Universidad del País Vasco Euskal Herriko Unibertsitatea

NLP Meets Agronomy: Document Classification for Plant Health Surveillance

Author: Luis Antonio VASQUEZ REINA
Advisors: Rodrigo AGERRI (UPV/EHU)
Robert BOSSY (MaIAGE Research Group at INRAE)
Miguel COUCEIRO (IDMC Institute at Université de Lorraine)

hap/lap

Hizkuntzaren Azterketa eta Prozesamendua
Language Analysis and Processing

Final Thesis

August 2023

Departments: Computer Systems and Languages, Computational Architectures and Technologies, Computational Science and Artificial Intelligence, Basque Language and Communication, Communications Engineer.

Abstract

In the field of Plant Health Epidemiological Surveillance, accurately analyzing written reports of events affecting agriculture is crucial. This master's thesis in Natural Language Processing leverages the power of Automatic Text Classification, specifically focusing on its application in monitoring plant health. Conducted as part of the *TIERS-ESV* project, this work is a joint effort between the Bibliome team at the INRAE MaIAGE Laboratory and the VSI team at the PESV Plant Health Surveillance Platform. Central to our research is a dataset curated and annotated by the VSI team, consisting of reports extracted from online sources. This diverse, multilingual dataset was subjected to thorough preprocessing, applying methods for noise elimination, error message removal, and addressing scrapping errors and annotation discrepancies. We employed various BERT models for Text Classification, tailored to our dataset via Fine-tuning and pattern-based training method. After extensive classifier training, we selected the top-performing models. The efficacy of the models obtained as a result of this work will result in the deployment of BERT-based classifiers, poised to assist the VSI experts in their monitoring mission.

Laburpena

Landare Osasunaren Zaintza Epidemiologikoaren arloan, funtsezkoa da nekazaritzari eragiten dioten gertaeren idatzizko txostenak zehaztasunez aztertzea. Hizkuntza Prozesamendurako master-tesi honek testu sailkapenaren ahalmena aprobetxatzen du, bereziki landareen osasuna kontrolatzeko duen aplikazioan zentratuz. Masterreko tesi hau *TIERS-ESV* proiektuaren barne garatu da eta INRAE MaIAGE Laborategia eta PESV Landare Osasuneko Zaintza Plataformaren VSI taldearen arteko ahalegina da. Gure ikerketan funtsezkoa da VSI taldeak eskuz etiketatutako datu-multzo bat, Internet-en iturrietatik ateratako txostenez osatua. Datu-multzo anitz eta eleaniztun honi aurreprozesamendu sakona egin zitzaion, zarata kentzeko metodoak aplikatuz, errore-mezuak kentzeko eta scrapping akatsak eta anotazioen desadostasunak zuzentzeko. Testu sailkapenerako hainbat BERT eredu mota erabili ditugu, gure datu multzora egokitutako fine-tuning-en eta PET-en oinarritutako metodoaren bidez. Sailkatzaileen entrenamendu anitz egin ondoren, errendimendurik handiena duten ereduak aukeratu ditugu. Lan honen ondorioz lortutako eredu eraginkorrenak hedatzea ekarriko du, VSI adituei beren jarraipen-misioan laguntzeko prest.

Resumen

En el área de la Vigilancia Epidemiológica de la Salud de las Plantas, el análisis preciso de reportes escritos sobre eventos que afectan a la agricultura es crucial. Este Trabajo de Fin de Máster en Análisis y Procesamiento del Lenguaje aprovecha las capacidades de la Clasificación Automática de Textos, específicamente, enfocándose en su aplicación al monitoreo la salud de las plantas. Realizado en el marco del proyecto *TIERS-ESV*, este trabajo es el resultado de una colaboración entre el equipo Bibliome de la unidad MaIAGE del laboratorio INRAE y el equipo VSI de la Plataforma de Vigilancia de la Salud de las Plantas PESV. Nuestra investigación se apoya en un conjunto de datos curados y anotados por el equipo VSI, que consiste en reportes extraídos de fuentes recopiladas en línea. Este conjunto diverso y multilingüe de datos fue sometido a un preprocesamiento exhaustivo, aplicando métodos para la eliminación de ruido, la supresión de mensajes de error y el tratamiento de errores de recopilación y discrepancias en las anotaciones. Utilizamos diversos modelos BERT para la Clasificación de Textos, adaptados a nuestro conjunto de datos mediante Fine-tuning y un método de entrenamiento basado en plantillas. Después de un entrenamiento intensivo de los clasificadores, seleccionamos los modelos de mejor rendimiento. La eficacia de los modelos obtenidos como resultado de este trabajo conducirá a la implementación de clasificadores basados en BERT, preparados para asistir a los expertos del equipo VSI en su misión de monitoreo.

This thesis is dedicated to everyone who has not dedicated their thesis to themselves.

Acknowledgments

Starting this Master's degree has been a great personal step. Moving abroad and really learning about other cultures is a both an opportunity and a challenge. Facing completely different expectations around social norms, education, and bureaucracy has been a test. I appreciate everyone that supported me during this journey, especially, I would like to thank, ordered by name length and then alphabetically, Anna, Flor, Diego, Elisa, María, Shane, Isabel, Joanna, Karolin, Maxime, William, and Luis Felipe. Additional thanks go to the LCT students and people that made our time more enjoyable.

I also thank my supervisors for the helpful discussions and support they offered me for this thesis, and the Saclay-IA platform for the computational resources provided.

Finally, I would like to acknowledge the LCT Consortium for putting together this Erasmus Mundus Master Program, and especially for the scholarship I received, without which, pursuing this degree would not have been feasible.

Contents

List of Figures

List of Tables

Abbreviations

$SoftMax$ Soft-ArgMax. 7, 12, 20, 57

σ Logistic function. 7

tanh Hyperbolic Tangent. 7, 20

AI Artificial Intelligence. 1–3, 14, 24

AUC Area Under the Curve. 59, 81, 82

BERT Bidirectional Encoder Representations from Transformers. 17–21, 24, 52–57, 59, 60, 63–66, 71

BLURB Biomedical Language Understanding and Reasoning Benchmark. 53

BPE Byte-Pair Encoding. 14, 15, 53

FFNN Feed-Forward Neural Network. 12

ML Machine Learning. 2, 4, 6, 12, 15, 54

MLM Masked Language Modelling. 20, 55, 57

NLP Natural Language Processing. 1, 2, 4, 13–15, 18, 24, 54, 70, 71

PESV Plateforme d'Epidémiosurveillance en Santé Végétale. 2, 25, 27, 59, 67, 68, 71

PET Pattern-Exploiting Training. 54–63, 65, 66, 71

ReLU Rectified Linear Unit. 7

RNN Recurrent Neural Network. 17

TIERS-ESV Traitement de l'Information et Expertise des Risques Sanitaires pour l'Epidémiosurveillance en Santé Végétal. 2, 67

VSI Veille Sanitaire Internationale. 2, 3, 25–27, 34, 37, 41, 45, 67–71

WP Word Piece. 15, 19, 52, 53

1 Introduction

1.1 Motivation

Natural Language Processing (NLP) is a subfield of Artificial Intelligence (AI) that focuses on the interaction between computers and humans through natural language. Its primary aim is to enable computers to understand, interpret, and generate human language in a way that is both meaningful and useful. Recently, NLP has been revolutionized by the widespread adoption of Neural Networks. As in many domains, AI and NLP have been starting to be adopted for Biomedical purposes, with promising results (Alrowili and Shanker, 2021; Hakala and Pyysalo, 2019).

In this thesis, we present an application of Text Classification to Plant Health Surveillance, which refers to the systematic observation, detection, and analysis of plant diseases and pests to prevent their spread and ensure the health and productivity of plants. It involves tracking disease patterns, assessing risks, and implementing measures to protect plant ecosystems and agricultural systems.

1.2 Hosting Institution : INRAE Laboratory

INRAE [1] (*Institut National de Recherche pour l'Agriculture, l'Alimentation et l'Environnement*) is a French research institute dedicated to agriculture, food, and the environment.

INRAE conducts scientific research and innovation activities to address various challenges related to sustainable agriculture, food production, and environmental conservation. Its research efforts span a wide range of disciplines, including agronomy, biology, ecology, genetics, forestry, hydrology, and applied mathematics and computer science.

Through its multidisciplinary approach and emphasis on sustainability, INRAE plays a crucial role in driving scientific progress, innovation, and policy development in France and beyond, with the ultimate aim of ensuring sustainable and resilient agricultural and environmental systems.

Within INRAE, the Mathematics and Numerics division (MathNum[2]) focuses on advancing research in various fields including applied mathematics, statistics, bioinformatics, AI, and information technology. The division's research units engage in theoretical, methodological, and applied research. They actively collaborate with teams from different divisions within INRAE and external organizations, fostering interdisciplinary partnerships. MaiAGE[3], one of these research units, is associated to the Paris-Saclay University, and brings together mathematicians, computer scientists, bioinformaticians, and biologists to address challenges in the fields of biology, agronomy, and ecology. This project was carried out with alongside the Bibliome[4] team, which specializes in the advancement and

[1] `https://www.inrae.fr/`
[2] `https://www.inrae.fr/departements/mathnum`
[3] `https://maiage.mathnum.inrae.fr`
[4] `https://maiage.mathnum.inrae.fr/fr/bibliome`

adaptation of NLP and Machine Learning (ML) techniques tailored for textual data within the fields of biology and agronomy.

1.3 Overarching Project: TIERS-ESV

The Bibliome team, as part of the Traitement de l'Information et Expertise des Risques Sanitaires pour l'Epidémiosurveillance en Santé Végétal (TIERS-ESV) project[5] (*Processing of Health Risk Information and Knowledge for Epidemiological Surveillance in Plant Health*), collaborates frequently with the Plateforme d'Epidémiosurveillance en Santé Végétale (PESV)[6] (*Plant Health Epidemiological Surveillance Platform*), and it is within this collaboration that we conducted this work.

The PESV Platform was created in 2018 by 7 French national public and private actors (INRAE, the Anses Laboratory[7], the French Ministry of Agriculture, the Cirad Center[8], the Acta reseach association [9], the French Chamber of Agriculture [10], and the FREDON Network[11]) with the objective to ensure the efficiency of epidemiological surveillance in plant health (Ministère de L'Agriculture et de L'Alimentation, 2018).

According to its mission statement, 'the PESV Platform leverages its scientific and technical expertise to engage in three key areas: monitoring, analysis, and advisory services. Its services cater to both public policies and all professionals within the plant health sector'.

As an integral part of its monitoring endeavors, the PESV Platform includes the Veille Sanitaire Internationale (VSI)[12] (*International Health Monitoring*) project. The main focus of the VSI team is "to address potential threats that may impact plant health by engaging in monitoring activities on a global scale". Its responsibilities encompass ongoing surveillance of events like notifications, reports, and changes in surveillance strategies across the globe. Furthermore, it conducts scientific monitoring on diverse subjects, including phylogeography, spatial distributions of species, habitat suitability assessments, innovative disease prevention measures, as well as surveillance and control methodologies.

1.4 Objective

Our objective is to leverage AI to support the Plant Health Monitoring efforts of the PESV Platform.

The primary objective of this project is to facilitate and enhance the monitoring of Plant Health at the PESV Platform by introducing a degree of automation to the traditionally manual processes. Recognizing the critical importance of timely and accurate Plant Health

[5] `https://plateforme-esv.fr/node/24638` and `https://maiage.inrae.fr/index.php/fr/node/2111`

[6] `https://plateforme-esv.fr/`

[7] `https://www.anses.fr/fr`

[8] `https://www.cirad.fr/`

[9] `https://www.acta.asso.fr/`

[10] `https://chambres-agriculture.fr/`

[11] `https://fredon.fr/`

[12] `https://plateforme-esv.fr/thematiques/GTVSI`

monitoring, our aim is to harness the data provided to us by experts at the VSI team in order to develop a detection system based on AI. By doing so, we aspire to offer a more efficient, data-driven approach to detect early signs of plant diseases, pests, or any other anomalies. This would not only ensure a faster response and intervention but also alleviates the workload of professionals in the field. This endeavor is not merely about replacing human intervention but assisting it. By automating certain repetitive and time-consuming tasks, we aim to alleviate experts to focus on more nuanced and complex aspects of plant health, ensuring that their expertise is utilized where it is most needed.

This thesis has led to the development of a Text Classification system designed to support the VSI experts. We have carefully picked six different BERT models based on their respective characteristics. Moreover, we've explored two training techniques: fine-tuning and few-shot learning with PET, a prompt-based approach. After undergoing two rounds of training, we generated a considerable number of candidate classifiers. These classifiers then underwent a filtering process, resulting in the selection of a limited set of the most high-performing ones, which will be deployed for use by the VSI experts in their Plant Health Surveillance mission.

1.5 Structure of this Thesis

After this Introduction, Section 2 provides foundational knowledge and context for understanding our work. Section 3 offers a review of existing literature, highlighting previous methodologies and their gaps. In Section 4 we delve into the Dataset, detailing its sources and characteristics, and then continue to Section 5, which elucidates the preprocessing techniques employed. Section 6 outlines the research methods and tools used, leading to Section 7, where our findings are presented and analyzed. Section 8 sheds light on the real-world relevance of the findings, and the study culminates with Section 9, summarizing key takeaways. Additionally, an Appendix is provided, rich with tables and detailed statistics, serving as a valuable resource for reference.

This thesis is submitted in fulfillment of the requirements for the Erasmus Mundus Language And Communication Technology[13] from the European Union, as a student assigned to the University of Lorraine (France), and the University of the Basque Country (Spain).

[13] `https://lct-master.org/`

2 Background

This Section introduces the conceptual tools necessary to understand our project, our methodology, and our results.

2.1 Text Classification

Text Classification is a fundamental task in NLP that involves automatically assigning predefined categories or labels to text documents based on their content. The goal of Text Classification is to develop computational models that can accurately classify text documents into predefined categories, enabling automated organization, retrieval, and analysis of large volumes of textual data.

Formally, given a set V called the *Vocabulary*, and a set C of *categories*, a *Text Classification System* is a function

$$f : V^* \times C \mapsto \{\texttt{True}, \texttt{False}\}$$

where V^* is the set of finite sequences of elements of V, that is, the set of *documents*. While the general definition of Text Classification accommodates multi-label classification scenarios, the specific focus of this project is on *Binary Text Classification*. In this particular case, the classification task involves assigning documents to one of two mutually exclusive categories (Shen, 2009; da Costa et al., 2023).

In our particular scenario, we will be handling entire documents for analysis rather than, for example, individual words or proper names. Thus, the task is termed *Document Classification*. In the context of Document Classification, the fundamental objective is to leverage a training set, comprising a collection of training documents (d_i, c_i), and employ a learning method or algorithm to derive a *Classifier* γ. The primary purpose of this classifier is to establish a mapping from documents to two classes. Here, for simplicity, we refer to them as the positive and negative classes. This is denoted as

$$\gamma : V^* \mapsto \{Positive, Negative\}$$

2.2 Neural Networks

Neural Networks are essential tools in ML and NLP. Given their complexity, this section introduces core concepts. Starting with a practical example, we introduce the concept of 'neuron', and then we proceed to explain neural 'training' based on data, its common challenges, and the solutions used in our work. Subsequently, our exposition continues to the organization and interconnection of neurons within a structure or *architecture* referred to as a Neural Network. Finally, the use of Neural Networks in NLP is also examined. We draw on the Open Access material from Zhang et al. (2023) and Wolfram (2023).

2.2.1 Motivating Example

To understand the rationale behind using Neural Networks, let's consider a scenario where we aim to determine how the temperature and humidity of a place relate to the perceived temperature by living beings, such as humans. In other words, given these two metrics across various locations, our goal is to forecast the felt temperature or Heat Index for each site. A simplified version of the model from Steadman (1979), can provide us with an equation for the Heat Index (HI).

$$HI = w_{Temp}T + w_{Hum}H + c$$

The main idea behind this model is that the heat index, our *target*, can be calculated as a weighted sum of the temperature T and the humidity H. The influence of T on the HI is given by w_{Temp} and that of H by w_{Hum}, and we correct by a constant value c.

In this model, H and T are termed the *features* of a location, serving as its defining characteristics. Essentially, locations with identical features will yield the same output. The terms w_{Temp} and w_{Hum} are referred to as *weights* or *parameters*. They signify the relevance of the features to the model. For instance, if w_{Hum} is zero, the HI would solely be influenced by T. Additionally, c is labeled as the *bias* or *offset*, representing the output value when the features are null. The equation's right-hand side, $w_{Temp}T + w_{Hum}H + c$, is recognized as the model's *output* or *prediction*.

In practice, the values for the weights and bias of this model have been determined to be:

$$\begin{aligned} w_{Temp} &= 1.1 \\ w_{Hum} &= 0.047F^\circ \\ c &= -10.3F^\circ \end{aligned}$$

Yet, this basic representation has its limitations. Interestingly, by introducing an additional computational step, we can achieve more accurate results. Let's consider incorporating a minor squared term through the function $f(u) = u + 0.0002u^2$. With this, our model evolves to:

$$HI = f(w_{Temp}T + w_{Hum}H + c)$$

In this enhanced model, f is termed the *activation function*. The model's prediction or output is derived by applying f to the weighted sum. This added computation step allows for a more authentic representation of how living beings respond to significant shifts in temperature and humidity. In general, this kind of model allows us to use numerical features, to obtain a prediction for a variable of interest.

2.2.2 Neurons

When dealing with increasingly complex phenomena, it becomes challenging to explicitly define the parameters for a model. This requires a broader approach. In the realm of Machine Learning, the concept of *neurons* serves as a generalization of the model described earlier.

(a) Neuron overview (b) Neuron with weights

Figure 1: Basic Neuron

Consider a model where the inputs encompass n distinct features, represented as $(x_1, x_2, \ldots, x_n)$. The diagram depicted in Figure 1 represents the following equation:

$$z = f(w_1 x_1 + w_2 x_2 + \ldots + w_d x_d + b)$$

Here, $w_1, w_2, \ldots, w_n$ denote the weights, b signifies the bias, f is the activation function, and z stands for the prediction.

One of the main insights in ML is that one does not need to manually define the parameters $(w_1, w_2, \ldots, w_n, b)$ for this kind of model. Instead, with ample data comprising input features and observed target values, it is feasible to determine the best parameter values. This ensures the model's predictions closely align with the observed target values. Once established, this model can be applied to new data. The process of finding the optimal parameters is called *training*.

In order to find the optimal parameters, we first need to decide how to produce our final predictions, how to measure how well the model is performing, and how to update the model's parameters.

2.2.3 Activation functions

The method used to derive our final predictions is contingent on the specific problem to be addressed. Various activation functions cater to different problems. For instance, in *regression*, where our objective is to compute a continuous quantity, like in the example in Section 2.2.1, we may handle values that can range from extremely small to exceedingly large. On the other hand, classification tasks aim to predict a categorical label, either by taking a final decision or by producing probabilities for each category.

Table 1 and Figure 2 enumerate some of the most common activation functions.

Activation Function	**Description**
Identity	Returns input as-is
Rectified Linear Unit (ReLU)	Retains positive values, sets negatives to zero
Logistic function (σ)	Maps input to a value between 0 and 1
Soft-ArgMax ($SoftMax$)	Converts input into a probability distribution over multiple classes
Hyperbolic Tangent (tanh)	Maps input to a value between -1 and 1

Table 1: Common activation functions

(a) Identity (b) ReLU (c) Sigmoid (d) Tanh

Figure 2: Common activation functions

2.2.4 Loss Functions

Loss functions quantify the deviation of predicted values from actual target values. The goal during training is to adjust the parameters to minimize this deviation. The process of inputting data into the neuron to compute the loss is termed the *forward pass*.

For regression tasks, the squared error is frequently employed. It calculates the squared difference between the predicted and actual values:

$$loss_{SE}(x, w, b) = \frac{1}{2}(z_{pred}(x, w, b) - z_{real})^2 \tag{1}$$

where $z_{pred} = f(xw + b)$.

In classification tasks, the *cross-entropy* loss is common. The essence of cross-entropy is to maximize the model's confidence in its category assignments[14]. Given features $x = x_1, x_2, \ldots, x_n$ and predicted probabilities $p_1, p_2, \ldots, p_k$ for categories $c_1, c_2, \ldots, c_k$, and representing actual categories as $(q_1, q_2, \ldots, q_k)$ where $q_j = 1$ for category j and $q_j = 0$ otherwise, cross entropy is:

$$loss_{CE}(x, w, b) = -\sum_j q_j(x) \log\left(p_j(x, w, b)\right) \tag{2}$$

For datasets with uneven category distribution, or *unbalanced* datasets, a modified version called *weighted cross-entropy* can be used. It adjusts the cross-entropy based on category representation. If category j has a weight of r_j, then:

[14]Deducing the formula for Cross-Entropy is outside the scope of this work

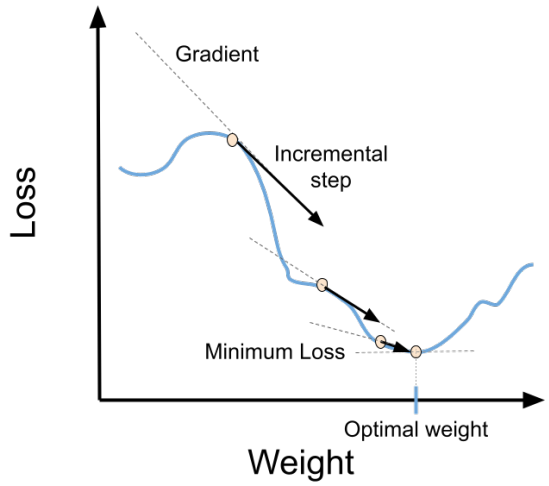


Figure 3: Gradient descent illustration

$$loss_{WCE}(x, w, b) = -\sum_{j} r_j q_j(x) \log\left(p_j(x, w, b)\right) \tag{3}$$

2.2.5 Backpropagation

After measuring the performance of a neuron through the loss, our aim is to adjust its parameters to reduce this loss. The process of *updating* the weights to achieve this reduction is termed *backpropagation*. A common optimization technique to achieve this is *gradient descent*. For illustrative purposes, let's assume that the neuron has just one weight, and thus, there's just one weight influencing the loss (as depicted in Figure 3). The key insight is that moving against the slope of the loss curve brings us closer to its minimum. This slope is the derivative or the *gradient* of the loss function. Through several iterative steps, updating the weight each time, we eventually reach the ideal weight that minimizes the loss.

Having identified the direction of adjustment, it's also crucial to regulate the magnitude of this adjustment. This is achieved using the *learning rate* η, which moderates the gradient's impact on weight updates. That is, our weight updating process is calculated in the following way:

$$w_{new} = w_{old} - \eta \nabla loss(x, w_{old}) \tag{4}$$

2.2.6 Training

Up to this point, we've discussed using just one feature vector as input for a neuron. But for a comprehensive representation of our task, multiple examples are essential. This means, for effective training, we require multiple data points, or in other words, a *Dataset*

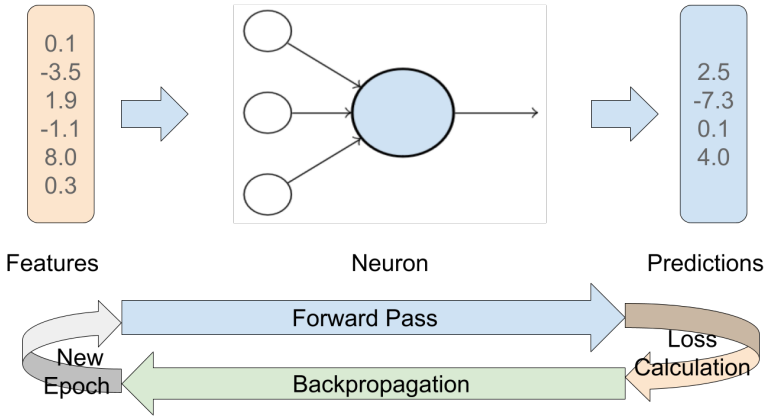


Figure 4: Neuron Training Process

comprising both features and their corresponding observed target values (x, z_{real}). If our Dataset D contains N items, our goal becomes minimizing the *average* loss across the dataset:

$$Loss(w, b) = \frac{1}{N} \sum_{x \in D} loss(x, w, b) \tag{5}$$

The entire process of executing the forward pass, determining the loss, and updating weights through backpropagation is termed an *epoch* (as illustrated in Figure 4).

In most cases, our computing resources might not possess the capacity to process the entire Dataset in one run. A common workaround is segmenting the Dataset into equally sized *batches*. The forward pass and loss computation are then executed batch by batch until the entire dataset is covered. Subsequently, the losses from each batch are combined to determine the final loss, followed by backpropagation. Typical batch sizes include powers of 2: $2, 4, 8, 16, 32, 64, \ldots, 2^n, \ldots$

Often, a single epoch doesn't suffice to yield satisfactory outcomes. Hence, it's a standard practice to train a neuron over multiple epochs. With each epoch, the loss typically reduces (as depicted in Figure 5).

2.2.7 Overfitting

As we progressively reduce our loss, we encounter a challenge. There are instances when the neuron will produce excellent predictions for the data it was trained on, but performs poorly on entirely new data, rendering the neuron useless for real-world applications. This issue is termed *overfitting*.

A prevalent strategy to counteract overfitting is to *divide* the dataset into three distinct portions:

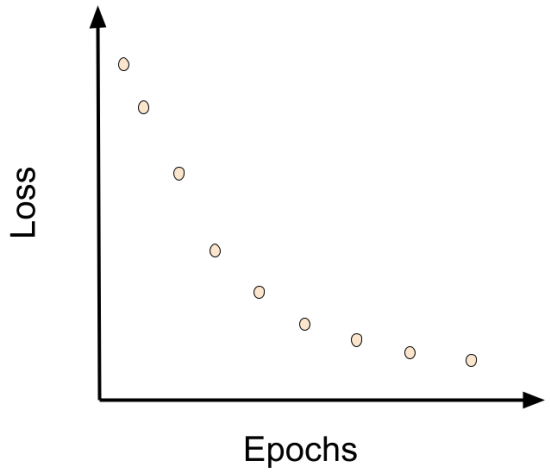


Figure 5: Evolution of loss during training

1. The *training* split, which constitutes a portion of the dataset dedicated to the core training processes, including the forward pass, loss evaluation, and crucially, backpropagation.
2. The *development* split, a portion of the dataset designated for tracking potential overfitting. For this split, we execute the forward pass and compute the loss, and occasionally other metrics, but refrain from backpropagation. Recognizing the beginning of overfitting allows us to adjust our setup for the optimal number of epochs.
3. The *test* split, a portion of the dataset reserved for assessing the efficacy of the chosen training setup, especially since this subset wasn't involved in its selection.

When monitoring the trajectory of the loss across the training and development splits, a pattern emerges. After a specific number of epochs, the loss on the training split continues to drop, but the development split's loss starts to climb. This shift marks the beginning of overfitting (depicted in Figure 6).

Other techniques for avoiding overfitting, like cross-validation, neuron dropout, gradient clipping, etc., are beyond the scope of this work.

2.2.8 Optimizers

We may generalize Gradient Descent (Equation 4), by noticing that we only need to calculate some way to update our weights:

$$w_{new} = w_{old} - \texttt{Update}(x, w_{old}) \tag{6}$$

Different optimization methods calculate the `Update` in different ways. Software implementations of these optimization methods are called *optimizers*. Some common methods are listed in Table 2, listed by increasing performance.

Figure 6: Evolution of loss on the training and development splits

Optimization method	**Description**
Gradient Descent	Classical Gradient Descent
Stochastic Gradient Descent (SGD)	Estimates the gradient by only using a randomly selected subset of the data
SGD with Momentum	Uses the estimated gradient and the previous value of `Update`
Averaged SGD	Takes into account several past values of `Update` and averages them while giving less importance to older values
Adaptive Gradient (AdaGrad)	A version of SGD where each weight is updated with its own learning rate, instead of all weights using the same one. It calculates `Update` by averaging and normalizing over the history of updates
Adam	AdaGrad + Momentum
AdamW	Extension of Adam, controlling for the magnitude of each weight

Table 2: Common optimization methods for backpropagation

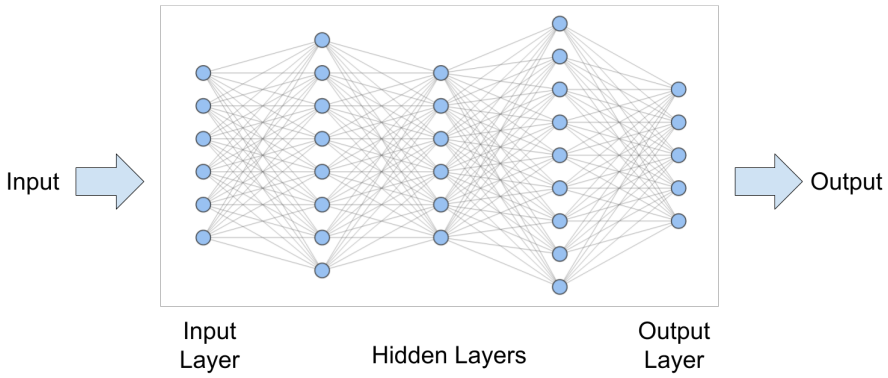


Figure 7: A Feed-Forward Neural Network

2.2.9 Feed-Forward Neural Networks

As we have seen, an individual neuron may take several inputs to give an output, and this provides us with a model useful for some applications. One of the groundbreaking insights in ML was the idea of linking neurons together. This involves using the output of one neuron as the input for others. By organizing neurons into *linear layers* and interconnecting them, we create a Feed-Forward Neural Network (FFNN) (Refer to Figure 7)[15].

