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. Je remercie aussi l'ensemble des membres du groupe BiTeM pour leur accueil et leurs conseils précieux et instructifs tout au long de cette période, ainsi que mes encadrants académiques pour leurs retours pertinents et leur disponibilité. Enfin, j'exprime ma gratitude aux partenaires du projet, notamment les équipes impliquées dans le projet BioMoQA et l'IPBES, pour les échanges scientifiques et l'accès aux données.

Contents

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 deep-learning 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. One of its secondary objective is to enhance SIBiLS by improving access to biodiversity-related scientific literature through deep-learning powered analytical services. 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.

1.2 IPBES Literature Classification

The Intergovernmental Science-Policy Platform on Biodiversity and Ecosystem Services (IPBES) produces assessments that synthesize evidence on biodiversity and ecosystem services to inform policy. In parallel to BioMoQA, I developed a multi-label classifier tailored to IPBES-related literature. The objective is to automatically assign documents to assessment-relevant thematic axes: IAS (Invasive Alien Species), SUA (Sustainable Use Assessment), and VA (Values Assessment). Unlike the binary BioMoQA task, each abstract may belong to none, one, or multiple labels.

The modeling pipeline mirrors BioMoQA: transformer encoders with a task-specific classification head, stratified cross-validation, random-search hyperparameter optimization, and rigorous statistical comparisons. Inputs concatenate title and abstract when available, and training optimizes threshold-independent ranking quality (ROC-AUC, Average Precision) alongside per-label discrimination. This classifier is being integrated into SIBiLS to support large-scale literature triage and consistent tagging of assessment themes; the integration is underway. Detailed methods and results are presented in the Methods and Results chapters.

Chapter 2

Introduction

2.1 Context

During this internship, conducted from March to July 2025, I collaborated with the BiTeM group to -as said earlier- build two classification models. 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 developed and evaluated a multi-label classifier for the IPBES (Intergovernmental Panel on Biodiversity and Ecosystem Services) dataset, which results are also reported here. Both projects share the common goal of enhancing researchers' ability to efficiently navigate the ever-growing corpus of scientific literature. Both models are in the process of being integrated into the SIBiLS interface; this integration is underway and still ongoing at the time of writing.

The central challenge addressed in this project is the development of automated systems that can effectively rank scientific documents by relevance rather than simply 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 Motivation

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 [18]. Advances in natural language procsesing (NLP) have opened new avenues for distilling large volumes of textual data into concise summaries and extracting high-quality, relevant information [21, 26]. Modern transformer-based models enable state-of-the-art performance in tasks including summarization, information retrieval, and question answering [25, 4].

2.3 Objectives

The first 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:

- Build two production-ready classifiers: a BioMoQA binary relevance ranker and an IPBES multi-label thematic tagger (IAS, SUA, VA)
- Optimize ranking quality using threshold-independent metrics: ROC-AUC for BioMoQA; weighted Average Precision and ROC-AUC for IPBES
- Compare BERT pre-trainedmodels (BERT-base, RoBERTa, BioBERT, BiomedBERT) and ensembles under stratified 5-fold cross-validation with fixed seeds
- Perform random-search hyperparameter optimization per model/loss, then train fold-specific final models and ensembles
- Mitigate imbalance: add curated synthetic negatives for BioMoQA; use per-label metrics and thresholding for IPBES
- Use title+abstract inputs and quantify the impact of titles on ranking performance
- Prepare reproducible deployment and integration into the SIBiLS platform (environment pinning, deterministic runs)

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.

In parallel, we design a multi-label classifier for the IPBES corpus to automatically tag documents with one or more assessment axes (IAS, SUA, VA). As with BioMoQA, the emphasis is on ranking quality rather than hard classification: we optimize threshold-independent metrics (weighted Average Precision and ROC-AUC) so that axis-relevant documents are surfaced early. The IPBES pipeline reuses the same transformer encoders, training procedures, and statistical validation, and is intended for integration into SIBiLS to standardize large-scale thematic triage across assessments.

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

Literature Review

Before exploring the model's implementation details, let's establish a brief literature review 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

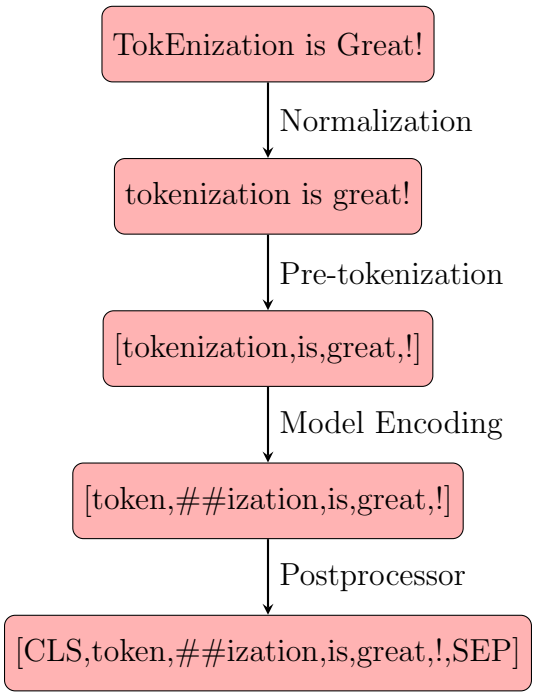

Figure 3.1: Example of a tokenization process

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. 3.1.2)	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 primary tokenization methods used by the transformer models in this project.

3.1.1 Byte Pair Encoding

Byte-Pair Encoding (BPE) was first introduced in 1994 by Philip Gage. It was then modified by [22] as a tokenizer for Large Language Models (LLMs). BPE builds a vocabulary by iteratively merging the most common pair of characters until the desired vocabulary size is reached. The process starts with individual characters and progressively combines frequent character pairs into larger subword units, creating a data-driven vocabulary that balances between character-level and word-level representations.

3.1.2 WordPiece

WordPiece, as it is used in recent models, was introduced in 2016 by Google ([27]). In 2018 Google used it again to pre-train BERT model [4], followed by other models such as DistilBERT, MobileBERT, Funnel Transformers, and MPNET.

WordPiece training is similar to BPE but differs in key ways. When splitting words into characters, it adds the prefix "##" to characters that follow others within a word. Instead of merging the most common token pairs, WordPiece prioritizes pairs that maximize a likelihood-based score, focusing on merging pairs where individual components are less frequent in the vocabulary. Unlike BPE which builds merging rules, WordPiece directly constructs a vocabulary. For encoding, WordPiece uses a greedy longest-match approach to split words according to the longest subwords found in its learned vocabulary.

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 [17] 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 challenge in training these models is the big 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 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 [20], 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 $x_i = (x_{i,1}, \ldots, x_{i,m}) \in \mathbb{R}^m$ 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:

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

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:

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

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 [8]. 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}.

