EMODnet Biology

EMFF/2019/1.3.1.9/Lot 6/SI2.837974

EMODnet Phase IV

D 2.5: Feasibility Study on Ecological Traits and sampling devices/methodologies identification with text mining

The European Marine Observation and Data Network (EMODnet) is financed by the European Union under Regulation (EU) No 508/2014 of the European Parliament and of the Council of 15 May 2014 on the European Maritime and Fisheries Fund.

Disclaimer

Document info

Title	D2.5 Feasibility study for recognition of specific ecological traits and/or sampling devices/methodologies in text.
WP title	WP2: Access to marine biological data
Tasks	Task 1: Maintain and improve a common method of access to data held in repositories
Authors	on behalf of Advance Services Nikos Minadakis, Lydia Kalaitzaki, Melina Loulakaki revision by Dimitra Mavraki (HCMR), Savvas Paragkamian (HCMR), Vasilis Gerovasileiou (HCMR), Joana Beja (VLIZ)
Dissemination level	Public
Submission date	12/04/2023
Deliverable due date	25/02/2023

Contents

1. Preface

The study was assigned to Advance Services (https://www.advancesvs.com/) through a sub-contract signed in September 2022 with a duration of six months. The work was performed in close collaboration with the HCMR team (Ms. Dimitra Mavraki, Mr. Savvas Paragkamian, Dr. Vasilis Gerovasileiou). Dr. Evangelos Pafilis (Bioinformatics and Biodiversity Informatics Researcher C, HCMR) also supported this work by proposing appropriate tools for text mining, by giving an overview of the status of text mining tools and related bibliography, and by setting the starting structure of the feasibility study.

During the period of six months, several meetings in person were organised -at least once per month-; while for more immediate response to any problems or queries that occurred, several video calls took place. Mr. Savvas Paragkamian (PhD student on Microbial Ecology), who has experience in text mining and on the curation process of historical literature (Paragkamian et al., 2022), closely monitored the work progress, the development and implementation of the text mining workflow on traits, and co-organised with Advance Services the online and free training session (see section 6). On the other hand, Ms. Dimitra Mavraki, as Data manager in HCMR, tried to combine both the technical and practical parts of the work. Although the nature of this study was technical (e.g., source codes, tools), emphasis was also given towards the ultimate target of assisting the work of data curators through an easy-to-use tool for recognising ecological traits and devices/methodologies on the working papers/pieces of text.

The process of selecting Advance Services started with a call published on the 27th of June 2022, in which the EMODnet Biology IV WP2 Leaders expressed their interest in external collaborators who might be interested in facilitating a feasibility study on Ecological Traits and sampling devices/methodologies identification with text mining. A detailed description of the project was provided along with the call. The call was also broadcasted in a community of trait-based researchers via https://github.com/open-traits-network/open-traits-network.github.io/issues/178. The expression of interest was open until the 31st of July. On the 26th of August, an interview with the only applicant took place, during which Advance Services was selected as a qualified collaborator.

Advance Services is a high-tech enterprise that is active in designing and implementing advanced software solutions, ETL (Extract, Transform and Load) processes and semantic big-databases. It was founded in 2018 in Heraklion, Crete and has already been established as one of the innovators in its field. The Advance Services team includes Computer Scientists and Software Engineers with more than 10 years' experience in large scale infrastructure projects supported by either private, European, or national funding. Team members have strong expertise in designing and implementing high quality integrated information systems and applications. Furthermore, the scientific background of the team is justified by several publications (>50) in international conferences and journals.

2. Introduction

This report is related to D2.5 entitled "Feasibility study for recognition of specific ecological traits and/or sampling devices/methodologies in text". The main aim of the study was to assess the accuracy of text mining prototypes regarding these entities; advise on next actions; examine whether further development and fine tuning should be sought. More specifically, it focused on ecological traits, life stages, and body length extraction with text mining and was built upon previous work on traits and text mining of EMODnet Biology.

Traits refer to the characteristics of a species, including its morphology, anatomy, physiology, biochemistry, and phenology. Scientists use traits to describe the characteristics of the taxa they study, and they are closely related to Essential Biodiversity Variables (EBVs) and ecosystem function. However, the meaning of traits is often inconsistent in literature due to their vast diversity and long history. To address this issue, trait databases have been created for many taxonomic groups, and standard vocabularies have been established to standardise trait terms. The Open Traits network has also been launched to promote open science standards and practices in the field.

