Rapport de Stage

BiTEM group - HES-SO Genève

Léandre Catogni

Bachelor en Mathématiques, Informatique et Sciences Numériques

Université de Genève
Mai 2025

Remerciements

Je souhaite tout d'abord remercier Julien Knafou, grâce à qui j'ai pu trouver ce stage, pour l'aide qu'il m'a accordé dans mon apprentissage.
Je suis également particulièrement reconnaissant pour la confiance que m'a accordé Patrick Ruch, directeur du stage. TODO : Completer

Contents

1 BiTEM group presentation 4
1.1 Biodiversity Monitoring via Question Answering 4

2 Introduction 6
2.1 Context 6
2.2 Motivations 6
2.3 Objectives 6

3 State of the art 8
3.1 Biodiversity Monitoring via Question Answering 8

4 State of the art 10
4.1 Tokenization 10
4.1.1 Byte Pair Encoding 12
4.1.2 WordPiece 13
4.2 Word Embeddings and Word2Vec 15
4.2.1 Term Frequency–Inverse Document Frequency (TF-IDF) 15
4.2.2 Goal 16
4.2.3 Word2Vec 16
4.2.4 Negative Sampling 17
4.2.5 GloVe 17
4.3 Neural Networks 18
4.4 Deep Contextual Models 19
4.4.1 Recurrent Neural Networks (RNNs) 19
4.4.2 Long Short-Term Memory (LSTM) 19
4.4.3 Attention for Neural Networks 20
4.4.4 Seq2seq 20
4.4.5 Transformers 21
4.4.6 BERT 21
4.4.7 BERT Variations 22
4.4.8 Heads 23
4.4.9 Fine-tuning 23
4.4.10 Optimization Algorithms 23
4.5 Evaluation 24
4.5.1 Data splitting 24
4.5.2 K-Fold Cross Validation 25
4.5.3 Metrics 26

4.5.4 Hyperparameter Optimization 29
4.5.5 Statistical Test 30

5 Data 33
5.1 Description 33
5.2 Cross-Validation 33
5.3 Pre-Processing 34
5.4 Contamination 34
5.5 Repartition 35

6 Méthodes 37
6.1 Overview 37
6.2 Gestion des donnés 38
6.2.1 Data Splitting 38
6.2.2 Handling Imbalanced Data 38
6.3 Architecture 39

7 Résultats 40

8 Analyse et discussion 43
8.0.1 Ablation 43
8.0.2 Contamination 44
8.1 Limitations 45

9 Conclusion 46

10 Bibliography 47

11 Annexes 48
11.0.1 Workflow 49

Chapter 1

BiTeM group presentation

The BiTeM group (Bibliomics and Text Mining Group), in which I conducted this internship, is part of the Information Science Department of the HES-SO/HEG Geneva. This research group's open-science projects focus primarily on clinical and biological data analysis through advanced text mining and natural language processing techniques.

The group is also affiliated with the text mining group of the Swiss Institute of Bioinformatics (SIB), which currently works on the Swiss Institute of Bioinformatics Literature Services (SIBiLS), a platform providing personalized Information Retrieval in the biological literature. SIBiLS includes daily updated and semantically enriched data from four collections (MEDLINE, PubMedCentral (PMC), Plazi treatments, and PMC supplementary files), a customizable search tool, a question answering service, customized literature triage for cell lines characterization, and a freely accessible API allowing researchers worldwide to integrate these services into their workflows.

BiTeM's lab provides a robust and scalable infrastructure based on cloud and federated architectures, enabling advanced text mining, semantic search, and AI-powered metadata curation across large biomedical and biodiversity datasets. As a BiTeM intern, I was assigned tasks within two distinct but complementary projects that will be detailed below.

1.1 Biodiversity Monitoring via Question Answering

The Biodiversity Monitoring via Question Answering (BioMoQA) project is an initiative gathering researchers from different institutions across Europe. Its primary objective is to enhance SIBiLS by improving access to biodiversity-related scientific literature through AI-powered analytical services, such as question-answering systems and SPARQL endpoints. These tools aim to assist researchers in addressing critical biodiversity questions related to climate change, habitat loss, and invasive species.

The main application motivating this project is a collaboration with the University of Neuchâtel to monitor biodiversity on island ecosystems. Researchers at this institution require efficient tools to identify relevant publications among the vast corpus of environmental and biological literature.

In this context, my primary task was to build a binary classifier model that labels biodiversity scientific journal abstracts based on their relevance to island ecosystems, as defined by domain experts at the University of Neuchâtel. The classifier is designed to function as a ranking system in practice, prioritizing the identification and high ranking of relevant documents over perfect classification accuracy. This approach recognizes that in information retrieval applications, it is more important to ensure relevant documents appear at the top of search results than to perfectly classify every document.

Chapter 2

Introduction

2.1 Context

During this internship, conducted from March to July 2025, I collaborated with the BiTeM group on two complementary projects within the broader scope of AI-powered scientific literature analysis. The primary focus of this report details my work on the Biodiversity Monitoring via Question Answering (BioMoQA) project, where I developed a transformer-based classifier for ranking biodiversity-related scientific abstracts. Additionally, I contributed to the development of a multi-label classifier for the IPBES (Intergovernmental Panel on Biodiversity and Ecosystem Services) dataset. While the IPBES project results are not yet available due to ongoing long computation times, both projects share the common goal of enhancing researchers' ability to efficiently navigate the ever-growing corpus of scientific literature.

Over the past decade, the exponential growth of scientific publications has created significant challenges for researchers seeking to stay current with developments in their fields. The volume of scientific literature published annually has increased by over 8% in the life sciences alone, reaching more than two million articles per year as of 2024 [**nih:stats2024**]. Advances in natural language processing (NLP) have opened new avenues for distilling large volumes of textual data into concise summaries and extracting high-quality, relevant information [**citeulike:123456**]. Modern transformer-based models enable state-of-the-art performance in tasks including summarization, information retrieval, and question answering.

The central challenge addressed in this work is the development of automated systems that can effectively rank scientific documents by relevance rather than merely classifying them. This ranking-focused approach is crucial for practical information retrieval applications, where the goal is to present users with the most relevant documents at the top of their search results.

2.2 Motivations

Researchers studying specialized domains, such as island ecosystems in biodiversity research, often face difficulty locating precise, high-quality data among a noisy corpus. This challenge is particularly acute in interdisciplinary fields where relevant information may be scattered across multiple domains and publication venues. The traditional approach of manual literature review becomes increasingly impractical as the volume of potentially relevant publications grows exponentially.

The need for automated literature triage is especially critical in biodiversity research, where timely access to relevant findings can inform conservation decisions and policy making. Island ecosystems, being particularly vulnerable to environmental changes and human impacts, require specialized monitoring approaches supported by comprehensive literature reviews.

2.3 Objectives

The primary objective of this project, conducted within the Biodiversity Monitoring via Question Answering initiative, is to develop an automated transformer-powered classifier specifically designed for ranking scientific abstracts based on their relevance to island biodiversity research. Unlike traditional classification systems that focus solely on accuracy, our approach prioritizes ranking performance to ensure that the most relevant documents are positioned at the top of search results.

More specifically, the project aims to:

- Develop a robust binary classification system capable of distinguishing between abstracts relevant and irrelevant to island ecosystem research
- Optimize the classifier for ranking performance rather than strict classification accuracy, recognizing that in practical information retrieval scenarios, the relative ordering of documents is more important than perfect binary classification
- Evaluate multiple state-of-the-art transformer architectures, with particular attention to domain-specific models pre-trained on biomedical literature
- Address the inherent class imbalance present in information retrieval datasets through appropriate loss functions and evaluation metrics
- Integrate the resulting classifier into the SIBiLS platform to assist researchers worldwide in efficiently identifying relevant biodiversity literature

This work contributes to the broader goals of the BioMoQA project (detailed in Chapter 1) by providing a foundational component for automated literature triage. The ultimate vision is to enable researchers to pose complex questions about island biodiversity and receive ranked lists of relevant publications, thereby accelerating scientific discovery and supporting evidence-based conservation efforts.

The practical impact of this work extends beyond academic research, as the resulting system will be deployed within the SIBiLS platform, making it freely accessible to the global research community and actively contributing to making science more open and accessible.

Chapter 3

State of the art

Before exploring the model's implementation details, let's establish a brief state of the art for Natural Language Processing (NLP).

3.1 Tokenization

Since Machine Learning models need proper numbers in order to work, we need to answer the following crucial problematic : How can we turn an infinite variety of raw text into a finite set of numbers ?
The first part of the answer to the problematic above is tokenization that is, breaking raw text into discrete units called ”tokens” (can be seen as subwords).
At first glance we could naively think that we could build a vocabulary (the set of tokens) by just considering each word of the corpus (the text on which we train a tokenizer) as a token but this would kill the purpose of tokenization.

What makes tokenization meaningful and what we need is :

- Representing rare or unseen words by splitting them into known sub-tokens, hence giving more flexibility and semantic when we encounter new words.

- Narrowing the huge number of exisiting words to a smaller tokens vocabulary, Which lowers compute time and ressources when training a model (we need less time to find common semantics between words that shares sub-tokens).

This is what makes tokens a robust and generalized representation of text.

Let's now clarify the overall process of building a tokenizer, described in the following figure.

The initial text processing step is normalization, which involves cleanup tasks such as lowercasing, trimming whitespace, and potentially removing accents.
Following normalization, pre-tokenization is essential, as raw text is not directly suitable for tokenizer training. This step splits the text based on whitespace and punctuation

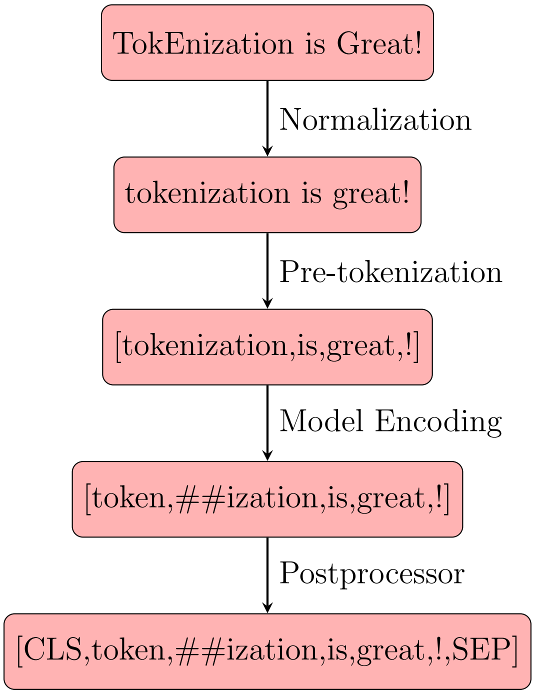

Figure 3.1: Example of a tokenization process

rules, producing preliminary sub-tokens (words and punctuation).