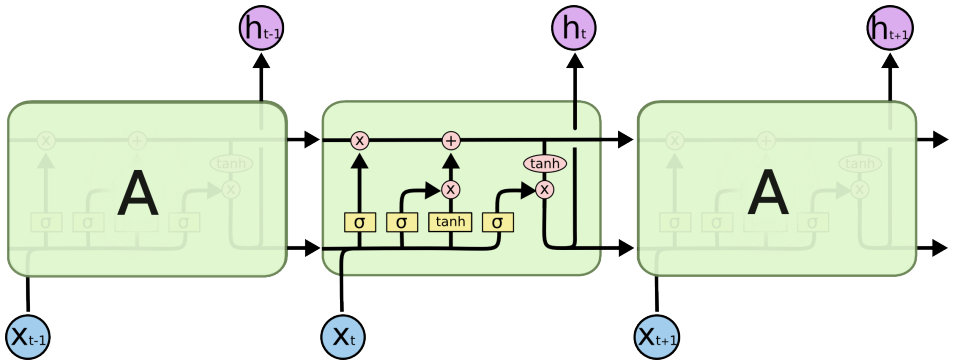

Figure 3.2: The structure of an LSTM cell, showing the cell state and three gates.

- **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 [1]. 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 [23], 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 famous paper "Attention Is All You Need," [25] proposed the **Transformer**, a non recurrent architecture that relies only 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.

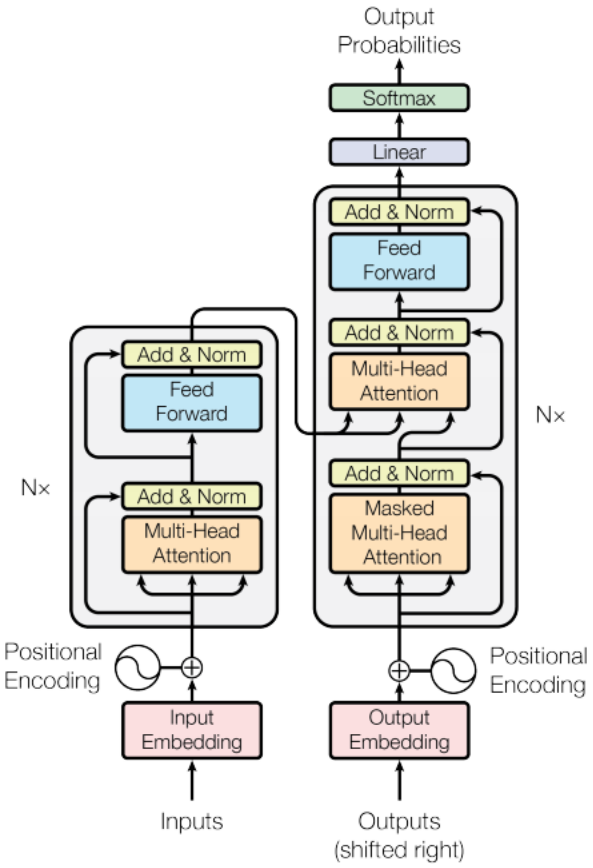

Figure 3.3: The Transformer architecture, highlighting the Multi-Head Attention and Encoder-Decoder stacks.

The architecture is built from stacks of encoders and decoders. Since there is no recurrence, the model has no fundamental 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 hardware than LSTMs, allowing the development of models pre-trained on a huge amount of data.

3.4.6 BERT

BERT, or **Bidirectional Encoder Representations from Transformers**, represents a shift in how deep learning is applied to NLP [4]. 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 provoked 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 [13] 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 [11] and BioMedBERT [6] 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 3.1 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 summary in Table 6.1 quantifies these trends: the BCE ensemble achieves the highest ROC-AUC, AP, and F1 with modest variance, while the strongest single models (bert-base, RoBERTa) are close behind. This aligns with the title-inclusive BCE advantage indicated by the critical difference analysis.

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 strength 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 [9] 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 [14] 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 [10].

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 :

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

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 (HPO) is a crucial step in machine learning that involves systematically searching for the best configuration of hyperparameters—settings that control the learning process but are not learned from data. Examples include learning rates, batch sizes, and regularization parameters.

Modern deep learning models, particularly transformers, have numerous hyperparameters that significantly impact performance. Rather than relying on default values or manual tuning, automated HPO methods like random search [2], grid search, or Bayesian optimization can efficiently explore the hyperparameter space to find optimal configurations. This approach is especially important when comparing different models, as it ensures fair evaluation by giving each model its best possible configuration.

The choice of optimization metric for HPO is essential and reflect the final evaluation objective. For ranking applications like ours, threshold-independent metrics such as ROC-AUC are particularly appropriate as they measure the model's ability to rank examples correctly across all possible classification thresholds.

3.5.5 Statistical Test

In 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 defining a null hypothesis, H_0, which typically claims 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 [5], 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 [19] 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 [16] 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.

The process begins with the raw dataset, which is then prepared and split into 5 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 (BioMoQA) or weighted AP (IPBES, see below). 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.

Chapter 4

Methods

4.1 Overview

Our experimental workflow is designed to ensure a robust and fair comparison of different modeling approaches. The entire pipeline, illustrated in Figure 4.1, is automated by a shell script that orchestrates data preparation, model training, hyperparameter optimization, and evaluation.