Text mining tools can assist in curating and discovering knowledge from multiple documents, but the training and evaluation of these tools rely on curated and annotated documents (corpora) and dictionaries (e.g., controlled vocabularies, ontological term and/or database entry names and synonyms).

One example of successful application of text mining tools is the recognition of species names in text. Further research is needed to examine which traits are feasible to mine from text and which methods are most promising for the task.

The process of text mining involves several activities that allow the user to derive information from unstructured text data. Before the user can apply various text mining techniques, they must begin with text pre-processing, i.e., cleaning and converting text data into a usable format. This practice is a core aspect of natural language processing (NLP) and typically involves techniques such as language identification, tokenization, part of speech tagging, chunking, and syntax parsing to appropriately format the data for analysis (IBM. (n.d.). Text mining).

The steps in determining whether it is possible to extract ecological species traits, from marine biodiversity documents, are the following: a. vocabulary establishment, b. corpus establishment, c. corpus annotation, d. data format preparation, e. model training for NER (Named Entity Recognition) and f. model results evaluation.

In text mining, a corpus refers to a large and structured set of pieces of text that are collected and used for linguistic analysis and computational processing. A text corpus can be thought of as a database of natural language texts that are typically stored in electronic form and can be accessed and analysed using software tools. Text corpora are often used in natural language processing (NLP) and machine learning applications to train and evaluate models for various text-related tasks, such as sentiment analysis, named entity recognition, and text classification (Manning et al., 1999).

An annotation tool is a software application that allows users to add or edit annotations or notes to text, images, or other digital content. These annotations can be used for a variety of purposes, including annotating, marking content for revision, or highlighting important information, and training custom NER models as required by this study. Using annotation tools curators can annotate corpora in order to train their NER models (Asgari et al., 2021).

Named Entity Recognition (NER) is a sub-task of NLP that involves identifying and categorising entities in text into predefined categories such as species names, people's names, organisations, places, and others. The goal of NER is to extract structured information from unstructured text so that it can be more easily analysed and understood. NER can be used in various applications such as information extraction, question answering systems, and text classification (Jurafsky et al., 2020).

NER is facilitated by a document tagger which is a software tool that automatically annotates or categorises text documents according to certain criteria, such as topics, sentiment, entities, etc. Document tagging involves

assigning one or more tags or labels to a document that describe its content, purpose or structure. This process helps to organise and categorise documents, so that they are easier to search, retrieve and analyse.

This work is built upon previous work of EMODnet Biology, namely EMODnet Phase III D3.7: Scientific document on the design of the workflow of text mining technologies in data archaeology, which identified traits as an underdeveloped entity type in the field of text mining. EMODnet projects on traits include Benthic occurrences, habitat maps, and species traits, EMODnet Biology thermal traits and the recently launched Btrait, which analyse and/or provide the means to analyse trait data.

Structure of the Report

The document starts with a Preface and an Introduction, which provide an overview and background of the research work.

The Methodology section is divided into several subsections, including Steps, Criteria for vocabulary and corpus establishment, Criteria for tool evaluation, Programming Environment, and Reproducibility Code. This section describes the methods, techniques, and tools used in the research work.

The Results - Discussion section contains several subsections, which present and discuss the findings of the research work. The subsections include Vocabulary, Corpus, Annotation, Taggers and NLP, Model training, Model evaluation and performance, Summary, and Other tools for further exploration.

The section on Future work and Further improvements outlines potential areas of research and development for future work. The document concludes with the Online training session report, the References, and an Appendix that contains the installation instructions for the tools used in the study.

3. Methodology

3.1 Steps

During the development and training of the text mining models, the following steps were followed (Figure 1):

1. Vocabulary Establishment
2. Corpus Establishment
3. Tools Selection
4. Corpus Annotation
5. Data format preparation
6. Model Selection
7. Model Training for NER (Named Entity Recognition)
8. Model Evaluation