Once we have these pre-tokenized elements, and assuming we proceed to train the tokenizer in order to build the tokens vocabulary, we can encode our input sequence using the tokenizer's learned vocabulary, which maps text to subword units and their corresponding token IDs.

Now that we have seen how a tokenizer works, let's explore some common types.

Tokenizers Overview			
Phase	BPE	WordPiece	Unigram
Training	Builds merging rules from individual characters	Builds vocabulary from individual characters by learning merging rules	Starts with a large vocabulary and prunes it down by removing tokens that reduce the likelihood the least
Training step	Iteratively merges the most occurring pair of tokens until a vocabulary size is reached	Iteratively merges the pair with the best score (cf. 4.1.2)	Uses an Expectation-Maximization algorithm to optimize token probabilities and reduce vocabulary
Learns	A new vocabulary and merge rules	A vocabulary only	A vocabulary with a probability distribution over tokens
Encoding	Apply learned merges to the sequence of character of a word	Iteratively look for the longest subword in the trained vocabulary until we tokenized the whole word	Tries multiple segmentations and chooses the most probable according to the learned token probabilities

Below, we focus only on BPE and WordPiece, as these are the only two methods used in the project.

3.1.1 Byte Pair Encoding

Byte-Pair Encoding (BPE) was first introduced in 1994 by Philip Gage as a text compression algorithm. It was then modified in [**sennrich2015neural**] as a tokenizer for Large Language Models (LLMs).
It builds a vocabulary by iteratively merging the most common pair of characters until the desired vocabulary size is reached, as shown in the algorithm below.

Algorithm 1 BPE Tokenizer Training

```
Require: Corpus of strings $\mathcal{C}$, desired vocabulary size $V_{\text{target}}$
Ensure: BPE merge operations $M = [\,m_1, m_2, \ldots\,]$
Initialize vocabulary $V \leftarrow$ all unique characters in $\mathcal{C}$
Split each word in $\mathcal{C}$ into a sequence of characters plus an end-of-word marker </w>
Compute pair frequencies: for each adjacent symbol pair $(x, y)$ in $\mathcal{C}$, count occurrences $f(x, y)$
while $|V| < V_{\text{target}}$ do
    Find most frequent pair $(a, b) = \arg\max_{(x,y)} f(x, y)$
    if $f(a, b) = 0$ then
        break
    end if
    Append merge $m \leftarrow (a, b)$ to $M$ and add new symbol $ab$ to $V$
    For each word in $\mathcal{C}$, replace all occurrences of $(a, b)$ by $ab$
    Recompute frequencies $f(x, y)$ for all pairs
end while
return $M$
```

Once we have a vocabulary of tokens (learned merges), we can use them to encode a given text sequence into tokens :

Algorithm 2 BPE Tokenizer Encoding

```
Require: Word $w$, learned merges $M = [\,m_1, m_2, \ldots, m_K\,]$
Ensure: Sequence of subword tokens
Initialize token sequence $T \leftarrow$ characters of $w$ with </w> appended
for $i = 1$ to $K$ do
    Let $(a, b) = m_i$
    while there exists adjacent tokens $T_j = a$, $T_{j+1} = b$ in $T$:
        Merge them into a single token $ab$
end for
return $T$
```

3.1.2 WordPiece

Wordpiece, as it is used in recent models, was introduced in 2016 by Google ([**wu2016google**]). In 2018 Google used it again to pre-train BERT model [**devlin2019bert**], followed by other models such as DistilBERT, MobileBERT, Funnel Transformers, and MPNET.

WordPiece training is very similar to BPE, but differs in three main ways :

- When splitting the word into individual characters, it adds the prefix ”##” to every character that follows another one, inside the word.

- Instead of merging the most common pair of tokens it merges the pair of tokens that maximize the following score : Let $a, b \in \mathcal{V}$ be 2 tokens of the vocabulary,

$$score(a, b) = \frac{|\{v \in \mathcal{V} \mid v = ab\}|}{|\{v \in \mathcal{V} \mid v = a\}| \times |\{v \in \mathcal{V} \mid v = b\}|}$$

This allows the tokenizer to focus in merging pairs for which the individual parts are less frequent in the vocabulary, hence prioritizing more ”atomic” tokens.

- In the algorithm, unlike BPE which builds merging rules, WordPiece builds a vocabulary (instead of merging rules).

Below is the detailed algorithm of the training process :

Algorithm 3 WordPiece Vocabulary Training

```
Require: Corpus of words $\mathcal{C}$, target vocabulary size $V_{\text{target}}$, minimum substring frequency $\tau$
Ensure: Learned vocabulary $V$
Initialize $V \leftarrow$ all unique characters in $\mathcal{C}$ plus special token [UNK]
Represent each word in $\mathcal{C}$ as a sequence of characters
while $|V| < V_{\text{target}}$ do
    Compute frequency $f(s)$ of every substring $s$ of length $\geq 2$ occurring in $\mathcal{C}$
    For each substring $s$ with $f(s) \geq \tau$, compute score:
        $\text{score}(s) = f(s)\,\big(|s| - 1\big)$
    Let $s^* = \arg\max_s \text{score}(s)$
    if $\text{score}(s^*) \leq 0$ then
        break
    end if
    Add token $s^*$ to $V$
    In all words in $\mathcal{C}$, replace every occurrence of $s^*$ by the single symbol $s^*$
end while
return $V$
```

As for encoding a given word, WordPiece splits it according to the longest subword in the vocabulary it finds.

Algorithm 4 WordPiece Encoding (Greedy Longest-Match)

```
Require: Input word $w$, vocabulary $V$, unknown token [UNK], suffix marker ##
Ensure: Sequence of wordpiece tokens $T$
Set $T \leftarrow [\,]$, position $i \leftarrow 1$
while $i \leq |w|$ do
    Let $j \leftarrow |w|$
    while $i \leq |w|$ do
        if substring $s = w[i\colon j] \in V$ (or ##$s$ for $i > 1$) then
            Append $s$ (with ## if $i > 1$) to $T$
            Set $i \leftarrow j$
            break
        else
            $j \leftarrow j - 1$
        end if
    end while
    if no valid $s$ found then
        Append [UNK] to $T$
        break
    end if
end while
return $T$
```

3.2 Word Embeddings and Word2Vec

After a text has been tokenized, we still need to translate these tokens into a format that a machine learning model can understand: numbers. This translation step is known as *word embedding*.

3.2.1 Term Frequency–Inverse Document Frequency (TF-IDF)

One of the earliest and most straightforward approaches to transforming text into numerical values is the TF-IDF method, which stands for *Term Frequency–Inverse Document Frequency*. The idea behind TF-IDF is to assign a weight to each word based on how important it is to a particular document, while also accounting for how common that word is across the entire corpus.

Mathematically, it is defined as:

$$\text{TF-IDF}(w, d, D) = \text{tf}(w, d) \cdot \log\left(\frac{|D|}{|\{d' \in D : w \in d'\}|}\right)$$

Here, $\text{tf}(w, d)$ represents the frequency of word w in document d, and the second part—the inverse document frequency—penalizes words that appear in many documents, reducing the weight of overly common terms.

While TF-IDF is still useful in many applications, it has an important limitation: it does not capture the meaning or context of words. For example, it treats the words “king” and “queen” as unrelated, even though they are semantically close. This is where word embeddings come into play.

3.2.2 Goal

The whole point of modern word embeddings is to capture a word's meaning in a vector of numbers. The guiding principle is simple: a word is defined by the company it keeps. If two words repeatedly appear in similar contexts, their vector representations should be close. This allows a model that learns something about ”doctors” to intuitively apply that knowledge to ”nurses” or ”physicians,” simply because their vectors occupy a similar neighborhood in the embedding space.

So how do we create these vectors? We don't define the relationships manually. Instead, we train a simple neural network to learn them. We start by assigning every word w in our vocabulary a random vector $\mathbf{v}_w \in \mathbb{R}^d$, where d is the embedding dimension (e.g., 300). Then, we give the network a task: read through a massive text corpus and, for each word, either predict it from its neighbors or predict the neighbors from the word.

As the network learns, it constantly adjusts the word vectors to get better at its prediction task. This process naturally forces the vectors of words used in similar contexts to move closer together. Eventually, the randomly initialized vectors converge into a rich, meaningful representation of the vocabulary. After training, words like ”apple” and ”orange” will have vectors that are very close, i.e., $\|\mathbf{v}_{\text{apple}} - \mathbf{v}_{\text{orange}}\| \approx 0$, while being far from the vector for ”car”. These final, learned vectors are the word embeddings.

This basic approach of learning from a word's immediate surroundings is powerful, but it only captures a very local sense of context. This limitation helped motivate the development of more robust methods like **Word2Vec**, which use smarter training strategies to create even richer word representations.

3.2.3 Word2Vec

The Word2Vec framework isn't a single algorithm, but rather a package of models and training techniques that produce high-quality word embeddings. Its two most famous architectures are the Continuous Bag of Words (CBOW) and Skip-Gram. They are essentially opposites:

- **CBOW** is like a fill-in-the-blank task. It takes a set of context words (e.g., ”the cat sat on the”) and tries to predict the missing target word (”mat”). The ”bag of words” part means the model treats the context as an unordered collection of words.

- **Skip-Gram** flips this around. It takes a single word (like "mat") and tries to predict its surrounding context words (like "the", "sat", "on").

While both achieve a similar goal, they have different strengths. CBOW is faster to train and tends to be slightly better for frequent words. Skip-Gram, on the other hand, is slower but does a better job of learning representations for rare words and performs well even with smaller amounts of data.

3.2.4 Negative Sampling

A major challenge in training these models is the sheer size of the vocabulary. When the Skip-Gram model tries to predict a context word, it technically has to calculate a probability for every single word in the entire vocabulary (which could be tens of thousands of words) using a softmax function. This is incredibly expensive and slow.