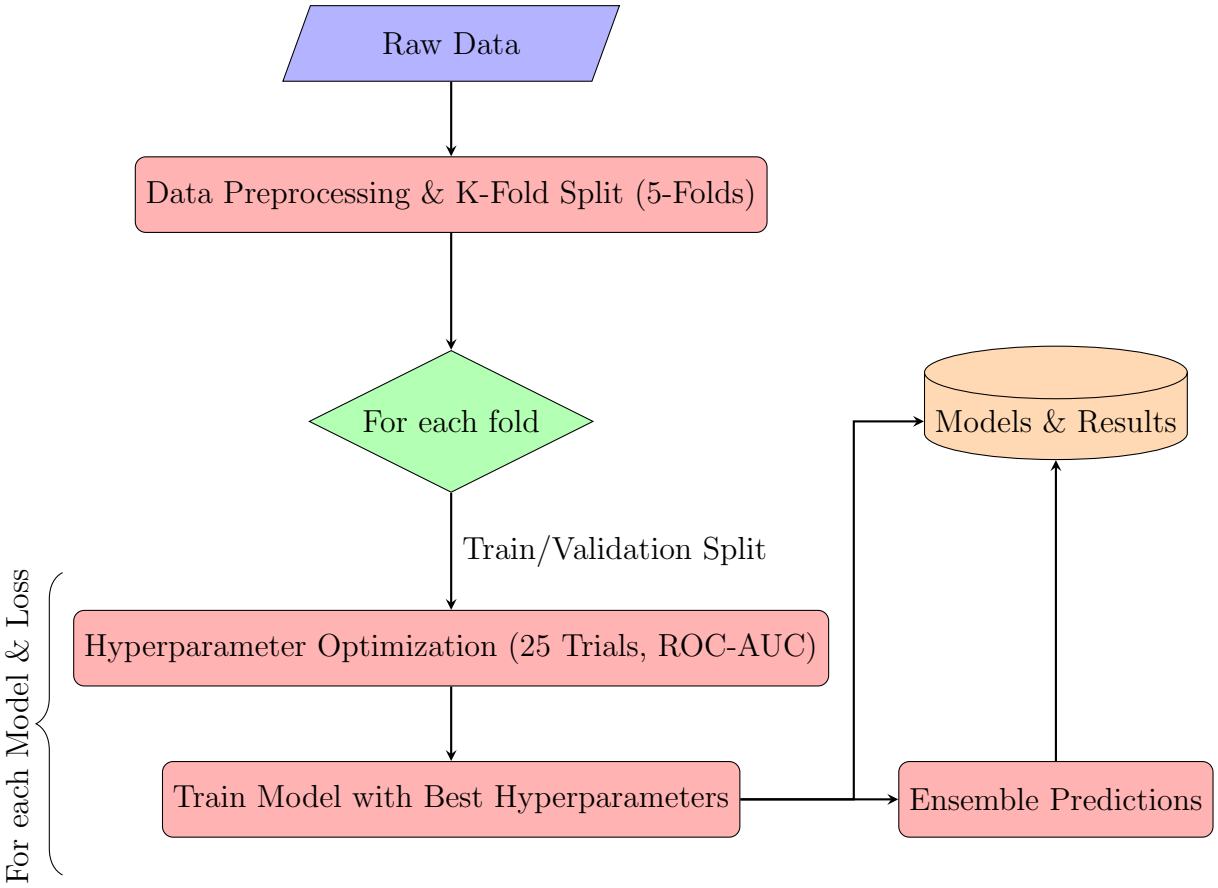

Figure 4.1: Experimental workflow for model training and evaluation.

The process begins with the raw dataset, which is then prepared and split into 5

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 a chosen 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.

4.2 Hyperparameter Optimization

Hyperparameter optimization aims to optimize model performance and ensure fair comparisons between architectures. Rather than relying on default settings, we systematically search for the best combination of hyperparameters for each model and loss function combination.

Common protocol

We employ random search with 25 trials per model. The 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 Focal Loss variants (when used), we also optimize gamma (1.0 to 3.0) and alpha (0.1 to 0.5). Early stopping monitors the development metric with a patience of 3 epochs. After HPO, we train the final model on the train+dev split for that fold and optimize an operating threshold for F1 on the dev part for reporting thresholded metrics.

BioMoQA

For the binary BioMoQA task, HPO optimizes ROC-AUC on the development set. This aligns with the ranking objective of surfacing relevant island biodiversity abstracts at the top of results. We additionally record F1, precision, recall, MCC, and AP.

IPBES

For the multi-label IPBES task, HPO optimizes weighted-average Average Precision. This choice reflects the need to reward ranking relevant papers early in the list and to penalize missing relevant papers.

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

4.3 Dataset Management

This section summarizes dataset handling for both tasks to avoid repetition. We highlight only task-specific aspects beyond the common setup described earlier (stratified 5-fold CV, fixed seeds, strict train/dev/test separation per fold).

4.3.1 Description (BioMoQA)

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 1434 manually curated documents: 996 positives (relevant to island biodiversity) and 438 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) to avoid any bias caused by the remembering of some models trained on scientific literature posterior to 2021, 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 500 synthetic negatives were sampled from the query results and added to the training data, bringing the total corpus to 1934 documents. Importantly, all testing and evaluation is performed exclusively on the original 1434 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.

4.3.2 Pre-Processing (BioMoQA)

The data preprocessing pipeline, implemented in `preprocess_biomoqa.py` in the code, focuses on data cleaning, integration, and cross-validation setup rather than text nor-

malization, which is handled by the transformer tokenizers themselves.

The preprocessing workflow consists of several key steps:

- **Data Cleaning**: The pipeline removes duplicates based on title, abstract, and DOI fields while handling conflicts where identical content has different labels. Missing values in essential fields (title, abstract, keywords, labels) result in row removal to ensure data quality.
- **Synthetic Negative Integration**: The 500 synthetic negatives retrieved from PubMed are combined with the original 1434 manually curated documents. These synthetic negatives are labeled with -1 initially to distinguish them from original negatives (labeled 0) during the splitting process.
- **Stratified Cross-Validation Setup**: A 5-fold stratified cross-validation is implemented exclusively on the original 1434 documents to ensure representative class distributions across folds. The stratification preserves the proportion of relevant vs. irrelevant documents in each fold.
- **Train/Dev/Test Splitting**: For each fold, the original data is split into training (˜56%), development (˜19%), and test (˜25%) sets using stratified sampling. Crucially, all 500 synthetic negatives are added only to the training sets, ensuring that evaluation metrics are computed solely on manually curated data.
- **Reproducibility Controls**: All random processes use fixed seeds derived from a master seed to ensure reproducible fold assignments and data splits across experimental runs.

The pipeline also validates fold integrity by ensuring no overlap between test sets across folds and converts all synthetic negative labels from -1 to 0 after the splitting process. This approach maintains clear separation between training and evaluation data while maximizing the use of available synthetic negatives for model training.

4.3.3 Description (IPBES)

The IPBES corpus consists of policy- and assessment-oriented biodiversity literature annotated with three non-exclusive labels: IAS (Invasive Alien Species), SUA (Sustainable Use Assessment), and VA (Values Assessment). Documents may bear zero, one, or multiple labels. We apply the same stratified 5-fold CV protocol, preserving label distributions across folds and ensuring no leakage between folds. Title and abstract are concatenated as model input when available.

4.3.4 Pre-Processing (IPBES)

Pre-processing mirrors BioMoQA: deduplication, integrity checks, fold assignment with fixed seeds, and generation of train/dev/test splits per fold. Because labels are multi-

hot, we compute and log both per-label supports and overall label cardinality per split to monitor balance.

4.3.5 Handling Imbalanced Data

As noted, as we add more an more synthetic negatives isntance our dataset become 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 [12]. 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.

4.4 Architecture

4.4.1 Common encoder and training loop

Our models are based on Transformer encoders (Table 3.1) from Hugging Face. Fine-tuning uses AdamW, linear warmup/decay, and early stopping on the development metric. Inputs concatenate title and abstract when available.

4.4.2 BioMoQA: binary head and losses

For BioMoQA, we add a single-logit classification head on the ‘[CLS]‘ representation with dropout regularization. We evaluate BCE and Focal Loss; reporting includes ROC-AUC, AP, F1, precision/recall, and MCC.

4.4.3 IPBES: multi-label head and losses

For IPBES, the head outputs one logit per label (IAS, SUA, VA) with sigmoid activations; training uses per-label BCE with logits.