When performing the forward pass, input features are fed into the FFNN via the *input layer*, they traverse through intermediate layers until they reach the *output layer*. As the FFNN undergoes training, every intermediate layer crafts new features for the subsequent layers. This mechanism enables the FFNN to *learn* new representations for the data. These in-between, or *hidden*, features detect and leverage patterns from earlier features to determine the final output of the Neural Network. A Neural Network with just one hidden layer is termed *shallow*, whereas those with several hidden layers are called *deep*.

For classification tasks, the $SoftMax$ activation function is typically employed in the output layer since it yields a probability distribution across various categories. Such a layer is termed a $SoftMax$ *layer* or a *classification layer*. The inputs to this layer are often referred to as *logits*. Given that we're working with probabilities, the objective is to minimize the cross-entropy loss between the predictions of the Neural Network and the true labels. (Refer to Figure 8)

2.2.10 Hyperparameters

When training a Neural Network, the goal is to progressively adapt its parameters or weights to minimize the loss. However, several critical choices need to be made by the designers of the Neural Network.

Some decisions concern the *architecture* of the Neural Network, such as:

[15] All diagrams of Neural Network were made with the $NN - SVG$ tool by LeNail (2019)

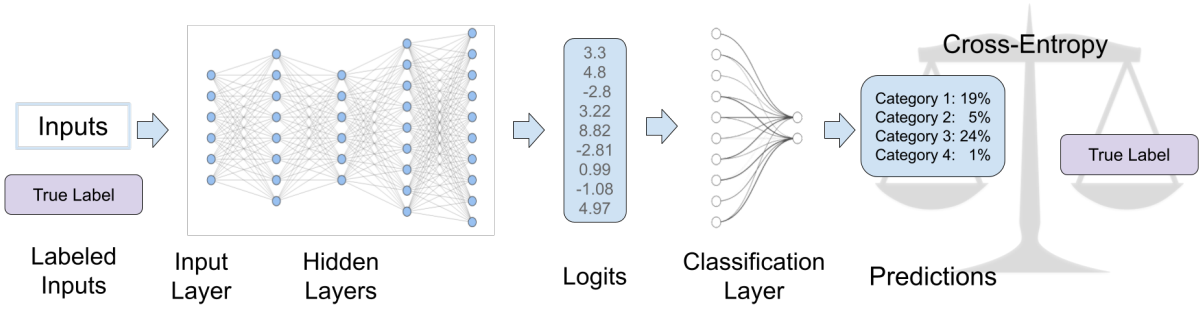


Figure 8: Neural Networks for Classification Tasks

- Number of layers
- Number of Neurons per layer
- Activation functions for each layer
- Dimension of output features
- Loss function
- Optimizer

There are also decisions about specific numerical values that affect the training method, such as:

- Number of epochs
- Learning rate
- Batch size
- Sizes for the dataset splits

All these are known as *hyperparameters*. Unfortunately, there is no one-size-fits-all answer to the question of which hyperparameters to use because the optimal hyperparameters can vary depending on the specific dataset, model architecture, and task at hand. The search space for hyperparameters can be vast, and finding the best combination can be computationally expensive and time-consuming. As a result, the choice of hyperparameters is often determined through trial and error, experimentation, and a combination of domain knowledge and experience.

2.2.11 Neural Networks for NLP tasks

For Neural Networks to handle NLP tasks, textual data must be transformed into numerical features to feed to the network. A significant advancement in NLP came when experts discovered that instead of handcrafting algorithms for feature generation, they could train Neural Networks to produce these features for subsequent Neural Networks. The next section elaborates on this approach.

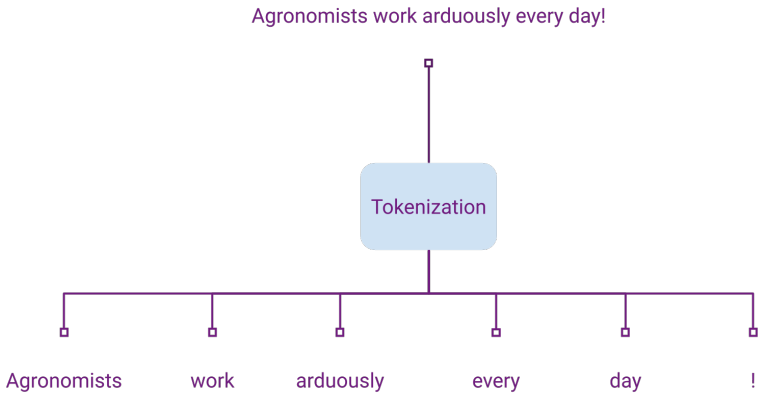


Figure 9: Naive tokenization

2.3 Tokenization and Embeddings

Most NLP tasks for text start with a corpus of documents. The documents may have been manually collected by humans; automatically collected, for example, using web scrapping tools, generating text from templates, or, most recently, sampling outputs from AI tools like GPT-3 (Wang et al., 2021) and ChatGPT (Huang et al., 2023); or constructed using a mixture of human and automatic input.

Considering the ever-changing nature of natural language, the initial phase in Text Classification involves defining the system's Vocabulary. *Tokenization* refers to the act of segmenting text into smaller units, known as *tokens*. As a first approach, one may use the intuitive idea of *word* for tokens, in languages with a morphosyntax similar to English (Figure 9). The collection of unique tokens derived from a text corpus forms the Vocabulary.

However, naive word tokenization[16] can lead to challenges with out-of-vocabulary (OOV) words when presented with new data. A more modern tokenization approach involves using *subword units*. This strategy addresses OOV words by breaking words down into smaller segments. In English, for instance, one might consider prefixes and suffixes as subwords (like `pre-`, `-ing`, `-ed`, and so on), enhancing the NLP system's ability to understand morphology. Nevertheless, instead of manually defining rules to divide words into subwords, there are two widely-used algorithms based on character distribution to create such tokenizers:

- *Byte-Pair Encoding (BPE)* (Sennrich et al., 2016) builds a vocabulary by iteratively merging the most frequent pair of characters or character sequences in a given text corpus. It starts with a character-level vocabulary and merges the most frequent pair of tokens until a desired vocabulary size is reached.

[16] Another simple tokenization technique is *character-level tokenization*, where text is simply divided into individual characters. This tokenization technique falls outside the scope of this work.

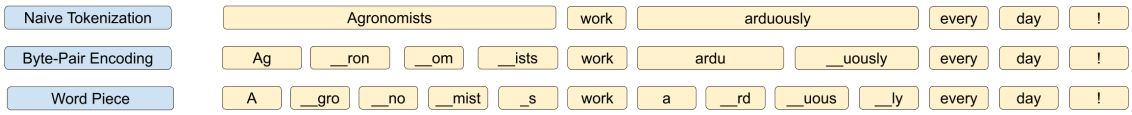


Figure 10: Different tokenization approaches:naive, Byte-Pair Encoding, and Word Piece

- *Word Piece (WP)* (Wu et al., 2016) begins by creating a vocabulary that consists of all the characters found in the training data. It then proceeds to learn a specific number of merge rules. Unlike BPE, Word Piece Tokenization selects symbol pairs that maximize their frequency relative to its constituent symbols, rather than choosing the most frequent symbol pair.

Typically, these algorithms result in different tokenizations for the same text, leading to different vocabularies. Figure 10 displays a sample output from the WP method in Devlin et al. (2019) and the BPE method in Liu et al. (2019) for English content.

Even though there are Text Classification approaches which only consider the distribution of tokens in a corpus, like *Naive Bayes Classification* (Manning and Schutze, 1999), recent breakthroughs in NLP have been enabled by *text embeddings*, which are vector representations of text in high-dimensional space.

Embeddings are intended to capture the semantic relationships between texts, allowing computational models to understand and manipulate textual data. Typically, embeddings serve as *input features* for textual data, which are then fed into ML algorithms (Figure 12).

Word embeddings are a type of text embedding that assign a vector representation to each token in a text. Within Text Classification, while word embeddings are prevalent, Document Classification especially benefits from *Document embeddings*. These embeddings provide a unique vector representation for an entire document, enabling a holistic understanding and categorization of texts based on their full content and context. Essentially, text embeddings ensure that texts with analogous meanings are closely aligned, while those with divergent meanings are distinctly separated (Figure 11).

Text embeddings have undergone significant evolution over the years, with several techniques and models contributing to their development. We present a brief overview of the key milestones in the evolution of word embeddings:

- Early approaches focused on frequency-based representations, such as Bag-of-Words (BOW) one-hot encoding[17] and term frequency-inverse document frequency (TF-IDF). These methods assigned weights or binary values to words based on their occurrence in the corpus, without capturing semantics.

[17] A one-hot vector has all its components null, except one which has value one

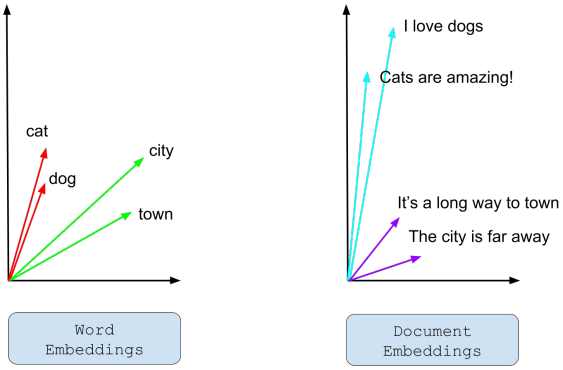


Figure 11: Word and Document embeddings

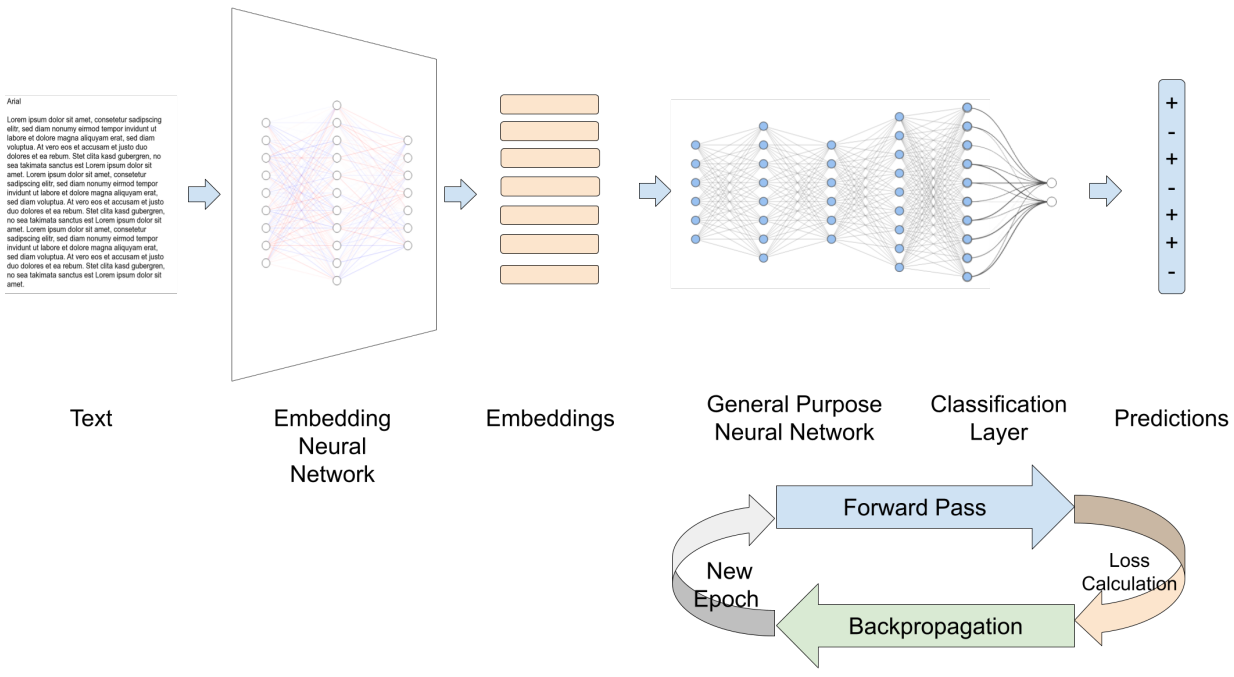


Figure 12: Using Embeddings to train Neural Networks for Classification

- *Word2Vec* (Mikolov et al., 2013) popularized the concept of word embeddings trained using unsupervised learning. These models utilized shallow neural networks to learn word embeddings by predicting neighboring words or contexts.
- Recurrent Neural Network (RNN) architectures, such as GRU (Gated Recurrent Unit), LSTM (Long Short-Term Memory), and Bi-LSTM (bidirectional LSTM), provided a breakthrough by processing sentences sequentially. For example, with *InferSent* document embeddings (Conneau et al., 2017). However, they faced challenges with long-term dependencies and were computationally expensive for longer sentences.
- Contextualized word embeddings introduced the idea of generating word representations that varied depending on the context in which the tokens appeared. Models like *ElMo* (Embeddings from Language Models) (Peters et al., 2018) and OpenAI's GPT (Generative Pre-trained Transformer) (Radford et al., 2018) were able to produce dynamic word embeddings.
- Transformer models, including Google's Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019), OpenAI's GPT-2 (Radford et al., 2019) and GPT-3 (Brown et al., 2020), revolutionized the field by leveraging attention mechanisms and large-scale pre-training. These models introduced contextualized word embeddings on a larger scale and achieved state-of-the-art performance across various NLP tasks.

In this work, our primary focus is on employing the BERT family of models for Text Classification. The details of these models will be explored in the next section.

2.4 BERT Models for Text Classification

A *Language Model* is a Neural Network that can be used for predicting the distribution of tokens, given a corpus. The main goal of a Language Model is to capture the underlying patterns and structure of tokens in the corpus. The architecture for the BERT (Bidirectional Encoder Representations from Transformers) family of models was introduced Devlin et al. (2019) as one capable of learning embeddings aware of the left and right context for each token[18].

> The BERT family of models has gained significant popularity due to its ability to efficiently generate contextualized embeddings for both individual tokens and entire documents at a low cost (Koroteev, 2021).

While the original BERT is a general-purpose model, many tasks benefit from domain-specific knowledge. The multitude of BERT models arises from the need to cater to different languages, domains, tasks, and computational constraints, combined with ongoing research efforts to improve model performance and efficiency.

[18]Hence the name "Bidirectional".

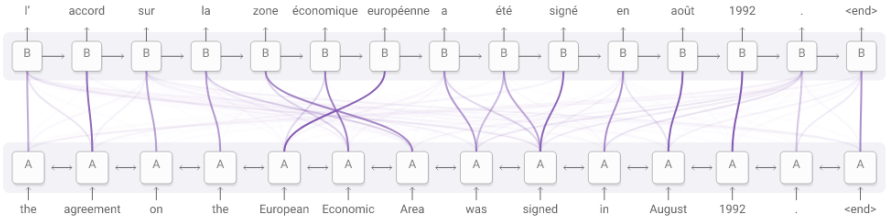


Figure 13: Attention Mechanism for Machine Translation, from Olah and Carter (2016)

Many BERT models are freely and readily accessible and set for deployment across a range of NLP tasks. Given this availability, they will serve as primary tools for this work.

2.4.1 Attention and Transformers

For many years, NLP methods faced challenges with understanding distant relationships within texts. For example, consider the sentence, "The cat, which was adopted from the shelter last month, loves to play.". Here, the subject (cat) and the verb (loves) are spaced apart by several words, making it tricky for systems identifying grammatical roles. Similarly, when translating from, say, Japanese to English, it's crucial to recognize that the Japanese verb typically appears at the end of the sentence.

Initially introduced in NLP for Machine Translation (Bahdanau et al., 2015), the *Attention* mechanism enables the Neural Network to focus on particular words or phrases crucial for grasping the context. It achieves this by employing trainable parameters to determine coefficients between 0 and 1, indicating the significance of each input token to each output token (Figure 13). The part of the network responsible for determining these attention coefficients is termed an *Attention Head*.

Interestingly, attention can be computed between a sentence and itself to evaluate the significance of its various components relative to one another; this process is called *Self-Attention*. The outcome of the Self-Attention mechanism for a text comprising n tokens is an $n \times n$ matrix $A = (a_{ij})$, where a_ij is a value between 0 and 1, denoting the influence of token j on token i (typically, $p_{ij} \neq p_{ji}$). In the original architecture, which we employ in this work, the time and memory usage for Self-Attention grow in a quadratic manner on the text's length ($O(n^2)$). More recent Self-Attention designs aim to reduce computing time Dao et al. (2022).

Attention mechanisms are a fundamental component of *Transformer* models, a type of deep learning architecture that has revolutionized NLP tasks. The original Transformer model, introduced in the "Attention is All You Need" paper by Vaswani et al. (2017), uses multiple Self-Attention Heads, and is primarily used for sequence-to-sequence tasks, such as Machine Translation.

A classical Transformer architecture consists of two main neural modules, the *Encoder* and the *Decoder*. The encoder module takes embeddings as input features, computes self-attention, and produces attention-enriched embeddings. These enhanced embeddings are

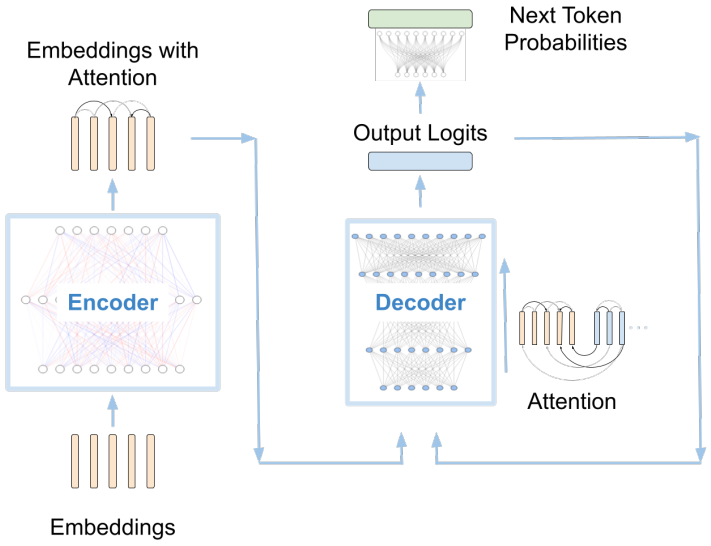


Figure 14: Classical Transformer Architecture

then channeled to the decoder, which outputs the logits of the probabilities of the next token. Notably, the decoder also incorporates its prior outputs as inputs, applies self-attention to them, and then calculates "cross" attention between these and the attention-augmented embeddings from the encoder. The design of this architecture is depicted in Figure 14.

After successful training, the encoder learns the optimal way to generate embeddings for each token considering all the other tokens in the input. At the same time, the decoder learns to produce tokens in a manner informed by the input and its own output. Due to the quadratic complexity of Self-Attention, Transformer architectures have a limited input size.

2.4.2 The training of BERT

The architecture for BERT essentially a series of encoder modules from the traditional Transformer design, layered sequentially[19]. Two distinct versions were introduced by the original auhors for English text: a base model with 12 encoder layers and 12 attention heads, yielding 768-dimensional embeddings, and a large model with 24 encoder layers and 16 attention heads, resulting in 1024-dimensional embeddings. The basic BERT architecture employs a WP tokenizer and can handle up to 512 tokens as input.

For training BERT to generate context-aware embeddings for individual tokens as well as entire sentences, it was subjected to two specific tasks. The datasets utilized for this purpose were Google's BookCorpus and the English portion of Wikipedia.

[19]Hence the name "Encoder Representations from Transformers".

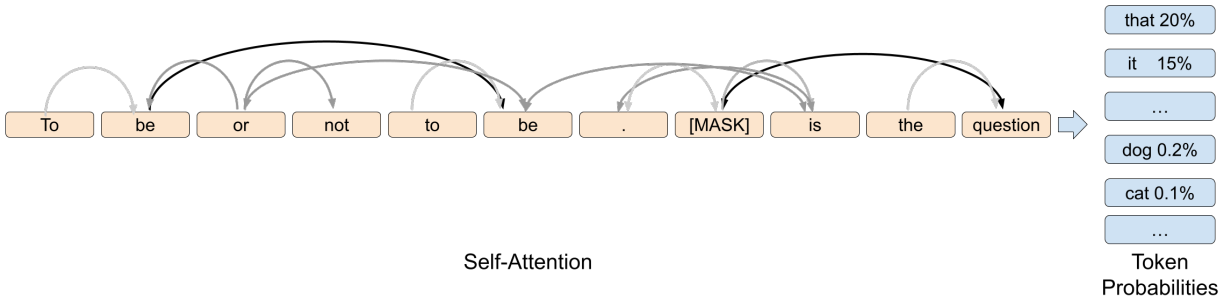


Figure 15: Self-Attention for MLM

Masked Language Modeling To learn embeddings for tokens, the authors introduce the Masked Language Modelling (MLM) task. At its core, the MLM task aims for the model to "fill-in the blank" or to "guess the missing word". For MLM, once a text is tokenized, 15% of the tokens are randomly substituted with a unique `[MASK]` token, symbolizing a blank space. This modified sequence is then processed by the encoder layers, relying on the self-attention mechanisms to provide sufficient contextual understanding (as depicted in Figure 15). To determine the omitted word, the final embedding corresponding to the `[MASK]` token is passed through a $SoftMax$ activation function, yielding a probability distribution over all the tokens in the vocabulary. By contrasting these probabilities with the known token, the cross-entropy loss is computed, which serves as the loss to be minimized. This procedure enables BERT to learn about the token distribution.

Next Sentence Prediction To learn embeddings for sentences, the Next Sentence Prediction task used by the authors involves joining two sentences or phrases and training the model to determine if they appeared consecutively in the training dataset. Specifically, for each example with sentences A and B, there's a 50% chance that B is the actual subsequent sentence to A, and a 50% chance it's a random sentence from the dataset. In order to do this, the authors introduce two special tokens, the `[CLS]` token, for classification, and the `[SEP]` token to separate the texts.

During training, the `[CLS]` token is placed at the start of the text, and the `[SEP]` token is positioned between the two sentences. After processing by the encoders, the `[CLS]` token's embedding undergoes further refinement via a *Pooling Layer*[20], resulting in what's termed the aggregated or *pooled* output. This pooled output aims to encapsulate the entire text's meaning. Ultimately, this output is fed into a classification layer, and the generated probabilities are matched against the known sentence distribution to compute the cross-entropy (as illustrated in Figure 2.4.2). Through this process, BERT is trained to generate embeddings that represent whole sentences.

[20]This layer comprises a linear layer with a tanh activation function

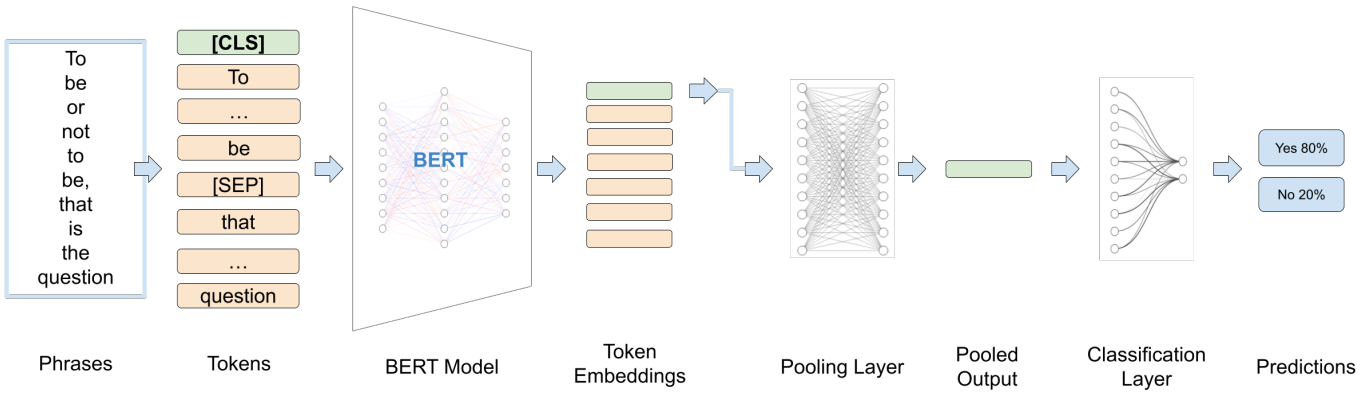


Figure 16: Next Sentence Prediction Task for BERT

2.4.3 Using BERT for Text Classification

The training process for BERT doesn't require the texts to be labeled or categorized, making it an *unsupervised* approach. This approach allowed BERT to learn general text representations that encompass a variety of syntactic and semantic patterns across diverse contexts.

Once the BERT model is trained, the conventional method for Document Classification using BERT involves feeding the text to the model[a]. Then, the pooled output serves as the document's embedding. This embedding is then fed to a classification layer, which is further trained for the specific task at hand (as illustrated in Figure 17).

[a] keeping the second sentence empty after the `[SEP]` token

Finally, the parameters of BERT models may be adapted to new data and tasks. In Section 6, we detail the techniques used for fitting BERT to our task.

2.5 Evaluation Metrics for Classification

After training a classifier for a Classification task, including Text Classification, the subsequent phase is to assess its performance. This involves leveraging a dataset with predefined class labels and contrasting the classifier's *predicted* with these *true* labels.

Appendix A contains the motivations and definitions for five classic metrics for evaluating binary classifiers: accuracy, precision, recall, F_1, and AUC. In this section, we take a step further and discuss the F_β score. The F_β score extends the concept of F_1 by introducing a parameter β, which allows us to control the trade-off between precision and recall. A

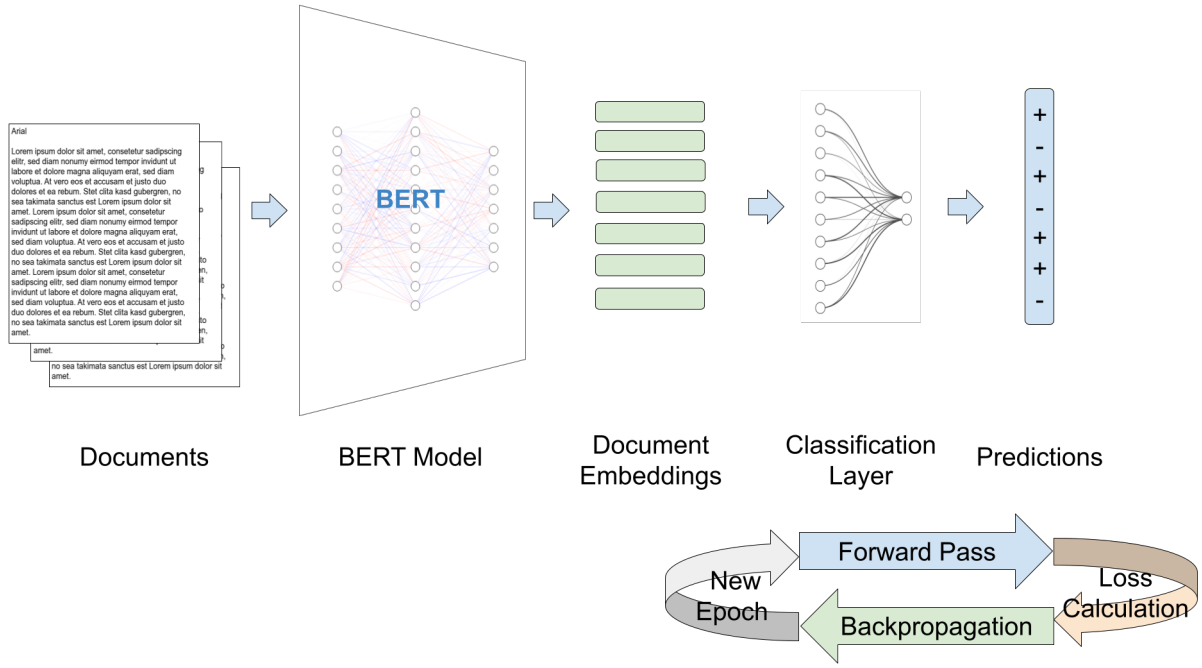


Figure 17: Using BERT Embeddings for Document Classification

higher value of beta emphasizes recall, while a lower value emphasizes precision. The F_1 score is a special case of the F_β score when β is set to 1.

$$F_\beta = (1 + \beta^2) \frac{P * R}{\beta^2 * P + R} \tag{7}$$

Example: If β is set to 2, the F_2 score will give more two times more weight to recall than precision, making it suitable for tasks where recall is more important than precision, such as ours.

Our task involves distinguishing between events representing health risks and impacting agriculture (the positive class, "Relevant") from harmless events (the negative class, "Irrelevant"). In this context, it becomes vital to identify as many true positive cases as possible, as they represent actual risks with potential consequences for agriculture.

Once our system flags a potential risk, subject experts will be notified to make the final assessment. A low precision would indicate that only a small fraction of the documents presented to the experts are actually relevant. Conversely, a low recall would imply that many actually relevant documents would be overlooked by the system, requiring experts to reexamine and asses the original data, a costly and time consuming task.

As a result, our focus leans more towards improving recall over precision. Therefore, we choose to assess our system using the F_2 score.