This is where **Negative Sampling** comes in as a clever efficiency hack. Instead of framing the task as a massive multi-class classification problem, it reframes it as a much simpler binary classification problem. For a given pair of words '(input, output)', such as '("mat", "sat")', the model is trained to predict whether they are a true context pair (output 1). At the same time, we generate a few "negative" samples by pairing the input word with random words from the vocabulary, like '("mat", "banana")'. The model is trained to identify these as fake pairs (output 0).

This way, at each training step, the model only has to update the weights for the one true "positive" example and a small number of manufactured "negative" examples, rather than all the weights for the entire vocabulary. This dramatically speeds up training without a significant loss in the quality of the final embeddings.

Note: BERT can be understood as a contextualized word embedding model, which we will discuss in detail in the following sections.

3.2.5 GloVe

Another major development in word embeddings came from Stanford with **GloVe**, which stands for Global Vectors for Word Representation. The creators of GloVe argued that while methods like Word2Vec were great at capturing local context (the words immediately surrounding a target word), they didn't make good use of the overall statistical information of the entire corpus. On the other hand, older methods that did use global stats (like Latent Semantic Analysis) weren't as good at the analogy tasks that made Word2Vec famous (e.g., "king - man + woman = queen").

GloVe was designed to get the best of both worlds. Its core idea is to learn word vectors by looking at a global **word-word co-occurrence matrix**. This is basically a giant table that counts how frequently each word appears in the context of every other word across all the text data. The model is trained so that the dot product of any two word vectors equals the logarithm of their probability of co-occurring.

The key insight is that the ratio of co-occurrence probabilities can reveal interesting relationships between words. By training directly on these global statistics, GloVe produces a vector space that often excels at capturing subtle semantic relationships. It represents a different and powerful approach to the same fundamental goal: turning words into meaningful numbers.

3.3 Neural Networks

Neural Networks are a class of machine learning models inspired by the structure and functioning of the human brain. They consist of layers of interconnected nodes (neurons), each performing simple computations that, when composed together, can model complex, non-linear relationships. Here we define an *Artificial Neural Network (ANN)* in a general form.

Let $x_i = (x_{i,1}, \dots, x_{i,m}) \in \mathbb{R}^m$ be the i-th input sample from a dataset of size n. An ANN is a function $p_\alpha : \mathbb{R}^m \to \mathbb{R}^d$, parameterized by weights and biases denoted collectively as $\alpha = \{(W_k, b_k)\}_{k=1}^{I}$, and defined as a composition of I layers:

$$p_\alpha(x_i) = (p_{\alpha,1}(x_i), \dots, p_{\alpha,d}(x_i)) = S_I \circ S_{I-1} \circ \cdots \circ S_1(x_i)$$

Each layer $S_k : \mathbb{R}^{d_{k-1}} \to \mathbb{R}^{d_k}$ transforms the input vector v (with $d_0 = m$, the input dimension) as follows:

$$S_k(v) = \begin{cases} \sigma(W_k v + b_k) & \text{for } k \in \{1, \dots, I-1\} \\ g(W_k v + b_k) & \text{for } k = I \end{cases}$$

Here:

- $W_k \in \mathbb{R}^{d_k \times d_{k-1}}$ and $b_k \in \mathbb{R}^{d_k}$ are the weight matrix and bias vector for the k-th layer,
- $\sigma(\cdot)$ is a non-linear activation function, commonly ReLU, sigmoid, or tanh,
- $g(\cdot)$ is the output activation function, e.g., softmax for classification or the identity function for regression.

This architecture allows the network to learn hierarchical representations of the input data through successive transformations, enabling powerful approximations of complex functions.

3.4 Deep Contextual Models

While learned word embeddings like Word2Vec marked a significant leap forward, they have a fundamental limitation: they are static. The vector for the word "bank" is the same in "river bank" as it is in "investment bank." To truly understand language, a model needs to interpret words based on the context in which they appear. This requires models that can process sequences of text and produce *contextualized* representations. This section traces the evolution of these models, from early recurrent networks to the modern Transformer architecture.

3.4.1 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) were the first architecture designed specifically to handle sequential data. Unlike standard feed-forward networks, which process inputs independently, an RNN maintains a 'memory' of past information. It achieves this by using a recurrent connection where the output from one step is fed back as an input to the next.

At each timestep t, the network's hidden state h_t is a function of the current input x_t and the previous hidden state h_{t-1}. This can be expressed as:

$$h_t = f(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

where W_{hh} and W_{xh} are weight matrices, b_h is a bias term, and f is a non-linear activation function like tanh. The hidden state h_t acts as a compressed representation of the entire sequence up to that point. However, in practice, standard RNNs suffer from the **vanishing and exploding gradient problem**, which makes it extremely difficult for them to capture long-range dependencies in the text. As the error signal is propagated back through many timesteps, it either diminishes to zero or grows uncontrollably.

3.4.2 Long Short-Term Memory (LSTM)

To address the limitations of simple RNNs, the Long Short-Term Memory (LSTM) network was introduced (Hochreiter & Schmidhuber, 1997). LSTMs are a special kind of RNN that are explicitly designed to avoid the long-term dependency problem. They introduce a more complex recurrent unit containing a **cell state** C_t and three regulatory **gates**: the forget gate, input gate, and output gate.

The cell state acts as a conveyor belt for information, allowing it to flow down the sequence largely unchanged. The gates, which are composed of a sigmoid layer and a pointwise multiplication, regulate what information is added to or removed from this cell state.

- **Forget Gate** (f_t): Decides what information to discard from the previous cell state C_{t-1}.

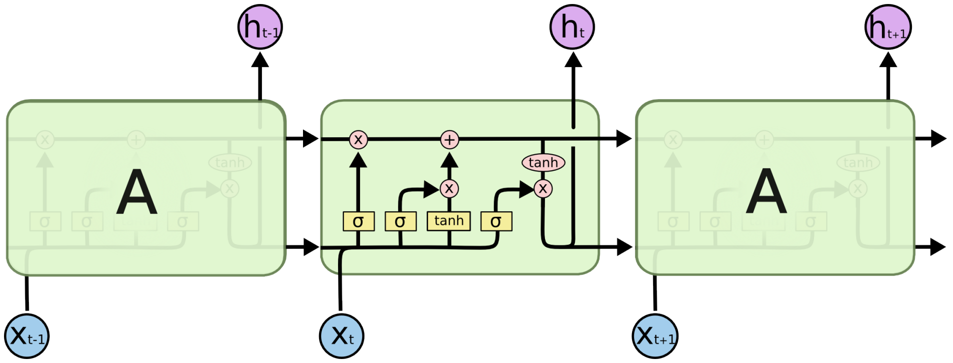

Figure 3.2: The structure of an LSTM cell, showing the cell state and three gates. Source: Christopher Olah's blog.

- **Input Gate** (i_t): Determines which new information to store in the current cell state C_t.
- **Output Gate** (o_t): Controls what part of the cell state is used to produce the output hidden state h_t.

These mechanisms allow LSTMs to selectively remember or forget information over long sequences, making them far more effective than vanilla RNNs for most NLP tasks.

3.4.3 Attention for Neural Networks

A key weakness of the traditional encoder-decoder architecture based on RNNs or LSTMs is the information bottleneck. The encoder must compress the entire meaning of a source sequence into a single fixed-length vector, which is then passed to the decoder. This is particularly challenging for long sequences.

The attention mechanism was introduced to solve this problem (Bahdanau et al., 2014). Instead of relying on a single context vector, attention allows the decoder to ”look back” at the hidden states of the entire input sequence at each step of the decoding process. It learns a set of attention weights, α, which determine how much focus to place on each input word when generating the next output word. This allows the model to draw connections between specific words in the input and output, dramatically improving performance on tasks like machine translation.

3.4.4 Seq2seq

Sequence-to-sequence (seq2seq) models provide a general framework for mapping an input sequence to an output sequence, where the lengths may differ. First proposed for machine translation (Sutskever et al., 2014), the architecture consists of two main components:

1. **An Encoder**: An RNN (or LSTM) that processes the input sequence token by token and encodes it into a context vector (its final hidden state).

2. **A Decoder**: Another RNN that is initialized with the encoder's context vector and generates the output sequence token by token.

The integration of the attention mechanism into this framework was a critical breakthrough, creating the model architecture that dominated NLP for several years.

3.4.5 Transformers

In their landmark paper "Attention Is All You Need," Vaswani et al. (2017) proposed the **Transformer**, an architecture that dispensed with recurrence entirely and relied solely on attention mechanisms. The core idea is that to understand a word's context, you don't need to process the sentence sequentially; you just need to know which other words are important to it. The Transformer achieves this with a mechanism called **self-attention**, which allows the model to weigh the importance of all other words in the input sequence for a given word.

The architecture is built from stacks of encoders and decoders. Since there is no recurrence, the model has no inherent sense of word order. This is solved by injecting **positional encodings** into the input embeddings, which provide information about the relative or absolute position of tokens in the sequence. By parallelizing the attention mechanism into multiple "heads" (**Multi-Head Attention**), the model can learn different types of relationships simultaneously. This parallelizable, non-recurrent design made the Transformer far more efficient to train on modern hardware than LSTMs, setting the stage for pre-training on truly massive datasets.

3.4.6 BERT

BERT, or **Bidirectional Encoder Representations from Transformers**, represents a paradigm shift in how deep learning is applied to NLP (Devlin et al., 2018). Instead of training a model from scratch for a specific task, BERT is a language representation model that is pre-trained on a massive amount of unlabeled text, and can then be quickly fine-tuned for various downstream tasks.

Architecturally, BERT is composed of a stack of Transformer **encoders**. Unlike traditional language models that process text from left-to-right or right-to-left, BERT is deeply bidirectional. It achieves this through two novel pre-training tasks:

1. **Masked Language Model (MLM)**: In this task, 15% of the tokens in the input text are randomly masked. The model's objective is to predict the original identity of these masked tokens based on the full, unmasked context from both the left and the right. This forces the model to learn a rich, bidirectional representation of language.

2. **Next Sentence Prediction (NSP)**: The model receives pairs of sentences and must predict whether the second sentence is the actual sentence that follows the first in the original text. This teaches the model to understand sentence-level relationships.