Baselines. We also trained SVMs and Random Forests with TF-IDF features as a non-neural baseline for comparison in BioMoQA.

Chapter 5

Implementation

This chapter outlines the code organization and execution flow. We summarize modules for data processing, model training/evaluation, orchestration, and environment setup. The IPBES multi-label pipeline reuses the same components as BioMoQA, with a multi-label classification head and weighted-average metrics.

5.1 Reproducibility and Code Availability

The complete implementation is available in the public GitHub repository: `https://github.com/CTGN/BioMoQA-Classifier.git`. The repository includes all source code, configuration files and shell scripts need to reproduce the results.

All experiments are fully reproducible through the use of fixed random seeds and deterministic algorithms. The `uv` package manager is used to ensure consistent dependency versions across different environments. Detailed installation and execution instructions will be provided in the repository's README file.

The modular design of the codebase facilitates extension to other domains or datasets, making it a valuable resource for future research in scientific literature classification and ranking applications.

For the IPBES model, use the same `uv` environment and orchestration. Its implementation is available in another public GitHub repository : `https://github.com/CTGN/IPBES-Classifier.git`

Chapter 6

Results

In this chapter, we present the empirical results of our experiments. We compare the performance of the different BERT models and the SVM and Random Forest baselines across the two loss functions, using the metrics established in the evaluation chapter. All results are reported as the average over the 5 folds of the cross-validation.

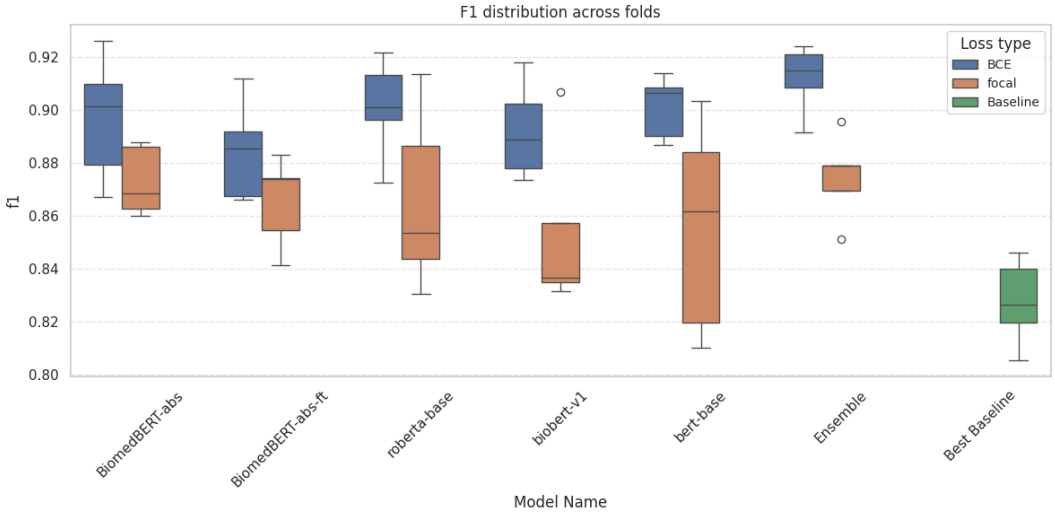

Figure 6.1: Distribution of F1 scores for BioMoQA across folds and models.

Figure 6.1 summarizes the F1 performance for all models (with titles). Several trends are apparent. First, all Transformer models outperform the best of baselines. Second, the ensemble performs clearly better than any other stand alone model as expected. Third the RoBERTa and BERT-base model seems to be the strongest standalone performers, while domain-specific models remain competitive. Finally, the BCE loss function performs consistently better than the loss function with less variance.

Figure 6.2 shows the distribution of ROC-AUC across folds and models. Variability across folds remains moderate. Overall, we can do the same observations as above.

On figure 6.3 what is interesting to see is that BioBERT, this time, seems to perform slightly better accross thresholds.

To assess whether the observed differences among model variants exceed random variation, we compared 10 variants (5 models $\times$ 2 loss functions). For each of the 5 cross-validation folds, we ranked the 10 variants by ROC-AUC on that fold's test split,

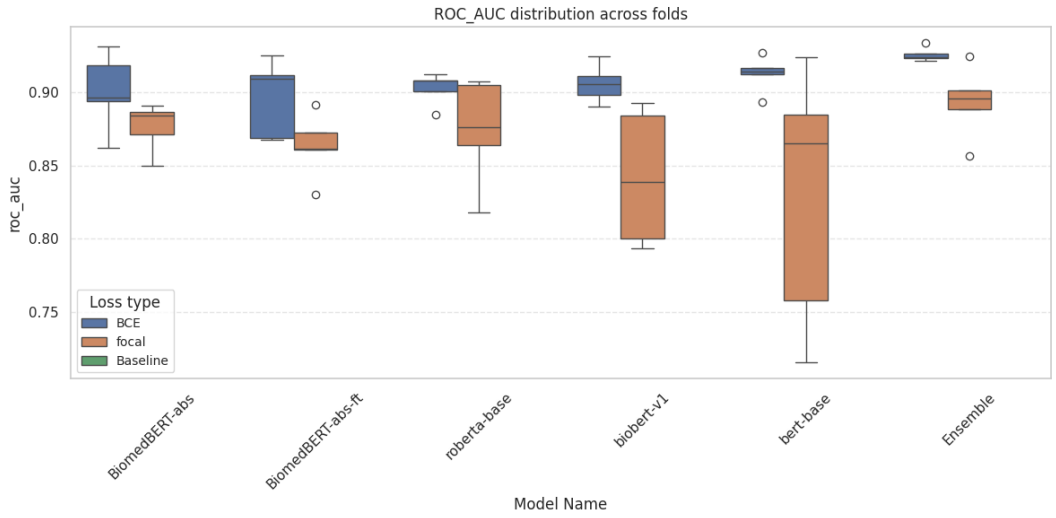

Figure 6.2: Distribution of ROC-AUC scores across folds and models for BioMoQA.

