	0.94
B-Vector	0.83	0.71	0.77
I-Vector	0.86	0.82	0.84
B-Location	0.84	0.94	0.89
I-Location	0.84	0.94	0.89
B-Date	0.74	0.78	0.76
I-Date	0.86	0.90	0.88
B-Dissemination_pathway	0.41	0.83	0.55
I-Dissemination_pathway	0.53	0.55	0.54
Macro avg	0.78	0.87	0.82

tends to over-predict Located_in due to its 57% share of fine-tuning examples. Per class thresholding partially addresses this, but the fundamental imbalance remains.

Discontinuous and nested entities. The BIO tagging scheme cannot represent discontinuous entity mentions (e.g. a pest name split across a clause boundary) or nested entities (e.g. a disease name that contains a plant name). While the EPOP annotation guidelines do not explicitly allow such cases, any such mentions in the corpus would be silently mislabelled, potentially contributing to the precision gap observed in Table 4.

Local context window. I experimented with providing only a local context window of 300 characters around the two entities rather than the full document text, motivated by the truncation problem. This approach (submission v7) achieved a public F1 of 0.329 compared to 0.360 without context windowing, a degradation of 0.031 points. I attribute this to two factors: first, relations are sometimes expressed across longer distances in the text than 300 characters allow; second, the substring search used to locate entities finds only the first occurrence in the document, which may not be the most contextually relevant one for a given relation.

8. Conclusion

I presented a two stage NER and Relation Extraction pipeline for knowledge graph extraction from agricultural pest documents. The central finding is that fine-tuning the RE model on entities detected by the NER model, rather than on gold annotated entities, is essential for consistency between fine-tuning and inference conditions. This design decision alone raised the private F1 from 0.192 to 0.425. Combined with longer NER fine-tuning, careful checkpoint selection based on development set ablation, and per class confidence thresholding, my final system achieved a private F1 score of 0.475, placing second in the PestCLEF 2026 competition.

Future work should investigate longer context models such as Longformer [4] to address document truncation, which affects more than half of the documents in the corpus. Learning rate scheduling would also be beneficial to stabilise NER fine-tuning and avoid the loss oscillations observed after epoch 30.

Acknowledgments

The author thanks the PestCLEF 2026 organisers for providing the EPOP corpus and for running the evaluation campaign.

Declaration on Generative AI

During the preparation of this work, the author used an AI assistant as a debugging and research tool during code development. After using this tool, the author reviewed and edited all content and takes full responsibility for the publication's content.

References

[1] Robert Bossy, Marine Courtin, Louise Deléger, Xinzhi Yao, Claire 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] Lukas Picek, Lukáš Adam, Stefan Kahl, Robert Bossy, Laura Chrobak, Hervé Goëau, Kostas Papafitsoros, Holger Klinck, Willem-Pier Vellinga, Robert Planqué, Tom Denton, Kevin Barnard, Claire Nédellec, Louise Deléger, Marine Courtin, Giulio Martellucci, Ilyass Moummad, Fabrice Vinatier, Pierre Bonnet, Alexis 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] Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, Hoifung Poon. Domain-Specific Language Model Pretraining for Biomedical Natural Language Processing. *ACM Transactions on Computing for Healthcare*, 3(1), 2021.

[4] Iz Beltagy, Matthew E. Peters, Arman Cohan. Longformer: The Long-Document Transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT 2019*, 2019.