3 Related Work

As in almost all industries, Neural Networks-based AI is being adopted in Agriculture (Javaid et al., 2023). Some of the most prominent uses of AI and Deep Learning have been the automatization of decisions or alerts produced by sensors (Wongchai et al., 2022); and the use of the Internet of Things[21] to integrate sensors and software, in order to warn farmers of possible issues with their crops (Bu and Wang, 2019).

On the other hand, the application of NLP to Agriculture, and especially to Epidemiological Surveillance, has been rather timid. A popular approach has been to use data mining from social media to gather textual data (Turenne et al., 2015). Twitter has been used as a source of crowdsourced knowledge about ecological events (Aramaki et al., 2011; Charles-Smith et al., 2015). However, there have been concerns about the reliability of social media data for Epidemiological Surveillance, as the public and the agricultural workers may lack the specialized knowledge to accurately assess risks (Welvaert et al., 2017).

Moreover, until recently, the advancements in Neural Network-based Language Models didn't significantly influence NLP applications in Epidemiological Surveillance (Alrowili and Shanker, 2021; Hakala and Pyysalo, 2019). This landscape shifted during the COVID-19 Pandemic when both computer vision and NLP emerged as vital tools for diagnosis, prevention, and notably for our work, monitoring the spread of the disease. Still, these technological strides haven't been extensively applied in the field of Plant Health Surveillance.

Shankar et al. (2020) introduced the idea of using social media data to monitor crop health. They collected a corpus of 5530 Tweets and used Binary Text Classification to detect "agriculturally relevant tweets", using embeddings and a Support Vector Machine Classifier, and obtained an accuracy of 86.5%. Even though the authors used established embedding techniques, namely *Word2Vec* and *Doc2Vec*, nowadays, far more powerful Neural Networks exist. Based on their provided confusion matrix, additional metrics for their results can be deduced as: Precision at 72.4%, Recall at 21.1%, F_1 at 32.7%, and F_2 at 24.6%. The stark contrast between their reported accuracy and the other metrics likely stems from the skewed nature of their dataset, which consists of 24% relevant tweets and 76% that are not. Our research offers a marked improvement over these findings.

To the best of our knowledge, the research most aligned with ours is by Jiang et al. (2022), who introduce *ChouBERT*, a BERT model tailored for Plant Health Surveillance. *ChouBERT* emerges from adapting the French BERT model, *CamemBERT* (Martin et al., 2020), to fit a custom Plant Health Dataset. This dataset comprises Tweets and French French Plant Health Bulletins[22]. Notably, the authors only disclosed the Precision metric for *ChouBERT*, which stands at 88.7%. However, since *ChouBERT* is designed exclusively for French content, its applicability to International Plant Health Monitoring remains limited.

[21] `https://en.wikipedia.org/wiki/Internet_of_things`
[22] `https://agriculture.gouv.fr/bulletins-de-sante-du-vegetal`

4 The VSI Dataset

Within this Section, we provide a detailed description of the VSI Dataset. As mentioned in Section 1.3, the PESV Platform collects information on risks to Plant Health or events that could potentially affect agriculture. The VSI experts achieve this by surveying online documents, which they subsequently screen and collect into the VSI dataset.

4.1 Dataset Collection

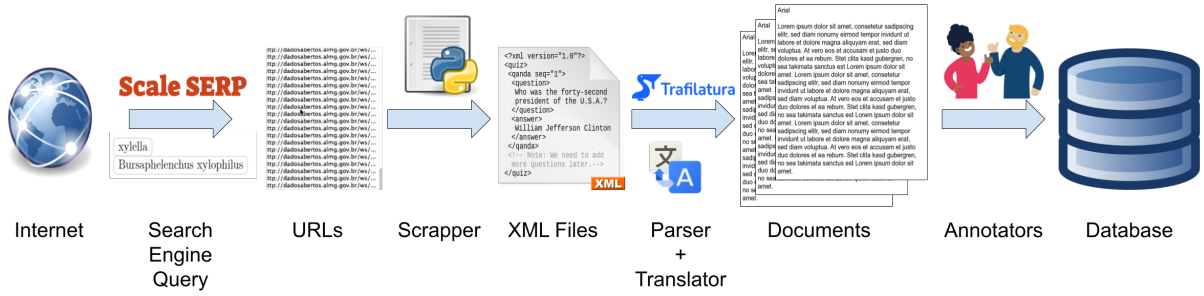


Figure 18: VSI Data Collection Pipeline

The main tool for collecting documents is the web scrapping tool *ScaleSERP*[23]. Every week, several queries are sent to various services, including the Google Search Engine, using the keywords and key phrases related to pathogens of interest (Table 27).

As part of the VSI pipeline, shown in Figure 18, the websites corresponding to the URLs obtained from *ScaleSERP* are downloaded using a Python script. For each website, its source (in HTML) is downloaded, and the tool Trafilatura (Barbaresi, 2021) parses it to XML/TEI, while attempting to discard "non-contents", such as menus, headings, ads, etc. Then, Trafilatura takes the XML/TEI and attempts to extract a title and an abstract from the metadata[24] and the full text from the XML body.

Given that these articles are extracted from public websites, they remain in their original language. The Google Translate API is used to assist the annotators with foreign-language articles. This API also automatically detects the document's language. Due to budget constraints, only the Title is translated into English.

Since documents are added each week, the VSI database is in constant construction. For our work, we utilized with the documents collected from the 11th of July 2022 to the 16th of April 2023, that is, the data collected over 39 weeks, comprising approximately 35,000 documents, which corresponds to more than 800 documents per week. We focus on a subset of the VSI database, specifically its textual data, to obtain the VSI dataset (Table 3).

[23] `https://www.scaleserp.com/`

[24] The title and the abstract are extracted from the `metadata.title` and `metadata.description` fields, respectively.

Content Source	Tool
Title	Trafilatura
Abstract	Trafilatura
Full text	Trafilatura
Translated Title	Google Translate

Table 3: Sources of text content in the VSI Dataset

4.2 Dataset Annotation

There are four VSI experts dedicated to annotating the dataset. Regrettably, while constructing the dataset, there was no record maintained to track the identity of the annotator responsible for labelling each document. For this reason, we are not able to provide any inter-annotator agreement measures on this dataset (Artstein and Poesio, 2008).

For annotation, the results of the VSI pipeline mentioned above are shared with the VSI experts. They utilize the interface shown in Figure 19 to inspect the results for various fields, including the *Date de publication* (publication date), *Auteurs* (authors), *Organism nuisible* (pathogen), etc. Additionally, they manually input information in other fields such as *Sujet* (subject), *Fiabilité* (reliability), and come up with a *Titre* (title). It is important to note that the title written by the VSI experts is usually different from the one extracted by Trafilatura (for example, the wording may vary).

After inspection, we found that that annotators exclusively provide titles for articles they find relevant, that is, there are no titles for irrelevant articles. Since our objective is to automate the process of identifying relevant articles, we disregard the titles provided by the annotators. Instead, we rely solely on the titles generated by Trafilatura and Google Translate, which are stored even in the case of the documents being rejected.

Special attention must be given to the *Sujet* (subject), as this field will be crucial for preprocessing the dataset (see Section 5.3).

When the VSI experts encounter an event that catches their attention (that is, that they consider **relevant**), they assign a **subject** to the respective document and utilize the interface presented in Figure 20 to allocate a subject ID and description to the document. Each subject corresponds to a specific health risk event that was reported during the weeks before the experts reviewed the documents. It is possible for multiple documents to share the same subject. In the case of documents considered **irrelevant** to the monitoring of plant health, the subject field for the document is left empty.

Subsequently, after a document is deemed relevant by the VSI experts, it is officially

published in the bulletin of the PESV Platform[25] and can be accessed through the VSI Document Search Engine[26]. Some sample entries containing all sources of content can be found in Table 4.

[25] https://plateforme-esv.fr/bulletins_et_points_sur_VSI
[26] https://plateforme-esv.fr/moteur-de-recherche-vsi

Entry ID	Title	Translated Title	Abstract	Full text	Subject ID
68	Xylella, da giugno 47 nuovi casi	Xylella, 47 new cases since June	In Puglia la Xylella fa...	Xylella, da giugno 47 nuovi casiIn Puglia...	4279
305	Agro - EL HERALDO - Edizione digitale	Agro - EL HERALDO - Digital edition	EL HERALDO - Edizione digital...	El presidente de la Federación del Citrus...	None
320	Misure fitosanitarie di controllo della Popillia japonica - 2022	Phytosanitary measures to control Popillia japonica - 2022	Il Servizio Fitosanitario Regionale per...	Popillia japonica è un insetto originario del...	4286
177	Traps set to catch invasive, destructive fruit fly in Pinellas County	Traps set to catch invasive, destructive fruit fly in Pinellas County	Florida Agriculture leaders are on...	PINELLAS COUNTY, Fla. — Florida Agriculture leaders...	None
159	Preocupan los efectos del cambio climático en la variedad picual	The effects of climate change on the picual variety are of concern	Podría ver reducido su rendimiento...	1. ¿Qué son las cookies? La Web...	None
89	Frantoi di Puglia danneggiati dalla Xylella, in arrivo 35 milioni di euro	Puglia oil mills damaged by Xylella, 35 million euros on the way	I frantoi oleari della Puglia...	I frantoi oleari della Puglia potranno presto...	None
82	Dal Mipaaf un miliardo per i terreni colpiti da Xylella	From Mipaaf one billion for the land affected by Xylella	Dal Mipaaf un miliardo per...	Il V bando per i Contratti di...	None
61	Le ministère veut alerter sur les espèces invasives	The ministry wants to warn about invasive species	Le ministère de l'Agriculture lance...	« Plantes en danger » Le ministère...	None
78	Xylella, casi di Polignano portano più a nord il limite della Puglia	Xylella, cases of Polignano bring the limit of Puglia further north	Il piano di Emergenza evidenzia...	In Puglia e Basilicata SOS AGRICOLTURA Resta...	4279

Table 4: Sample entries from the VSI dataset

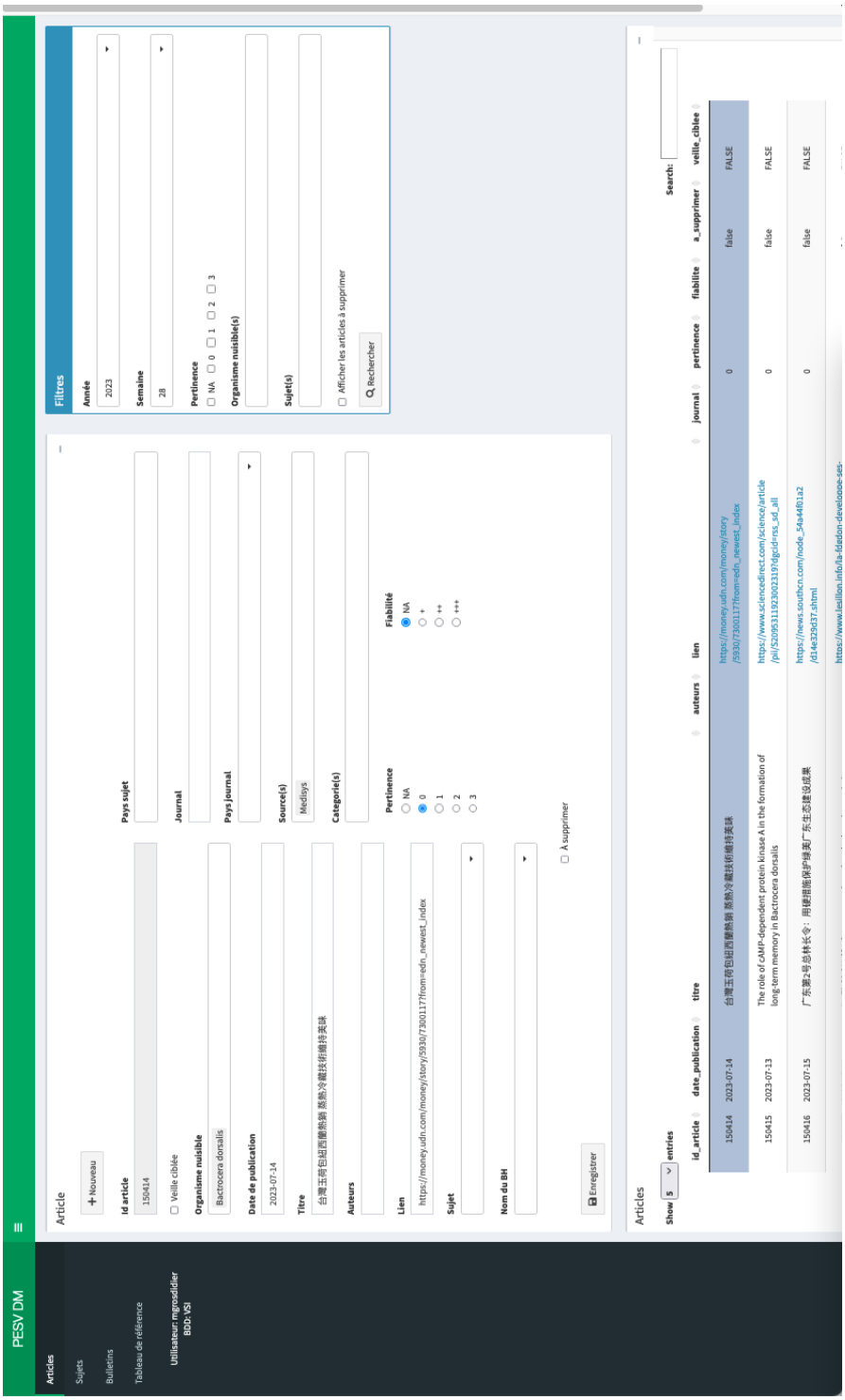


Figure 19: VSI Annotation Interface for Articles

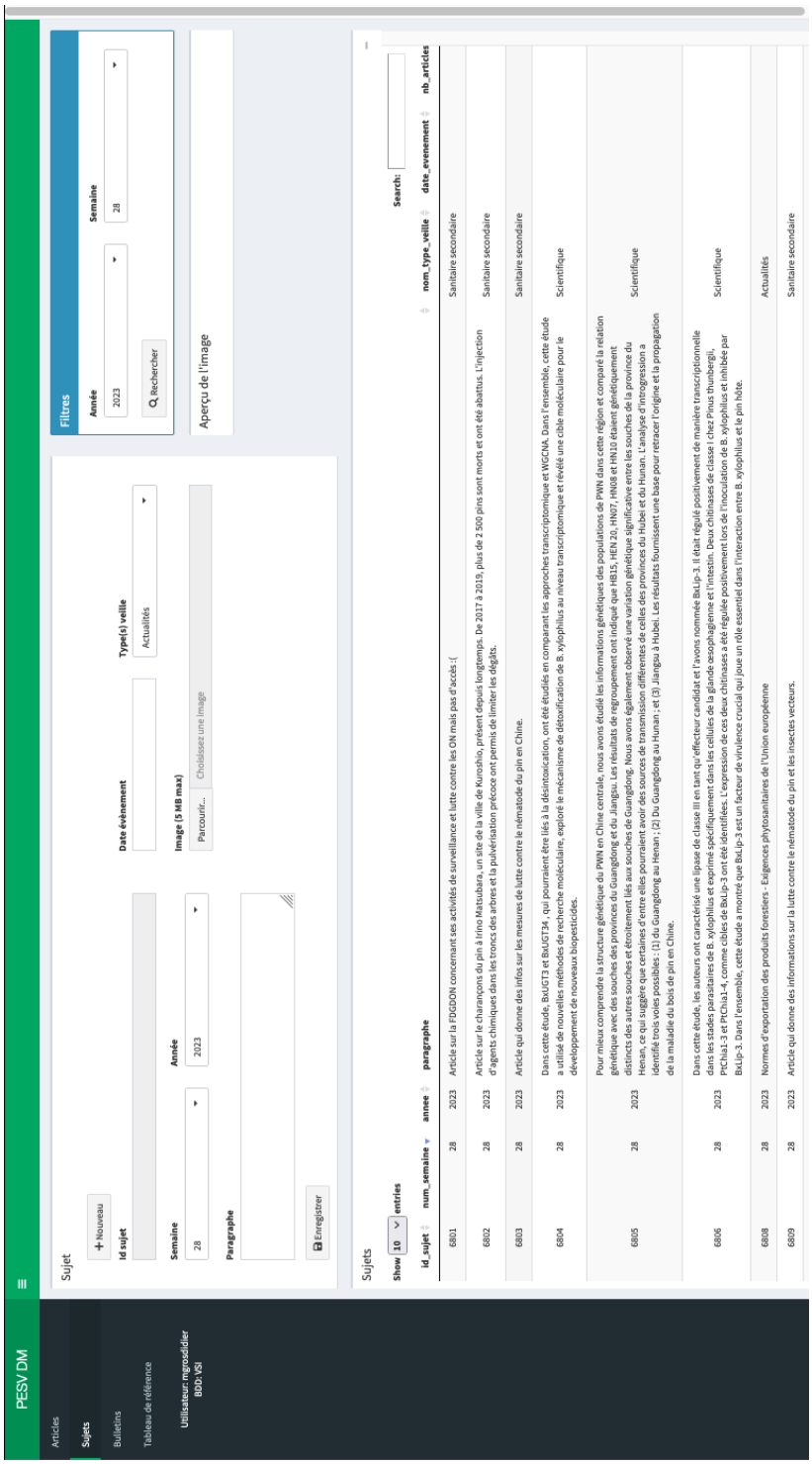


Figure 20: VSI Annotation Interface for Subjects

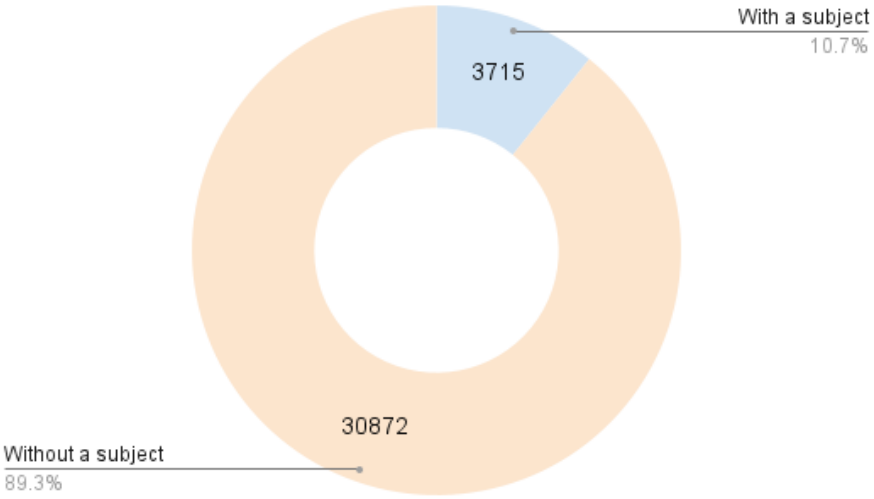


Figure 21: Distribution of subjects in the VSI dataset without preprocessing

4.3 Dataset Statistics

There are 34,587 entries in the dataset, out of which 3,715 have an assigned subject and 30,872 do not, producing a highly unbalanced dataset (Figure 21).

The language automatically detected by the Google Translate API on the Title can be used to approximately study the distribution of the entries in the dataset. In Figure 22 it can be seen that around one third of the entries are in English, followed by Italian, Spanish, and French. There is a considerable amount of failed translation attempts, for around one fifth of the entries. The complete data on documents per language is available in Appendix B.3.

The presence of some Latin documents is surprising. Upon inspecting the data, we discovered that this occurrence was due to scrapped Titles with several scientific names (Table 28 in the Appendix).

Often, the Trafilatura scrapper is not able to retrieve content for the Title, Abstract, and Full text of a website, or the Google Translate API fails. Thus, there are several entries without content for all fields. If we check the unique entries for each source of content, we can see that there are numerous duplicate entries, since there are at least around 20,000 unique entries per source of content (Figure 23).

As theTranslated Title is derived from the Translated Title, and due to the possibility of failure in the Google Translate API, the number of unique entries for the Translated Title is lower than that for the Title. At the same time, the Abstract field has fewer unique entries than all the other Content Sources, while the Full text has the most. This may be explained by the fact that, usually, the creators of a website omit filling the metadata, and when they do, they frequently include only a title and not a description, while most times, the body of the website will be available.

Using naive tokenization (splitting on whitespace), we can study the distribution of the

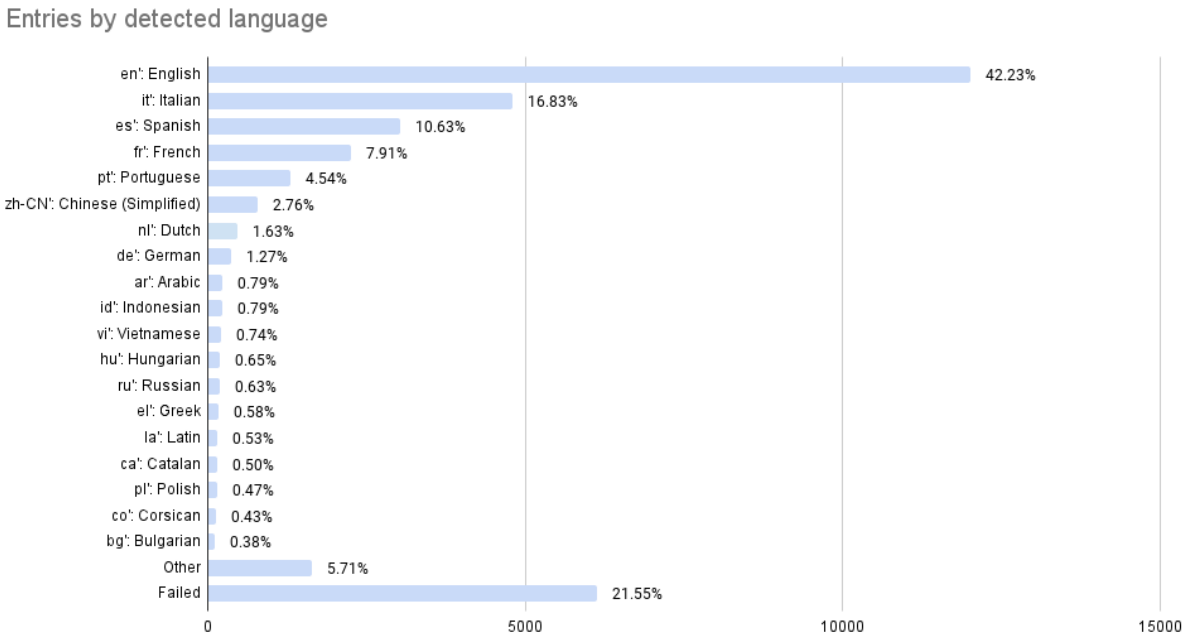


Figure 22: Language distribution of entries in the VSI dataset

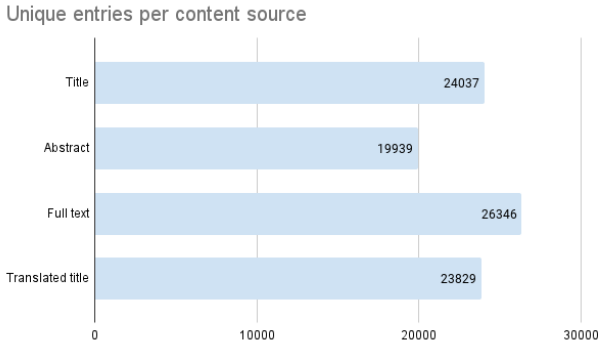
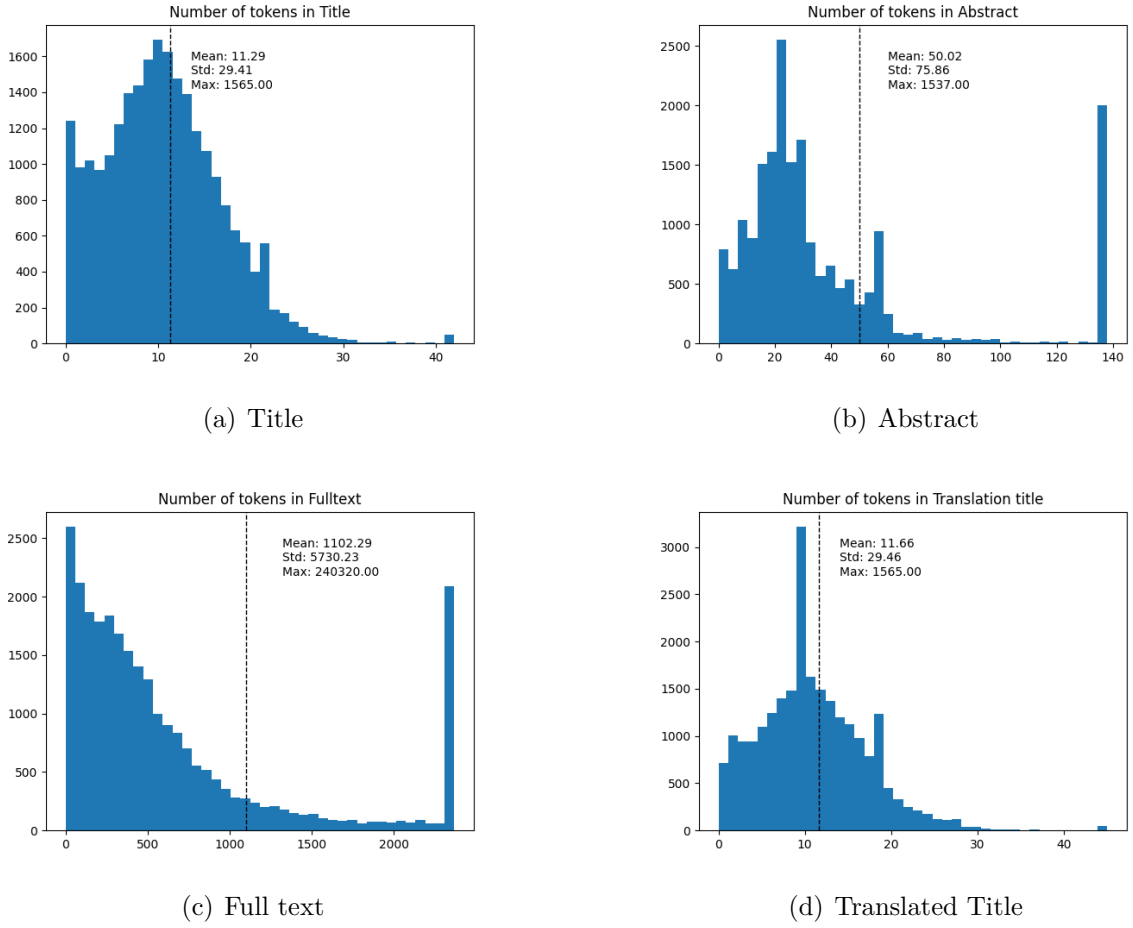


Figure 23: Unique entries per source of content in the VSI dataset

tokens for all sources of content. In the histograms in Figure 24 the longest documents have been grouped together to facilitate plotting [27]. As one can see, the Title and Translated Title have a very similar distribution, both are shorter than the Abstract, while the content from the Full text is the longest.



(a) Title

(b) Abstract

(c) Full text

(d) Translated Title

Figure 24: Length (in n. of tokens) of the Content Sources in the VSI dataset

4.4 Dataset Issues

Upon examining the data, we encountered numerous issues that we proceed to describe and which will be addressed during preprocessing (Section 5).

[27] This explains the long bars at the end of the right tails of the histograms.

4.4.1 Duplicate Entries

As mentioned in Section 4.3, there are some duplicate entries in the dataset (Figure 21 VS Figure 23). According to the VSI experts, often the Google Search for a query will return the same results from one week to the next, and this introduces duplicate entries.

4.4.2 Scrapping Failures

On some instances, the Trafilatura scrapper either fails or is blocked by website servers that do not allow bots. This introduces error messages to the dataset, some samples of which can be seen in Table 5.

Error Message
Not Found
404
Page Not Found
Your data. Your experience.
Loading...
None
nan
JavaScript n'est pas disponible.
JavaScript is not available.
Please update your browser
Do you accept cookies ?
Verify you are not a robot
Discuz! Database Error

Table 5: Some error messages in the VSI dataset

4.4.3 Scrapping errors

In some other instances, the web scraper successfully extracts content from the website, but the parsing algorithm performs poorly, retrieving only the headers or the website's name (Tables 6 and 7).

This type of scrapping error introduces inconsistencies into the VSI dataset, as multiple documents end up having the same text for several Content Sources, as in Table 8[28].

[28] X-MOL is a Chinese search engine for scholars.