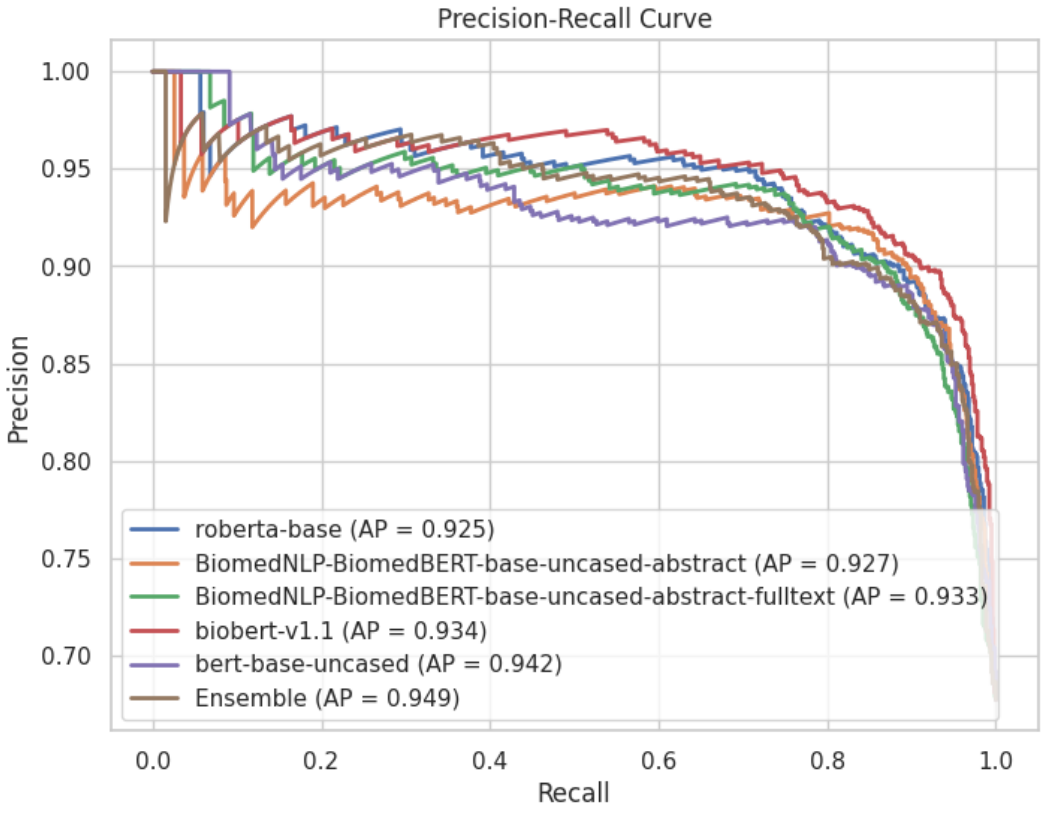

Figure 6.3: Precision–Recall behavior for BioMoQA across thresholds and models.

and then applied the Friedman test treating folds as blocks and variants as treatments. This yielded $p < 0.005$ for the null of equal average ranks. Because cross-validation folds

come from the same underlying dataset and are not strictly independent, we treat this result as indicative rather than definitive; see Section 7.2.4. We therefore emphasize effect sizes and the critical difference diagram (Figure 6.4), using Nemenyi post-hoc comparisons primarily as a descriptive aid.

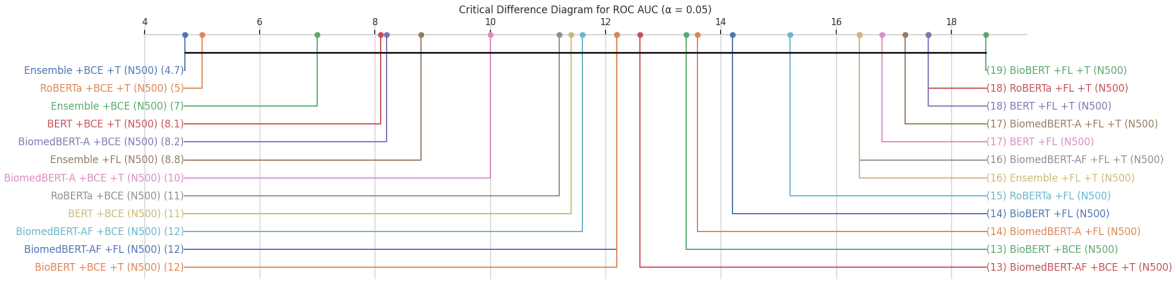

Figure 6.4: Critical difference diagram for BioMoQA ROC-AUC (alpha=0.05).

The diagram suggests that including the title with BCE loss performs better on average. Within the above caveats, RoBERTa appears to be the strongest standalone model, followed by BERT-base. A plausible explanation is RoBERTa's optimized pre-training (longer training, dynamic masking, larger batches) yielding stronger sentence-level representations that pair well with short titles; domain-specific gains may be muted here because island biodiversity relies on broad language and geographical cues rather than biomedical jargon.

Table 6.1 quantifies these trends: the BCE ensemble achieves the highest ROC-AUC, AP, and F1 with modest variance, while the strongest single backbones (bert-base, RoBERTa) are close behind. This aligns with the title-inclusive BCE advantage indicated by the critical difference analysis.

Table 6.1: BioMoQA: mean ± SD across 5 folds (BCE, with title).

Model	ROC-AUC	AP	F1	Accuracy	MCC
BiomedBERT-abs	0.900 ± 0.026	0.940 ± 0.023	0.897 ± 0.024	0.856 ± 0.035	0.661 ± 0.085
BiomedBERT-abs-ft	0.896 ± 0.027	0.935 ± 0.024	0.884 ± 0.019	0.840 ± 0.027	0.627 ± 0.065
biobert-v1	0.906 ± 0.013	0.943 ± 0.013	0.892 ± 0.018	0.849 ± 0.025	0.645 ± 0.060
bert-base	0.913 ± 0.012	0.949 ± 0.008	0.901 ± 0.012	0.862 ± 0.019	0.676 ± 0.049
roberta-base	0.903 ± 0.011	0.938 ± 0.006	0.901 ± 0.019	0.865 ± 0.025	0.691 ± 0.058
Ensemble	0.926 ± 0.005	0.956 ± 0.007	0.912 ± 0.013	0.877 ± 0.021	0.714 ± 0.052

6.1 IPBES Results

Figure 6.5 shows that all BERT models are competitive, but with the BCE-based ensemble being particularly and surprisingly low. What is very clear tho, is that the baseline performs very poorly compared to the BERT based models.

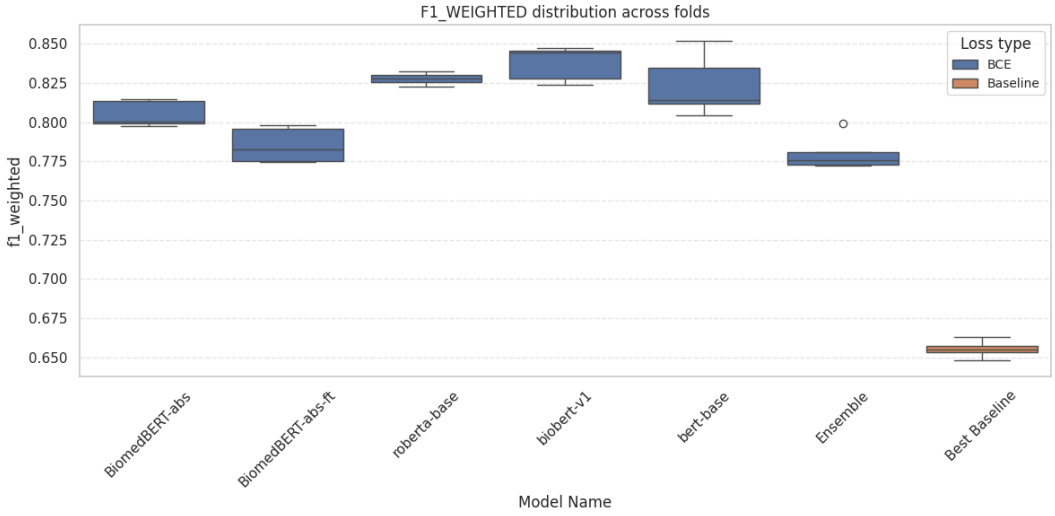

Figure 6.5: Distribution of weighted-average F1 for IPBES across folds and models.