After pre-training on a huge corpus (like Wikipedia and the BookCorpus), the result is a powerful model that produces contextualized embeddings. For any given input text, BERT outputs a vector for each token that encapsulates its meaning within that specific sentence. This pre-trained model can then be adapted for a wide range of tasks with minimal architectural changes.

3.4.7 BERT Variations

The success of BERT sparked a wave of research into pre-trained language models. Many variations have been proposed, often tweaking the pre-training objectives, architecture, or training data. The key innovation of RoBERTa (Liu et al., 2019) demonstrated that the original BERT was significantly undertrained. By training for longer, on more data, with larger batches, removing the NSP objective (which they found to be of limited benefit), and using a dynamic masking strategy, RoBERTa was able to substantially outperform BERT on many benchmarks.

It was soon discovered that pre-training on a corpus tailored to a specific domain could yield significant performance gains on tasks within that domain. Models like BioBERT and BioMedBERT are trained on vast collections of biomedical literature (e.g., PubMed abstracts). This allows them to learn the specific syntax, vocabulary, and semantic relationships prevalent in medical and biological texts.

Table **??** provides a comprehensive comparison of the transformer models used in this work, highlighting their key characteristics and domain-specific adaptations.

Table 3.1: Comparison of transformer models used in this study.

Model	**Description**	**Parameters**	**Context**	**Pre-training Data**	**Domain**
BERT-base	Original bidirectional transformer	110M	512	BookCorpus + Wikipedia	General
RoBERTa-base	Optimized BERT training	125M	512	BERT data + additional text	General
BioBERT-v1.1	BERT fine-tuned on biomedical text	110M	512	BERT + PubMed + PMC	Biomedical
BiomedBERT-abstract	BERT pre-trained on PubMed abstracts	110M	512	PubMed abstracts only	Biomedical
BiomedBERT-fulltext	BERT pre-trained on full-text articles	110M	512	PubMed + PMC full-text	Biomedical

The domain-specific models (BioBERT and BiomedBERT variants) are particularly relevant for our biodiversity classification task, as they have been exposed to scientific terminology and writing styles characteristic of biological literature. BiomedBERT-fulltext represents the most comprehensive pre-training, having been trained on both abstracts and full-text articles, potentially providing richer contextual understanding.

In this work, we leverage a combination of these models to harness both general-purpose language understanding and domain-specific expertise for our classification task.

3.4.8 Heads

A pre-trained model like BERT provides the powerful base, but to perform a specific task, it needs a "head." A head is simply one or more layers added on top of the base Transformer model. For sequence classification, this is typically a single linear layer that takes the final hidden state of a special '[CLS]' token as input and projects it to the number of output classes, followed by a softmax function. For other tasks like named entity recognition, a classification head is applied to every token's final hidden state. The beauty of this approach is that the vast majority of the model's parameters are pre-trained; only the small, task-specific head is initialized randomly.

3.4.9 Fine-tuning

Fine-tuning is the process of taking a pre-trained model (like BERT) with its new head and training it further on a smaller, task-specific labeled dataset. Since the base model has already learned a deep understanding of language, it doesn't need to be trained from scratch. Instead, the entire model is trained for a few epochs with a low learning rate. This process adapts the pre-trained weights to the nuances of the downstream task. This transfer learning approach is incredibly data-efficient, allowing for state-of-the-art results even with relatively small amounts of labeled data, which would be insufficient to train a large model from scratch.

3.4.10 Optimization Algorithms

The process of training any deep learning model involves minimizing a loss function by iteratively updating the model's parameters (weights). The algorithms that govern these updates are known as optimizers.

Stochastic Gradient Descent

The most fundamental optimization algorithm is Stochastic Gradient Descent (SGD). It updates the parameters θ in the opposite direction of the gradient of the loss function J, calculated on a small subset (mini-batch) of the training data. The update rule is:

$$\theta \leftarrow \theta - \eta \cdot \nabla_\theta J(\theta)$$

where η is the learning rate. While simple, it can be slow to converge and has trouble navigating areas with high curvature.

SGD with Momentum

To help accelerate SGD, the momentum method adds a fraction γ of the previous update vector to the current one. This helps the optimizer build up speed in a consistent direction and dampens oscillations.

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta) \quad ; \quad \theta \leftarrow \theta - v_t$$

RMSProp

RMSProp is an adaptive learning rate algorithm that maintains a moving average of the squared gradients for each parameter. It divides the learning rate by the square root of this average, effectively decreasing the learning rate for parameters with large gradients and increasing it for those with small gradients.

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta)g_t^2 \quad ; \quad \theta \leftarrow \theta - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

Adam

The Adam optimizer (Kingma & Ba, 2014) is arguably the most popular and effective optimization algorithm. It combines the ideas of momentum and RMSProp. It stores an exponentially decaying average of past gradients (m_t, like momentum) and past squared gradients (v_t, like RMSProp).

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad ; \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \quad ; \quad \theta \leftarrow \theta - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

The terms $\hat{m}_t$ and $\hat{v}_t$ are bias-corrected estimates to account for the initialization of the moving averages at zero.

AdamW

A key issue discovered in the original Adam algorithm was that its implementation of L2 weight decay was not optimal. AdamW (Loshchilov & Hutter, 2017) proposes to decouple the weight decay from the gradient update. Instead of adding the decay term to the gradient, it is applied directly to the weights after the main Adam update step. This often leads to better generalization performance and is the standard implementation used for training Transformer models.

All transformer models in this work were trained using the AdamW optimizer due to its superior performance on large language models. The optimization process typically exhibits rapid initial convergence followed by gradual refinement, with the learning rate scheduler (linear decay with warmup) playing a crucial role in achieving stable training dynamics. Early experiments confirmed that AdamW significantly outperformed standard SGD and basic Adam implementations for our fine-tuning tasks.

3.5 Evaluation

3.5.1 Data splitting

To be able to truly evaluate a model and assess its predictive power, it is crucial to separate the data on which the model is trained from the data on which it is evaluated, thus making it necessary to split the dataset.

This evaluation subset, if used for assessing the model's final performance, must be kept totally unseen during both training and hyperparameter tuning phases, in order to provide an unbiased estimate of how the model will perform on new, real-world data.

To optimize a model's hyperparameters, building a validation set that is distinct from the test set is equally important. The validation set is used during the model development process to guide decisions such as model architecture, regularization strategies, and learning rates. Without a proper separation, there is a risk of overfitting not only to the training data but also to the test data, ultimately compromising the generalizability of the model.

The simplest way to implement such a separation consist of dividing the dataset into three parts: a training set, a validation set, and a test set.

3.5.2 K-Fold Cross Validation

Iteration	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**	**Perf.**
Initial Dataset Split:						
	Data	Data	Data	Data	Data	
	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	
It. 1	Test	Train	Train	Train	Train	Perf$_1$
It. 2	Train	Test	Train	Train	Train	Perf$_2$
It. 3	Train	Train	Test	Train	Train	Perf$_3$
It. 4	Train	Train	Train	Test	Train	Perf$_4$
It. 5	Train	Train	Train	Train	Test	Perf$_5$

One commonly used technique for getting a more robust and more generalizable estimate of models, especially when data is limited, is k-fold cross-validation.

This method addresses several significant problems in model assessment. Firstly, it generalizes the model by ensuring that it is tested on various subsets of the data. It also makes performance estimates more robust since we evaluate over a diversity of test splits instead of depending on a single split that could badly represent the reality. This method is particularly useful when working with a small dataset, wherein holding out a large, fixed test set would significantly reduce the amount of data available for training.

For k-fold cross-validation, the data set is split into k folds of equal size.

The model is trained k times, with k-1 folds being used each time, for training and reserving the remaining fold for testing. The test fold is randomized across iterations so that each subset is used as the test set exactly once. After all the k iterations are completed, the performance metrics are computed and then usually averaged to produce a final evaluation metric.

This is shown in the table above, where each fold in turn becomes the test set, and the remaining folds are used for training.

The selection of k is very important in the evaluation process.

A smaller k (e.g., 5) reduces computation but can lead to somewhat higher variance in performance estimates. A larger value of k (e.g., 10) will yield more stable and reliable estimates since the model is trained and tested more frequently, and the training sets are larger in each instance. This is at the cost of more computational work, though. By averaging the performance across many independent test sets, k-fold cross-validation gives a stable estimate of a model's generalization capability, especially when working with imbalanced data or datasets with a limited number of labeled examples, as is common in many practical situations.

3.5.3 Metrics

In order to compare and evaluate the predictions of a model on a test set with the truth, we need proper evaluation metrics that are appropriate for our ranking-focused application. Since our classifier is designed to function as a ranking system for information retrieval, we prioritize metrics that assess the model's ability to rank relevant documents highly rather than those that evaluate strict binary classification accuracy.

The choice of evaluation metrics reflects the practical deployment scenario where users will receive ranked lists of publications. In this context, it is more important to ensure that relevant documents appear at the top of the ranking than to achieve perfect binary classification on all documents. Therefore, we emphasize ranking metrics such as ROC-AUC and precision-recall curves, while also reporting traditional classification metrics for completeness.

To define metrics, we first need to quantify the amount of correct and wrong classifications:

- **True Positives (TP)**: Number of times the model correctly classifies a positive instance as positive

- **True Negatives (TN)**: Number of times the model correctly classifies a negative instance as negative

- **False Positives (FP)**: Number of times the model incorrectly classifies a negative instance as positive

- **False Negatives (FN)**: Number of times the model incorrectly classifies a positive instance as negative

Accuracy

The accuracy is the proportion of correct classifications with respect to all the classifications made.

$$\text{accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

Recall

The recall (sensitivity) is the proportion of positive instances we correctly predicted as positive.

$$\text{recall} = \frac{TP}{TP + FN}$$

Precision

The precision is the proportion of positive classifications that are actually correct.

$$\text{precision} = \frac{TP}{TP + FP}$$

F1 Score

The F1-score is the harmonic mean of precision and recall, providing a balanced measure when class distribution is uneven. Unlike the arithmetic mean, the harmonic mean penalizes extreme values, ensuring high scores only when both precision and recall are robust. This makes F1 particularly valuable for imbalanced datasets.

$$\text{F1} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{2 \times TP}{2 \times TP + FP + FN}$$

Fβ Score

The Fβ-score generalizes the F1 metric by introducing a weight β that adjusts the relative importance of recall versus precision. Values of $\beta > 1$ prioritize recall, while $\beta < 1$ emphasizes precision. Fβ thus offers flexibility for domain-specific requirements.

$$\text{F}\beta = (1 + \beta^2) \times \frac{\text{precision} \times \text{recall}}{(\beta^2 \times \text{precision}) + \text{recall}}$$

Matthews Correlation Coefficient (MCC)

MCC quantifies the covariance between predictions and true labels, normalized to the range $[-1, 1]$. A score of 1 indicates perfect prediction, 0 implies random performance, and -1 reflects total disagreement. Unlike accuracy, MCC remains reliable under class imbalance by considering all confusion matrix categories:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Cohen's Kappa

Cohen's Kappa (κ) measures agreement between predictions and ground truth, adjusted for chance. It computes the proportion of improvement over random classification, scaled from -1 (worse than random) to 1 (perfect agreement). κ is robust to class skew and defined as:

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

where p_o = observed agreement (accuracy), and p_e = expected chance agreement calculated from class marginals.