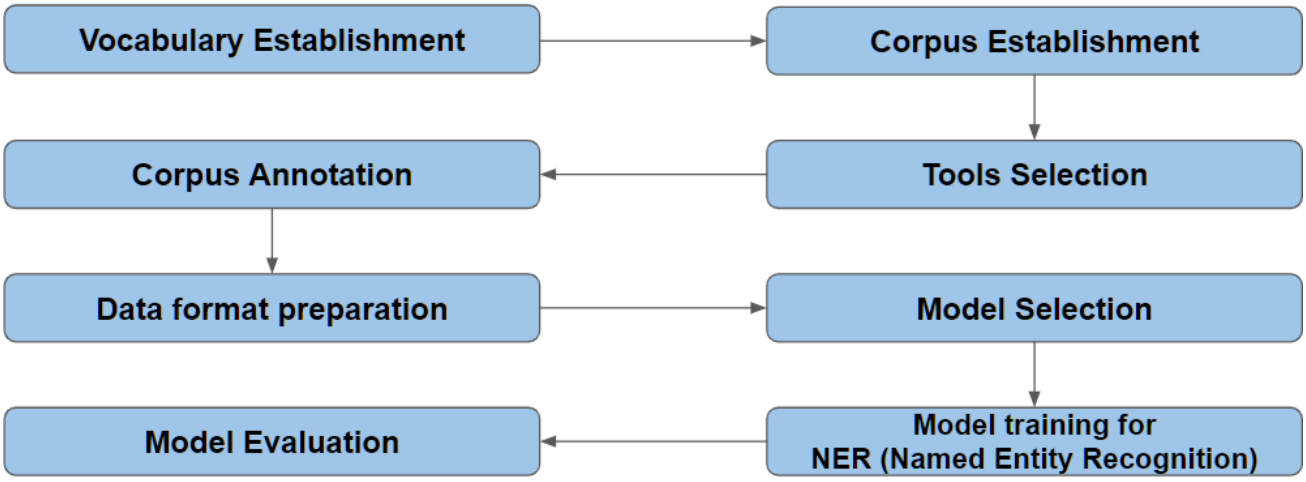


Figure 1. Steps followed for the feasibility study

3.2 Criteria for vocabulary and corpus establishment

The main criteria used to establish the corpus were: a. that the texts included as the corpus had to be scientific articles and b. they had to comprise information about traits of marine species.

In addition, the primary criterion for the establishment of vocabularies was that four dictionaries had to be created, one for each trait category of interest (life stages, distribution descriptors, body length), and one for sampling devices. Furthermore, all entities had to be linked through a descriptor link to marine species traits.

3.3 Criteria for tool evaluation

In Named Entity Recognition (NER) model training, the common evaluation metrics used to measure the performance of the model are the following: a. accuracy, b. precision, and c. recall (Reimers et al., 2020).

a. **Accuracy:**

measures the percentage of correctly predicted tokens in the dataset. It is calculated as the number of correctly predicted tokens divided by the total number of tokens.

b. **Precision:**

measures the proportion of true positive entity predictions out of all predicted entities. It is calculated as the number of true positive entity predictions divided by the sum of true positive and false positive entity predictions.

c. **Recall:**

measures the proportion of true positive entity predictions out of all actual entities in the dataset. It is calculated as the number of true positive entity predictions divided by the sum of true positive and false negative entity predictions.

In general, a good NER model should have high accuracy, precision, and recall values. However, the specific metric(s) that are most important will depend on the particular use case and the associated costs and benefits of different types of prediction errors. For example, in some applications, such as information extraction or named entity linking, recall may be more important than precision, while in others, such as sentiment analysis, precision may be more important than recall. Precision and recall representation are shown in Figure 2.

Precision and Recall are complementary metrics that have an inverse relationship. Therefore, if both are of interest then it is better to use the F1 score to combine precision and recall into a single metric. This score is calculated as the harmonic mean of precision and recall, where a score of 1 indicates perfect precision and recall, and a score of 0 indicates poor performance. The formula for F1 score is:

F1 score = 2 * (precision * recall) / (precision + recall)

F1 score is often used as a summary statistic to compare the overall performance of different models or to optimise hyperparameters of a model. In general, a higher F1 score indicates better performance of the model, and a model with high F1 score means it is able to balance between precision and recall.