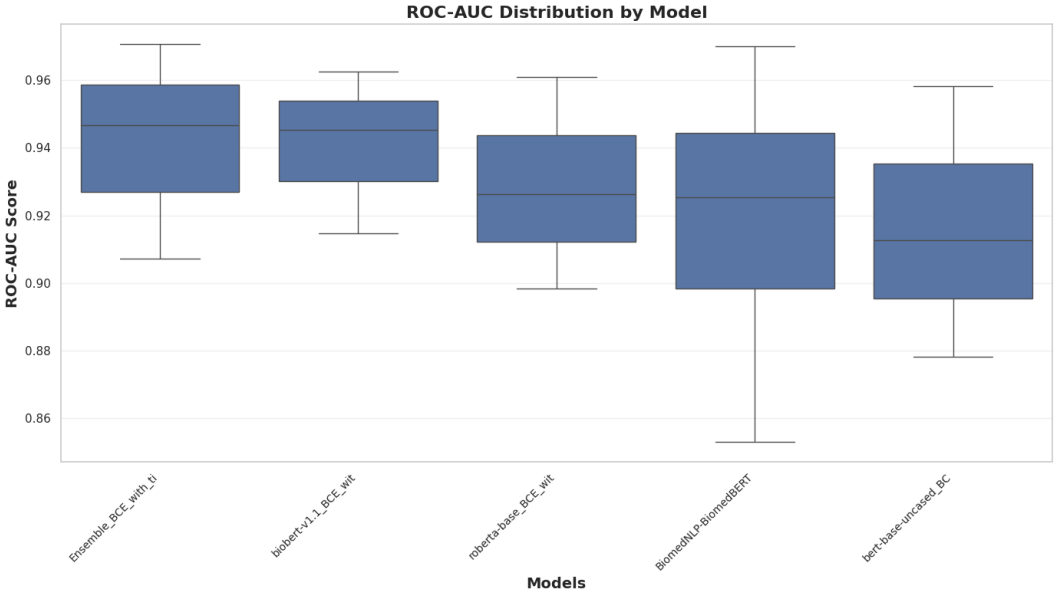

Figure 6.6: Distribution of weighted-average ROC-AUC across folds and models for IPBES.

Weighted ROC-AUC distributions in Figure 6.6 are high and tight for the best models, indicating stable ranking quality across thresholds. The ensemble consistently leads, while differences between the top single models are small, since the variance is notable.

Figure 6.7 highlights the precision–recall trade-off for all threshold and for each model. What strengthen what we saw earlier is that the ensemble learning is again performing well along the different thresholds. BioBERT this time seems better than

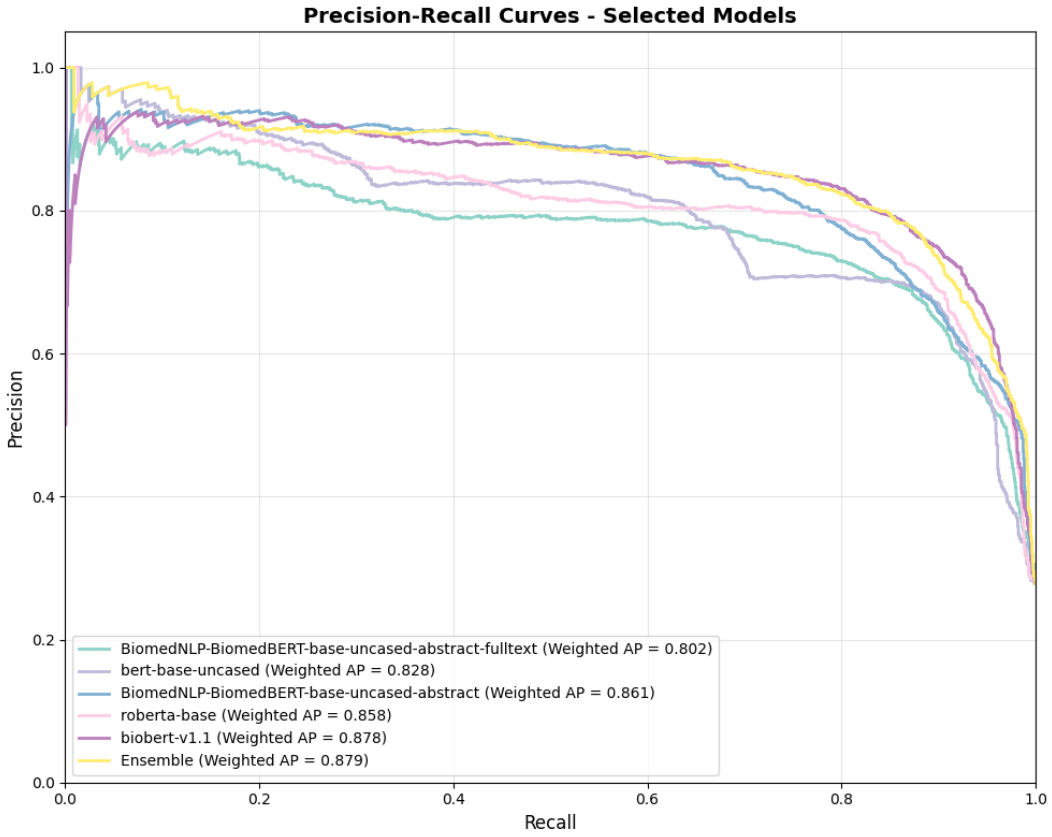

Figure 6.7: Weighted-average precision–recall behavior for IPBES across models. Note the relative difficulty of SUA compared to IAS and VA.

the other standalone models.

For IAS (Figure 6.8), the ROC curve approaches the top-left corner, consistent with strong separability (mean ROC-AUC ≈ 0.97) and high AP, indicating reliable detection of IAS-relevant documents.

For SUA (Figure 6.9), the curve is farther from the top-left than IAS/VA, reflecting the lower separability (mean ROC-AUC ≈ 0.91) and lower AP. This corroborates that SUA is the hardest axis and benefits most from ensembling and recall-oriented thresholding.

For VA (Figure 6.10), performance sits between IAS and SUA (mean ROC-AUC $\approx$ 0.95; AP ≈ 0.87), indicating good rankability with moderate headroom for improvement compared to IAS.