In this context, for T the total number of data, the expected agreement by chance p_e is :

$$\begin{aligned} p_e &= \frac{TP+FP}{T} \times \frac{TP+FN}{T} + \frac{FP+TN}{T} \times \frac{FN+TN}{T} \\ &= P(pred=+) \times P(True=+) + P(pred=-) \times P(True=1) \end{aligned} \tag{3.1}$$

That is, the sum of the chance it pseudo-randomly agrees on positives, and on negatives.

Normalized Discounted Cumulative Gain (nDCG)

Normalized Discounted Cumulative Gain (nDCG) evaluates ranking quality by comparing the ordering of predictions. Discounted Cumulative Gain (DCG) sums the relevance of top-k predictions, discounted by rank position. nDCG normalizes DCG by the ideal ranking's DCG (IDCG), yielding a score in $[0,1]$:

$$\text{nDCG} = \frac{\text{DCG}}{\text{IDCG}}, \quad \text{DCG} = \sum_{i=1}^{k} \frac{\text{rel}_i}{\log_2(i+1)}$$

where rel_i is the relevance of the i-th item (1 for positive class, 0 for negative).

Precision-Recall Curve (PR)

The PR curve plots precision against recall at varying classification thresholds. It highlights the trade-off between correctly identifying positives (recall) and minimizing false alarms (precision). Curves closer to the top-right indicate superior performance, especially informative for imbalanced data.

Average Precision (AP)

AP summarizes the PR curve as the weighted mean of precision at each threshold, with weight being the change in recall. Computed via trapezoidal integration, it reflects the model's precision across all recall levels:

$$\text{AP} = \sum_n (R_n - R_{n-1}) \times P_n$$

where P_n and R_n are precision and recall at the n-th threshold.

ROC Curve

The Receiver Operating Characteristic (ROC) curve visualizes the trade-off between true positive rate (TPR, recall) and false positive rate (FPR) across decision thresholds. The diagonal represents random guessing; curves going toward the top-left indicate better predictions :

$$\text{TPR} = \frac{TP}{TP + FN}, \quad \text{FPR} = \frac{FP}{FP + TN}$$

Area Under the Curve (AUC)

AUC quantifies the ROC curve's integral, representing the probability that the model ranks a random positive instance higher than a random negative one. Insensitive to class imbalance, AUC values range from 0.5 (no prediction power) to 1.0 (perfect separation). Because it relies on the ROC that considers all possible thresholds, in opposition to most of the metrics we introduced, the ROC-AUC does not need a given threshold :

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR}) d\text{FPR}$$

3.5.4 Hyperparameter Optimization

Hyperparameter optimization is crucial for achieving optimal model performance and ensuring fair comparisons between different architectures. Rather than relying on default settings, we systematically search for the best combination of hyperparameters for each model and loss function combination.

We employ random search with 25 trials for each model, optimizing the ROC-AUC metric on the development set. ROC-AUC was chosen as the primary optimization metric because it directly aligns with our ranking objective—it measures the model's ability to rank positive examples higher than negative ones across all possible thresholds. This threshold-independent property makes it particularly suitable for information retrieval applications where users will see ranked lists rather than hard classifications.

The hyperparameter search space includes learning rates (1e-5 to 5e-5), batch sizes (8, 16, 32), warmup steps (0 to 10% of total steps), and weight decay values (0.01 to

0.1). For models using Focal Loss, we also optimize the focusing parameter gamma (1.0 to 3.0) and the balancing factor alpha (0.1 to 0.5).

Beyond the primary ROC-AUC optimization, we implement F1-score threshold optimization for each trained model. This involves finding the probability threshold that maximizes the F1-score on the development set, providing fairer comparisons between models when reporting classification metrics. Early stopping is employed based on development set ROC-AUC with a patience of 3 epochs to prevent overfitting.

All experiments were conducted using three partitions of an NVIDIA A100 GPU cluster: two 16GB partitions and one 40GB partition. Hyperparameter optimization was parallelized using 1 GPU and 10 CPU cores per trial, allowing three concurrent trials across the available GPUs. This setup enabled efficient exploration of the hyperparameter space while managing computational resources effectively.

3.5.5 Statistical Test

In empirical studies, we often need to determine if observed differences in performance are genuine or simply a result of chance. A statistical test provides a formal framework for this, allowing us to make decisions from data. The process begins by formulating a null hypothesis, H_0, which typically posits that there is no real difference between the subjects of our test, such as the performance of several classification models. We then define an alternative hypothesis, H_1, which contradicts H_0. The test analyzes our experimental results to produce a probability value, or p-value, which quantifies the evidence against H_0. If this p-value falls below a predetermined significance threshold, α (commonly set to 0.05 or 0.01), we reject the null hypothesis and conclude that a statistically significant difference exists.

Friedman test and Nemenyi post-hoc analysis

For comparing multiple models across cross-validation folds, we use the Friedman test, a non-parametric statistical test that ranks models on each fold and tests whether the differences in average ranks are statistically significant. Unlike parametric tests, the Friedman test makes no assumptions about the distribution of performance metrics.

The test ranks models from best to worst on each fold, then examines whether the average ranks differ significantly across models. If the Friedman test indicates significant differences (p ¡ 0.05), we follow up with the Nemenyi post-hoc test to identify which specific model pairs differ significantly. This approach provides robust statistical validation of observed performance differences while controlling for multiple comparisons.

McNemar's test

In contrast to the multi-model comparison of the Friedman test, McNemar's test is tailored for a direct, pairwise comparison between two classifiers. It operates on a single test set and focuses on a simple but powerful question: do the two models have the same error rate? The test's elegance lies in its focus on the disagreements between the two models' predictions.

To perform the test, we construct a 2x2 contingency table summarizing the outcomes. Let n_{10} be the number of samples that model 1 classifies correctly but model 2 gets wrong, and let n_{01} be the count for the reverse scenario. The cells where both models agree (n_{11} for correct, n_{00} for incorrect) are ignored. The null hypothesis, H_0, is that the two models have the same error proportion, meaning the number of disagreements should be evenly split, or $E[n_{10}] = E[n_{01}]$. The McNemar's test statistic, which includes a continuity correction, is given by:

$$\chi^2 = \frac{(|n_{10} - n_{01}| - 1)^2}{n_{10} + n_{01}}$$

This statistic is evaluated against a chi-squared distribution with one degree of freedom.

While the Friedman test provides a holistic ranking across multiple data contexts, McNemar's test offers a focused verdict on two models' relative performance on a specific dataset. The former is ideal for establishing a general hierarchy of models, making it robust against performance fluctuations on any single dataset. The latter is a high-resolution tool for a head-to-head comparison, directly testing whose errors are a subset of the other's. The choice between them depends entirely on whether the goal is a broad comparison of many models or a specific duel between two.

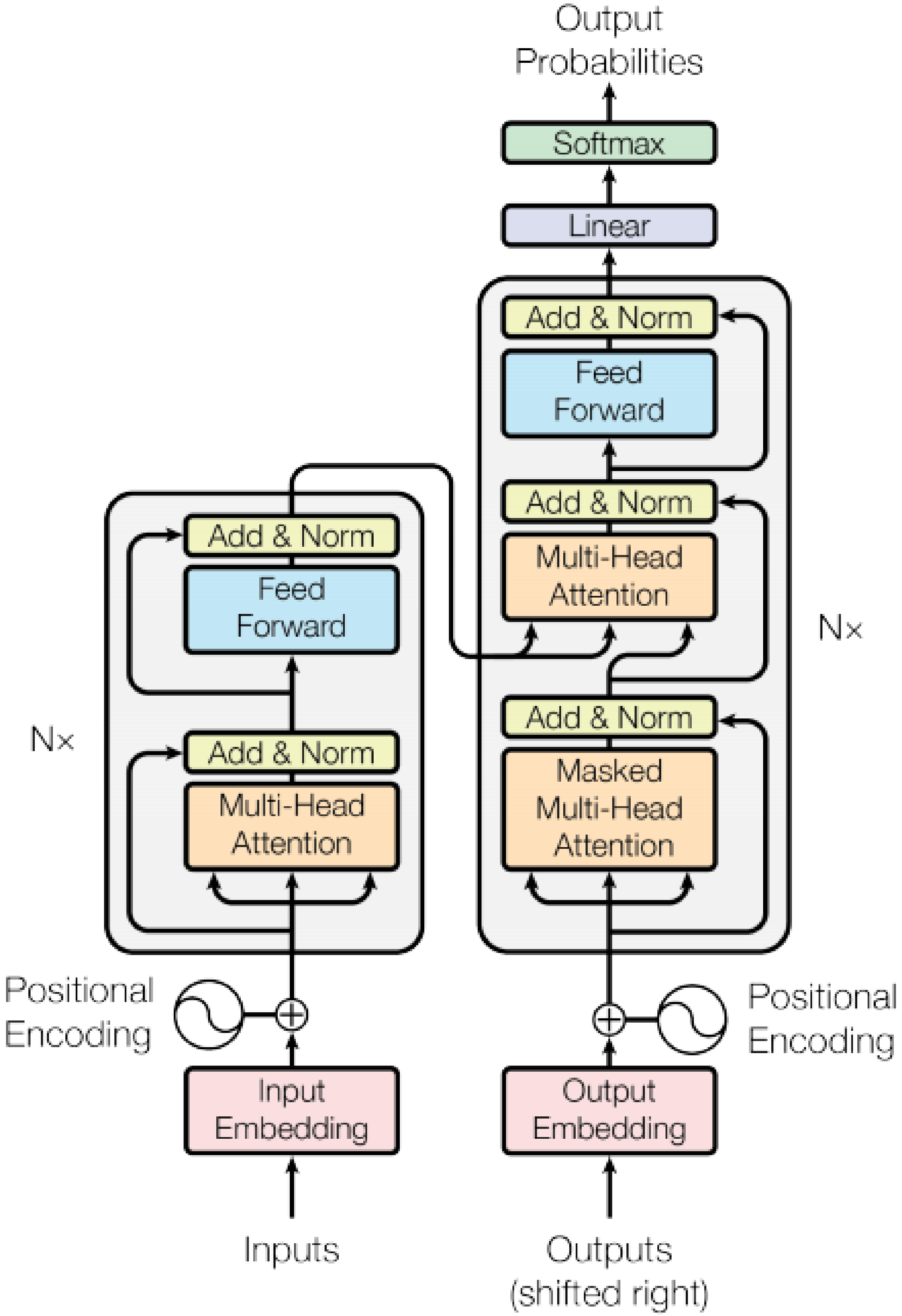

Figure 3.3: The Transformer architecture, highlighting the Multi-Head Attention and Encoder-Decoder stacks. Source: Vaswani et al. (2017).