Header
Search results
Welcome
Advance articles
shop
Green Blog
Pesquisa (*Query*)
Link Diretti (*Direct Links*)
Búsqueda (*Query*)
Actualidad (*Current events*)
Browsing
Project Listing
Research articles
Pagamenti Online (*Online Payments*)
Notícias (*News*)
Category: events
News

Table 6: Some headers parsed by Trafilatura

Website name
Wall Street Journal
Bloomberg
TikTok
Facebook
Portal Embrapa
Argentina.gob.ar
Notizie dal Comune (*Town News*)
zmianynaziemi.pl
Pakistan Journal of Zoology
SAARC Journal of Agriculture
MONTSAME News Agency
ORCID

Table 7: Some website names parsed by Trafilatura

Entry ID	Title	Translated Title	Abstract	Full text	Subject
12895	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	None
13860	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	4827
14614	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	4874
15033	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	4896
15271	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	None
15683	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	4930
16355	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	4984
16472	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	None
19065	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	None
19541	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	None
21913	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	5186
22152	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	None
23491	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	None
23745	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	None
25632	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	None
26284	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	5448
29990	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	5694
30419	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	5686
30456	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	None
30994	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	5743
31122	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	None
31543	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	None
32019	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	5757
32096	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	5799
32279	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	None
32495	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	None
33089	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	5829
34431	X-MOL	X-MOL	Web site created using create-react-app	X-MOLYou need to enable JavaScript to run this app.	None

Table 8: Inconsistencies introduced by scrapping errors

4.4.4 Data Noise

Even in those cases when Trafilatura is able to retrieve content from the website, there are some expected types of noise in the data, considering it comes from the internet. Namely, the following (Table 9) :

- URLs
- HTML Tags
- Emojis
- Encoding errors
- Generic noise (different quotation or division characters, strings containing only dates, etc…)

Furthermore, there exists a source of noise that may not be immediately apparent: occasionally, the content concludes with the website's name, leading to different entries that vary by only a few tokens as a suffix (Table 10).

4.4.5 Annotation Inconsistencies

Finally, human error also introduces inconsistencies into the VSI dataset. In some cases, there are entries with the exact same content, with different or no assigned subjects (Table 11). After consulting with the VSI experts, they explained that this is due to two factors:

- Different annotators labeling the same content.
- After a subject loses relevance (see Section 4.1), the annotators usually ignore articles related to it, which is effectively equivalent to assigning no subject.

Due to all the issues mentioned, one must take the statistics in Section 4.3 with a grain of salt. In particular, naively checking the entries for the presence of an assigned subject (as in Figure 21) may lead to very similar or exactly equal content labelled in two different categories, thus rendering the classification task significantly more challenging. These issues will addressed during preprocessing (Section 5).

URLs
Regione attiva piano anti Popillia japonica - www.lombardianotizie.online - Conflombardia
SciELO Argentina - www.scielo.org.ar
https://twitter.com/julia_lopez9/status/1601582135923863553/photo/1
Ornitho.it homepage - www.ornitho.it
Dzienna porcja warzyw - Nasza Rola - www.naszarola.pl

HTML Tags
Fall armyworm in Europe: how can we use biocontrol?
Xylella, CIA Puglia: «Presidente Emiliano se ci sei batti un colpo»
Japankäfer Popillia japonica
Categoría: Revista
Corteva Agriscience introduces seed treatment product Dermacor® in Brazil
Current Research on Fusarium
[Other] Highly Sensitive and Rapid Detection of Citrus Huanglongbing Pathogen (Candidatus Liberibacter asiaticus') Using Cas12a-Based Methods

Emojis
Esther Ogunbayo on LinkedIn: https://lnkd.in/dEwvQD_A Adeoye Opeyemi kindly like, follow and repost 😀
[⭐] on TikTok
"#Leonardo #italy🇮🇹 #puglia #taranto #volgopuglia #volgoitalia | Lion sculpture, Sculpture, Taranto"
👩‍🔬🔬🧪Gen L de resistencia al Virus Rugoso del Tomate (ToBRFV)🦠 | Agrocultivos TV innovando

Encoding Errors
<ººÇ¥ 3> ¹®Àå(Á¦6Á¶°ü · Ã).
âç½çº¢âé³éé³éâå ¨å½å´æâ é²æ²»ä¸è½âä¸æäºäºâ
æµè¥¿ä¹¡æææºåå¤æ ç®¡æ¤ è¶ åç¾æ£µç¾å¹´å¤æ âèæææä¾â
çå±±äººé¿åºâç§æç¿¼â æ¹åå²³é³å¿âçºc è¢ç« ââ®æ¤âæ£®æç»¿â
Î¤Î¿ ÎÎÎ©Î¤ÎÎ ÎÎÎ®ÏÎ · Ï Î´Î¯Î½ÎµÎ¹ ÏÏÏÏÎ¿Ï Ï Î±Î½ÏÎ¹Î¼ÎµÏÏÎ¹ÏÎ · Ï ÏÎ¿Ï Î´Î¬Îºº Î¿Ï ÏÎ · Ï ÎµÎ»Î¹Î¬Ï - §Î±Î½Î¹ÏÏÎ¹ÎºÎ± ÎÎÎ±
å®¢æ · åè¡¨æç®

Table 9: Noise in the VSI dataset

Title
Cousin of crop-killing bacteria mutating rapidly
Cousin of crop-killing bacteria mutating rapidly - MixPoint
Cousin of crop-killing bacteria mutating rapidly - My Droll
Cousin of crop-killing bacteria mutating rapidly - Newsprepare
Cousin of crop-killing bacteria mutating rapidly - Sky News: The Latest News from the World
Cousin of crop-killing bacteria mutating rapidly - timetotimes.com
A look at 2023's new garden plants
A look at 2023's new garden plants | Times News Online
Modeling climate change impacts on potential global distribution of Tamarixia radiata
Modeling climate change impacts on potential global distribution of Tamarixia radiata - PubMed
Portugal detecta Xylella en cítricos por primera vez en la UE y en Italia se consolida el foco de 'Mosca oriental'
Portugal detecta Xylella en cítricos por primera vez en la UE y en Italia se consolida el foco de 'Mosca oriental' - Agrodigital
Portugal detecta Xylella en 75 especies vegetales
Portugal detecta Xylella en 75 especies vegetales - FruitToday
Tornano le Giornate Fai di Primavera: 750 luoghi aperti in tutta Italia
Tornano le Giornate Fai di Primavera: 750 luoghi aperti in tutta Italia - LegnanoNews
Giornate Fai di Primavera. I segreti dell'edizione 2023. Da Bolzano alla Sardegna, 750 luoghi da scoprire
Giornate Fai di Primavera. I segreti dell'edizione 2023. Da Bolzano alla Sardegna, 750 luoghi da scoprire - Notizie italiane in tempo reale!
Milleproroghe, soddisfazione di Confagricoltura per gli emendamenti a favore del settore primario - Agricolae
Milleproroghe, soddisfazione di Confagricoltura per gli emendamenti a favore del settore primario - Comunicati | Confagricoltura

Table 10: Similar entries with website names as a suffix

Entry ID	Title	Subject
4662	Cousin of crop-killing bacteria mutating rapidly	None
5885	Cousin of crop-killing bacteria mutating rapidly	4472
58	Danger pour les végétaux : première détection de la bactérie Xylella fastidiosa dans le Gard	None
850	Danger pour les végétaux : première détection de la bactérie Xylella fastidiosa dans le Gard	4259
26873	Commodity risk assessment of ash logs from the US treated with sulfuryl fluoride to prevent the entry of the emerald ash borer Agrilus planipennis	5524
27196	Commodity risk assessment of ash logs from the US treated with sulfuryl fluoride to prevent the entry of the emerald ash borer Agrilus planipennis	5530
322	Anche a Varese l'invasione della Popillia Japonica, l'insetto devastatore di campi e giardini	None
335	Anche a Varese l'invasione della Popillia Japonica, l'insetto devastatore di campi e giardini	4286
343	Anche a Varese l'invasione della Popillia Japonica, l'insetto devastatore di campi e giardini	4286

Entry ID	Abstract	Subject
13997	A CABI-led study involving 57 scientists from 46 different institutions …	4829
24902	A CABI-led study involving 57 scientists from 46 different institutions …	None
4662	A bacterial species closely related to deadly citrus greening disease …	4472
5388	A bacterial species closely related to deadly citrus greening disease …	None
22658	Ministry of Agriculture activated batteries in the fight against xylella.…	5245
24067	Ministry of Agriculture activated batteries in the fight against xylella.…	5331

Entry ID	Full text	Subject
32437	Altre 23 piante infette dal batterio Xylella fastidiosa subsp. pauca ceppo ST53 sono state individuate tra Fasano (Brindisi) e Castellana Grotte (Bari), …	5781
32712	Altre 23 piante infette dal batterio Xylella fastidiosa subsp. pauca ceppo ST53 sono state individuate tra Fasano (Brindisi) e Castellana Grotte (Bari),…	5820

Table 11: Annotation inconsistencies in the VSI dataset

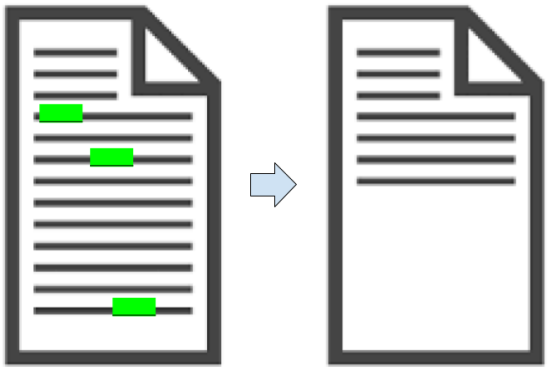


Figure 25: Extracting phrases with Keywords

5 Preprocessing the VSI dataset

In this Section, we outline the approaches taken to resolve the challenges found in the VSI Dataset. After reviewing the problems described in Section 4.4, we adopted various tactics to achieve a preprocessed dataset suitable for Text Classification.

5.1 Leveraging Keywords

Leveraging the available keywords and key phrases used for the Search Engine queries used to construct the dataset (Table 27), we obtained four more sources of content.

Using the Python package `NLTK` (Bird and Loper, 2004), we divided each document into phrases and selected those containing at least one of the keywords or key phrases. This process was carried out for both the Abstract and Full text sources of content (Figure 25). However, considering the diverse range of languages in the documents, it was not always apparent whether we should discard the content entirely or retain it. For example, even if the keyword is not present in the text, maybe its translation is; or we could be facing a language written in a non-Latin script. Consequently, we pursued two different approaches: in one case, we discarded the original content (O.C.), and in the other, we retained it, resulting in four new sources of content (Table 12).

Figure 26 offers a comparison of the number of unique entries per content source. In both cases, keeping only the phrases containing keywords dramatically reduces the size of the data, especially for the Abstract. Possible reasons for this will be explained in Section 5.2. Note that, in the case of keeping the original content, there are slightly fewer unique entries for both the Abstract and the Full text ($19,939 - 19,798 = 141$ and $26,346 - 24,447 = 1,899$, respectively). This may be explained by the fact that some data noise may be removed by the sentence extraction process, especially those cases where several entries differ by a small amount of tokens (see Table 10).

Content Source	Origin
Title	Trafilatura
Abstract	Trafilatura
Full text	Trafilatura
Translated Title	Google Translate
Phrases with Keywords (Abstract)	Preprocessing
Phrases with Keywords + O.C (Abstract)	Preprocessing
Phrases with Keywords (Full text)	Preprocessing
Phrases with Keywords + O.C (Full text)	Preprocessing

Table 12: Sources of text content in the VSI Dataset after preprocessing

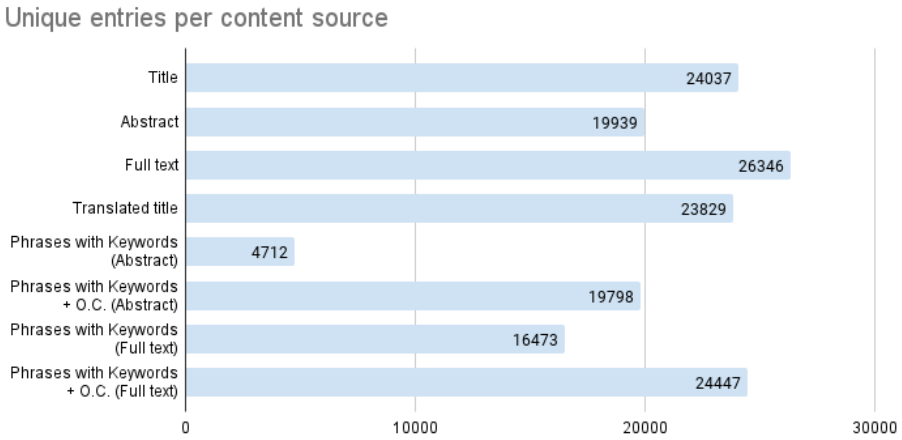


Figure 26: Unique entries per source of content in the VSI dataset

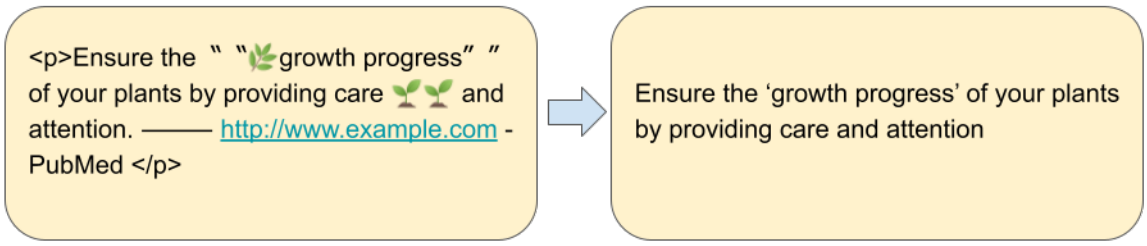


Figure 27: Removing noise from multilingual strings

5.2 Data Cleaning

5.2.1 Noise Removal

To address the text noise mentioned in Section 4.4.4, we use regular expressions to:

1. Remove characters not in any human alphabet. This handles encoding errors.
2. Remove emojis, URLs, HTML tags, hours, dates, extra white spaces.
3. Remove punctuation at beginning and end of the string.
4. Remove digits at beginning and end of the string.
5. Standardize quotation characters into the ASCII simple quotation character (’).
6. Standardize hyphen-like characters into the ASCII hyphen (-).
7. Remove sentence suffixes that may be the name of the website, by deleting short text (≤ 3 tokens) that follow an ASCII hyphen or bar (|) near the end of the string.

See Table 31 in the Appendix for some examples of the regular expressions used to clean the multilingual strings. Figure 27 shows the effect of string cleaning on an example text.

5.2.2 Deleting Error Messages

Fortunately, error messages in the dataset are very consistent. Given that they do not provide information on the relevance of a document, we may delete them using regular expressions. See Table 30 in the Appendix for some examples of the regular expressions used to remove the error messages.

In Table 13, we present the number of error messages identified using our regular expressions. Notably, the proportion of error messages generally hovers around one-fifth of all entries. In 18.50% ($\approx 20\%$) of cases Trafilatura completely fails to obtain any content from the website. This proportion almost doubles ($36.50 \approx 40\%$) for the Abstract. This

Content Source	Error Count	Percentage of Raw dataset
Title	7224	20.89%
Abstract	12624	36.50%
Full text	6609	19.11%
Translated Title	7232	20.91%
All Content Sources	6399	18.50%

Table 13: Error Message Count per Content Source in the VSI dataset

discrepancy in the Abstract could be attributed to the common occurrence of missing metadata, as discussed in Section 4.3.

5.2.3 Handling scrapping errors

In order to handle scraping errors from Trafilatura, such as obtaining the name of the website or certain HTML headers, our first approach was to use regular expressions to filter them out. After removing the error messages from all sources of content, we proceeded to count the occurrences of each entry and ranked them in descending order. Given that our dataset contains approximately 35,000 documents, we focused on entries that appeared at least twice.

Although we attempted to identify common patterns and create regular expressions to match frequent entries, this approach proved unsuccessful. It only filtered around 500 unique entries, accounting for a mere 2% of all unique entries in the best case (Abstract). Consequently, it became apparent that several scraping errors likely appeared only once in the dataset, rendering our previous strategy ineffective.

Nevertheless, along this examination, we noticed that scraping errors tend to be short in length (see Tables 6 and 7). As a result, we devised a new filtering strategy based on the length of the entries. However, simply counting naive tokens was insufficient, as languages like Chinese often lack whitespace in their text. Therefore, we needed a more robust algorithm to determine when a string is considered suspiciously "too short."

Our heuristic to filter by length consists in:

1. First, the string is split on white spaces.
2. Then, if there is only one token, and the string has less than 20 characters, it is considered "too short".
3. Next, if there are less than four tokens, the string is considered "too short"
4. Finally, in all other cases, we keep the string.

Upon removing error messages, we observed that parsing errors occur in entries for the Title and the Translated Title approximately 1 in 7 times ($\approx 16\%$), as can be seen in Table 14. The Abstract and Full text are rarely short ($\leq 3\%$), and very few entries have more than one content source that is too brief ($\leq 3\%$).

Content Source	Short Message Count	Percentage of Raw dataset
Title	5691	16.45%
Abstract	1023	2.96%
Full text	181	0.52%
Translated Title	5306	15.34%
All Content Sources	17	0.05%
Title & Abstract	589	1.70%
Title & Full text	141	0.41%
Full text & Abstract	50	0.06%

Table 14: Short Message Count per Content Source in the VSI dataset

5.3 Resolving Inconsistencies and Duplicate Documents

The initial preprocessing steps focused solely on cleaning the textual content of the dataset. Now, our attention turns to addressing the labeling challenges, specifically, instances where entries have identical content but differ in subject assignment or lack a subject (as in Tables 8 and 11).

Considering our main goal is to automate the identification of articles relevant to the VSI experts, where subject assignment signifies their interest, we propose the following strategy:

> For each of the eight content sources, we group the dataset based on the text content. If a particular text has been assigned a subject at least once, we categorize it as **relevant**. Conversely, if no subject has been assigned to a text, we consider it **irrelevant**.

We believe this strategy reflects the intention of the annotators as they labeled the documents (see Section 4.2). Additionally, by design, this strategy eliminates duplicate content. Table 15 illustrates this process for the content from the Title.

Entry ID	Title	Subject
4662	Cousin of crop-killing bacteria mutating rapidly	None
5885	Cousin of crop-killing bacteria mutating rapidly	4472
19661	Modeling climate change impacts on potential …	None
19361	Modeling climate change impacts on potential …- PubMed	None
58	Danger pour les végétaux : première détection de …	None
850	Danger pour les végétaux : première détection de …	4259
26873	Commodity risk assessment of ash logs from the US …	5524
27196	Commodity risk assessment of ash logs from the US …	5530
29896	Tornano le Giornate Fai di Primavera …	None
29899	Tornano le Giornate Fai di Primavera …- LegnanoNews	None
⋮	⋮	⋮

Title	Relevance
Cousin of crop-killing bacteria mutating rapidly	1
Modeling climate change impacts on potential …	0
Danger pour les végétaux : première détection de …	1
Commodity risk assessment of ash logs from the US …	1
Tornano le Giornate Fai di Primavera …	0
⋮	⋮

Table 15: Determining Relevance for Title

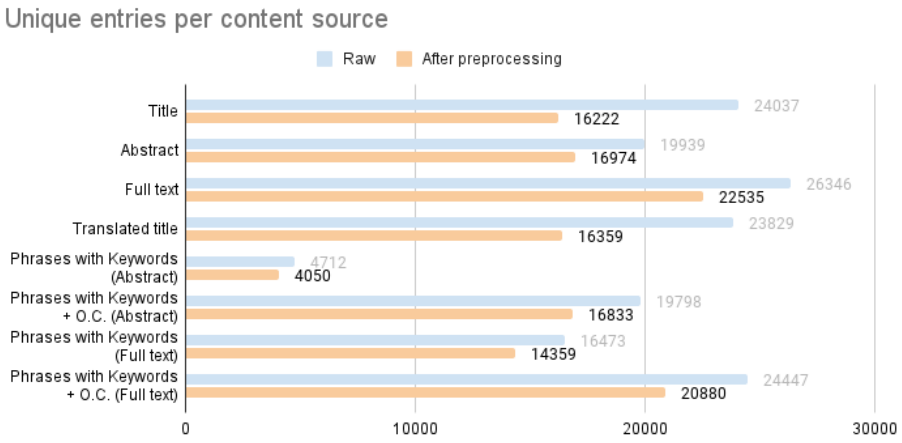


Figure 28: Unique Entries per content source before and after preprocessing

5.4 Preprocessing Results

In this section, we present the outcomes of our preprocessing techniques, yielding refined datasets suitable for text classification. Additional numerical results can be found in Appendix D.1.

Figure 28 demonstrates that preprocessing leads to a reduction in the number of unique entries. For the majority of Content Sources, approximately 5 out of 6 unique entries are retained after the filtering process ($\approx$ 85%), except for the Title and Abstract, where approximately 4 out of 6 unique entries are preserved ($\approx$ 67%).

Unique entries per Content Source	**Percentage left**
Title	67.49%
Abstract	85.13%
Full text	85.53%
Translated Title	68.65%
Phrases with Keywords (Abstract)	85.95%
Phrases with Keywords + O.C (Abstract)	85.02%
Phrases with Keywords (Full text)	87.17%
Phrases with Keywords + O.C (Full text)	85.41%

Table 16: Unique Entries per content source before and after preprocessing

Regarding the average token count of the content, the histograms in Figure 29 show that distributions across the four original Content Sources align with the original observations in Figure 24. The same trend is observed for the Phrases with Keywords as well. Upon comparing the means before and after preprocessing (Table 17), we observe that the means are higher for all the original content sources, except for the Full text.

The higher means can be attributed to the filtering of error messages, which tend to be

short in length, and short entries that may correspond to scrapping errors. As for the Full text, upon closer inspection of the filtered-out *long* texts identified as error messages, it was observed that they often pertain to topics such as website cookies or terms and conditions. These texts tend to be quite lengthy and are not typically found in website metadata (from where the Title and the Abstract are extracted). This observation explains the lower mean for the Full text after preprocessing, which we interpret as a positive indication of our filtering approach.

When analyzing the Phrases with Keywords, a significant decrease in the mean is noted, particularly for the Phrases with Keywords extracted from the Full text. This reduction can be attributed to the fact that only a few sentences in an article usually contain the exact keywords used for locating them through a Search Engine. Consequently, after preprocessing, the mean of the Phrases with Keywords decreases substantially when compared to their original sources.

Overall, these observations highlight the effectiveness of our preprocessing in removing undesirable content while retaining relevant information.

Content Source	**Before preprocessing**	**After preprocessing**
Title	11.29	12.46
Abstract	50.02	50.55
Full text	1102.29	1097.76
Translated Title	11.66	12.82
Phrases with Keywords (Abstract)	(50.02)	34.92
Phrases with Keywords + O.C (Abstract)	(50.02)	35.77
Phrases with Keywords (Full text)	(1102.29)	115.66
Phrases with Keywords + O.C (Full text)	(1102.29)	217.71

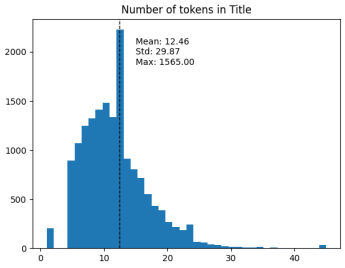
Table 17: Mean token counts before and after preprocessing the VSI dataset

By applying our preprocessing approach, we construct eight labelled datasets for binary Text Classification (Figure 30). These datasets exhibit a balance of positives and negatives ranging from 12 to 14% positives, and from 86 to 88% negatives[a], representing a slight improvement compared to the original dataset, which had a balance of 10% positives to 90% negatives (Figure 21).

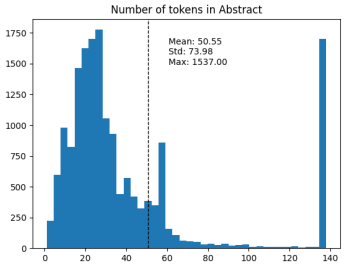
[a]For more details, see Table 34 in Appendix D

The only exception are the Phrases with Keywords from the Abstract discarding Original Content ("Phrases with Keywords (Abstract)"). This specific dataset displays a 27% positive and 73% negative balance. The reasons behind this imbalance can be attributed to the factors previously explained. As the metadata required for extracting the Abstract may be empty or uninformative; and few entries may contain the exact search keywords we are looking for; the dataset is substantially reduced, resulting in only 4,050 unique entries for the Phrases with Keywords (Abstract) as opposed to the original 19,939 unique entries

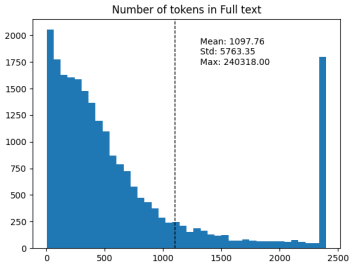
for the Abstract ($\approx 20\%$ left). Consequently, the significantly smaller dataset size has led to a noticeable change in the balance of the labels.



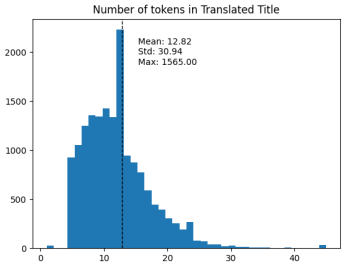
(a) Title



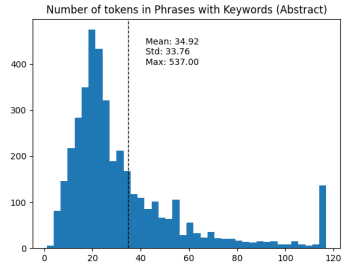
(b) Abstract



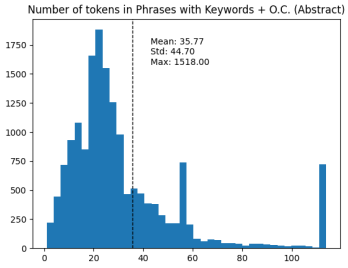
(c) Full text



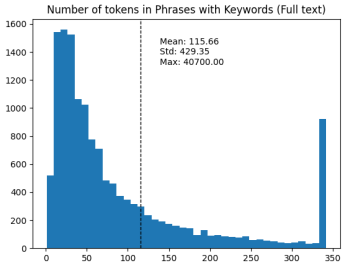
(d) Translated Title



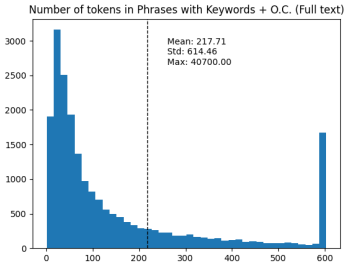
(e) Phrases with Keywords (Abstract)



(f) Phrases with Keywords + O.C (Abstract)

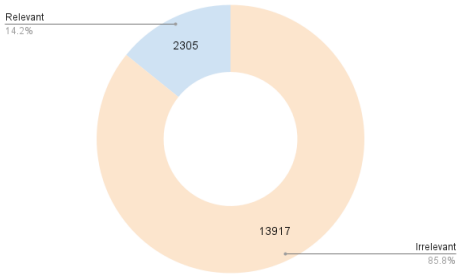


(g) Phrases with Keywords (Full text)

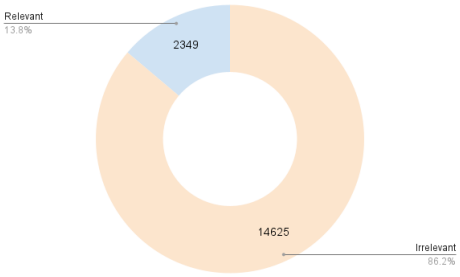


(h) Phrases with Keywords + O.C (Full text)

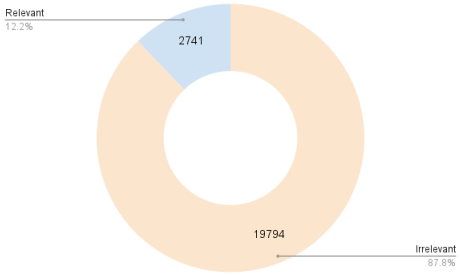
Figure 29: Length (in n. of tokens) of the Content Sources in the VSI dataset after preprocessing



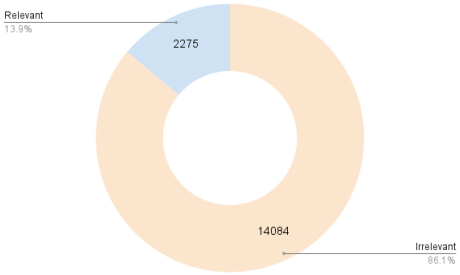
(a) Title



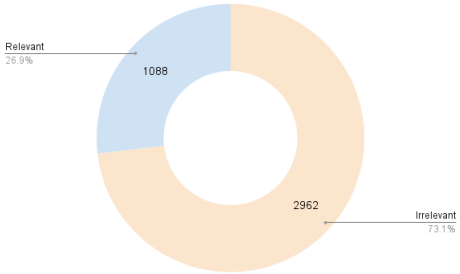
(b) Abstract



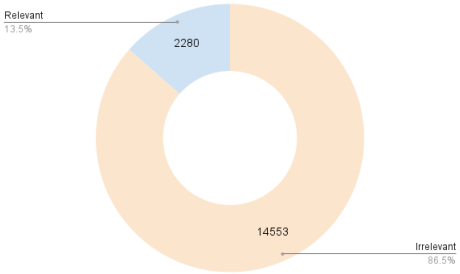
(c) Full text



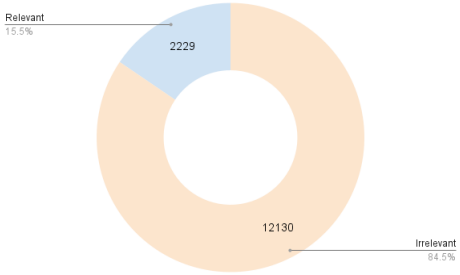
(d) Translated Title



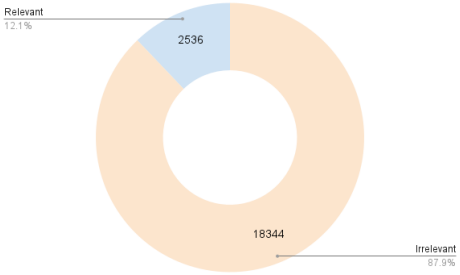
(e) Phrases with Keywords (Abstract)



(f) Phrases with Keywords + O.C (Abstract)



(g) Phrases with Keywords (Full text)



(h) Phrases with Keywords + O.C (Full text)

Figure 30: Positive/Negative balance after preprocessing the VSI dataset

Model	Languages	Tokenization	Encoder Layers	Attention Heads	Vocabulary	Parameters
BERT-base	1	WordPiece	12	12	30k	110M
mBERT	104	WordPiece	12	12	110k	172M
Bio-Link-BERT	1	WordPiece	12	12	30k	110M
SciBERT	1	WordPiece	12	12	30k	110M
RoBERTa-base	1	BPE	12	8	50k	125M
XLM-RoBERTa	100	SentencePiece	12	12	250k	270M

Table 18: Sizes of BERT models used for this project. Adapted from Conneau et al. (2020)

6 Methodology

In this Section, we delve into the techniques we used to train BERT for Document Classification. We first introduce different pre-trained BERT models, and then proceed to explain the training approaches used for this work.