Quantitatively, the BCE ensemble achieves mean macro ROC-AUC of 0.943 (SD ˜0.003) and mean macro AP of 0.886 (SD ˜0.004) across folds. Per-label means for the ensemble are: IAS ROC-AUC 0.971 and AP 0.972; SUA ROC-AUC 0.907 and AP 0.818; VA ROC-AUC 0.948 and AP 0.867. The spread is narrow for IAS/VA and wider for SUA (AP spanning roughly 0.807–0.836), indicating SUA as the hardest axis.

Table 6.2 corroborates the preceding figures: the BCE ensemble attains the highest

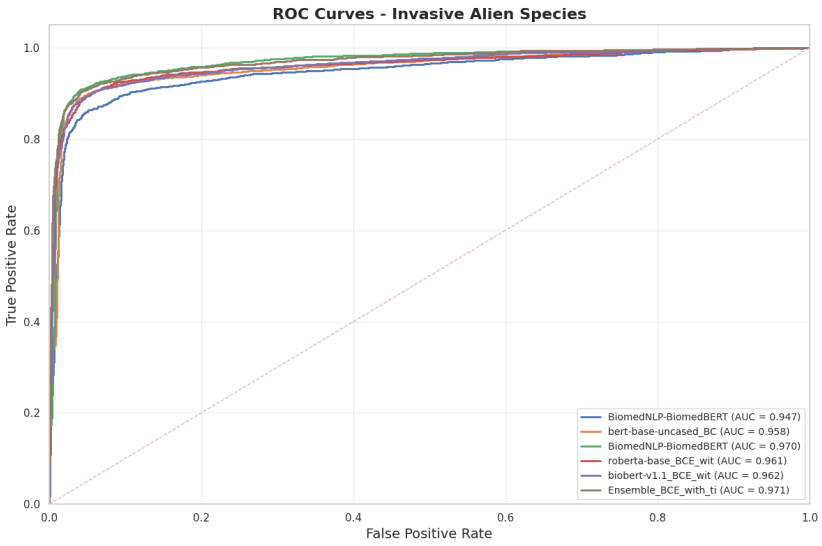

Figure 6.8: Per-label ROC curve: IAS.

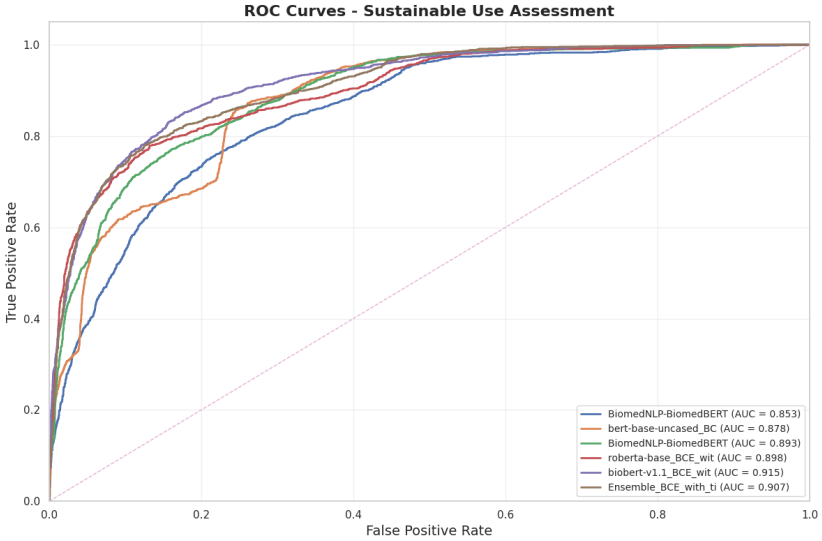

Figure 6.9: Per-label ROC curve: SUA.

macro ROC-AUC and AP with tight standard deviations, while its weighted F1 remains below the top single models (e.g., BioBERT, RoBERTa). This pattern matches our ranking-oriented objective and the earlier observation that the ensemble's F1 is comparatively lower despite superior ranking metrics.

Figure 6.11 visually confirms the modest but consistent advantages of BioBERT and the ensemble; most pairwise gaps remain below the significance threshold at $\alpha = 0.05$.

What we can conclude from this results for the IPBES dataset is that our expections are confirmed : the ensemble model performed the best while the best standalone model

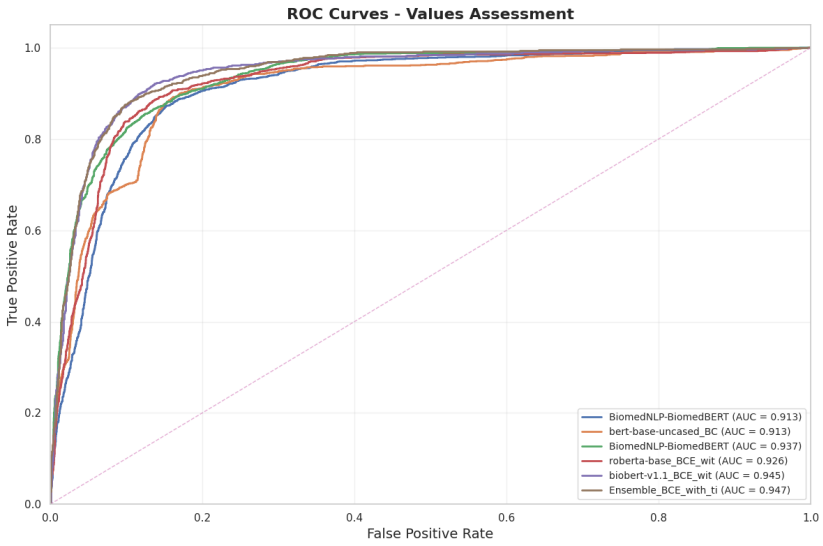

Figure 6.10: Per-label ROC curve: VA.

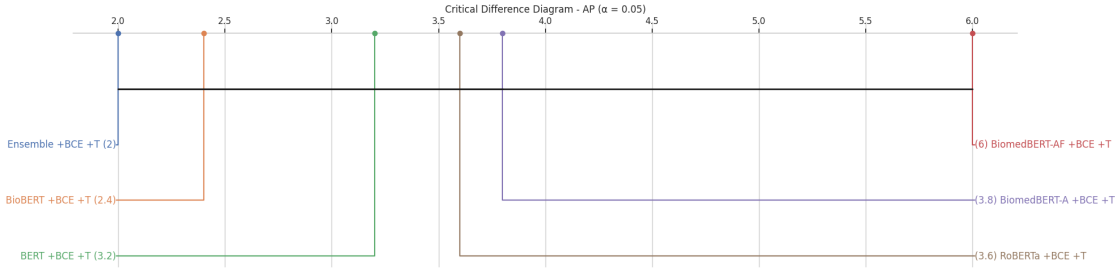

Figure 6.11: Critical difference diagram for IPBES Average Precision (alpha=0.05). Only BCE-loss models with title are considered.