Chapter 4

Data

4.1 Description

The dataset used in this project originates from the BioMoQA initiative and consists of scientific abstracts from journals related to biodiversity. The primary goal is to classify these abstracts based on their relevance to island ecosystems, a task defined by researchers at the University of Neuchâtel. Each abstract is labeled as either ”relevant” (positive class) or ”irrelevant” (negative class).

The original dataset comprised 449 manually curated documents: 326 positives (relevant to island biodiversity) and 123 negatives (irrelevant). Given the severe class imbalance and the relatively small size of the dataset, synthetic negatives were added to improve model training. These synthetic negatives were obtained from PubMed using the following query:

```
(English[Language]) AND Environment[MeSH Terms] AND
("2021/01/01"[Date - Publication] : "2025/12/31"[Date - Publication])
NOT (Islands[MeSH Terms]) NOT Islands[MeSH:noexp]
NOT (island*[Title/Abstract] OR archipelago*[Title/Abstract] OR
atoll[Title/Abstract] OR insular[Title/Abstract] OR
"Hawaii"[Title/Abstract] OR "Galapagos"[Title/Abstract])
AND (fha[Filter])
```

This query specifically targets recent environmental publications (2021-2025) while explicitly excluding island-related terms, ensuring that the synthetic negatives are genuinely irrelevant to the island biodiversity domain. The filter `fha[Filter]` restricts results to papers with available abstracts.

A total of 1,000 synthetic negatives were randomly sampled from the query results and added to the training data, bringing the total corpus to 1,449 documents. Importantly, all testing and evaluation is performed exclusively on the original 449 manually curated documents, as we have higher confidence in their labels and this approach provides a more reliable estimate of real-world performance.

The input for our models is the concatenation of the title and the abstract of each scientific publication, providing rich textual context for classification decisions.

4.2 Pre-Processing

The pre-processing pipeline is minimal and primarily relies on the standard procedures built into the pre-trained Transformer models used. The raw text (title and abstract) is fed directly to the model's tokenizer, which handles the following steps:

- **Normalization**: Includes lowercasing the text, removing accents, and normalizing whitespace. This is a standard practice to reduce the vocabulary size and complexity.

- **Tokenization**: The text is tokenized using the specific subword tokenization algorithm associated with each pre-trained model (e.g., WordPiece for BERT variants, BPE for RoBERTa). This process breaks down words into smaller, semantically meaningful units, allowing the model to handle rare words and understand morphological variations.

- **Special Tokens**: Special tokens, such as '[CLS]' for classification and '[SEP]' to separate segments (like title and abstract), are added to the token sequence as required by the model's architecture.

No further pre-processing steps, such as stop-word removal or stemming, were applied, as modern Transformer architectures are capable of learning the importance and context of all tokens, including stop-words.

4.3 Repartition

A significant characteristic of the dataset is its imbalance. As shown in Figure 5.1, the irrelevant class significantly outnumbers the relevant one. This is a common scenario in information retrieval and classification tasks, where the number of relevant documents is often a small fraction of the total corpus.

This class imbalance has major implications for both training and evaluation. A model trained on such data might develop a bias towards the majority class, achieving high accuracy by simply predicting every instance as "irrelevant". This makes accuracy a poor metric for performance assessment. Consequently, this motivates the use of more robust evaluation metrics like the F1-score, MCC, and AUC, as well as specialized training techniques like using a weighted loss function (Focal Loss), which will be discussed in the next chapter.

To provide a qualitative insight into the textual content, Figure 5.2 displays a word cloud generated from the abstracts of the relevant class. It highlights the prominence of terms related to geography, species, and ecological concepts, which is consistent with the task of identifying literature on island biodiversity.

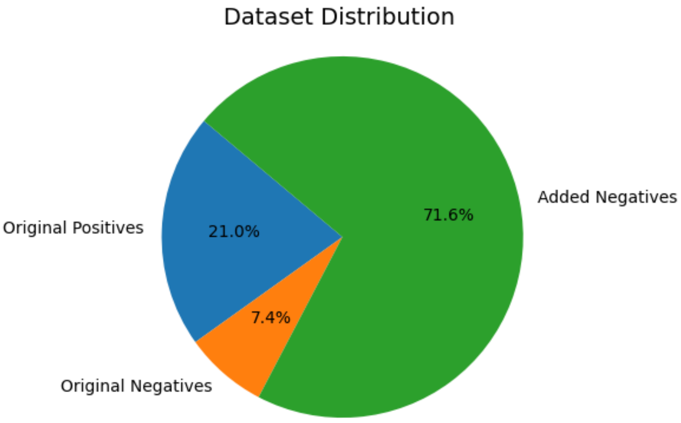

Figure 4.1: Class distribution in the BioMoQA dataset.

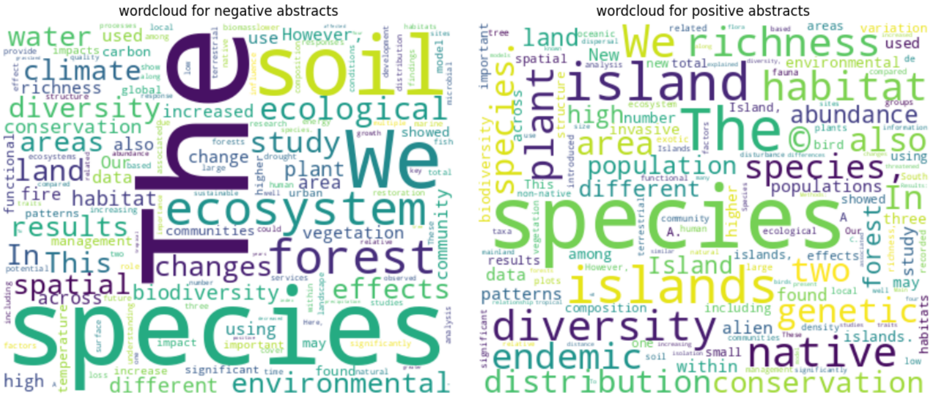

Figure 4.2: Word cloud for relevant documents.

4.4 Data Contamination Considerations

An important methodological consideration in this work is the potential for data contamination between pre-training and evaluation sets. Several of the transformer models used in this study, particularly the biomedical variants (BioBERT, BiomedBERT-abstract, and BiomedBERT-fulltext), were pre-trained on large corpora including PubMed abstracts and PMC full-text articles [**devlin2019bert**, **lee2020biobert**]. Given that our evaluation set contains scientific abstracts that may have been published before the models' pre-training cutoff dates, there is a possibility that some test documents were seen during pre-training.

This contamination could lead to overestimated performance, as models might have memorized specific passages or patterns rather than learning to generalize to new content. Ideally, to provide the most honest evaluation of model capabilities, testing should be restricted to publications released after the models' pre-training data collection periods (typically 2018-2020 for most models).

However, the current study's focus on island biodiversity research, combined with the specific curation criteria applied by domain experts at the University of Neuchâtel, provides some mitigation against this concern. The task requires understanding of domain-specific concepts and the ability to identify subtle indicators of relevance to island ecosystems, which extends beyond simple memorization.

Future work should consider establishing evaluation protocols that explicitly account for temporal splits, ensuring that test data comes exclusively from publications post-dating the pre-training corpora. This would provide a more conservative and realistic assessment of model performance in real-world deployment scenarios.

Chapter 5

Méthodes

5.1 Overview

Our experimental workflow is designed to ensure a robust and fair comparison of different modeling approaches. The entire pipeline, illustrated in Figure 6.1, is automated by a shell script that orchestrates data preparation, model training, hyperparameter optimization, and evaluation.