6.1 BERT Models

As previously mentioned in Section 2.4, the BERT architecture from Devlin et al. (2019) has been customized for various tasks, resulting in a wide array of models to select from. For our project, we have opted to employ six different BERT models, based on their specialized task and the characteristics of our data. Since our data is in different languages, we utilize BERT models that differentiate between upper case and lower case characters (`cased` versions). Additionally, due to computational constraints, we opt for the basic (`base`) versions of the models rather than the `large` ones (Table 18).

6.1.1 BERT-base

The original BERT model developed by Google in Devlin et al. (2019). It was trained on a large corpus of English text from Wikipedia and Google's BooksCorpus dataset, using a WP tokenizer.

6.1.2 mBERT

The Multilingual BERT (Devlin et al., 2019) is a version of the original BERT model that is designed to handle text in multiple languages. It was trained on the top 104 languages with the biggest Wikipedia sizes by number of articles, using a WP tokenizer.

6.1.3 Bio-Link-BERT

This model was introduced in Yasunaga et al. (2022) as the result of training a classical BERT architecture while enriching the context of the documents by using two corpora of documents linked to one another by hyperlinks. For documents doc_1, doc_2 linked by a hyperlink $doc_1 \rightarrow doc_2$, during the training phase, the sections of doc_2 that are reachable by the hyperlink will be concatenated to their corresponding sections from doc_1 for the Next

Sentence Prediction task (See Section 2.4.2). This process increases the context available for the model to learn embeddings.

The authors present a model for the general domain (Link-BERT), trained on the English Wikipedia with hyperlinks; and a model for the biomedical domain (Bio-Link-BERT), trained on PubMed[29] abstracts (in English) containing hyperlinks, and used a WP tokenizer. As of the culmination of our work, Bio-Link-BERT is still one of the top performing BERT models in the Biomedical Language Understanding and Reasoning Benchmark (BLURB) (Gu et al., 2021).

6.1.4 SciBERT

This model was introduced in Beltagy et al. (2019) as a specialized model for scientific texts. It was trained with the classical BERT architecture using 1.14 million papers in English from Semantic Scholar[30], 82% of which belong to the biomedical domain. The authors use the full text from the papers.

6.1.5 RoBERTa-base

This model was introduced in Liu et al. (2019) and is an extension and improvement of the original BERT model. RoBERTa-base follows the same architecture as BERT but undergoes a more extensive pre-training process, specifically, it uses much larger batch sizes, longer sequences, and removes the Next Sentence Prediction" task used in BERT. Additionally, it is trained on 10 times more data than BERT, by merging five datasets: Google's BooksCorpus, the English Wikipedia, Common-Crawl[31] English News (Hamborg et al., 2017), OpenWebText (Gokaslan et al., 2019), and the "Stories" subset of the Common-Crawl (Trinh and Le, 2018). It uses a BPE tokenizer.

6.1.6 XLM-RoBERTa

This model was introduced in Conneau et al. (2020) as the multilingual version of RoBERTa-base. For training, the authors constructed the 100-CC corpus for training XLM-RoBERTa, which consists of monolingual data in 100 languages from the Common-Crawl, and used a SentencePiece tokenizer (Kudo and Richardson, 2018), which extends the classical BPE tokenization algorithm.

6.1.7 Excluded Models

Initially, we considered three other BERT models; however, after careful consideration, we decided not to include them due to various reasons.

[29] `https://pubmed.ncbi.nlm.nih.gov/`
[30] `https://www.semanticscholar.org/`
[31] `https://commoncrawl.org/`

ChouBERT This model was introduced in Jiang et al. (2022) as a specialized model for Plant Health Monitoring. However, it was trained exclusively on French texts, which raises strong concerns about its ability to generalize to our linguistically and thematically diverse dataset.

DistilBERT This model was introduced in Sanh et al. (2019) as a smaller version of the original BERT model. It uses a process called "distillation", which compresses the knowledge of a larger BERT model into a smaller one, while still retaining decent performance.

During our preliminary training experiments with the BERT models mentioned above, DistilBERT consistently underperformed. Regardless of the configurations used, it defaulted to predicting the majority class. Therefore, we chose to omit DistilBERT from our final model selection.

BioBERT This model was introduced in Lee et al. (2020) as a BERT variant specifically designed for biomedical text and tasks. Once again, this model uses the classical BERT architecture on PubMed abstracts in English. Since it has been vastly outperformed by Bio-Link-BERT in several NLP tasks (Yasunaga et al., 2022), we decided not to include it in our final model selection.

6.2 Fine-tuning

Finetuning, in the context of NLP, refers to the process of taking a pre-trained language model and further training it on a specific task or dataset. At the outset, we use a pre-trained model, which has undergone training on a vast dataset, enabling it to grasp general features and patterns applicable to a broad spectrum of tasks. Fine-tuning allows us to adapt the pre-trained model to a more specific domain by updating its parameters using task-specific data.

In our scenario, BERT embeddings are used for document classification. To tailor the model to our particular task, we enable the backpropagation process to adjust not only the classification layer but also the weights of the BERT model (refer to Figure 31 in contrast to Figure 17). This technique allows us to make the BERT model adapt the way it produces embeddings and learn a representation of our documents which is more convenient for our task.

Given the imbalance in our datasets, we employ weighted cross-entropy as the loss. Yet, our initial experiments revealed that artificially balancing the dataset enhanced performance. Further details can be found in Section 6.4.

6.3 Pattern-Exploiting Training

Few-shot learning is a ML set of techniques that aim to train models to generalize and perform well on tasks with very limited labeled data. Pattern-Exploiting Training (PET) was introduced by Schick and Schütze (2021) as a technique for Few-shot learning in NLP

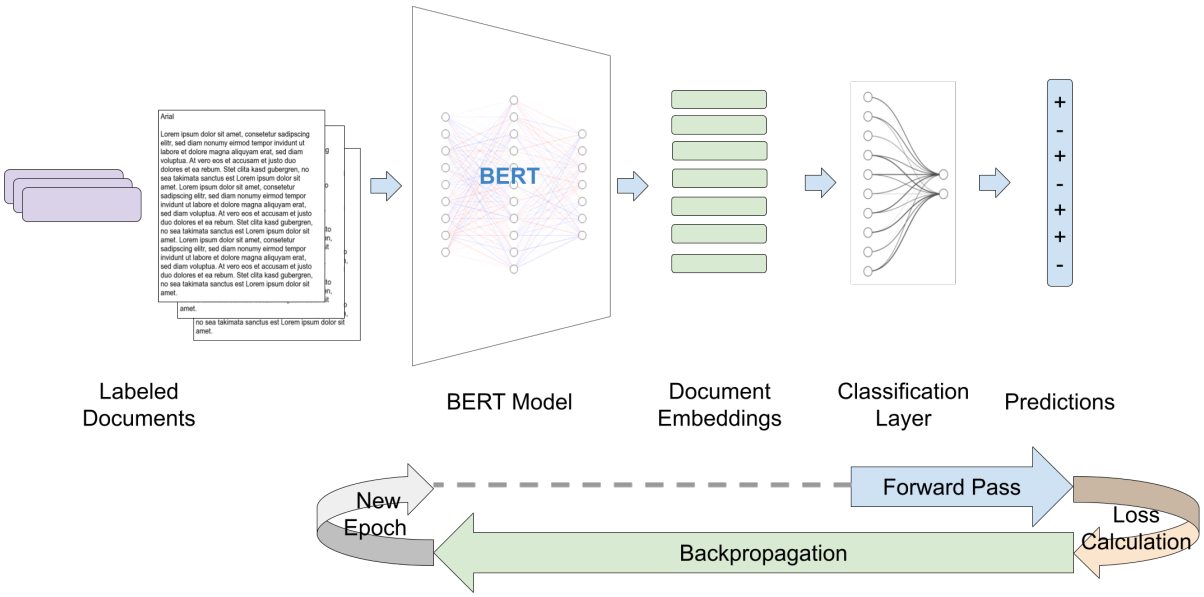


Figure 31: Fine-tuning BERT for Document Classification

using basic prompt engineering, and has provided improvements over standard training approaches.

The intuition behind PET is to use *task descriptions* to improve the model's performance in Text Classification. Leveraging BERT's MLM feature, we design a *pattern*, which takes a document and produces a text with one concealed token (using the `[MASK]` token). The model then tries to "fill-in the blank" by estimating the likelihood of all tokens. The pattern is intended to guide the model towards specific predictions. Next, we use a *verbalizer*, a list of words tied to a category. We then combine these predictions to determine the final likelihood for each category, as illustrated in Figure 32. Finally, we aggregate the probabilities for the relevant tokens into artificial or *synthetic* scores for each category (Figure 32).

Implementing PET is more intricate. It requires creating multiple patterns, with each pattern aimed at guiding the model's token predictions by emphasizing a different aspect of the task. Additionally, each category requires its own verbalizer. It's important to remember that the model's vocabulary is tied to its tokenization method. For instance, in Figure 32, while we use terms like "Definitely", in reality it might be tokenized differently, like "Definite-ly". We're generally limited to concealing a single token, even though there are ways to expand this[32].

In our case, we aim to determine if a document is relevant to Plant Health Surveillance using the BERT models discussed in Section 6.1. Each BERT model has a unique vocabulary and tokenization technique, so we had to pick words that would be consistently tokenized across models. As our task is Binary Classification, we opted to create patterns

[32]The PET paper's authors suggest a method for multiple tokens, however, the trade-off is that the training time increases significantly

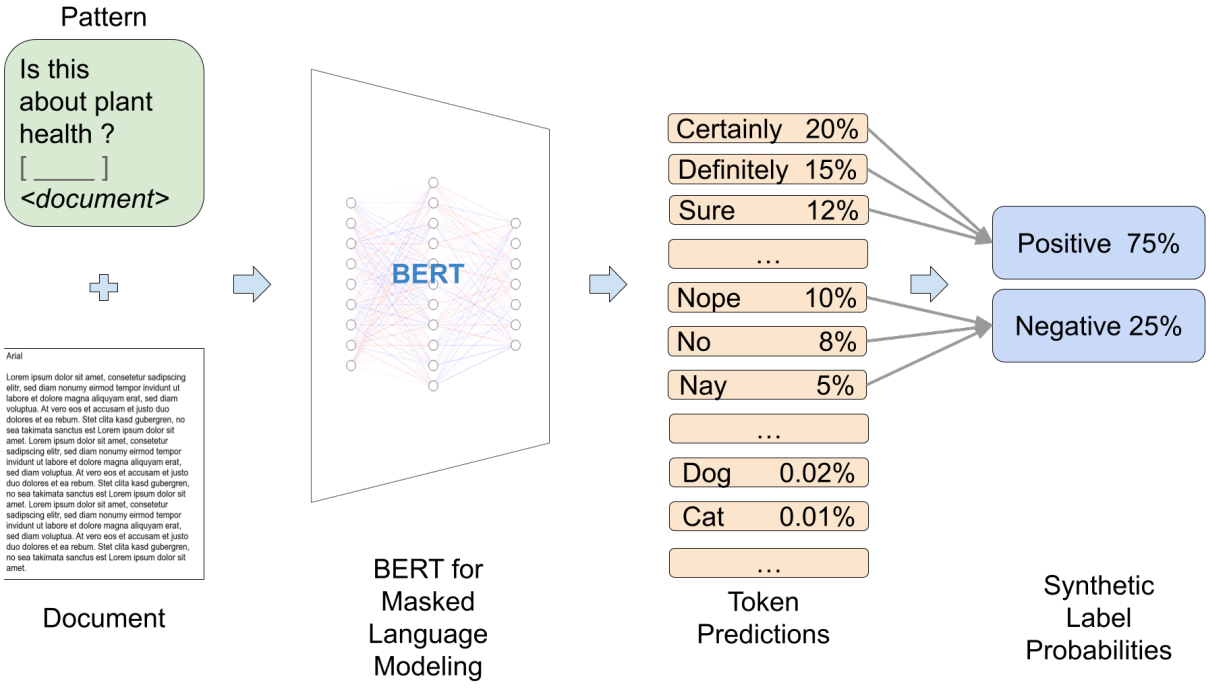


Figure 32: Intuition behind PET

Pattern	Verbalizers
Is this about plant health? [____] < *document* >	Relevant → `yes`
Is this relevant? [____] < *document* >	Irrelevant → `no`
Is this about epidemic surveillance? [____] < *document* >	

Table 19: Patterns and Verbalizers used for PET

using polar questions[33]. Given the 512-token input limit of BERT models and the length of some documents, we ensured the masked token was positioned early in the input, avoiding patterns like the following:

$$< document > < question > [\_\_\_\_]$$
$$< document > [\_\_\_\_] < question >$$

After initial tests revealed that PET training took much longer than Fine-tuning, we settled on just three patterns. The specific patterns and verbalizers we used are detailed in Table 19.

The PET method involves two steps: first training multiple models to generate synthetic labels, followed by fine-tuning a model for classification.

Unlike conventional training approaches, PET requires dividing the dataset into *four* distinct splits: training, development, testing, and an additional split of unlabeled documents known as the *unlabeled* split. The rationale behind this will become clear as we

[33]Polar questions typically have a “yes/no” answer

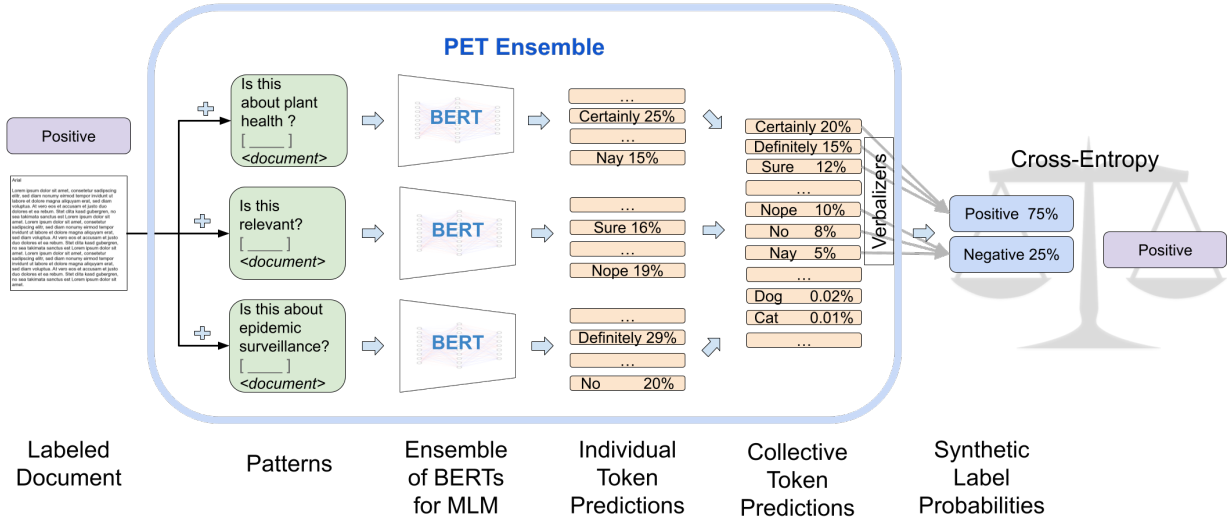


Figure 33: Training a PET Ensemble

delve deeper into the method. Given that this technique is designed for Few-shot learning, the training split typically has only few entries per category.

The first phase involves training a set, or *ensemble*, of models. For every pattern, we take a copy of a BERT model, such as mBERT, and prompt it to predict the concealed token for a specific document in the training segment. Each ensemble model yields different probability estimates for the tokens in the vocabulary.

These probabilities are then combined by averaging their logits for each token[34] and using $SoftMax$ to determine a probability for every vocabulary token. Subsequently, the verbalizers transform the probabilities of pertinent tokens into category label probabilities. These synthetic label probabilities are then compared against the true labels in order to calculate a cross entropy loss. This procedure is depicted in Figure 33[35].

Simultaneously, during this phase, the ensemble models undergo training for MLM on our dataset to learn the typical contexts for our task. This method of training an ensemble for both synthetic annotation and MLM on a dataset requires a custom loss function, which the authors define as a weighted sum of the cross-entropy loss and the MLM loss.

The second phase of PET involves utilizing the trained PET ensemble to generate synthetic labels for the unlabeled documents. This synthetically labelled dataset is then employed to fine-tune a new copy of the BERT model (mBERT in our example) for Document Classification, as detailed in Section 6.2 above. The fine-tuned model, along with its trained classification layer, will act as our Document Classifier (See Figure 34).

It's worth noting that each phase in PET could, in principle, have distinct hyperparameters. Yet, mirroring the original authors' approach, we opted to use the same

[34]Meaning, the aggregated logit for a token is the average of the logits generated by the ensemble models.
[35]This approach bears some similarity to the "distillation" method introduced by Sanh et al. (2019)

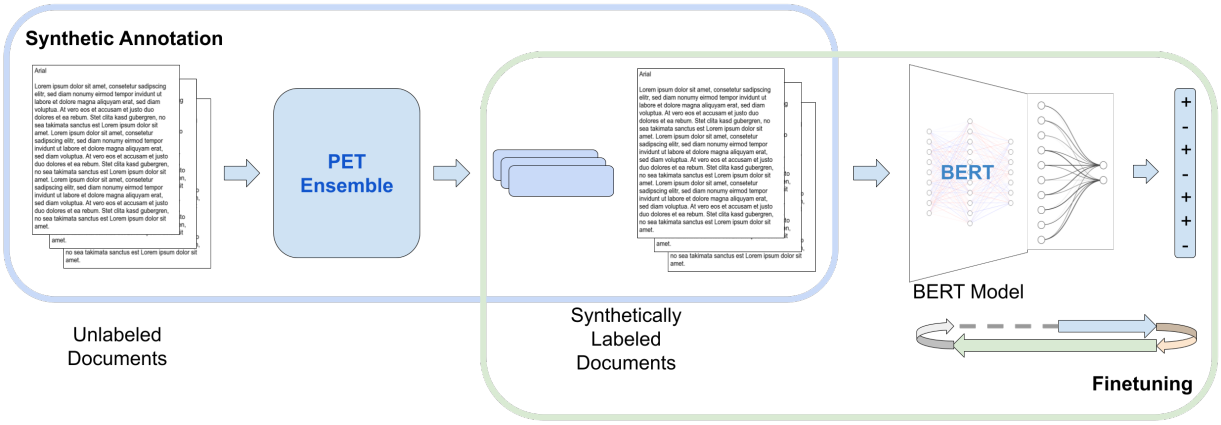


Figure 34: Creating a Synthetically Labeled Dataset with the PET Ensemble, and subsequently Fine-tuning

hyperparameters across both phases for our experiments.

In this work, we opted to utilize PET with varying training sizes, including 50, 100, 200, 500, and 1000 documents *per category*.

Next, we detail the method we used to divide the dataset for Fine-tuning and PET.

6.4 Splitting the Dataset

As mentioned at the end of Section 5.4, after preprocessing, we obtain eight datasets for Binary Text Classification. For each dataset, we produce seven splits, two for training using Fine-tuning and five for training using PET.

The most straightforward splitting method is for Fine-tuning on an unbalanced dataset. For each Content Source, such as the Translated Title, we randomly split the dataset into 80%-10%-10% for training, development, and testing, maintaining the original dataset's category distribution.

For Fine-tuning on a balanced dataset, we retain the split sizes and replicate the test split from the previous step. We then create balanced development and training segments by oversampling the positive category and randomly discarding from the negative category. Considering the original distribution was 90% negatives to 10% positives, we believe there will still be a sufficient representation of the negative category.

For PET, we select a specific number of documents per category, for instance, 50. We then replicate the test and development splits from the balanced Fine-tuning split. From the balanced Fine-tuning training split, we select the first 50 documents from both the positive and negative categories to form the PET training split (keeping in mind that PET was first proposed for Few-shot learning). The remaining documents are used to construct the unlabeled split, by discarding their labels. Likewise for 100, 200, 500, and 1000 documents per category.

A detailed breakdown of the split sizes can be found in Appendix E.2.

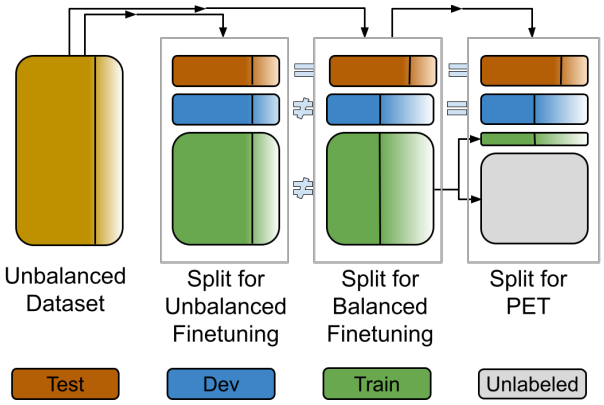


Figure 35: Strategies for Dataset Splitting

Repository	URL
Preprocessing and Training	GitLab Repository
Original PET	GitHub Repository
PET for PESV	GitLab Repository

Table 20: Code repositories for this work

This approach guarantees that for every Content Source, the test split remains consistent across all training methods. Maintaining the same test split for different training techniques ensures the **comparability** of our results. This is crucial, as performance on the test split assesses the classifier's effectiveness in real-world situations.

6.5 Implementation, Hyperparameters, and Hardware

For this work, we worked in Python 3.9.2 and implemented a package for preprocessing the datasets and training BERT language models. This package is built around the `Transformers` library from the 🤗 HuggingFace company for Fine-tuning (Wolf et al., 2020), `Scikit-learn` for metrics and statistics (Pedregosa et al., 2011), `pandas` for handling the datasets (Team, 2020), `BeautifulSoup4` for parsing XML (Richardson, 2019), among others. We also modified the code repository for PET to include our own task-specific patterns and verbalizers. Our implementation is designed to run on one GPU.

During training, we keep track of the accuracy, precision, recall, F_1, and F_2 on the training and development splits per epoch. For the final models, we calculate all these metrics along with the Area Under the Curve (AUC) on the development and test splits. Memory constraints prevented us from saving model checkpoints during training. Therefore, we conducted two training rounds: one to determine the best number of epochs and another to train only up to those identified optimal epochs.

Hyperparameter	Value
Input size	300 tokens
Batch size	32
Optimizer	AdamW
Train-dev-test splits	80%-10%-10%
Epochs for Fine-tuning	10
Learning rate for Fine-tuning	5e-5
Epochs for PET Ensemble	5
Epochs for PET Document Classifier	5
Learning rate for PET	1e-5

Table 21: Hyperparameters

We performed some experiments on the MaiAGE local computers, but the bulk of the training was performed on the `Lab-IA` cluster[36]. This is a computing platform associated to the University of Paris-Saclay, and serves researchers in the Paris region. For our experiments, we conducted various tests and executed our code, but the GPU selection was beyond our control, contingent on the availability of GPUs. Table 35 shows the specifications of this cluster.

After several preliminary experiments, it became clear that the size of our datasets and the implementations of our training methods (specifically, PET) would pose a challenge for the available hardware. These preliminary tests also shed light on certain patterns in the training results, which informed our hyperparameter choices. We found that Fine-tuning was faster than PET, that larger batch sizes maxed out our GPU memory, and that extremely high or low learning rates led to erratic learning. Table 21 lists the hyperparameters we used for our final setup. For any hyperparameters not highlighted in this table, we relied on the default settings offered by the `Transformers` and the original PET implementations.

6.6 Methodology Summary

To summarize, we have eight Content Sources (each one has a labeled dataset), seven training setups (two for Fine-tuning and five for PET), and six different BERT models. This gives us $8 \times 7 \times 6 = 336$ training scenarios.

Table 22 shows the approximate training times for each Content Source for single instances of Fine-tuning and PET. Since there are two Fine-tuning setups and five PET setups, the total training time was approximately $2 \times 41 + 5 \times 102 = 594$ GPU hours, or about 3.5 GPU weeks. Factoring in the two training cycles, the total comes to about 7 GPU weeks.

[36] `http://hebergement.universite-paris-saclay.fr/lab-ia/`

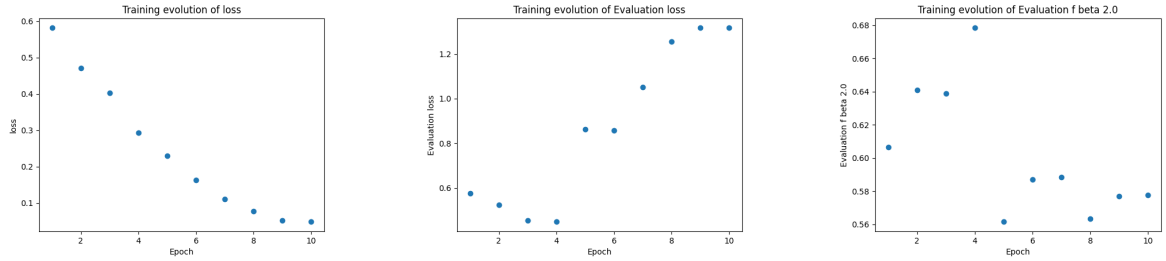
Content Source	**Fine-tuning Time**	**PET Time**
Title	3	10
Abstract	4	11
Full text	12	24
Translated Title	3	10
Phrases with Keywords (Abstract)	1	4
Phrases with Keywords + O.C (Abstract)	4	11
Phrases with Keywords (Full text)	3	12
Phrases with Keywords + O.C (Full text)	11	20
Total	41	102

Table 22: Approximate Training times per Content Source, in GPU hours

7 Results and Discussion

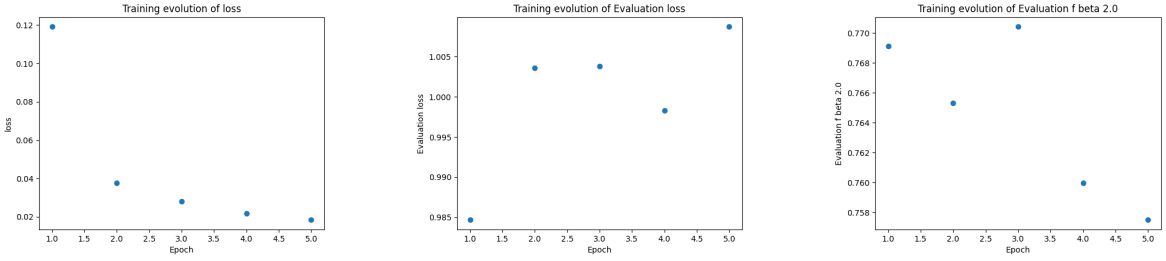
In this Section, we present the results of our training experiments, and provide insights on them.

Comprehensive training plots and logs can be found in our GitLab repository. For a glimpse into the results from our initial training round, refer to Figures 36 and 37. In the first figure, we can see that the optimal epoch to avoid overfitting in that particular scenario is the fourth epoch, which coincidentally also maximizes the F_2. In contrast, in the second image, the first epoch is optimal, while it does not have the best F_2.



(a) Training Loss (b) Development Loss (c) Development F_2

Figure 36: Unbalanced Fine-tuning - Losses and Development F_2 evolution for mBERT on the Title



(a) Training Loss (b) Development Loss (c) Development F_2

Figure 37: PET 1000 - Losses and Development F_2 evolution for RoBERTa-base on the Title

The optimal epochs for each training scenario are detailed in Appendix F.1. It's evident that Unbalanced Fine-tuning takes the longest to reach the optimal epoch, especially when compared to the other methods, where the majority reach the optimum within the first or second epoch. The disparity between the two fine-tuning techniques is significant, with Balanced Fine-tuning often reaching its peak performance in just one epoch. When it comes to PET, all five scenarios exhibit a consistent pattern, typically achieving their optimal epoch before the fourth one. This suggests that any variations in the rate of convergence might occur during the training of the PET ensembles. Given that all ensembles undergo

training for five epochs, it must be that this sufficient for achieving satisfactory results. Unfortunately, we did not record the evolution of the custom losses used to train the PET ensembles.