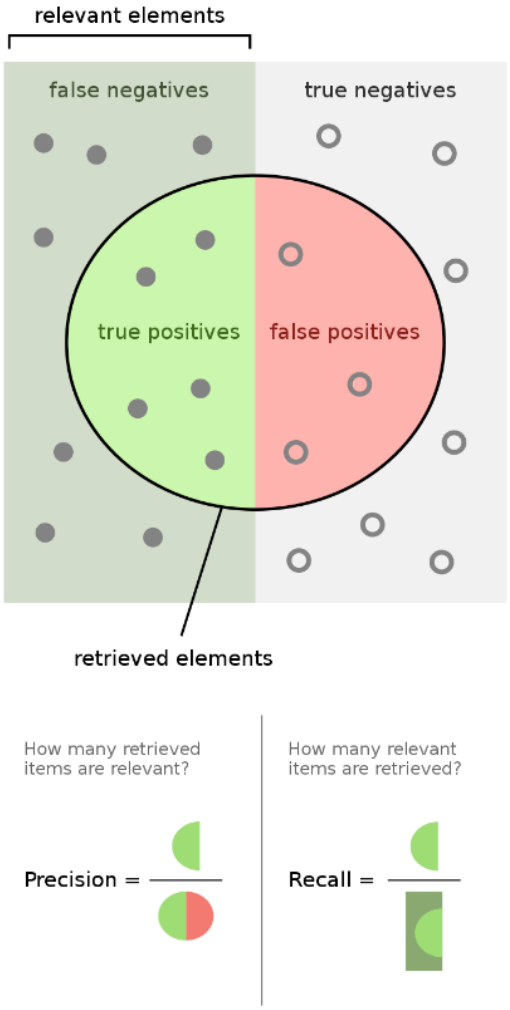


Figure 2. A representation of precision and recall metrics. Precision is calculated as the number of true positive entity predictions divided by the sum of true positive and false positive entity predictions. Recall is calculated as the number of true positive entity predictions divided by the sum of true positive and false negative entity predictions (Walber (2014). Precision and Recall [Graph illustration])

3.4 Programming Environment

In order to train the models, a programming environment called Google Colaboratory is used for notebook production, edit and run processes. Google Colaboratory, known as "Colab", is a data analysis and machine learning application that enables the integration of rich text, charts, photos, executable Python code, and more into a single document stored in Google Drive. It was selected due to its ability to handle notebooks and attach a google drive folder. Such ability gives the convenience of having a folder in a drive account and just connecting it with the notebook in Google Colab, set the path and run the cells without need for example of uploading the different input files.

3.5 Reproducibility Code

Google Colab notebooks are reproducible in the way that are implemented, while users can connect the project folder with their google account by adding a shortcut, run the code blocks (python snippets) and train a model without any other process. The code can be found in this git repository, where a link is given for the google drive folder in order to run the code.

4. Results - Discussion

4.1 Vocabulary

Dictionaries for marine species traits were selected and manually extracted from the WoRMS marine species traits wiki (Marine Species Traits Wiki) and the BIOTIC traits dictionary and categorised into four entities/categories with a descriptor link to marine species traits. The entities of the vocabulary are as follows:

i. Distributional descriptors (194 entities), that refer to environment, habitat, province, vertical biological zone and depth of the species,

ii. Life stages (38 entities), which refer to the life stage of the species,

iii. Body size (23 entities), which refers to the qualitative and quantitative body size of the species and

iv. Sampling device (28 entities), i.e. sampling tools for pelagic and benthic organisms, dredges, corers and hyper benthic sleds and nets.

Dictionaries can be found here.

4.2 Corpus

After conducting an online search of publications and their corresponding citations, we have arrived at the conclusion that no curated corpus on marine species traits currently exists. Thus, it was necessary to develop it.

To create the corpus, the literature on marine species traits (Marine Species Traits Wiki) was surveyed, a literature dedicated to texts on the ecological traits that were suitable for this study. Since it was difficult to find texts with multiple references related to the entities of interest, the best choice was to keep the summaries of related texts that contained the most information. The first corpus consisted of 14 abstracts. Additionally, to assist the work of the corpus development, the HCMR team provided historical literature on marine species, which included information related to sampling devices, locations, body size and life stage of specific species. As a result, the final corpus consisted of five full text scientific documents along with the previous 14 abstracts.

Corpus texts can be found here.

4.3 Annotation

In order to provide proper training for the selected NLP libraries, a very important step was the selection of the annotation tool. The annotation tool is required for the categorisation and the labelling of the desired words into the four following categories, as it was required by the Feasibility Study:

1. Body Size
2. Distribution Descriptors
3. Life Stages
4. Sampling Devices

4.3.1 Brat

BRAT (Rapid Annotation Tool) (Stenetorp et al., 2012) is a free, web-based text annotation tool designed for annotating text with structural information such as named entities, events, relationships, and co-references. It is widely used for natural language processing and information extraction tasks, especially in computational linguistics. BRAT supports multiple annotators working simultaneously on the same document and provides visualisations of annotation results for improved control of data quality and inter-annotator agreement.

BRAT provides built-in support for annotating named entities, such as people, organisations, and locations, as well as events and relationships between entities. BRAT can also annotate user-defined entities, thus allowing the annotation of information that is project specific. The process of annotating entities in BRAT involves selecting text in the document and tagging it with the appropriate entity type. The annotated entities can then be visualised and organised so that the relationships between entities in the text can be easily understood.