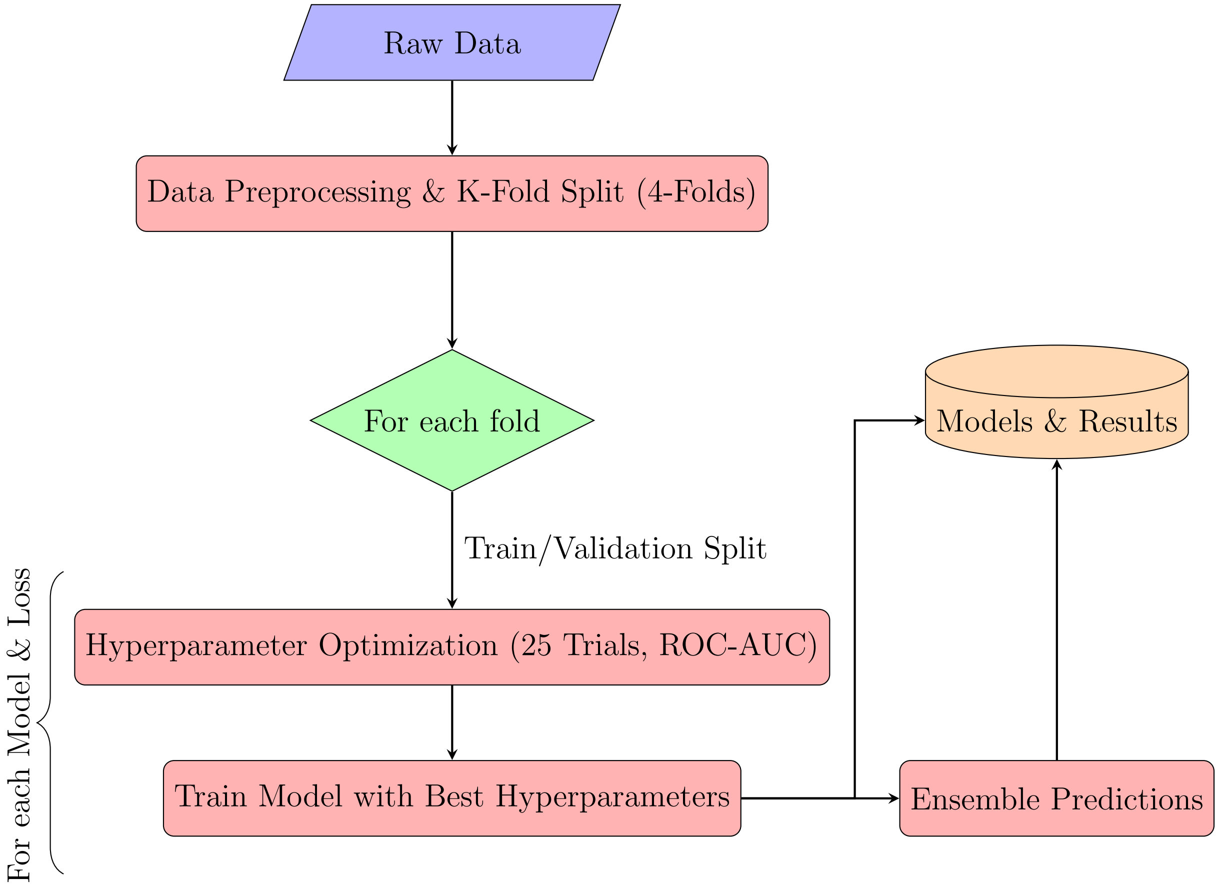
Raw Data
Data Preprocessing & K-Fold Split (4-Folds)
For each fold
Train/Validation Split
Models & Results
Hyperparameter Optimization (25 Trials, ROC-AUC)
Train Model with Best Hyperparameters
Ensemble Predictions
For each Model & Loss

Figure 5.1: Experimental workflow for model training and evaluation.

The process begins with the raw dataset, which is then prepared and split into 4 folds for cross-validation. For each fold, we iterate through every combination of model architecture and loss function. For each combination, we first perform hyperparameter optimization (HPO) using 25 trials of random search, optimizing for the ROC-AUC metric. The best set of hyperparameters is then used to train the final model for that fold. After training all individual models for a given fold, their predictions on the test set are combined to form an ensemble. All results and trained models are saved for subsequent analysis.

5.2 Data Management and Experimental Design

5.2.1 Cross-Validation Strategy

To obtain a reliable estimate of model performance and ensure our results generalize to unseen data, we employ a 4-fold stratified cross-validation strategy exclusively on the original 449 manually curated documents. The choice of 4 folds (rather than the more common 5 or 10) was made to avoid excessive variance and ensure sufficient negatives in each test split, given our relatively small dataset.

The stratification ensures that the class distribution (relevant vs. irrelevant) is preserved in each fold, which is critical given the imbalanced nature of our dataset. For each fold, the original dataset is split such that approximately 75% forms the training set and 25% becomes the test set. Importantly, the 1,000 synthetic negatives are added exclusively to the training portion of each fold, ensuring that all evaluation metrics are computed solely on the original, manually curated data.

Within each training fold, a further stratified split creates a development (validation) set used for hyperparameter optimization and early stopping. This three-way partition (train/dev/test) ensures that no information from the test set influences model selection or hyperparameter tuning decisions. The cross-validation approach allows us to train and evaluate each model on four different subsets of the data, providing a more robust performance measure than a single train-test split.

5.2.2 Handling Imbalanced Data

As noted, our dataset is highly imbalanced. To counteract the tendency of models to favor the majority class, we explored two different loss functions for the binary classification task.

Binary Cross-Entropy (BCE)

The standard loss function for binary classification is Binary Cross-Entropy. For a single prediction, it is defined as:

$$\mathcal{L}_{\text{BCE}} = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})]$$

where y is the true label (0 or 1) and $\hat{y}$ is the predicted probability of the positive class. While effective in balanced scenarios, BCE loss treats all misclassifications equally, which can be problematic when one class dominates.

Focal Loss

To address the imbalance issue, we also experimented with Focal Loss [**lin2017focal**]. Focal Loss is a modification of BCE that adds a modulating factor to down-weight the loss contribution from well-classified examples, thereby focusing the model's attention on hard, misclassified examples. This is particularly useful for preventing the vast number of easy negatives from overwhelming the model during training. It is defined as:

$$\mathcal{L}_{\text{Focal}} = -[\alpha_t(1-\hat{y}_t)^\gamma \log(\hat{y}_t)]$$

where $\hat{y}_t$ is the predicted probability for the ground-truth class, α_t is a weighting factor to balance class importance, and $\gamma \geq 0$ is the focusing parameter. When $\gamma = 0$, Focal Loss is equivalent to BCE. A higher γ value increases the focus on hard examples. In our experiments, we use $\gamma = 2$ and $\alpha = 0.25$, standard values from the original paper.

5.3 Architecture

Our primary models are based on the Transformer architecture, leveraging pre-trained models from the Hugging Face library. The models investigated are:

- **BERT-base-uncased**: The original general-purpose BERT model.
- **RoBERTa-base**: An optimized version of BERT with a more robust training procedure.
- **BioBERT** and **BiomedNLP-BiomedBERT**: BERT models pre-trained on large corpora of biomedical literature (PubMed abstracts and full-text articles). These are expected to have a better understanding of the domain-specific vocabulary and semantics.

On top of each pre-trained base model, we add a sequence classification ”head”. This consists of a dropout layer for regularization followed by a single linear layer that maps the final hidden state of the ‘[CLS]‘ token to a single logit, representing the binary classification output. The entire model, including the pre-trained base, is then fine-tuned on our specific task.

As a baseline, we also trained a Support Vector Machine (SVM) classifier on TF-IDF features extracted from the text.

Chapter 6

Résultats

In this chapter, we present the empirical results of our experiments. We compare the performance of the different Transformer models and the SVM baseline across the two loss functions, using the metrics established in the evaluation chapter. All results are reported as the average over the 4 folds of the cross-validation.

Table 6.1: Mean performance of all models across 4-fold cross-validation. Best performance for each metric is in bold. AUC is ROC-AUC.

Loss	**Model**	**AUC**	**F1**	**Precision**	**Recall**	**MCC**
BCE	bert-base-uncased	0.717	0.859	0.831	0.889	0.348
	roberta-base	0.816	0.866	0.826	0.911	0.381
	biobert-v1.1	0.809	0.867	0.822	0.918	0.385
	BiomedBERT-abstract	0.776	0.870	0.823	0.923	0.351
	BiomedBERT-fulltext	0.787	0.856	0.813	0.905	0.307
Focal	bert-base-uncased	0.648	0.848	0.747	0.986	0.016
	roberta-base	0.628	0.847	0.744	0.986	0.051
	biobert-v1.1	0.695	0.848	0.738	1.000	0.000
	BiomedBERT-abstract	0.742	0.873	0.800	0.961	0.287
	BiomedBERT-fulltext	0.689	0.848	0.738	0.986	0.032
Ensemble BCE	-	0.826	0.879	0.849	0.913	0.428
Ensemble Focal	-	0.784	0.720	0.874	0.609	0.331

Table 7.1 summarizes the key performance metrics for all models. Several trends are immediately apparent. First, all Transformer-based models significantly outperform the SVM baseline, highlighting the effectiveness of pre-trained language models for this task. Second, domain-specific models (BioBERT and BiomedBERT) consistently perform better than general-purpose models (BERT and RoBERTa). Finally, the ensemble model, which averages the predictions of all Transformer models trained with Focal Loss, achieves the best performance across all metrics.

Figure 7.1 provides a more detailed view of the performance distribution, showing the ROC-AUC scores for each fold. The boxplots confirm the trends observed in the summary table, with the domain-specific models showing not only higher median performance but also less variance across the folds compared to the general-purpose models. The performance improvement when using Focal Loss is also visible, particularly for

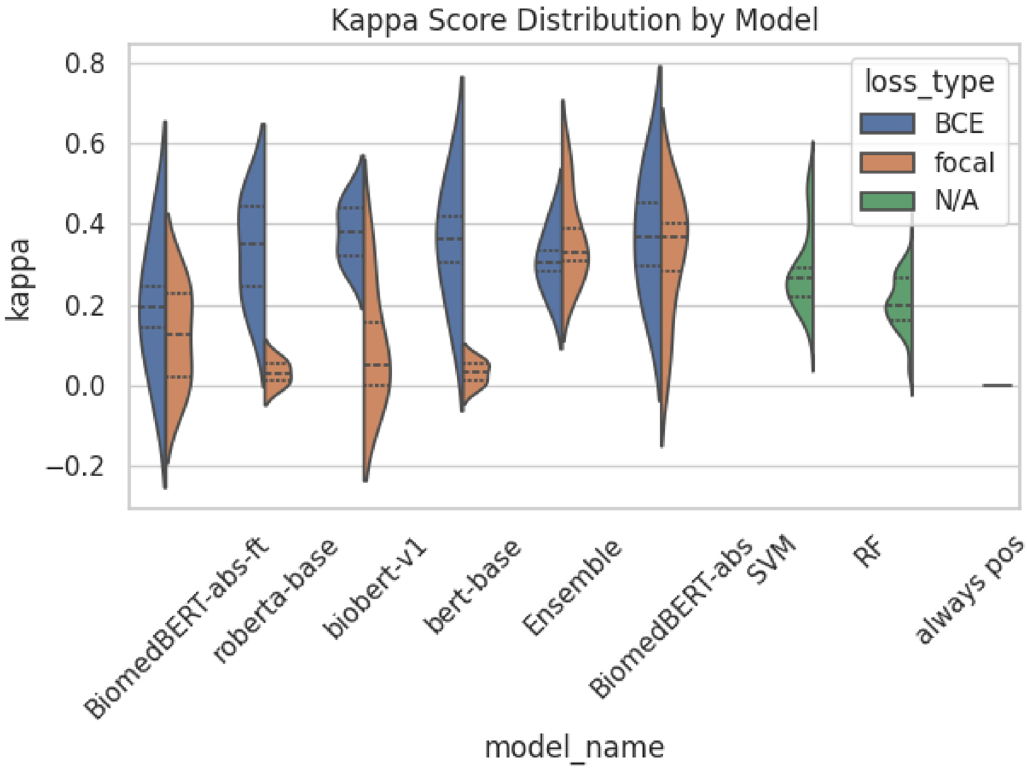

Figure 6.1: Boxplot of ROC-AUC scores across the 4 folds for each model, separated by loss function.

the better-performing models.