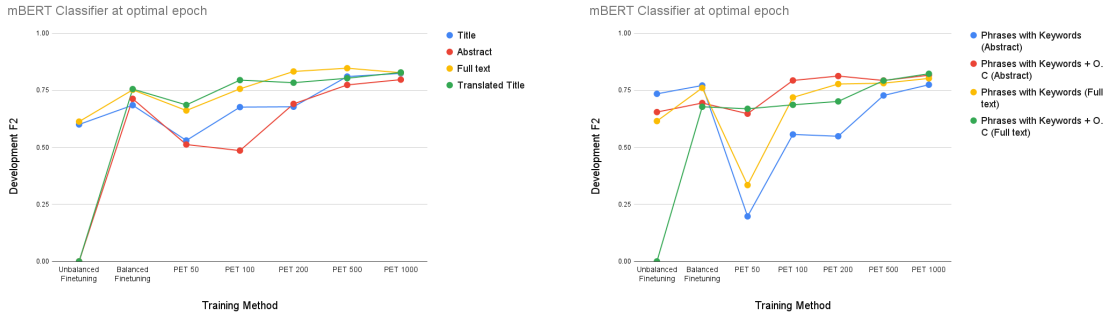


Figure 38: Development F_2 for mBERT at the optimal epochs, across all the training methods

Based on these findings, we moved forward to the second round, trained models up to their optimal epochs, and evaluated their final performance. The metrics for the best classifiers across each training scenario and for every BERT model are detailed in Appendix F.2. A direct comparison of the Development F_2 outcomes reveals that Unbalanced Fine-tuning consistently underperforms compared to other training techniques [37] (See Figure 38 for sample results for mBERT). In fact, Balanced Fine-tuning outperforms Unbalanced Fine-tuning in nearly all instances, with only one exception[38].

When examining PET, a noticeable trend emerges: performance improves as the document count per category grows. Specifically, the Development F_2 for PET 50 generally lags behind other PET variants and seldom matches their levels. Conversely, both PET 500 and PET 1000 consistently register superior results.

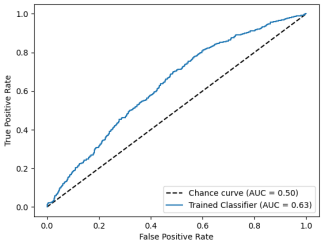
Given the high number of training scenarios (totaling 336), we may impose a stringent performance threshold, focusing only on classifiers that achieve a Development F_2 of at least 80%. Out of all the scenarios, only 72 classifiers meet this criterion, which is roughly one-fifth of the total. A summary of these top-performing classifiers is provided in Table 62. It's evident from the table that PET training typically surpasses Fine-tuning[39], and that Unbalanced Fine-tuning never makes it to the list. Among the PET configurations, both PET 500 and PET 1000 consistently exceed the set threshold.

When ranking the BERT models based on the number of top-performing classifiers, XLM-RoBERTa leads with 22, followed closely by mBERT and Bio-Link-BERT, each with 16. This suggests that the multilingual nature of our dataset might play a more pivotal role in determining relevance than its biomedical aspect. This aligns with the fact that only

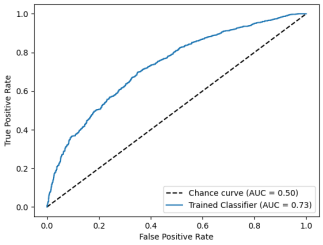
[37] A F_2 score of 0% indicates the model defaulted to predict the predominant class (negative)
[38] This specific instance pertains to BERT-base on the Title, showing a marginal difference of about 3%
[39] The sole exception being XLM-RoBERTa on the Full text

4 top classifiers are associated with SciBERT, and a mere 3 with BERT-base. RoBERTa-base, with its 11 top classifiers, occupies a middle ground, possibly indicating its superior capabilities compared to BERT-base and SciBERT. However, its training exclusively on English text might be a limiting factor.

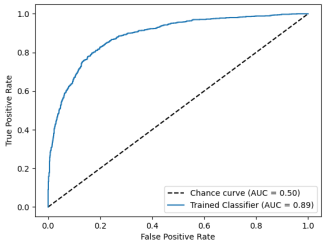
At the same time, it's concerning to observe that several top classifiers exhibit worryingly low Development AUC values, ranging between 60% and 80% (Figure 39). A low AUC indicates challenges faced by these models in differentiating between positive and negative instances. Yet, their high F_2 scores suggest a pronounced recall relative to precision. Piecing these insights together, it's plausible that these models achieve elevated F_2 scores by predominantly categorizing most documents as positive.



(a) Bio-Link-BERT with PET 50 on the Title



(b) RoBERTa-base with PET 200 on the Full text



(c) XLM-RoBERTa with PET 1000 on the Full text

Figure 39: Sample AUCs for some classifiers

Prioritizing models that genuinely grasp the concept of relevance rather than indiscriminately marking everything as positive, we set a threshold of 80% for the Development AUC. By doing so, we narrow down to 49 classifiers, detailed in Table 63. Interestingly, the Content Source with the highest number of top-performing classifiers is the Full text, exhibiting 15 top classifiers across all BERT models. This indicates that the Full text, typically longer than other Content Sources, offers ample content, enabling all models to effectively learn the task.

Conversely, it's worth noting that the Phrases with Keywords (Abstract) doesn't feature on the list. As discussed in Section 5, this dataset undergoes significant reduction during preprocessing, potentially leaving it with insufficient content. Given that we still have four top-performing classifiers for the Abstract, we opt to **exclude** the Phrases with Keywords (Abstract) as a Content Source.

To avoid classifiers that flag almost everything as a positive, we impose a 75% threshold on the Development F_1. This criterion narrows our selection to 17 classifiers. Table 64 presents the Development and Test metrics for these classifiers, emphasizing the top-performing metrics. Once more, the Full text dominates with the most classifiers on this list (9), and its best three metrics are highlighted.

Predictably, the Test metrics are lower than the Development metrics. Yet, it's striking to see a significant drop in the F_1 and precision, with most falling below the 50% mark. This trend can be attributed to our emphasis on the F_2, which leans towards Recall at the

expense of Precision. At the same time, the highest Test Recalls hover around or exceed 80%.

Observe that the Phrases with Keywords (Full text) is absent from the list. This can be attributed to the significant reduction in dataset content when only sentences containing keywords are considered. Additionally, the peak metrics for Phrases with Keywords + O.C (Abstract) and Phrases with Keywords + O.C (Full text) consistently fall short of those for Abstract and Full text, respectively. These findings suggest that incorporating Phrases with Keywords as Content Sources does **not** enhance performance, contrary to our hypothesis in Section 5.1. Consequently, we opt to **exclude** these sources of content moving forward, and thus, we are left with the four original Content Sources.

Considering the best performing classifiers (Table 23), it's noteworthy that, except for one, all top classifiers are derived from PET 1000. Moreover, the best classifiers are consistently obtained by training multilingual BERT models. This highlights the significance of the multilingual nature of our dataset in determining relevance over its biomedical aspect.

Content Source	Title	Abstract	Full text	Full text	Full text	Translated Title
Model	mBERT	XLM-RoBERTa	mBERT	XLM-RoBERTa	XLM-RoBERTa	mBERT
Training Method	PET 1000	PET 1000	PET 1000	Balanced Fine-tuning	PET 1000	PET 1000
Development F_2	82.44 %	82.34 %	82.75 %	83.62 %	85.73 %	82.87 %
Development AUC	83.46 %	84.15 %	88.82 %	90.57 %	88.85 %	85.27 %
Development F_1	77.55 %	77.41 %	81.52 %	83.95 %	81.6 %	78.81 %
Development Accuracy	75.09 %	74.91 %	81.06 %	84.06 %	79.99 %	76.92 %
Development Precision	70.58 %	70.4 %	79.56 %	84.52 %	75.53 %	72.86 %
Development Recall	86.06 %	85.98 %	83.58 %	83.39 %	88.73 %	85.82 %
Test F_2	60.56 %	60.57 %	66.97 %	63.72 %	63.79 %	63.81 %
Test AUC	82.5 %	84.6 %	89.61 %	87.22 %	89.66 %	84.2 %
Test Precision	27.83 %	28.74 %	50.21 %	51.1 %	44.34 %	31.86 %
Test Recall	85.78 %	83.76 %	79.24 %	82.24 %	72.49 %	85.15 %
Test F_1	42.03 %	42.79 %	35.44 %	38.42 %	29.4 %	46.37 %
Test Accuracy	66.73 %	69.14 %	86.13 %	76.28 %	90.15 %	72.9 %

Table 23: Best classifiers by Content Source

In summary, among all the training methods examined in our work, Unbalanced Fine-tuning yields the least favorable performance, while PET demonstrates increasingly improved performance as more documents per category are introduced. In terms of Content Sources, incorporating keywords from the documents does not enhance performance, whereas utilizing the Full text proves to be the most effective in providing substantial content for the models to learn from, with the Title, Translated Title and Abstract lagging behind. Additionally, the multilingual aspect of our dataset appears to hold greater significance than their biomedical content, as evidenced by the superior performance of classifiers trained with multilingual BERT models.

Finally, we put forward a reason for the enhanced performance of PET compared to Fine-tuning. We theorize that the use of task descriptions in PET enables the model to have a clearer comprehension of the task. To illustrate this idea, we reference an example from the foundational PET paper by Schick and Schütze (2021).

Take into account the next three statements: the first two have labels, and our challenge is to assign a label to the third.

1. Category A: This *was* the best *pizza* I've ever had.
2. Category B: You *can* get better sushi for half the *price*.
3. Category ?: *Pizza was* average. Not worth the *price*.

Based solely on this data, determining a label is challenging: Is the focus on pizza or price? Or is it about identifying verb tenses (Past vs. Present)? Yet, if we're informed that the objective is to identify sentences discussing pizza, we'd promptly label the third sentence in category A. Conversely, if it's about prices, category B would be the appropriate label.

For our task, consider the three following (artificial) sentences:

1. Relevant: Oriental Fruit fly (Bactrocera dorsalis) detected in a new location in Central Asia.
2. Irrelevant: Check out the best decorative plants for your garden this summer, and beware of insects!
3. Category ?: What to do if you find fruit flies plaguing your field - Agriculture News Online.

For the task of identifying document relevance for Plant Health Surveillance, we categorize the third sentence as 'Irrelevant'. However, such subtleties might be overlooked by a language model without additional context.

Indeed, referring to Table 23, it's evident that Fine-tuning yields one of the top classifiers only when applied to the Full text. Throughout our analysis, we've observed that many top-performing classifiers were also trained on the Full text. This suggests that due to its length (averaging 1102 tokens, but cut at 300 input tokens), the Full text provides ample content, enabling any BERT model to grasp the task effectively. In contrast, other Content Sources, being much shorter than the Full text, might not offer sufficient context for the models. For instance, the Title averages 11 tokens, the Abstract 50 tokens, and the Translated Title 12 tokens. As a result, classifiers benefit from the added context given by PET patterns.

Furthermore, it's worth noting that all other best classifiers predominantly use PET 1000. This suggests that exposing the model to more document examples combined with task-specific knowledge enhances its performance. However, the performance improvement stales over time, as the incremental gains from transitioning from PET 500 to PET 1000 are marginal, as seen in Tables 50-55.

In conclusion, by employing PET and its patterns, we inject task-specific knowledge, thereby guiding the model.

8 Practical Implications and Future Work

In this Section, we discuss the real-world impact of our research, detailing its application, updates, and the effects it will have on the work of the VSI experts.

8.1 Overview

Upon determining the optimal hyperparameters for the most effective classifiers (as seen in Table 23), the PESV team plans to integrate our findings into their data collection pipeline as a component of the TIERS-ESV initiative.

For the TIERS-ESV project, we have developed two key services:

- A service for inference.
- A service for training.

8.2 Inference Service

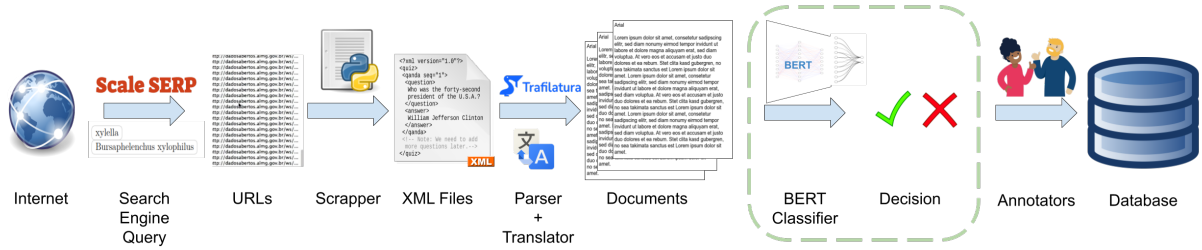


Figure 40: Our classifiers added to the VSI Data Collection Pipeline

Our Document Classification system will be integrated into the VSI data collection pipeline, as depicted in Figure 40. This system is composed of two primary components:

1. Heuristic methods

2. Neural-based Text Classification

The heuristic component is designed to eliminate entries that lack substantial content. For those documents that do contain content, the Text Classification component of the system applies our trained models. The results produced by this filter will be fed into the VSI database. Overall, we anticipate that this addition will significantly reduce the manual effort required by the VSI experts.

The implementation of this filter will utilize the Python package developed during this thesis.

1. First, the results from web scraping, as an XML/TEI file, are processed using our tool to extract the Title, the Abstract, the Full text, and the Translated Title.

2. Subsequently, any noise such as HTML tags, URLs, website names, and the like are filtered out from the text for each Content Source.

3. The heuristic module then employs the tools we developed for preprocessing. By using functions that identify error messages and scraping issues, one can eliminate document that lack substantial content.

4. Then, for each Content Source, the cleaned text is given as input for the Text Classification model.

5. Finally, the output of the filter for a particular document consists of
    - A boolean flag indicating whether the document was kept.
    - The cleaned text
    - The prediction: Relevant VS Irrelevant
    - The label probabilities

Given our observations on addressing error messages and managing scraping errors (as discussed in Sections 5.2.2 and 5.2.3), we anticipate that the heuristic model will exclude between 20% to 30% of unique documents, depending on the Content Source.

This filtering system will be applied to the documents scraped weekly by the PESV Platform. For user convenience, it will be encapsulated within a `REST` API, hosted on the MaiAGE servers, and integrated into the *AlvisNLP-ML*[40] toolkit of the Bibliome group (Ba and Bossy, 2016). Lastly, when publishing selected documents in the official bulletin of the PESV Platform, they will be arranged in order of decreasing relevance.

8.3 Training Service

As the VSI Database expands with new entries, there will be a need to update the models utilized for Text Classification. This responsibility falls on the training service, which implements the entire pipeline to train the best performing classifiers from Table 23.

The frequency and data selection for training will be at the discretion of the VSI experts. They must determine whether training should occur weekly, bi-weekly, etc. Additionally, as their priorities and interests evolve, they might find it beneficial to limit the training data, perhaps only focusing on data from recent months.

The training service is designed to run on the `Lab-IA` cluster. All scripts related to this service can be found in our GitLab repository.

8.4 Consequences for Plant Health Surveillance

Until now, the VSI experts at the PESV Platform have been meticulously reviewing every scraped document. Integrating Text Classification into the annotation process will significantly reduce the burden on the VSI experts, ultimately aiding in long-term Plant Health Surveillance.

[40] `https://bibliome.github.io/alvisnlp/`

By incorporating the Heuristics module, the VSI experts can identify which queries and websites pose challenges for their scraping tools.

Given that the VSI experts will review documents marked as Relevant by our system, we decided to calculate the positive ratio for the top classifiers on their respective test splits. Moreover, we aim for a reduced rate of false negatives, meaning the percentage of genuinely Relevant documents that our system mistakenly labels as Irrelevant (refer to Table 24). The rationale behind this is that overlooking an event that poses a risk to Plant Health (a positive) is more dangerous than mistakenly identifying a harmless event (a negative) as Relevant.

Content Source	Model	Training Method	Positives/Total	FN/Total
Title	mBERT	PET 1000	43.33%	2.0%
Abstract	XLM-RoBERTa	PET 1000	40.16%	2.23%
Full text	mBERT	PET 1000	26.55%	1.66%
Full text	XLM-RoBERTa	Balanced Fine-tuning	24.16%	2.89%
Full text	XLM-RoBERTa	PET 1000	37.27%	1.20%
Translated Title	mBERT	PET 1000	34.62%	1.69%

Table 24: Positives and FN ratios for the best classifiers

The minimal False Negative rates indicate that our systems will overlook only around 2 out of every 100 Relevant (unique) documents, which we view as an impressive outcome. As mentioned in Section 5.4, the actual proportions in the VSI Dataset fluctuate between 12% and 14% positives. The closer our classifiers approach to this balance, the fewer documents the VSI experts will need to review manually.

In the most favorable scenario, with a Positive ratio of 24%, the VSI experts will encounter roughly $76\% \approx 3/4$ fewer documents than they currently do. Conversely, in the least favorable scenario, with a Positive ratio of 43%, they will see about $57\% \approx 1/2$ fewer documents.

This demonstrates that integrating our system into their monitoring mission will significantly decrease the number of documents requiring manual review. This will allow the VSI experts to either assess the same volume of documents more quickly, allocate the same time to evaluate a larger set of documents, or broaden their surveillance to additional pathogens. We consider this outcome to be satisfactory and a fitting conclusion to our work.

8.5 Future Work

There are some ways for improving our work, here, we highlight three.

Translating the Full text One significant improvement could involve translating the Full text of the documents. Our findings indicate that using the Full text provides the most substantial content for training our models. Considering that the most advanced language models are typically developed initially for English text, creating a new Content Source by translating the Full text into English seems like a logical step.

However, this approach comes with its own set of challenges:

- Using the Google Translate API for extensive content could become costly.
- Determining whether a particular species is associated to an agricultural event is important to determine its relevant to Plant Health Surveillance. Given that agricultural reports tend to use the common names of species, using automatic translation carries the risk of inaccurately rendering the common names of various organisms. For instance, the Coccinellidae beetle is referred to as 'ladybug' in English, while in French, it is known as 'Bête à bon Dieu' (literally, 'god's creature'). Such discrepancies present a substantial challenge for general-purpose translation systems.

Leveraging Metadata As mentioned in Section 4, the VSI Database also contains metadata for each article, which presents an opportunity to enhance our results.

For instance, considering that the interests of the VSI experts evolve over time, we could leverage the publication and annotation dates associated with each article to craft additional features for Machine Learning. This could potentially allow the model to adapt more effectively to the shifting focus of the VSI experts.

Additionally, we could parse the URL of each article and use it to help determine which sources are the most trustworthy.

Clustering instead of Classification A final suggestion is to complement Text Classification with *Clustering* techniques. This approach would enable us to analyze the similarities between documents and understand their distribution more comprehensively.

For instance, we could investigate how documents are related across different subjects, or assess whether one subject is encompassed within another. This could provide valuable insights into the structure and relationships within the dataset, and to improve the efficiency of the VSI experts.

9 Conclusions

This master thesis has explored an application of NLP in the field of Plant Health Surveillance, a critical area that impacts agriculture, food security, and environmental sustainability. Conducted at the INRAE Laboratory, a renowned French research institute, and

in collaboration with the PESV Platform this work stands at the intersection of advanced computational methods and biological research.

The research was aimed at automating the process of determining the relevance of documents for Plant Health Surveillance, thereby significantly reducing the manual effort required by the experts of the VSI team at the PESV Platform. The thesis thoroughly studies the properties and statistics of the data on phytosanitary events collected and annotated by the VSI experts and develops methods to preprocess it for NLP applications.

Furthermore, the thesis outlines the development of a Document Classification system, which is intended to be integrated into the VSI data collection pipeline. This system comprises two primary components: Heuristic methods and Neural-based Text Classification. The heuristic component is designed to filter out entries that lack substantial content. For documents that do contain content, the Text Classification component of the system applies trained models to categorize the documents as 'Relevant' or 'Irrelevant' for Plant Health Surveillance. The results produced by this filter are intended to be fed into the VSI database.

The thesis provides a detailed analysis of the performance of various BERT models, with mBERT and XLM-RoBERTa producing the best results, due to the multilingual nature of the data. The Pattern Exploiting Training (PET) method, which was used extensively in this research, proved to be effective, especially when numerous examples (from 500 to 1,000 per category) were provided to the model during training. The thesis shows that using the document's Full text, due to its length and richness, allows the models to grasp the task effectively, highlighting the importance of context in Text Classification tasks.

As the VSI Database expands with new entries, there will be a need to update the models used for Text Classification. The thesis outlines a training service that is designed to be flexible and adaptive to the evolving needs and priorities of the VSI experts.

In summary, this thesis makes a contribution to the field of Text Classification in the context of Plant Health Surveillance. It not only provides a robust and automated solution to a real-world problem but also offers insights into the effectiveness of various training strategies and the importance of context in Text Classification tasks. The work presented in this thesis holds promise for significantly reducing the manual workload of the VSI experts, thus aiding long-term Plant Health Surveillance.

References

Sultan Alrowili and Vijay Shanker. BioM-transformers: Building large biomedical language models with BERT, ALBERT and ELECTRA. In *Proceedings of the 20th Workshop on Biomedical Language Processing*, pages 221–227, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.bionlp-1.24. URL https://aclanthology.org/2021.bionlp-1.24.

Eiji Aramaki, Sachiko Maskawa, and Mizuki Morita. Twitter catches the flu: Detecting influenza epidemics using Twitter. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1576, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics. URL https://aclanthology.org/D11-1145.

Ron Artstein and Massimo Poesio. Survey article: Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596, 2008. doi: 10.1162/coli.07-034-R2. URL https://aclanthology.org/J08-4004.

Mouhamadou Ba and Robert Bossy. Interoperability of corpus processing workflow engines: the case of. AlvisNLP/ML in OpenMinTeD. In *Meeting of working Group Medicago sativa*, page np, Portoroz, Slovenia, May 2016. URL https://hal.science/hal-01455853.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

Adrien Barbaresi. Trafilatura: A Web Scraping Library and Command-Line Tool for Text Discovery and Extraction. In *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 122–131. Association for Computational Linguistics, 2021. URL https://aclanthology.org/2021.acl-demo.15.

Iz Beltagy, Kyle Lo, and Arman Cohan. 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, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1371. URL https://aclanthology.org/D19-1371.

Steven Bird and Edward Loper. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain,

July 2004. Association for Computational Linguistics. URL https://aclanthology.org/P04-3031.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Fanyu Bu and Xin Wang. A smart agriculture iot system based on deep reinforcement learning. *Future Generation Computer Systems*, 99:500–507, 2019. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2019.04.041. URL https://www.sciencedirect.com/science/article/pii/S0167739X19307277.

Lauren E. Charles-Smith, Tera L. Reynolds, Mark A. Cameron, Mike Conway, Eric H. Y. Lau, Jennifer M. Olsen, Julie A. Pavlin, Mika Shigematsu, Laura C. Streichert, Katie J. Suda, and Courtney D. Corley. Using social media for actionable disease surveillance and outbreak management: A systematic literature review, 2015. URL http://dx.doi.org/10.1371/journal.pone.0139701.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1070. URL https://aclanthology.org/D17-1070.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.747. URL https://aclanthology.org/2020.acl-main.747.

Liliane Soares da Costa, Italo L Oliveira, and Renato Fileto. Text classification using embeddings: a survey. *Knowledge and Information Systems*, pages 1–43, 2023.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *arXiv*, 2022. doi: 10.48550/ARXIV.2205.14135. URL https://arxiv.org/abs/2205.14135.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. Domain-specific language model pretraining for biomedical natural language processing, October 2021. URL http://dx.doi.org/10.1145/3458754.

Kai Hakala and Sampo Pyysalo. Biomedical named entity recognition with multilingual BERT. In *Proceedings of the 5th Workshop on BioNLP Open Shared Tasks*, pages 56–61, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5709. URL https://aclanthology.org/D19-5709.

Felix Hamborg, Norman Meuschke, Corinna Breitinger, and Bela Gipp. news-please: A generic news crawler and extractor. In *Proceedings of the 15th International Symposium of Information Science*, pages 218–223, March 2017. doi: 10.5281/zenodo.4120316.

James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, 1982.

Fan Huang, Haewoon Kwak, and Jisun An. Is chatgpt better than human annotators? potential and limitations of chatgpt in explaining implicit hate speech. In *Companion Proceedings of the ACM Web Conference 2023*, WWW '23 Companion, page 294–297. Association for Computing Machinery, 2023. ISBN 9781450394192. doi: 10.1145/3543873.3587368. URL https://doi.org/10.1145/3543873.3587368.

Mohd Javaid, Abid Haleem, Ibrahim Haleem Khan, and Rajiv Suman. Understanding the potential applications of artificial intelligence in agriculture sector. *Advanced Agrochem*, 2(1):15–30, 2023. ISSN 2773-2371. doi: https://doi.org/10.1016/j.aac.2022.10.001. URL https://www.sciencedirect.com/science/article/pii/S277323712200020X.

Shufan Jiang, Rafael Angarita, Stéphane Cormier, Julien Orensanz, and Francis Rousseaux. ChouBERT: Pre-training French language model for crowdsensing with Tweets in phytosanitary context. In *International Conference on Research Challenges in Information Science*, pages 653–661. Springer, 2022.

M. V. Koroteev. Bert: A review of applications in natural language processing and understanding. *arXiv*, 2021. doi: 10.48550/ARXIV.2103.11943. URL https://arxiv.org/abs/2103.11943.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL https://aclanthology.org/D18-2012.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

Alexander LeNail. Nn-svg: Publication-ready neural network architecture schematics. *Journal of Open Source Software*, 4(33):747, 2019. doi: 10.21105/joss.00747. URL https://doi.org/10.21105/joss.00747.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL http://arxiv.org/abs/1907.11692. cite arxiv:1907.11692.

Christopher Manning and Hinrich Schutze. *Foundations of statistical natural language processing*. MIT press, 1999.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.645. URL https://aclanthology.org/2020.acl-main.645.

Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751, 2013.

Ministère de L'Agriculture et de L'Alimentation. Convention cadre portant définition et organisation de la plateforme national d'epidémiosurveillance en santé végétale. Convention cadre, 2018. Direction générale de l'Alimentation.

Chris Olah and Shan Carter. Attention and augmented recurrent neural networks. *Distill*, 2016. doi: 10.23915/distill.00001. URL http://distill.pub/2016/augmented-rnns.

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, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://aclanthology.org/N18-1202.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. *OpenAI Blog*, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Leonard Richardson. Beautiful soup 4 documentation. *dec*, 2019.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC2 Workshop*, 2019.

Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.20. URL https://aclanthology.org/2021.eacl-main.20.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units, 2016.

Priyamvada Shankar, Christian Bitter, and Marcus Liwicki. Digital crop health monitoring by analyzing social media streams. In *2020 IEEE / ITU International Conference on Artificial Intelligence for Good (AI4G)*, pages 87–94, 2020. doi: 10.1109/AI4G50087.2020.9310985.

Dou Shen. *Text Categorization*, pages 3041–3044. Springer US, Boston, MA, 2009. ISBN 978-0-387-39940-9. doi: 10.1007/978-0-387-39940-9_414. URL https://doi.org/10.1007/978-0-387-39940-9_414.

R. G. Steadman. The assessment of sultriness. part i: A temperature-humidity index based on human physiology and clothing science. *Journal of Applied Meteorology (1962-1982)*, 18(7):861–873, 1979. ISSN 00218952, 2163534X. URL http://www.jstor.org/stable/26179216.

The Pandas Development Team. pandas-dev/pandas: Pandas. https://doi.org/10.5281/zenodo.3509134, February 2020.

Trieu H. Trinh and Quoc V. Le. A simple method for commonsense reasoning. *ArXiv*, abs/1806.02847, 2018. URL https://api.semanticscholar.org/CorpusID:47015717.

Nicolas Turenne, Mathieu Andro, Roselyne Corbière, and Tien T. Phan. Open Data Platform for Knowledge Access in Plant Health Domain : VESPA Mining. working paper or preprint, 2015. URL https://hal.science/hal-01145489.

C J Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Oxford, England, 2 edition, January 1979.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon,

U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. Want to reduce labeling cost? GPT-3 can help. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4195–4205, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.354. URL https://aclanthology.org/2021.findings-emnlp.354.

Marijke Welvaert, Omar Al-Ghattas, Mark Cameron, and Peter Caley. Limits of use of social media for monitoring biosecurity events. *PLOS ONE*, 12(2):1–17, 02 2017. doi: 10.1371/journal.pone.0172457. URL https://doi.org/10.1371/journal.pone.0172457.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6.

Stephen Wolfram. What is ChatGPT doing and why does it work? https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/, February 2023.

Anupong Wongchai, Surendra Kumar Shukla, Mohammed Altaf Ahmed, Ulaganathan Sakthi, Mukta Jagdish, and Ravi kumar. Artificial intelligence - enabled soft sensor and internet of things for sustainable agriculture using ensemble deep learning architecture. *Computers and Electrical Engineering*, 102:108128, 2022. ISSN 0045-7906. doi: https://doi.org/10.1016/j.compeleceng.2022.108128. URL https://www.sciencedirect.com/science/article/pii/S0045790622003780.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, et al. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016.

Michihiro Yasunaga, Jure Leskovec, and Percy Liang. Linkbert: Pretraining language models with document links, 2022.

Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into Deep Learning. https://d2l.ai/, 2023.

A Evaluation Metrics for Classification

In Binary Classification, we deal with two classes. Our primary concern is to measure how well the positive class is predicted. The content of this section is an adaptation of Van Rijsbergen (1979).

When the classifier is applied to an item, there are four potential outcomes, contingent on the classifier's predictions and the true class labels. These four cases constitute a *Confusion Matrix* (as depicted in Table 26).