BRAT's entity annotation capabilities make it a useful tool for natural language processing, such as information extraction and text categorisation, and for creating training data for machine learning models. On the other hand, user-defined entities require manual annotation, which can be time consuming and error prone. Defining custom entity types can be challenging, as they must be meaningful, clearly defined, and relevant to the text being annotated when referred to the configuration. This can be a complex task that requires expertise and experience.

The BRAT server is implemented in Python (version 2.5), while the installation script assumes a UNIX-like environment.

The URL of the tool is: http://brat.nlplab.org/index.html

4.3.2 Tecoholic

Tecoholic (Tecoholic. (n.d.).) is another tool that is only for text annotation. To use it properly, the user must divide the content into paragraphs or text passages and use a consistent separator between passages, e.g., newline, empty line, or a text separator such as -. For large datasets, the text must be split into smaller files to tag them separately. Then the entity labels are simply created using the "new tag" button and the user can start annotating the text. After processing, the annotation file can be downloaded in JSON format. Annotation files from Tecoholic can be easily converted and used as training data in spaCy (description included in section 4.3.3).

The URL of the tool is: https://tecoholic.github.io/ner-annotator/

4.3.3 spaCy NER annotation tool

spaCy NER annotation tool is capable of annotating text data for spaCy NER to create a custom NER model. Copying and pasting the raw text and adding the entity names in the "named entities" section is followed by selecting the text and then the desired entity. Once the annotation is complete, it is worth noting that the annotations can be used as training data in spaCy by simply copying and pasting them into the spaCy environment.

4.3.4 UBIAI

UBIAI (UbiAi. (n.d.)) is a fast-labelling tool for annotating data, but it is not open source. Import data can be in the format of TXT, PDF, HTML or DOCX. UBIAI can also recognise JSON files with existing entities, CSV files containing one document per row and zip files containing TXT, PDF or HTML. UBIAI does not require any installation and can be found and used via the link. Once the user has created a project, uploaded the documents, and selected the type of project annotation, they can declare the entities and start annotating. The type of the annotation is about 'single character span based' or 'word span based', while the user can choose OCR annotation for PDF. After completing annotation, annotation files can be exported in different formats, such as spaCy format for training, JSON or IOB (short for inside, outside, beginning) format, which is a requirement for the training/fine tuning of different models.

4.4 Taggers and NLP

4.4.1 Gnfinder

Gnfinder (Mozzherin et al., 2023) is a tool that allows users to find scientific names on web pages, PDFs, MS Office documents, images, or texts. It is considered as a very fast finder of scientific names and it uses both

dictionary and NLP (Natural Language Processing) approaches, such as information extraction. Gnfinder can work with many file formats and includes name verification against many biological databases.

Gnfinder features:

• Includes REST API and a web-based user Interface.

• Extracts text from PDF files, MS Word, MS Excel, HTML, XML, RTF, JPG, TIFF, GIF etc. files for name recognition.

• Downloads web pages from a specified URL for name recognition.

• Optionally, automatically detects the language of the text, and adapts the Bayes algorithm for the language. English and German languages are currently supported.

• Uses complementary heuristic and natural language processing algorithms.

• Optionally checks found names against multiple biodiversity databases.

Gnfinder was studied and tested in different texts, URL and PDF to evaluate its performance and accuracy by manually annotating the texts with the scientific names and then comparing the annotations with the results of Gnfinder. It seemed to perform well, not only in searching for scientific names but also in their verification against biodiversity databases. Using the same example in the command line version and then online, it is observed that the online version of the tool could not recognise some non-normalized text names such as, "(Atherion)." This format in a sentence could not be recognised as the scientific name Atherion. Finally, it seems that Gnfinder cannot be helpful in the first steps of the study because it cannot recognise marine species traits.

4.4.2 JensenLab Tagger

With JensenLab Tagger (Juhl Jensen, L. (n.d.)) we refer to Tagcorpus, a dictionary based approach. Tagcorpus is a C++ program that allows users to tag a corpus of documents with the search term they enter. It is widely used to identify and label various entities in text data, such as proteins, species, diseases, tissues, chemicals, drugs, and Gene Ontology (GO) terms among others, in articles in the Medline corpus. Often two types of information are tagged simultaneously to find common mentions. For example, proteins and diseases, or human proteins and viral proteins.

JensenLab Tagger is a pre-trained Named Entity Recognition (NER) model. While it may work well for general purpose entity recognition in these specific domains, it may not perform optimally for custom NER tasks that involve different entity types or require specific domain knowledge.