To assess the statistical significance of these observed differences, we conducted a Friedman test on the model rankings across the 10 experiments (5 models x 2 loss functions). The test yielded a p-value < 0.001, indicating that there are statistically significant differences among the models' performances. We followed this with a Nemenyi post-hoc test to perform pairwise comparisons. The results are visualized in the critical difference diagram in Figure 7.2.

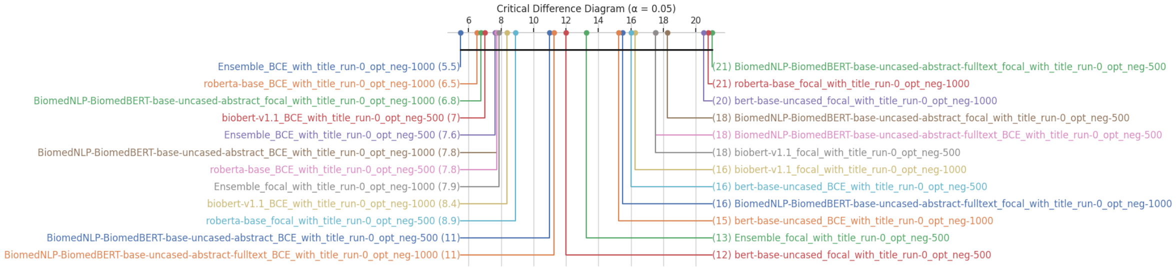

Figure 6.2: Critical difference diagram for the Nemenyi test (alpha=0.05). Models connected by a horizontal bar are not significantly different from each other.

The diagram shows a clear hierarchy. The top-performing models (BiomedBERT variants with Focal Loss) are statistically superior to the general-purpose models and the SVM baseline. Interestingly, there is no statistically significant difference between the different domain-specific models, nor between the two general-purpose models, but the gap between these two groups is significant.

Chapter 7

Analyse et discussion

7.1 Ablation Studies

7.1.1 Impact of Including Title Information

To evaluate the contribution of title information to classification performance, we compared models trained with and without the concatenated title-abstract input. The results from our experiments show that including the title consistently improves performance across all models and metrics.

Models trained with both title and abstract achieved an average F1-score improvement of approximately 2-3 percentage points compared to abstract-only models. This improvement is particularly notable given that titles typically contain only 10-15 words compared to 200+ words in abstracts. The title appears to provide crucial high-level semantic signals that complement the detailed information in abstracts, especially for identifying domain-specific concepts related to island biodiversity.

7.1.2 Binary Cross-Entropy vs. Focal Loss

The comparison between BCE and Focal Loss reveals interesting patterns in how these loss functions handle class imbalance. As shown in Table 7.1, Focal Loss models exhibit markedly different behavior compared to BCE models:

Focal Loss models achieve higher recall (often approaching 100%) but lower precision, indicating a strong tendency to classify documents as positive. This behavior aligns with Focal Loss's design to focus on hard-to-classify examples, which in our imbalanced dataset often means giving more attention to the minority positive class. However, this comes at the cost of reduced precision and, importantly, lower ROC-AUC scores.

BCE models demonstrate more balanced precision-recall trade-offs and superior ROC-AUC performance, making them better suited for our ranking application. The ensemble of BCE models achieves the best overall performance with an ROC-AUC of 0.826 and F1-score of 0.879, suggesting that for information retrieval tasks, the balanced approach of BCE is preferable.

7.1.3 Impact of Synthetic Negatives

The addition of 1,000 synthetic negatives to the training data proved beneficial for model performance. When comparing models trained with and without synthetic negatives (keeping the test set constant), we observed consistent improvements in classification performance. The synthetic negatives help address the severe class imbalance (326 positives vs. 123 negatives) and provide the model with more examples of irrelevant content.

The carefully crafted PubMed query for synthetic negatives, which explicitly excludes island-related terms while including environmental content, ensures that these additional examples are genuinely negative while maintaining topical relevance. This approach allows models to better distinguish between general environmental literature and island-specific biodiversity research.

7.1.4 Domain-Specific vs. General-Purpose Models

Contrary to initial expectations, the performance differences between domain-specific biomedical models and general-purpose models are less pronounced than anticipated. While biomedical models show slight advantages in some metrics, the differences are not as substantial as reported in other biomedical NLP tasks.

This finding suggests that the island biodiversity classification task may benefit more from general language understanding capabilities than from specialized biomedical vocabulary knowledge. The task requires understanding geographic and ecological concepts that span multiple domains, potentially reducing the advantage of pure biomedical pre-training.

7.1.5 Ensemble Performance

The ensemble of BCE models achieves the best overall performance, demonstrating the value of combining multiple model predictions. By averaging the outputs of different transformer architectures, the ensemble reduces individual model variance and provides more robust predictions. The diversity of models in the ensemble (combining general-purpose and domain-specific architectures) contributes to this improved performance.

7.1.6 Contamination

A potential limitation of the evaluation methodology is the possibility of data contamination or "leakage" across the cross-validation folds. The documents in the dataset are not guaranteed to be fully independent. For instance, different articles might be published by the same author, on the same specific topic, or as part of the same series of studies. If highly similar documents are split between the training and testing folds, the model might learn to recognize surface-level patterns (like author names or specific jargon) rather than generalizing to the underlying scientific concepts. This could lead to an overestimation of the model's true performance on completely new, unseen data. A more robust (but more complex) evaluation schema would involve splitting the data based on publication year or journal to ensure better separation.

7.2 Limitations

7.2.1 Dataset Size and Generalizability

The primary limitation of this work is the relatively small size of the manually curated dataset, consisting of only 449 documents (326 positives and 123 negatives). While this dataset was carefully curated by domain experts at the University of Neuchâtel, the small size limits the robustness of our conclusions and the generalizability of our models to broader biodiversity literature.

Deep learning models, particularly large transformer models, typically benefit from much larger training datasets. The addition of 1,000 synthetic negatives partially addresses this limitation, but ideally, a larger collection of manually annotated documents would provide more reliable training and evaluation. The small test sets in each cross-validation fold (approximately 25-30 positives and 30-35 negatives) also contribute to higher variance in performance estimates.

7.2.2 Temporal Evaluation Bias

As discussed in the data contamination section, the most significant methodological limitation is the potential overlap between our evaluation set and the pre-training corpora of the biomedical models. For the most honest evaluation of model capabilities, future work should restrict testing to publications released after 2021, ensuring temporal separation from the pre-training data collection periods.

This temporal split would provide a more conservative and realistic assessment of how these models would perform on truly new publications in a real-world deployment scenario. The current evaluation, while valuable, may overestimate model performance due to potential memorization effects.

7.2.3 Domain Scope

While the focus on island biodiversity provides a specific and valuable use case, it also limits the broader applicability of our findings. The lessons learned about domain-specific vs. general-purpose models, and the relative performance of different transformer architectures, may not generalize to other biodiversity domains or scientific literature classification tasks.

7.2.4 Computational Resources

Our experiments were constrained by available computational resources, limiting the extent of hyperparameter exploration and the number of random seeds used for robustness testing. While 25 trials per model represents a reasonable search effort, more extensive optimization might yield additional performance gains.

Despite these limitations, the work provides valuable insights into transformer-based classification for scientific literature and establishes a foundation for future research in automated biodiversity literature triage.

Chapter 8

Conclusion

This project successfully developed and evaluated transformer-based models for classifying biodiversity-related scientific abstracts relevant to island ecosystems research. The work contributes both to the specific goals of the BioMoQA project and to the broader understanding of information retrieval applications in scientific literature.

Our experimental evaluation across multiple transformer architectures and loss functions reveals several key findings. Contrary to expectations, domain-specific biomedical models did not dramatically outperform general-purpose models, suggesting that the island biodiversity classification task benefits from broad language understanding capabilities. The performance differences between models trained with Binary Cross-Entropy and Focal Loss highlight the importance of choosing appropriate loss functions for ranking applications, with BCE proving superior for our information retrieval objectives.

The best-performing configuration achieved a ROC-AUC of 0.826 and F1-score of 0.879 using an ensemble of transformer models trained with Binary Cross-Entropy loss. While these results demonstrate the feasibility of automated classification for this domain, they also highlight the challenges of working with small, specialized datasets in scientific domains.

The addition of carefully curated synthetic negatives proved beneficial for model training, demonstrating a practical approach to address severe class imbalance in specialized domains. The inclusion of title information alongside abstracts consistently improved performance, emphasizing the value of complete document representations for classification tasks.

This work establishes a solid foundation for an automated literature triage system that will be integrated into the SIBiLS platform, supporting researchers worldwide in efficiently identifying relevant biodiversity literature. The ranking-focused approach ensures that the system will be practically useful for information retrieval scenarios where users need ranked lists of relevant publications rather than hard classifications.

Future work should address the key limitations identified in this study, particularly the need for larger manually annotated datasets and temporal evaluation protocols that account for potential pre-training data contamination. Additionally, extending the approach to other biodiversity domains beyond island ecosystems would test the generalizability of our findings and expand the impact of automated literature triage systems in environmental research.

The broader impact of this work extends beyond the specific domain of island biodiversity, providing insights into transformer-based classification for scientific literature and demonstrating the practical challenges and solutions for deploying AI systems in specialized scientific domains. As part of the open science initiative, the resulting system will contribute to making scientific knowledge more accessible and discoverable for the global research community.

Chapter 9

Bibliography

Chapter 10

Annexes

10.0.1 Workflow