Case	**Description**
True Positives (TP)	The item *is* in the positive class, and the classifier *predicts* the positive class.
True Negatives (TN)	The item *is* in the negative class, and the classifier *predicts* the negative class.
False Negative (FN)	The item *is* in the positive class, and the classifier *predicts* the negative class.
False Positive (FP)	The item *is* in the negative class, and the classifier *predicts* the positive class.

Table 25: Classification Outcomes

Actual Class/Predicted Class	**Predicted Positive**	**Predicted Negative**
Actual Positive	TP	FN
Actual Negative	FP	TN

Table 26: Confusion Matrix for Binary Classification

There are four classic metrics for evaluating classifiers:

- Accuracy: Accuracy measures the overall correctness of a classification model. It is the ratio of *correctly* predicted instances to the total number of instances in the dataset.

$$A = \frac{TP + TN}{Positives + Negatives} \tag{8}$$

  Example: If a model classifies 90 out of 100 samples correctly, the accuracy would be 90%.

  Example: In an unbalanced dataset containing 80% of negatives, a classifier that flags everything as a negative would get an accuracy of 80%.
- Precision: Precision quantifies the ability of a model to correctly identify positive instances among the ones it predicted as positive. It is the ratio of true positive predictions to the total number of positive predictions (both true positives and false positives).

$$P = \frac{TP}{PredictedPositives} \tag{9}$$

$$= \frac{TP}{TP + FP} \tag{10}$$

Example: If a model predicts 100 instances as positive, and 80 of them are truly positive, the precision would be 80%.
Note that, with a model that almost always predicts the negative class, there will be minimal False Positives and True Positives ($FP, TP \to 0$), and thus, this flawed model would have nearly 0% precision.

- Recall (Sensitivity or True Positive Rate): Recall measures the ability of a model to correctly identify all positive instances in the dataset. It is the ratio of true positive predictions to the total number of actual positive instances in the dataset.

$$R = \frac{TP}{ActualPositives} \tag{11}$$

$$= \frac{TP}{TP + FN} \tag{12}$$

Example: If there are 100 positive instances in the dataset, and the model correctly identifies 80 of them, the recall would be 80%.
Note that, with a model that almost always predicts the positive class, there will be minimal False Negatives ($FN \to 0$), and thus, this flawed model would have nearly 100% recall.

- F_1 Score: The F_1 score is the harmonic mean of precision and recall. It balances the trade-off between precision and recall, penalizing disparities between these metrics, and provides a single metric to evaluate a model's performance.

$$F_1 = \left(\frac{1}{P} + \frac{1}{R}\right)^{-1} \tag{13}$$

$$= 2 * \frac{P * R}{P + R} \tag{14}$$

Example: If a model has a precision of 75% and recall of 80%, the F_1 score would be calculated as $2 * ((0.75 * 0.80)/(0.75 + 0.80)) = 77,4\%$.

A.1 Evaluating Probabilistic Classifiers

A probabilistic binary classifier is a classifier that produces the probabilities of an item belonging to the positive class, rather than giving a definitive verdict. To take a decision, a specific threshold or cut-off is needed. For example, if a 50% threshold is chosen for the positive class, items with probabilities above this are deemed positive, while those below are considered as negative. Additionally, this threshold can be adjusted to be more stringent, requiring a higher confidence level (e.g., 60%) for positive classification, or more lenient, allowing lower confidence (e.g., 40%) for positive classification, and so on (Hanley and McNeil, 1982).

When evaluating a probabilistic classifier using a labeled dataset, different thresholds lead to varied confusion matrices. Using this insight, we can formulate a metric to evaluate such classifiers. To do this, we first need to understand two key concepts:

- The True Positive Rate (TPR) is the ratio of True Positive predictions to all actual Positive cases. That is, it's the recall (Equation 12). It assesses the classifier's efficacy with respect to the positive portion of the dataset.

$$TPR = \frac{TP}{TP + FN} = R \tag{15}$$

- The False Positive Rate (FPR) is the ratio of False Positive predictions to all actual Negative cases. It assesses the classifier's efficacy concerning the negative portion of the dataset, but in a reverse manner. That is, a higher FPR means worse performance regarding the negatives, and lower FPR means better performance (Equation 17).

$$FPR = \frac{FP}{TN + FP} \tag{16}$$

$$= 1 - \frac{TN}{TN + FP} \tag{17}$$

The fundamental observation is that selecting a particular probability threshold gives rise to specific values in the confusion matrix, which in turn provides a pair of values (FPR, TPR). Plotting these points as the threshold varies creates the *Receiver Operating Characteristic* (ROC) curve.

To introduce the AUC measure, let's further explore the nuances of the ROC curve (Refer to Figure 41):

1. A threshold of 0% implies that every item is classified as a positive, thus, there are no False Negatives and no True Negatives, which means that the FPR and TPR are both 1.

2. A threshold of 100% implies that every item is classified as a negative, thus, there are no True Positives and no False Positives, which means that the FRP and TPR are both 0.

3. Combining these two observations, if we plot the curve by decreasing the threshold from 100% to 0%, we will obtain a curve starting from point $(0, 0)$ and ending at point $(1, 1)$.

4. A perfect classification system would make no mistakes, no matter the probability threshold[41]. That is, it would have no False Positives and no False Negatives. This would make the TPR always 1, which corresponds to a horizontal line 1 unit above the horizontal axis.

[41] Except for thresholds of 0% and 100%, as explained above

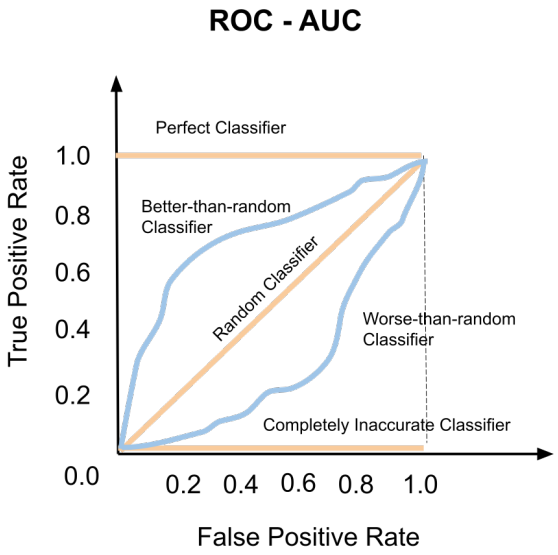


Figure 41: Area Under Curve

5. A completely inaccurate classification system is always wrong, no matter the probability threshold. That is, it would never have True Positives nor True Negatives. This would make the TPR always 0, which corresponds to a horizontal line on top of the horizontal axis.

6. A classification system that always takes decisions at random, is equally likely to correctly identify positive instances as it is to incorrectly label negative instances as positive. This means, the FPR and TPR would always be equal, which corresponds to a diagonal line.

7. A classification system that performs better than a random classifier will have a performance tending towards a perfect classifier, which corresponds to a curve above the diagonal.

8. A classification system that performs worse than a random classifier will have a performance tending towards a completely inaccurate classifier, which corresponds to a curve below the diagonal.

The AUC is simply the area under the ROC curve, and can be used to evaluate probabilistic classifiers, because:

- A perfect classifier has an AUC of 1.0
- A completely inaccurate classifier has an AUC of 0.0
- A random classifier has an AUC of 0.5
- A better-than-random classifier has an AUC above 0.5
- A worse-than-random classifier has an AUC below 0.5

Organism of interest	Type of organism	Search Query
Fusarium oxysporum f. sp. cubense tropical (race 4)	Fungus	fusarium oxysporum tropical
Xylella fastidiosa	Bacteril	xylella
Bursaphelenchus xylophilus	Nematode	Bursaphelenchus xylophilus
Bactrocera dorsalis	Insect	Bactrocera dorsalis
Candidatus Liberibacter spp.	Bacteria	huanglongbing
Popillia japonica	Insect	Popillia Japonica
Flavescence dorée	Bacteria	flavescence
Tomato brown rugose fruit virus	Virus	ToBRFV
Spodoptera frugiperda	Insect	spodoptera frugiperda
Bretziella fagacearum	Fungus	oak wilt Bretziella
Agrilus planipennis	Insect	Agrilus planipennis
Thaumatotibia leucotreta	Insect	Thaumatotibia leucotreta
Xylotrechus chinensis	Insect	Xylotrechus chinensis
Toumeyella parvicornis	Insect	Toumeyella parvicornis
Ceratocystis platani	Fungus	Ceratocystis platani chancre du platane
General Plant Health		first report plant disease new plant health

Table 27: Keywords and key phrases used for the VSIdataset

B VSI Dataset samples and language statistics

This appendix presents tables and statistics to provide a deeper insight into the characteristics of the VSI dataset.

B.1 Keywords for constructing the VSI Database

B.2 Sample entries detected as Latin

Text
Bactrocera (Daculus) oleae (Gmelin, 1790)
resting on a leaf - Popillia japonica
La Popillia japonica devasta colture in Piemonte
Regione attiva piano anti Popillia japonica - www.lombardianotizie.online …
Popillia japonica | Libero Quoditiano.it
Lombardy: anti Popillia japonica plan activated in Milan – Lombardy
Candidatus liberibacter ç¬¬1é¡µ
Milano, area di San Siro: Regione attiva piano anti Popillia japonica
A Milano il piano anti Popillia japonica | Milano.zone
Vithal Popillia japonica Polysect Ultra SL PFnPO Le Migliori Venditori
ultime notizie su ore popillia
Meridiem Seeds presenta Yuparanà nel Lazio
Lombardia: al via piano regionale contro la Popillia japonica a San Siro (2)
Milano, area di San Siro: Regione attiva piano anti Popillia japonica
Control of Anthracnose …by Colletotrichum musae …
…on Curcuma alismatifolia Gagnep using Antagonistic Bacillus spp.
Lombardia: al via piano regionale contro la Popillia japonica a San Siro
Popillia japonica , come difendersi
Japankäfer Popillia japonica
RETI INSETTICIDE PER Popillia japonica - Asso Web TV
La Popillia japonica
Citrus tristeza virus …of Candidatus Liberibacter Asiaticus by Diaphorina citri
Popillia japonica
Popillia devastante nel Santhiatese - Prima Vercelli

Table 28: Sample entries detected as Latin text

Language	Entries	Language	Entries	Language	Entries
af	18	id	226	ro	77
am	2	ig	3	ru	180
ar	226	ilo	24	rw	13
az	13	is	34	sd	1
be	3	it	4788	sk	66
bg	107	iw	13	sl	51
bn	6	ja	74	sm	1
bs	33	jv	2	sn	8
ca	142	jw	1	so	4
ceb	7	ka	13	sq	18
co	121	kk	7	sr	12
cs	94	kn	4	su	1
cy	5	ko	51	sv	53
da	35	kri	5	sw	14
de	362	ku	3	ta	4
el	164	ky	1	te	9
en	12016	la	152	tg	12
eo	7	lb	28	th	31
es	3025	ln	3	tk	3
et	31	lo	4	tl	3
eu	3	lt	26	tr	83
fa	36	lv	11	ts	3
fi	25	mg	4	tt	1
fil	11	mi	5	uk	30
fr	2251	mk	1	ur	1
fy	12	mo	8	uz	11
ga	8	mr	19	vi	210
gd	3	ms	17	yi	1
gl	18	mt	6	yo	1
gn	69	ne	2	zh	41
gu	6	nl	465	zh-CN	784
ha	2	no	29	zh-TW	72
haw	3	ny	6	zu	4
hi	39	om	11	Failed	6132
hmn	2	pl	135		
hr	77	ps	3		
ht	4	pt	1292		
hu	184	qu	4		
hy	2				

Table 29: Number of entries per language in the VSI Dataset

B.3 VSI Language statistics

Table 29 shows the distribution of languages (by ISO code) in the VSI dataset as automatically detected by the Google Translate API.

C Regular expressions used in this project

In this Appendix, one will find illustrative examples of the regular expressions utilized for various tasks in this project. It is important to note that our matching algorithms are case-insensitive.

C.1 Error Message Removal - VSI Dataset

Regular Expressions
^JavaScript is not available\.$
^Error$
^NA$
^Timeout error$
^None$
^Loading(\.)*$
^Not Found$
^Access Restricted$
^Page Not Found$
^\[\]$
^blacklisted
^'NoneType' object has no attribute 'get'?$
^PPlease Wait\.\.\.\s+ Cloudflare$
^HTTPSConnectionPool.*
^Checking your browser.*
^Please update your browser$
^JavaScript n'est pas disponible\.$
^Vos données\.% Votre expérience\.*$
^Before you continue to YouTube$
^Just a moment(\.)*
^Your data\. Your experience\.$
^Access Denied$
…

Table 30: Some Regular Expressions for Removing Errors Messages in the VSI dataset

C.2 Noise Removal - VSI Dataset

Description	**Regular Expression**	**Explanation**
Emojis	`[\U00010000-\U0010ffff]`	Unicode emojis
Hyphen-like Characters	`[\u002D\u058A\u2010\u2011\u2012 \u2013\u2014\u2015\u2E3A \u2E3B\uFE58\uFE63\uFF0D]`	Common hyphen-like characters in Unicode such as - ‐ ‑ ‒ – — ―, etc.
Quotation-mark-like Characters	`[\u00BB\u00AB\u25B7\u2018\u2019 \u201C\u201D\u2039\u203A \u300C\u300D\u300E\u300F \u301D\u301E\u301F\uFE41 \uFE42\uFE43\uFE44\uFF02 \uFF07\uFF62\uFF63]`	Common quotation-mark-like characters in Unicode, such as » « ‘ ’ “ ” ‹ ›
URL	`\bhttps?://\S+\b`	
Date	`\b{4}[.-/]{1,2}[.-/]{1,2}\b|\b{1,2}[.-/]{1,2}[.-/]{4}\b|\b{1,2}[.-/]{1,2}[.-/]{1,2}\b`	Date pattern for dates in formats YYYY-MM-DD, DD-MM-YYYY, DD.MM.YYYY, DD/MM/YYY, etc....
Hours	`\b{1,2}:{2}(?::{2})?\b`	Hours pattern e.g. 10:30 AM, 15:45, or 2023-05-15 10:30:00. Also, 15:45:30 and 2023-05-15T12:30:45+00:00, etc

Table 31: Regular Expressions for Noise Removal

D Preprocesing Statistics

In this Appendix, we present the full results of our preprocessing approaches, as explained in Section 5.

D.1 Preprocessing the VSI dataset

Unique entries per Content Source	Raw	After preprocessing	Percentage
Title	24037	16222	67.49%
Abstract	19939	16974	85.13%
Full text	26346	22535	85.53%
Translated Title	23829	16359	68.65%
Phrases with Keywords (Abstract)	4712	4050	85.95%
Phrases with Keywords + O.C (Abstract)	19798	16833	85.02%
Phrases with Keywords (Full text)	16473	14359	87.17%
Phrases with Keywords + O.C (Full text)	24447	20880	85.41%

Table 32: Unique Entries per content source before and after preprocessing

Content Source	Size	Mean	STD	Min	25%	50%	75%	max
Title	16222	12.464	29.867	1	8	11	15	1565
Abstract	16974	50.549	73.982	1	19	27	46	1537
Full text	22535	1097.756	5763.482	1	187	411	800	240318
Translated Title	16359	12.821	30.943	1	8	11	15	1565
Phrases with Keywords (Abstract)	4050	34.919	33.765	1	18	25	39	537
Phrases with Keywords + O.C (Abstract)	16833	35.768	44.700	1	17	25	38	1518
Phrases with Keywords (Full text)	14359	115.655	429.369	1	26	52	114	40700
Phrases with Keywords + O.C (Full text)	20880	217.711	614.478	1	32	71	201	40700

Table 33: Token Length statistics after preprocessing the VSI dataset

Content Source	Irrelevant	Relevant	Size	% Irrelevant	% Relevant
Title	13917	2305	16222	85.8%	14.2%
Abstract	14625	2349	16974	86.2%	13.8%
Full text	19794	2741	22535	87.8%	12.2%
Translated Title	14084	2275	16359	86.1%	13.9%
Phrases with Keywords (Abstract)	2962	1088	4050	73.1%	26.9%
Phrases with Keywords + O.C (Abstract)	14553	2280	16833	86.5%	13.5%
Phrases with Keywords (Full text)	12130	2229	14359	84.5%	15.5%
Phrases with Keywords + O.C (Full text)	18344	2536	20880	87.9%	12.1%

Table 34: Label Distribution after preprocessing the VSI dataset

Node Names	CPU	RAM	GPU
n[1-5]	2 x Intel Xeon E5-2620 v4 8 cores / 16 threads @ 2.40 GHz (Haswell)	128 GiB	2 x NVIDIA RTX A6000 with **48 GiB** GDDR6 (PCIe)
n[51-55]	2 x Intel Xeon Gold 5120 14 cores / 28 threads @ 2.2GHz (Skylake)	192 GiB	3 x NVIDIA Tesla V100 with **32 GiB** of RAM (PCIe)
n[101-102]	2 x Intel Xeon Gold 6148 20 cores / 40 threads @ 2.4 GHz (Skylake)	384 GiB	4 x NVIDIA Tesla V100 with **32 GiB** of RAM (NVLink)

Table 35: Specifications of `Lab-IA` Nodes

E Training Setup

E.1 GPU Specifications

Content Source	Train-Size	Positives	Negatives	Dev-Size	Positives	Negatives	Test-Size	Positives	Negatives
Title	12980	1845	11135	1621	230	1391	1621	230	1391
Abstract	13582	1881	11701	1696	234	1462	1696	234	1462
Full text	18031	2193	15838	2252	274	1978	2252	274	1978
Translated Title	13091	1821	11270	1634	227	1407	1634	227	1407
Phrases with Keywords (Abstract)	3242	872	2370	404	108	296	404	108	296
Phrases with Keywords + O.C (Abstract)	13469	1826	11643	1682	227	1455	1682	227	1455
Phrases with Keywords (Full text)	11491	1785	9706	1434	222	1212	1434	222	1212
Phrases with Keywords + O.C (Full text)	16706	2030	14676	2087	253	1834	2087	253	1834

Table 36: Split Statistics for Unbalanced Fine-tuning

Content Source	Train-Size	Positives	Negatives	Dev-Size	Positives	Negatives	Test-Size	Positives	Negatives
Title	12976	6488	6488	1622	811	811	1621	230	1391
Abstract	13578	6789	6789	1696	848	848	1696	234	1462
Full text	18028	9014	9014	2252	1126	1126	2252	274	1978
Translated Title	13086	6543	6543	1634	817	817	1634	227	1407
Phrases with Keywords (Abstract)	3240	1620	1620	404	202	202	404	108	296
Phrases with Keywords + O.C (Abstract)	13466	6733	6733	1682	841	841	1682	227	1455
Phrases with Keywords (Full text)	11486	5743	5743	1434	717	717	1434	222	1212
Phrases with Keywords + O.C (Full text)	16704	8352	8352	2088	1044	1044	2087	253	1834

Table 37: Split Statistics for Balanced Fine-tuning

E.2 Statistics for Splits

Content Source	Train-Size	Positives	Negatives	Dev-Size	Positives	Negatives	Test-Size	Positives	Negatives	Unlabeled
Title	100	47	53	1622	811	811	1621	230	1391	12876
Abstract	100	55	45	1696	848	848	1696	234	1462	13478
Full text	100	53	47	2252	1126	1126	2252	274	1978	17928
Translated Title	100	56	44	1634	817	817	1634	227	1407	12986
Phrases with Keywords (Abstract)	100	42	58	404	202	202	404	108	296	3140
Phrases with Keywords + O.C (Abstract)	100	59	41	1682	841	841	1682	227	1455	13366
Phrases with Keywords (Full text)	100	50	50	1434	717	717	1434	222	1212	11386
Phrases with Keywords + O.C (Full text)	100	51	49	2088	1044	1044	2087	253	1834	16604

Table 38: Split Statistics for PET 50

Content Source	Train-Size	Positives	Negatives	Dev-Size	Positives	Negatives	Test-Size	Positives	Negatives	Unlabeled
Title	200	102	98	1622	811	811	1621	230	1391	12776
Abstract	200	97	103	1696	848	848	1696	234	1462	13378
Full text	200	109	91	2252	1126	1126	2252	274	1978	17828
Translated Title	200	100	100	1634	817	817	1634	227	1407	12886
Phrases with Keywords (Abstract)	200	78	122	404	202	202	404	108	296	3040
Phrases with Keywords + O.C (Abstract)	200	115	85	1682	841	841	1682	227	1455	13266
Phrases with Keywords (Full text)	200	105	95	1434	717	717	1434	222	1212	11286
Phrases with Keywords + O.C (Full text)	200	96	104	2088	1044	1044	2087	253	1834	16504

Table 39: Split Statistics for PET 100

Content Source	Train-Size	Positives	Negatives	Dev-Size	Positives	Negatives	Test-Size	Positives	Negatives	Unlabeled
Title	400	201	199	1622	811	811	1621	230	1391	12576
Abstract	400	206	194	1696	848	848	1696	234	1462	13178
Full text	400	226	174	2252	1126	1126	2252	274	1978	17628
Translated Title	400	203	197	1634	817	817	1634	227	1407	12686
Phrases with Keywords (Abstract)	400	170	230	404	202	202	404	108	296	2840
Phrases with Keywords + O.C (Abstract)	400	206	194	1682	841	841	1682	227	1455	13066
Phrases with Keywords (Full text)	400	200	200	1434	717	717	1434	222	1212	11086
Phrases with Keywords + O.C (Full text)	400	184	216	2088	1044	1044	2087	253	1834	16304

Table 40: Split Statistics for PET 200

Content Source	Train-Size	Positives	Negatives	Dev-Size	Positives	Negatives	Test-Size	Positives	Negatives	Unlabeled
Title	1000	481	519	1622	811	811	1621	230	1391	11976
Abstract	1000	512	488	1696	848	848	1696	234	1462	12578
Full text	1000	513	487	2252	1126	1126	2252	274	1978	17028
Translated Title	1000	482	518	1634	817	817	1634	227	1407	12086
Phrases with Keywords (Abstract)	1000	481	519	404	202	202	404	108	296	2240
Phrases with Keywords + O.C (Abstract)	1000	524	476	1682	841	841	1682	227	1455	12466
Phrases with Keywords (Full text)	1000	499	501	1434	717	717	1434	222	1212	10486
Phrases with Keywords + O.C (Full text)	1000	492	508	2088	1044	1044	2087	253	1834	15704

Table 41: Split Statistics for PET 500

Content Source	Train-Size	Positives	Negatives	Dev-Size	Positives	Negatives	Test-Size	Positives	Negatives	Unlabeled
Title	2000	956	1044	1622	811	811	1621	230	1391	10976
Abstract	2000	1000	1000	1696	848	848	1696	234	1462	11578
Full text	2000	1002	998	2252	1126	1126	2252	274	1978	16028
Translated Title	2000	975	1025	1634	817	817	1634	227	1407	11086
Phrases with Keywords (Abstract)	2000	987	1013	404	202	202	404	108	296	1240
Phrases with Keywords + O.C (Abstract)	2000	1026	974	1682	841	841	1682	227	1455	11466
Phrases with Keywords (Full text)	2000	963	1037	1434	717	717	1434	222	1212	9486
Phrases with Keywords + O.C (Full text)	2000	996	1004	2088	1044	1044	2087	253	1834	14704

Table 42: Split Statistics for PET 1000

Model/Content Source	Title	Abstract	Full text	Translated Title	Phrases with Keywords (Abstract)	Phrases with Keywords + O.C (Abstract)	Phrases with Keywords (Full text)	Phrases with Keywords + O.C (Full text)
BERT-base	2	2	2	2	2	2	2	2
mBERT	4	2	4	4	2	2	4	10
Bio-Link-BERT	7	10	2	3	4	6	10	4
SciBERT	2	2	2	2	2	2	2	9
RoBERTa-base	8	7	7	4	4	9	3	4
XLM-RoBERTa	5	10	6	8	4	9	9	6

Table 43: Optimal epochs for Unbalanced Fine-tuning

Model/Content Source	Title	Abstract	Full text	Translated Title	Phrases with Keywords (Abstract)	Phrases with Keywords + O.C (Abstract)	Phrases with Keywords (Full text)	Phrases with Keywords + O.C (Full text)
BERT-base	1	1	1	1	1	1	3	1
mBERT	1	1	1	1	1	1	1	1
Bio-Link-BERT	1	1	1	2	1	1	1	1
SciBERT	1	1	1	1	1	1	1	1
RoBERTa-base	1	1	1	1	1	2	1	1
XLM-RoBERTa	3	1	1	1	2	1	1	1

Table 44: Optimal epochs for Balanced Fine-tuning

F Training Results

In this Appendix we present the numerical results and metrics from our training experiments.

F.1 VSI Optimal Epochs

Model/Content Source	Title	Abstract	Full text	Translated Title	Phrases with Keywords (Abstract)	Phrases with Keywords + O.C (Abstract)	Phrases with Keywords (Full text)	Phrases with Keywords + O.C (Full text)
BERT-base	3	1	2	3	4	1	3	2
mBERT	1	2	2	1	3	1	2	2
Bio-Link-BERT	1	1	3	2	2	1	3	3
SciBERT	2	1	2	5	3	2	1	2
RoBERTa-base	4	1	1	1	1	1	3	2
XLM-RoBERTa	2	1	1	1	1	2	2	1

Table 45: Optimal epochs for PET 50

Model/Content Source	Title	Abstract	Full text	Translated Title	Phrases with Keywords (Abstract)	Phrases with Keywords + O.C (Abstract)	Phrases with Keywords (Full text)	Phrases with Keywords + O.C (Full text)
BERT-base	1	1	1	1	1	1	1	1
mBERT	2	1	1	1	4	3	2	1
Bio-Link-BERT	2	4	1	4	2	3	1	1
SciBERT	2	2	3	1	3	1	1	3
RoBERTa-base	1	1	1	2	3	1	1	1
XLM-RoBERTa	3	1	1	4	3	1	1	2

Table 46: Optimal epochs for PET 100

Model/Content Source	Title	Abstract	Full text	Translated Title	Phrases with Keywords (Abstract)	Phrases with Keywords + O.C (Abstract)	Phrases with Keywords (Full text)	Phrases with Keywords + O.C (Full text)
BERT-base	4	3	1	1	2	1	1	1
mBERT	2	1	1	3	2	1	2	2
Bio-Link-BERT	3	4	4	2	3	3	4	4
SciBERT	3	3	2	1	1	4	1	2
RoBERTa-base	2	4	2	2	1	2	2	2
XLM-RoBERTa	3	3	1	5	1	4	1	1

Table 47: Optimal epochs for PET 200

Model/Content Source	Title	Abstract	Full text	Translated Title	Phrases with Keywords (Abstract)	Phrases with Keywords + O.C (Abstract)	Phrases with Keywords (Full text)	Phrases with Keywords + O.C (Full text)
BERT-base	4	1	3	2	1	3	2	3
mBERT	3	3	3	1	1	3	1	3
Bio-Link-BERT	3	4	2	3	1	3	3	3
SciBERT	3	2	1	1	1	2	4	2
RoBERTa-base	5	1	4	2	2	2	2	2
XLM-RoBERTa	2	2	3	2	1	1	2	1

Table 48: Optimal epochs for PET 500

Model/Content Source	Title	Abstract	Full text	Translated Title	Phrases with Keywords (Abstract)	Phrases with Keywords + O.C (Abstract)	Phrases with Keywords (Full text)	Phrases with Keywords + O.C (Full text)
BERT-base	1	1	3	1	1	2	1	1
mBERT	1	1	5	2	1	3	1	1
Bio-Link-BERT	2	3	2	1	1	1	1	4
SciBERT	2	2	2	2	1	1	1	1
RoBERTa-base	1	5	1	1	1	4	3	1
XLM-RoBERTa	1	2	5	1	1	5	1	3

Table 49: Optimal epochs for PET 1000

F.2 VSI Metrics at Optimal Epochs

Content Source	Unbalanced Fine-tuning	Balanced Fine-tuning	PET 50	PET 100	PET 200	PET 500	PET 1000	Content Source
Title	68.22%	64.97%	49.6%	38.3%	65.7%	71.14%	73.92%	Title
Abstract	61.15%	67.6%	62.74%	32.01%	61.64%	66.78%	74.35%	Abstract
Full text	68.94%	69.76%	53.02%	70.37%	73.16%	81.35%	83.1%	Full text
Translated Title	69.91%	70.72%	78.96%	75.88%	76.32%	74.07%	74.43%	Translated Title
Phrases with Keywords (Abstract)	65.22%	71.22%	47.74%	49.12%	44.06%	62.44%	65.59%	Phrases with Keywords (Abstract)
Phrases with Keywords + O.C (Abstract)	64.58%	67.59%	69.47%	76.91%	70.52%	77.68%	80.16%	Phrases with Keywords + O.C (Abstract)
Phrases with Keywords (Full text)	63.96%	64.55%	52.23%	67.21%	75.84%	71.83%	71.21%	Phrases with Keywords (Full text)
Phrases with Keywords + O.C (Full text)	62.25%	70.31%	70.28%	70.16%	70.95%	74.63%	75.47%	Phrases with Keywords + O.C (Full text)