To create a custom NER model, it is needed to train your own model using a labelled dataset that include examples of the entities you want to recognise. This involves selecting and preparing the appropriate data, choosing a suitable algorithm, and fine tuning the model on your specific task.

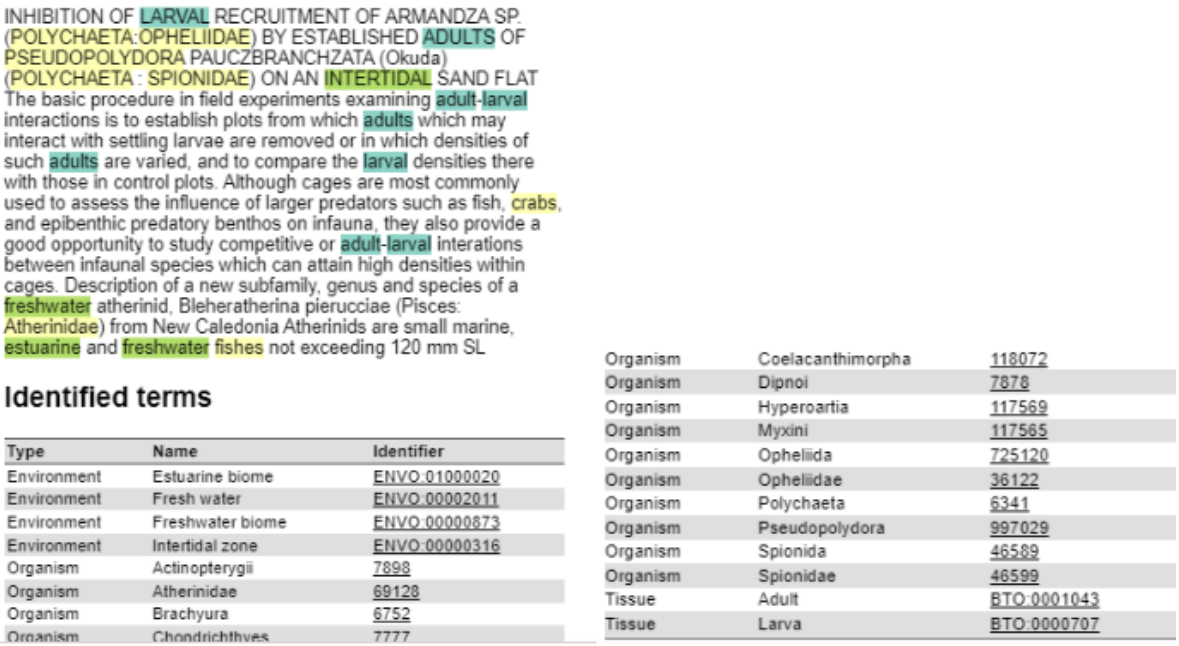
In summary, JensenLab Tagger is not designed for custom NER tasks and may not perform well on tasks that involve different entity types or require specific domain knowledge. To create a custom NER model, it needs to train your own model using a labelled dataset and an appropriate algorithm.

4.4.3 Extract

EXTRACT (JensenLab. (n.d.)) and (Pafilis et al., 2016) identifies genes/proteins, chemical substances, living things, habitats, tissues, illnesses, phenotypes, and Gene Ontology terms that are referenced in a text. It maps these terms to the appropriate ontology/taxonomy entries and returns a link with more information). The JensenLab tagger lies at the core of EXTRACT. Said differently, EXTRACT is a web-based front-end for the JensenLab tagger. Built as curator-assistant, among others it highlights the identified entity mentions (incl. support for Organism, Environments and Tissue dictionaries, available at Index of / (jensenlab.org)).

When testing EXTRACT in the literature related to the topic of our study, entities related to body size could not be detected, while the feature "Adult/Larva" (which refers to life stage) was assigned to the type "Tissue" as shown in Figure 3, and that is expected since it is not concluded in the established dictionaries. Adapting the code to the study approach would be a significant challenge due to the complexity of the process combined

with the limited study time. As a result, it has been decided that the EXTRACT will not be pursued for the purpose of the study.