Table 6.2: IPBES: mean ± SD across 5 folds (BCE, with title).

Model	ROC-AUC (macro)	AP (macro)	F1 (weighted)
BiomedBERT-abs	0.933 ± 0.002	0.863 ± 0.009	0.805 ± 0.008
BiomedBERT-abs-ft	0.917 ± 0.007	0.834 ± 0.010	0.785 ± 0.011
biobert-v1	0.942 ± 0.008	0.883 ± 0.016	0.838 ± 0.011
bert-base	0.931 ± 0.013	0.866 ± 0.023	0.824 ± 0.019
roberta-base	0.934 ± 0.004	0.872 ± 0.009	0.828 ± 0.004
Ensemble	0.942 ± 0.003	0.885 ± 0.005	0.780 ± 0.011

performance is made by a domain specific model (BioBER), suggesting that the model was able to grasp more meaning thanks to a specfic scientfic understanding.

Chapter 7

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 ROC-AUC score improvement of approximately 1-2 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 Figure 6.1, 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, suggesting that for information retrieval tasks, the balanced approach of BCE is preferable.

7.1.3 Impact of Synthetic Negatives

The addition of 500 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 (996 positives vs. 438 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.

7.2 Limitations

7.2.1 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 [4, 11]. 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 pre-training set having a really high-volume of data, we can assume that this contamination is a minor one.

7.2.2 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.

7.2.3 Computational Resources

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

7.2.4 Statistical Testing Assumptions

An important methodological limitation concerns the independence assumptions underlying our statistical tests. The Friedman test and subsequent Nemenyi post-hoc analysis assume that the performance measurements across folds are independent observations. However, in cross-validation, the folds are not truly independent as they are derived from the same underlying dataset and may contain related documents (e.g., papers from the same authors, research groups, or on highly similar topics).

This violation of the independence assumption could lead to inflated confidence in the statistical significance of observed differences between models. A more rigorous approach would require techniques specifically designed for dependent samples or larger-scale experiments with truly independent datasets. However, the effect sizes observed between model categories (e.g., domain-specific vs. general-purpose) suggest that the main conclusions remain valid despite this limitation.

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 developed BERT-based classifiers for two complementary literature-triage tasks and evaluated them under rigorous 5-fold cross-validation, ranking-oriented metrics, and statistical comparisons. For BioMoQA, the BCE ensemble with titles achieved strong ranking performance with mean ROC-AUC of 0.926 (SD ˜0.004), AP of 0.956 (SD ˜0.006), and F1 of 0.912 (SD ˜0.012), alongside accuracy 0.877, precision 0.890, recall 0.936, and MCC 0.714. These results confirm that transformer ensembles provide robust separation between relevant and irrelevant island-biodiversity abstracts.

For IPBES, the BCE ensemble attained a mean macro ROC-AUC of 0.942 (SD ˜0.003), mean macro AP of 0.885 (SD ˜0.004), and weighted F1 of 0.780 (SD ˜0.010) across folds, with clear per-label differences: IAS is easiest, VA intermediate, and SUA the hardest axis (lower AP and ROC-AUC). These findings validate the feasibility of reliable, large-scale multi-label tagging for assessment-relevant themes.

Across both tasks, BCE-trained BERT models and their ensembles consistently outperformed classical baselines. Domain-specific models were competitive, while general-purpose models also performed strongly, suggesting that broad language understanding complements specialized vocabulary for these applications. The addition of carefully curated synthetic negatives in BioMoQA contributed to improved robustness under class imbalance.

In the coming weeks, both models will be integrated into SIBiLS to deliver ranked, explainable literature triage to end users. This deployment will make the systems' capabilities broadly accessible within an open-science ecosystem and support faster, more consistent evidence discovery for biodiversity and policy assessments.

Bibliography

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[2] James Bergstra and Yoshua Bengio. "Random search for hyper-parameter optimization". In: *Journal of machine learning research* 13.2 (2012), pp. 281–305.

[3] Aaron M Cohen and William R Hersh. "An overview of research and evaluation in biomedical informatics". In: *Journal of biomedical informatics* 43.5 (2010), pp. 639–646.

[4] Jacob Devlin et al. "BERT: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[5] Milton Friedman. "A comparison of alternative tests of significance for the problem of m rankings". In: *The Annals of Mathematical Statistics* 11.1 (1940), pp. 86–92.

[6] Yu Gu et al. "Domain-specific language model pretraining for biomedical natural language processing". In: *ACM Transactions on Computing for Healthcare* 3.1 (2021), pp. 1–23.

[7] Haibo He and Edwardo A Garcia. "Learning from imbalanced data". In: *IEEE Transactions on knowledge and data engineering* 21.9 (2009), pp. 1263–1284.

[8] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[9] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[10] Ron Kohavi et al. "A study of cross-validation and bootstrap for accuracy estimation and model selection". In: 14.2 (1995), pp. 1137–1145.

[11] Jinhyuk Lee et al. "BioBERT: a pre-trained biomedical language representation model for biomedical text mining". In: *Bioinformatics* 36.4 (2020), pp. 1234–1240.

[12] Tsung-Yi Lin et al. "Focal loss for dense object detection". In: *arXiv preprint arXiv:1708.02002* (2017).

[13] Yinhan Liu et al. "RoBERTa: A robustly optimized BERT pretraining approach". In: *arXiv preprint arXiv:1907.11692* (2019).

[14] Ilya Loshchilov and Frank Hutter. "Decoupled weight decay regularization". In: *arXiv preprint arXiv:1711.05101* (2017).

[15] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge university press, 2008.

[16] Quinn McNemar. "Note on the sampling error of the difference between correlated proportions or percentages". In: *Psychometrika* 12.2 (1947), pp. 153–157.

[17] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[18] National Institutes of Health. "Statistics on scientific publications". In: (2024).

[19] Peter Nemenyi. "Distribution-free multiple comparisons". In: (1963).

[20] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global vectors for word representation". In: (2014), pp. 1532–1543.

[21] Fabrizio Sebastiani. "Machine learning in automated text categorization". In: *ACM computing surveys* 34.1 (2002), pp. 1–47.

[22] Rico Sennrich, Barry Haddow, and Alexandra Birch. "Neural machine translation of rare words with subword units". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2015, pp. 1715–1725.

[23] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems* 27 (2014).

[24] Anne E Thessen, Hong Cui, and Dmitry Mozzherin. "Applications of natural language processing in biodiversity science". In: *Advances in bioinformatics* 2012 (2012).

[25] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[26] David Westergaard et al. "A comprehensive and quantitative comparison of text-mining in 15 million full-text articles versus their corresponding abstracts". In: *PLoS computational biology* 14.2 (2018), e1005962.

[27] Yonghui Wu et al. "Google's neural machine translation system: Bridging the gap between human and machine translation". In: *arXiv preprint arXiv:1609.08144* (2016).