Table 50: BERT-base F_2 on Development split

Content Source	Unbalanced Fine-tuning	Balanced Fine-tuning	PET 50	PET 100	PET 200	PET 500	PET 1000	Content Source
Title	60.05%	68.5%	53.03%	67.66%	67.81%	81.06%	82.44%	Title
Abstract	0%	71.31%	51.25%	48.59%	69.07%	77.38%	79.71%	Abstract
Full text	61.29%	75.24%	66.17%	75.67%	83.29%	84.72%	82.75%	Full text
Translated Title	0%	75.57%	68.61%	79.5%	78.36%	80.32%	82.87%	Translated Title
Phrases with Keywords (Abstract)	73.48%	77.15%	19.74%	55.67%	54.88%	72.77%	77.48%	Phrases with Keywords (Abstract)
Phrases with Keywords + O.C (Abstract)	65.49%	69.45%	64.76%	79.35%	81.28%	79.34%	81.64%	Phrases with Keywords + O.C (Abstract)
Phrases with Keywords (Full text)	61.53%	76.07%	33.46%	71.9%	77.79%	78.15%	80.25%	Phrases with Keywords (Full text)
Phrases with Keywords + O.C (Full text)	0%	67.81%	66.93%	68.66%	70.16%	79.26%	82.29%	Phrases with Keywords + O.C (Full text)

Table 51: mBERT F_2 on Development split

Content Source	Unbalanced Fine-tuning	Balanced Fine-tuning	PET 50	PET 100	PET 200	PET 500	PET 1000	Content Source
Title	65.93%	79.97%	79.11%	74.01%	77.01%	81.94%	81.2%	Title
Abstract	41.85%	74.63%	22.3%	54.82%	65.14%	73.95%	76.57%	Abstract
Full text	69.93%	75.49%	66.94%	73.6%	76.35%	81.94%	83.25%	Full text
Translated Title	68.61%	74.59%	83.77%	81.31%	80.02%	78.83%	80.65%	Translated Title
Phrases with Keywords (Abstract)	70.29%	77.67%	58.07%	35.71%	59.58%	73.34%	72.89%	Phrases with Keywords (Abstract)
Phrases with Keywords + O.C (Abstract)	53.88%	74.05%	70.33%	81.95%	77.02%	81.2%	79.35%	Phrases with Keywords + O.C (Abstract)
Phrases with Keywords (Full text)	0%	75.78%	55.44%	68.61%	75.76%	77.37%	80.19%	Phrases with Keywords (Full text)
Phrases with Keywords + O.C (Full text)	61.97%	73.53%	63.54%	67.28%	71.73%	79.5%	79.74%	Phrases with Keywords + O.C (Full text)

Table 52: Bio-Link-BERT F_2 on Development split

Content Source	Unbalanced Fine-tuning	Balanced Fine-tuning	PET 50	PET 100	PET 200	PET 500	PET 1000	Content Source
Title	64.18%	69.04%	58.89%	60.75%	71.78%	75.91%	80.37%	Title
Abstract	57.89%	64.95%	58.02%	48.16%	67.4%	78.08%	75.47%	Abstract
Full text	68.18%	70.13%	59.93%	73.05%	71.87%	82.43%	83.95%	Full text
Translated Title	70.89%	73.72%	78.28%	78.12%	76.74%	74.26%	78.27%	Translated Title
Phrases with Keywords (Abstract)	53.6%	71.43%	55.61%	32.45%	40.86%	72.84%	67.11%	Phrases with Keywords (Abstract)
Phrases with Keywords + O.C (Abstract)	64.44%	70.34%	74.67%	78.99%	77.94%	78.04%	78.93%	Phrases with Keywords + O.C (Abstract)
Phrases with Keywords (Full text)	64.38%	71.96%	55.79%	73.47%	70.7%	71.55%	75.03%	Phrases with Keywords (Full text)
Phrases with Keywords + O.C (Full text)	54.55%	61.84%	73.97%	75.8%	75.86%	76.82%	80.13%	Phrases with Keywords + O.C (Full text)

Table 53: SciBERT F_2 on Development split

Content Source	Unbalanced Fine-tuning	Balanced Fine-tuning	PET 50	PET 100	PET 200	PET 500	PET 1000	Content Source
Title	0%	77.14%	48.3%	36.17%	57.09%	77.73%	76.28%	Title
Abstract	0%	79.77%	31.98%	35.32%	49.85%	68.98%	79.66%	Abstract
Full text	64.15%	79.41%	42.48%	73.41%	79.01%	82.72%	83.18%	Full text
Translated Title	64.66%	74.07%	83.54%	71.96%	71.55%	80.25%	81.02%	Translated Title
Phrases with Keywords (Abstract)	67.61%	68.03%	31.28%	55.11%	54.27%	75.48%	82.78%	Phrases with Keywords (Abstract)
Phrases with Keywords + O.C (Abstract)	0%	72.03%	57.81%	77.08%	73.51%	77.25%	80.51%	Phrases with Keywords + O.C (Abstract)
Phrases with Keywords (Full text)	0%	77.08%	46.61%	70.47%	70.85%	69.86%	74.53%	Phrases with Keywords (Full text)
Phrases with Keywords + O.C (Full text)	0%	71.84%	67.83%	59.86%	69.82%	73.4%	78.52%	Phrases with Keywords + O.C (Full text)

Table 54: RoBERTa-base F_2 on Development split

Content Source	Unbalanced Fine-tuning	Balanced Fine-tuning	PET 50	PET 100	PET 200	PET 500	PET 1000	Content Source
Title	0%	70.92%	22.25%	62.33%	70.07%	80.11%	83.12%	Title
Abstract	48.6%	75.74%	53.67%	63.15%	72.72%	78.69%	82.34%	Abstract
Full text	0%	83.62%	61.73%	82.62%	80.51%	85.64%	85.73%	Full text
Translated Title	48.99%	80.16%	82.89%	80.27%	81.22%	78.36%	79.91%	Translated Title
Phrases with Keywords (Abstract)	0%	76.44%	38.38%	58%	52.36%	78.04%	82.85%	Phrases with Keywords (Abstract)
Phrases with Keywords + O.C (Abstract)	0%	77.82%	75.58%	81.94%	78.55%	80.79%	81.13%	Phrases with Keywords + O.C (Abstract)
Phrases with Keywords (Full text)	48.65%	74.94%	43.41%	73.3%	80.04%	79.59%	79%	Phrases with Keywords (Full text)
Phrases with Keywords + O.C (Full text)	59.01%	63.35%	67.21%	69.72%	77.95%	82.49%	80.51%	Phrases with Keywords + O.C (Full text)

Table 55: XLM-RoBERTa F_2 on Development split

Content Source	Unbalanced Fine-tuning	Balanced Fine-tuning	PET 50	PET 100	PET 200	PET 500	PET 1000	Content Source
Title	87.58%	84.33%	60.55%	66.45%	68.61%	73.41%	78.43%	Title
Abstract	84.6%	84.01%	62.84%	63.3%	68.22%	70.49%	77.93%	Abstract
Full text	90.23%	87.46%	60.5%	66.04%	72.51%	79.66%	85.09%	Full text
Translated Title	88.8%	87.35%	61.03%	67.21%	73.89%	80.89%	82.94%	Translated Title
Phrases with Keywords (Abstract)	76.56%	74.04%	58.04%	57.61%	62.62%	63.03%	61.13%	Phrases with Keywords (Abstract)
Phrases with Keywords + O.C (Abstract)	85.87%	84.02%	64.03%	67.59%	69.42%	76.22%	80.2%	Phrases with Keywords + O.C (Abstract)
Phrases with Keywords (Full text)	86.38%	83.49%	64.86%	69.48%	70.67%	74.13%	78.18%	Phrases with Keywords (Full text)
Phrases with Keywords + O.C (Full text)	86.47%	85.65%	63%	68.14%	70.01%	75.19%	79.16%	Phrases with Keywords + O.C (Full text)

Table 56: BERT-base AUC on Development split

Content Source	Unbalanced Fine-tuning	Balanced Fine-tuning	PET 50	PET 100	PET 200	PET 500	PET 1000	Content Source
Title	89.58%	85.08%	62.69%	70.39%	71.14%	80.93%	83.46%	Title
Abstract	59.16%	85.66%	64.84%	65.77%	70.88%	79.8%	82.81%	Abstract
Full text	92.01%	90.38%	62.79%	71.3%	80.29%	86.75%	88.82%	Full text
Translated Title	63.71%	86.85%	66.74%	70.19%	73.87%	81.88%	85.27%	Translated Title
Phrases with Keywords (Abstract)	82.77%	80.77%	59.16%	59.64%	65.64%	68.93%	67.59%	Phrases with Keywords (Abstract)
Phrases with Keywords + O.C (Abstract)	86.77%	85.77%	61.12%	69.23%	74.41%	81.8%	84.51%	Phrases with Keywords + O.C (Abstract)
Phrases with Keywords (Full text)	87.56%	87.23%	62.65%	70.19%	73.28%	80.17%	82.43%	Phrases with Keywords (Full text)
Phrases with Keywords + O.C (Full text)	66.75%	86.51%	61.35%	68.66%	74.04%	81.62%	83.35%	Phrases with Keywords + O.C (Full text)

Table 57: mBERT AUC on Development split

Content Source	Unbalanced Fine-tuning	Balanced Fine-tuning	PET 50	PET 100	PET 200	PET 500	PET 1000	Content Source
Title	87.92%	84.18%	63.14%	70.27%	69.32%	76.65%	79.8%	Title
Abstract	83.07%	83.17%	61.32%	64.91%	71.08%	76.67%	79.93%	Abstract
Full text	90.3%	87.55%	62.36%	69.01%	74.07%	82.41%	85.16%	Full text
Translated Title	89.25%	87.81%	67.29%	71.45%	73.33%	81.22%	83.03%	Translated Title
Phrases with Keywords (Abstract)	76.2%	74.15%	57.97%	57.97%	64.61%	68.24%	64.57%	Phrases with Keywords (Abstract)
Phrases with Keywords + O.C (Abstract)	77.04%	83.74%	59.48%	67.77%	69.91%	77.49%	79.01%	Phrases with Keywords + O.C (Abstract)
Phrases with Keywords (Full text)	69.36%	82.23%	67.15%	69.43%	72.1%	76.44%	77.69%	Phrases with Keywords (Full text)
Phrases with Keywords + O.C (Full text)	87.07%	84.14%	58.85%	66.96%	71.86%	78.92%	82.23%	Phrases with Keywords + O.C (Full text)

Table 58: Bio-Link-BERT AUC on Development split

Content Source	Unbalanced Fine-tuning	Balanced Fine-tuning	PET 50	PET 100	PET 200	PET 500	PET 1000	Content Source
Title	85.92%	85.31%	60.48%	66.94%	71.19%	77.66%	80.88%	Title
Abstract	83.87%	84.7%	61.99%	62.9%	67.7%	76.74%	80.41%	Abstract
Full text	89.14%	88.28%	60.01%	65.82%	71.05%	79.82%	84.83%	Full text
Translated Title	90.07%	88.25%	63.6%	71.77%	72.58%	78.79%	83.33%	Translated Title
Phrases with Keywords (Abstract)	60.81%	74.59%	56.9%	57.95%	63.58%	65.37%	62.92%	Phrases with Keywords (Abstract)
Phrases with Keywords + O.C (Abstract)	85.99%	83.41%	62.9%	68.86%	70.51%	78.24%	79.8%	Phrases with Keywords + O.C (Abstract)
Phrases with Keywords (Full text)	87.06%	85.17%	63.77%	68.39%	70.87%	75.59%	79.12%	Phrases with Keywords (Full text)
Phrases with Keywords + O.C (Full text)	85.58%	83.59%	64.93%	69.68%	73.13%	78.18%	81.2%	Phrases with Keywords + O.C (Full text)

Table 59: SciBERT AUC on Development split

Content Source	Unbalanced Fine-tuning	Balanced Fine-tuning	PET 50	PET 100	PET 200	PET 500	PET 1000	Content Source
Title	70.16%	84.16%	61.59%	68.73%	71.31%	77.09%	80.67%	Title
Abstract	61.35%	85.35%	61.03%	60.8%	66.09%	73.05%	81.69%	Abstract
Full text	90.61%	89.72%	61.17%	66.13%	73.06%	81.03%	84.94%	Full text
Translated Title	88.78%	86.82%	64.43%	70.05%	74.71%	82.01%	84.2%	Translated Title
Phrases with Keywords (Abstract)	77.87%	70.52%	58.25%	58.55%	62.3%	68.92%	63.71%	Phrases with Keywords (Abstract)
Phrases with Keywords + O.C (Abstract)	64.25%	84.43%	63.24%	68.62%	70%	77.66%	81.8%	Phrases with Keywords + O.C (Abstract)
Phrases with Keywords (Full text)	67.9%	82.78%	64.45%	70.02%	71.33%	74.8%	80.06%	Phrases with Keywords (Full text)
Phrases with Keywords + O.C (Full text)	65.28%	85.97%	65.87%	69.46%	72.09%	77.1%	81%	Phrases with Keywords + O.C (Full text)

Table 60: RoBERTa-base AUC on Development split

Content Source	Unbalanced Fine-tuning	Balanced Fine-tuning	PET 50	PET 100	PET 200	PET 500	PET 1000	Content Source
Title	45.39%	84.85%	61.54%	69.62%	73.51%	82.48%	83.55%	Title
Abstract	84.81%	86.26%	62.55%	66.29%	74.06%	81.92%	84.15%	Abstract
Full text	41.19%	90.57%	60.9%	70%	78.04%	86.66%	88.85%	Full text
Translated Title	76.76%	84.24%	63.84%	70.52%	74.28%	82.48%	84.19%	Translated Title
Phrases with Keywords (Abstract)	61.8%	73.74%	59.34%	61.82%	61.68%	69.38%	70.57%	Phrases with Keywords (Abstract)
Phrases with Keywords + O.C (Abstract)	54.71%	86.08%	64.83%	72.02%	75.57%	82.09%	84.82%	Phrases with Keywords + O.C (Abstract)
Phrases with Keywords (Full text)	64.07%	78.83%	68.78%	72.97%	74.48%	79.52%	80.14%	Phrases with Keywords (Full text)
Phrases with Keywords + O.C (Full text)	85.01%	85.27%	62.15%	69.31%	73.44%	80.73%	83.46%	Phrases with Keywords + O.C (Full text)

Table 61: XLM-RoBERTa AUC on Development split

Model	Content Source	Training Method	Development F_2	Development AUC
BERT-base	Full text	PET 500	81.35%	79.66%
BERT-base	Full text	PET 1000	83.1%	85.09%
BERT-base	Phrases with Keywords + O.C (Abstract)	PET 1000	80.16%	80.2%
mBERT	Abstract	PET 1000	79.71%	82.81%
mBERT	Full text	PET 200	83.29%	80.29%
mBERT	Full text	PET 500	84.72%	86.75%
mBERT	Full text	PET 1000	82.75%	88.82%
mBERT	Phrases with Keywords + O.C (Abstract)	PET 100	79.35%	69.23%
mBERT	Phrases with Keywords + O.C (Abstract)	PET 200	81.28%	74.41%
mBERT	Phrases with Keywords + O.C (Abstract)	PET 500	79.34%	81.8%
mBERT	Phrases with Keywords + O.C (Abstract)	PET 1000	81.64%	84.51%
mBERT	Phrases with Keywords (Full text)	PET 1000	80.25%	82.43%
mBERT	Phrases with Keywords + O.C (Full text)	PET 500	79.26%	81.62%
mBERT	Phrases with Keywords + O.C (Full text)	PET 1000	82.29%	83.35%
mBERT	Title	PET 500	81.06%	80.93%
mBERT	Title	PET 1000	82.44%	83.46%
mBERT	Translated Title	PET 100	79.5%	70.19%
mBERT	Translated Title	PET 500	80.32%	81.88%
mBERT	Translated Title	PET 1000	82.87%	85.27%
Bio-Link-BERT	Title	Balanced Fine-tuning	79.97%	84.18%
Bio-Link-BERT	Title	PET 50	79.11%	63.14%
Bio-Link-BERT	Title	PET 500	81.94%	76.65%
Bio-Link-BERT	Title	PET 1000	81.2%	79.8%
Bio-Link-BERT	Full text	PET 500	81.94%	82.41%
Bio-Link-BERT	Full text	PET 1000	83.25%	85.16%
Bio-Link-BERT	Translated Title	PET 50	83.77%	67.29%
Bio-Link-BERT	Translated Title	PET 100	81.31%	71.45%
Bio-Link-BERT	Translated Title	PET 200	80.02%	73.33%
Bio-Link-BERT	Translated Title	PET 1000	80.65%	83.03%
Bio-Link-BERT	Phrases with Keywords + O.C (Abstract)	PET 100	81.95%	67.77%
Bio-Link-BERT	Phrases with Keywords + O.C (Abstract)	PET 500	81.2%	77.49%
Bio-Link-BERT	Phrases with Keywords + O.C (Abstract)	PET 1000	79.35%	79.01%
Bio-Link-BERT	Phrases with Keywords (Full text)	PET 1000	80.19%	77.69%
Bio-Link-BERT	Phrases with Keywords + O.C (Full text)	PET 500	79.5%	78.92%
Bio-Link-BERT	Phrases with Keywords + O.C (Full text)	PET 1000	79.74%	82.23%
SciBERT	Title	PET 1000	80.37%	80.88%
SciBERT	Full text	PET 500	82.43%	79.82%
SciBERT	Full text	PET 1000	83.95%	84.83%
SciBERT	Phrases with Keywords + O.C (Full text)	PET 1000	80.13%	81.2%
RoBERTa-base	Abstract	Balanced Fine-tuning	79.77%	85.35%
RoBERTa-base	Abstract	PET 1000	79.66%	81.69%
RoBERTa-base	Full text	Balanced Fine-tuning	79.41%	89.72%
RoBERTa-base	Full text	PET 200	79.01%	73.06%
RoBERTa-base	Full text	PET 500	82.72%	81.03%
RoBERTa-base	Full text	PET 1000	83.18%	84.94%
RoBERTa-base	Translated Title	PET 50	83.54%	64.43%
RoBERTa-base	Translated Title	PET 500	80.25%	82.01%
RoBERTa-base	Translated Title	PET 1000	81.02%	84.2%
RoBERTa-base	Phrases with Keywords (Abstract)	PET 1000	82.78%	63.71%
RoBERTa-base	Phrases with Keywords + O.C (Abstract)	PET 1000	80.51%	81.8%
XLM-RoBERTa	Title	PET 500	80.11%	82.48%
XLM-RoBERTa	Title	PET 1000	83.12%	83.55%
XLM-RoBERTa	Abstract	PET 1000	82.34%	84.15%
XLM-RoBERTa	Full text	Balanced Fine-tuning	83.62%	90.57%
XLM-RoBERTa	Full text	PET 100	82.62%	70%
XLM-RoBERTa	Full text	PET 200	80.51%	78.04%
XLM-RoBERTa	Full text	PET 500	85.64%	86.66%
XLM-RoBERTa	Full text	PET 1000	85.73%	88.85%
XLM-RoBERTa	Translated Title	Balanced Fine-tuning	80.16%	84.24%
XLM-RoBERTa	Translated Title	PET 50	82.89%	63.84%
XLM-RoBERTa	Translated Title	PET 100	80.27%	70.52%
XLM-RoBERTa	Translated Title	PET 200	81.22%	74.28%
XLM-RoBERTa	Translated Title	PET 1000	79.91%	84.19%
XLM-RoBERTa	Phrases with Keywords (Abstract)	PET 1000	82.85%	70.57%
XLM-RoBERTa	Phrases with Keywords + O.C (Abstract)	PET 100	81.94%	72.02%
XLM-RoBERTa	Phrases with Keywords + O.C (Abstract)	PET 500	80.79%	82.09%
XLM-RoBERTa	Phrases with Keywords + O.C (Abstract)	PET 1000	81.13%	84.82%
XLM-RoBERTa	Phrases with Keywords (Full text)	PET 200	80.04%	74.48%
XLM-RoBERTa	Phrases with Keywords (Full text)	PET 500	79.59%	79.52%
XLM-RoBERTa	Phrases with Keywords (Full text)	PET 1000	79%	80.14%
XLM-RoBERTa	Phrases with Keywords + O.C (Full text)	PET 500	82.49%	80.73%
XLM-RoBERTa	Phrases with Keywords + O.C (Full text)	PET 1000	80.51%	83.46%

Table 62: Development F_2 and AUC of top performing classifiers by BERT Model

Content Source	Model	Training Method	Development F_2	Development AUC	Development F_1
Title	Bio-Link-BERT	Balanced Fine-tuning	79.97 %	84.18 %	78.45 %
Title	XLM-RoBERTa	PET 500	80.11 %	82.48 %	76.25 %
Title	SciBERT	PET 1000	80.37 %	80.88 %	75.77 %
Title	mBERT	PET 500	81.06 %	80.93 %	74.84 %
Title	Bio-Link-BERT	PET 1000	81.2 %	79.8 %	74.99 %
Title	mBERT	PET 1000	82.44 %	83.46 %	77.55 %
Title	XLM-RoBERTa	PET 1000	83.12 %	83.55 %	77.3 %
Abstract	RoBERTa-base	PET 1000	79.66 %	81.69 %	76.12 %
Abstract	mBERT	PET 1000	79.71 %	82.81 %	76.53 %
Abstract	RoBERTa-base	Balanced Fine-tuning	79.77 %	85.35 %	79.16 %
Abstract	XLM-RoBERTa	PET 1000	82.34 %	84.15 %	77.41 %
Full text	RoBERTa-base	Balanced Fine-tuning	79.41 %	89.72 %	81.81 %
Full text	BERT-base	PET 500	81.35 %	79.66 %	74.28 %
Full text	Bio-Link-BERT	PET 500	81.94 %	82.41 %	74.62 %
Full text	SciBERT	PET 500	82.43 %	79.82 %	72.87 %
Full text	RoBERTa-base	PET 500	82.72 %	81.03 %	74.7 %
Full text	mBERT	PET 1000	82.75 %	88.82 %	81.52 %
Full text	BERT-base	PET 1000	83.1 %	85.09 %	79.13 %
Full text	RoBERTa-base	PET 1000	83.18 %	84.94 %	77.87 %
Full text	Bio-Link-BERT	PET 1000	83.25 %	85.16 %	77.82 %
Full text	mBERT	PET 200	83.29 %	80.29 %	74.04 %
Full text	XLM-RoBERTa	Balanced Fine-tuning	83.62 %	90.57 %	83.95 %
Full text	SciBERT	PET 1000	83.95 %	84.83 %	77.98 %
Full text	mBERT	PET 500	84.72 %	86.75 %	79.78 %
Full text	XLM-RoBERTa	PET 500	85.64 %	86.66 %	78.34 %
Full text	XLM-RoBERTa	PET 1000	85.73 %	88.85 %	81.6 %
Translated Title	XLM-RoBERTa	PET 1000	79.91 %	84.19 %	77.17 %
Translated Title	XLM-RoBERTa	Balanced Fine-tuning	80.16 %	84.24 %	79.23 %
Translated Title	RoBERTa-base	PET 500	80.25 %	82.01 %	75.93 %
Translated Title	mBERT	PET 500	80.32 %	81.88 %	76.1 %
Translated Title	Bio-Link-BERT	PET 1000	80.65 %	83.03 %	76.67 %
Translated Title	RoBERTa-base	PET 1000	81.02 %	84.2 %	78.47 %
Translated Title	mBERT	PET 1000	82.87 %	85.27 %	78.81 %
Phrases with Keywords + O.C (Abstract)	mBERT	PET 500	79.34 %	81.8 %	75.42 %
Phrases with Keywords + O.C (Abstract)	Bio-Link-BERT	PET 1000	79.35 %	79.01 %	74.14 %
Phrases with Keywords + O.C (Abstract)	BERT-base	PET 1000	80.16 %	80.2 %	75.63 %
Phrases with Keywords + O.C (Abstract)	RoBERTa-base	PET 1000	80.51 %	81.8 %	76.41 %
Phrases with Keywords + O.C (Abstract)	XLM-RoBERTa	PET 500	80.79 %	82.09 %	76.15 %
Phrases with Keywords + O.C (Abstract)	XLM-RoBERTa	PET 1000	81.13 %	84.82 %	78.3 %
Phrases with Keywords + O.C (Abstract)	mBERT	PET 1000	81.64 %	84.51 %	78.53 %
Phrases with Keywords (Full text)	XLM-RoBERTa	PET 1000	79 %	80.14 %	74.17 %
Phrases with Keywords (Full text)	XLM-RoBERTa	PET 500	79.59 %	79.52 %	73.32 %
Phrases with Keywords (Full text)	mBERT	PET 1000	80.25 %	82.43 %	76.63 %
Phrases with Keywords + O.C (Full text)	mBERT	PET 500	79.26 %	81.62 %	75.84 %
Phrases with Keywords + O.C (Full text)	Bio-Link-BERT	PET 1000	79.74 %	82.23 %	75.58 %
Phrases with Keywords + O.C (Full text)	SciBERT	PET 1000	80.13 %	81.2 %	75.16 %
Phrases with Keywords + O.C (Full text)	XLM-RoBERTa	PET 1000	80.51 %	83.46 %	77.22 %
Phrases with Keywords + O.C (Full text)	mBERT	PET 1000	82.29 %	83.35 %	78.19 %
Phrases with Keywords + O.C (Full text)	XLM-RoBERTa	PET 500	82.49 %	80.73 %	74.44 %

Table 63: Development F_2, AUC, and F_1 of top performing classifiers by Content Source

Content Source	Model	Training Method	Develop-ment F_2	Develop-ment AUC	Develop-ment F_1	Develop-ment Accuracy	Develop-ment Precision	Develop-ment Recall	Test F_2	Test AUC	Test Precision	Test Recall	Test F_1	Test Accuracy
Title	mBERT	PET 1000	82.44 %	83.46 %	77.55 %	75.09 %	70.58 %	86.06 %	60.56 %	82.5 %	27.83 %	85.78 %	42.03 %	66.73 %
Title	XLM-RoBERTa	PET 1000	83.12 %	83.55 %	77.3 %	74.3 %	69.22 %	87.52 %	58.4 %	80.8 %	26.13 %	84.48 %	39.92 %	64.24 %
Abstract	XLM-RoBERTa	PET 1000	82.34 %	84.15 %	77.41 %	74.91 %	70.4 %	85.98 %	60.57 %	84.6 %	28.74 %	83.76 %	42.79 %	69.14 %
Full text	mBERT	PET 1000	82.75 %	88.82 %	81.52 %	81.06 %	79.56 %	83.58 %	66.97 %	89.61 %	50.21 %	79.24 %	35.44 %	86.13 %
Full text	BERT-base	PET 1000	83.1 %	85.09 %	79.13 %	77.33 %	73.3 %	85.98 %	59.65 %	84.84 %	40.58 %	69.08 %	26.47 %	86.86 %
Full text	RoBERTa-base	PET 1000	83.18 %	84.94 %	77.87 %	75.24 %	70.39 %	87.13 %	57.83 %	85.96 %	38 %	64.82 %	24.18 %	88.69 %
Full text	Bio-Link-BERT	PET 1000	83.25 %	85.16 %	77.82 %	75.11 %	70.19 %	87.31 %	59.15 %	85.66 %	39.23 %	66.33 %	25.13 %	89.42 %
Full text	XLM-RoBERTa	Balanced	83.62 %	90.57 %	83.95 %	84.06 %	84.52 %	83.39 %	63.72 %	87.22 %	51.1 %	82.24 %	38.42 %	76.28 %
Full text	SciBERT	Fine-tuning PET 1000	83.95 %	84.83 %	77.98 %	75.02 %	69.72 %	88.46 %	56.23 %	85.13 %	36.49 %	62.78 %	23.02 %	87.96 %
Full text	mBERT	PET 500	84.72 %	86.75 %	79.78 %	77.6 %	72.7 %	88.38 %	61.25 %	87.61 %	41.6 %	69.48 %	27.1 %	89.42 %
Full text	XLM-RoBERTa	PET 500	85.64 %	86.66 %	78.34 %	74.76 %	68.6 %	91.3 %	56.84 %	87.59 %	36.31 %	61.09 %	22.67 %	91.24 %
Full text	XLM-RoBERTa	PET 1000	85.73 %	88.85 %	81.6 %	79.99 %	75.53 %	88.73 %	63.79 %	89.66 %	44.34 %	72.49 %	29.4 %	90.15 %
Translated Title	RoBERTa-base	PET 1000	0.8102069614	84.2 %	78.47 %	77.28 %	74.57 %	82.81 %	59.18 %	83.72 %	31.41 %	75.98 %	44.44 %	73.86 %
Translated Title	mBERT	PET 1000	82.87 %	85.27 %	78.81 %	76.92 %	72.86 %	85.82 %	63.81 %	84.2 %	31.86 %	85.15 %	46.37 %	0.7289663462
Phrases with Keywords + O.C (Abstract)	XLM-RoBERTa	PET 1000	81.13 %	84.82 %	0.7829977629	76.96 %	74 %	83.14 %	58.48 %	83.78 %	29.18 %	78.07 %	42.48 %	71.38 %
Phrases with Keywords + O.C (Abstract)	mBERT	PET 1000	81.64 %	84.51 %	78.53 %	77.08 %	73.85 %	83.85 %	58.69 %	83.37 %	28.73 %	79.39 %	42.19 %	70.55 %
Phrases with Keywords + O.C (Full text)	mBERT	PET 1000	82.29 %	83.35 %	78.19 %	76.22 %	72.2 %	85.26 %	56.41 %	82 %	25.31 %	81.42 %	38.61 %	68.65 %

Table 64: Development and Test metrics for Best Performing classifiers