INHIBITION OF LARVAL RECRUITMENT OF ARMANDZA SP. (POLYCHAETA:OPHELIIDAE) BY ESTABLISHED ADULTS OF PSEUDOPOLYDORA PAUCZBRANCHZATA (Okuda) (POLYCHAETA : SPIONIDAE) ON AN INTERTIDAL SAND FLAT
The basic procedure in field experiments examining adult-larval interactions is to establish plots from which adults which may interact with settling larvae are removed or in which densities of such adults are varied, and to compare the larval densities there with those in control plots. Although cages are most commonly used to assess the influence of larger predators such as fish, crabs, and epibenthic predatory benthos on infauna, they also provide a good opportunity to study competitive or adult-larval interations between infaunal species which can attain high densities within cages. Description of a new subfamily, genus and species of a freshwater atherinid, Bleheratherina pierucciae (Pisces: Atherinidae) from New Caledonia Atherinids are small marine, estuarine and freshwater fishes not exceeding 120 mm SL

Identified terms

Type	Name	Identifier
Environment	Estuarine biome	ENVO:01000020
Environment	Fresh water	ENVO:00002011
Environment	Freshwater biome	ENVO:00000873
Environment	Intertidal zone	ENVO:00000316
Organism	Actinopterygii	7898
Organism	Atherinidae	69128
Organism	Brachyura	6752
Organism	Chondrichthyes	7777
Organism	Coelacanthimorpha	118072
Organism	Dipnoi	7878
Organism	Hyperoartia	117569
Organism	Myxini	117565
Organism	Opheliida	725120
Organism	Opheliidae	36122
Organism	Polychaeta	6341
Organism	Pseudopolydora	997029
Organism	Spionida	46589
Organism	Spionidae	46599
Tissue	Adult	BTO:0001043
Tissue	Larva	BTO:0000707

Figure 3. Extract example in the browser.

4.4.4 spaCy

spaCy (Honnibal et al., 2017) is an open-source software library for advanced natural language processing (NLP) in Python. It is designed to process large amounts of text quickly and efficiently, and to provide easy access to NLP tasks such as tokenisation, text classification, and named entity recognition. The features of spaCy (Figure 4) have been evaluated for their performance using specific examples. Overall, spaCy is considered one of the most user-friendly libraries for natural language processing. It has a clean and simple API, pre-trained models, and supports multiple languages, making it a popular choice for NLP task processing.

spaCy's NER system is based on the statistical learning method, which uses machine learning algorithms such as Conditional Random Fields (CRF) to predict entities in text. spaCy's NER is fast, accurate, and easy to use, providing named entities as objects and allowing easy integration with NLP pipelines. spaCy also provides visualisation tools for understanding NER output and a variety of evaluation metrics for assessing NER performance.

Tokenization	Segmenting text into words, punctuations marks etc.
Part-of-speech (POS) **Tagging**	Assigning word types to tokens, like verb or noun.
Dependency Parsing	Assigning syntactic dependency labels, describing the relations between individual tokens, like subject or object.
Lemmatization	Assigning the base forms of words. For example, the lemma of "was" is "be", and the lemma of "rats" is "rat".
Sentence Boundary Detection (SBD)	Finding and segmenting individual sentences.
Named Entity Recognition (NER)	Labelling named "real-world" objects, like persons, companies or locations.
Entity Linking (EL)	Disambiguating textual entities to unique identifiers in a knowledge base.
Similarity	Comparing words, text spans and documents and how similar they are to each other.
Text Classification	Assigning categories or labels to a whole document, or parts of a document.
Rule-based Matching	Finding sequences of tokens based on their texts and linguistic annotations, similar to regular expressions.
Training	Updating and improving a statistical model's predictions.
Serialization	Saving objects to files or byte strings.

Figure 4. spaCy features

4.4.5. spaCy blank English model

spaCy (Honnibal et al., 2017) is an open-source natural language processing (NLP) library in Python. The "blank English model" refers to a version of the library that hasn't been pre-trained for any particular NLP task. This means that it doesn't have any pre-existing knowledge of the English language but can still be used to perform basic NLP operations such as tokenization, stemming, and lemmatization. To use the blank English model of spaCy, you need to install the library (see installing spaCy above) and load the model into your Python code environment.

4.4.6 NLTK

NLTK (Natural Language Toolkit) (Bird et al., (n.d.)) is an open-source library for Natural Language Processing (NLP) in the Python programming language. It provides a set of tools for tasks such as tokenisation, part of speech tagging, named entity recognition, sentiment analysis, and more. NLTK is used in academia and industry for NLP research and development, and it is considered one of the most comprehensive NLP libraries on the market but can be slower and more memory intensive compared to spaCy.

NLTK's NER system uses a sequence labelling approach in which each word in a sentence is assigned to a label indicating whether it is an entity or not. The NER system uses a Unigram Chunker, a simple rule-based NER system that uses a regular expression parser to identify entity phrases in text. It is worth noting that while NLTK provides NER capabilities, it may not be as fast or efficient as other NER libraries such as spaCy, and its performance may not be as good as more advanced NER systems. NLTK also provides tools to evaluate and improve NER performance, such as precision, recall, and F1 score metrics.

Neural network model option is not available in NLTK. In order to build a named entity recognition model in NLTK, the user has to write their own code to recognise named entities in the text based on the tokenised text

and the POS (part-of-speech) tags. This can involve defining patterns of words and POS tags that correspond to named entities and using NLTK's chunking and parsing functions to identify named entities in the text.

4.4.7 RoBERTa

RoBERTa (Liu et al., 2019) is a large Deep Learning based language model developed by Facebook AI. It is an improvement of the BERT (Bidirectional Encoder Representations from Transformers) model, which is one of the most popular pre-trained models for NLP tasks. RoBERTa is trained on a larger corpus of text data and incorporates several changes to the training procedure that should further improve the quality of the resulting speech representations. It has achieved top performance on several NLP benchmarks and is used in many NLP applications, such as sentiment analysis, question answering, and text classification. The RoBERTa transformer model has also been fine-tuned using spaCy transformers.

4.5 Model training

4.5.1 BERT/BioBERT

BERT (Bidirectional Encoder Representations from Transformers) (Wolf et al., 2021) is a pre-trained deep learning language model developed by Google. It is a transformer-based architecture trained on a large corpus of text data using an unsupervised learning approach. The model is trained to predict missing words in a sentence (masked language modelling) and predict the next sentence in a sentence pair (next sentence prediction). It is based on the architecture of BERT but has been specifically trained on a large corpus of scientific publications from the life sciences and medicine. The goal of BioBERT is to provide a high-quality NLP model capable of accurately processing and understanding the unique vocabulary and terminology used in the biological and medical fields. This makes BioBERT particularly useful for NLP tasks in these domains, such as named entity recognition, relation extraction, and question answering.

BERT is also unique as it processes text in a bidirectional manner, meaning that the model considers both the previous and subsequent words when processing a given word. This contrasts with traditional language models, which typically process text only from left to right. Thanks to this bidirectional approach, BERT is able to understand the context of words in a sentence in a more robust and sophisticated way. Pre-trained versions of BERT in multiple languages are freely available so that researchers and practitioners can use the model as a basis for their own NLP projects.

Modern pre-trained models can be simply downloaded and trained using the APIs and tools provided by Transformers. Pretrained models can save users the time and resources needed to train a model from scratch while lowering the users' computing financial expenses and carbon footprint. These models cover typical NLP tasks like text classification, named entity recognition, etc. in a variety of modalities.

Spacy-transformers package provides spaCy model pipelines that wrap Hugging Face's transformers package, so they can be used in spaCy. As a result, modern transformer topologies like BERT are easily accessible.

4.5.2 Data format preparation

As the very first step, text mining prototypes were selected and the input data format for each one was examined before training. The models to be trained or fine-tuned were the spaCy English model, the "BERT", "BioBERT" and the "roBERTa", as those were considered the most appropriate according to our evaluation process. The input data are considered as the annotations of the corpus, divided into three categories: training data, development data and evaluation data (Table1 presents the corpus range and the data division). The corpus is composed of PDFs and texts.

Table 1. Corpus Representation

	Training data	Development data	Evaluation data
Number of full texts-abstracts	2-11	2	1-3
Number of entities	814	455	707

For the corpus annotation at the second step, UBIAI was used in order to annotate the data and download them in IOB and spaCy's NER format, that are the required formats for the different models selected training. For training in spaCy the annotated data have to be in the following format: "sentence", (entity, begin char, end char), where begin and end character refer to the position of the word inside the sentence and start counting from 0 at each one. For RoBERTa, BERT and BioBERT fine tuning, annotations format is the 'IOB' labelled per word (a list of words with the annotated entities for each one, labelled with I for inside, B for beginning or O for outside, see Table 2).

Table 2. IOB format annotations' example

Entity	IOB label
Migratory	O
Travels	O
In	O
Big	O
shoals,	B-DISTRIBUTION_DESCRIPTOR
Appears	O
In	O
Considerable	O
Quantities	O
About	O

Examining the UBIAI output annotations for the desired formats, it was noticed that spaCy NER's format had wrong offsets for the beginning and the end character of the entity in the sentence, so some conflict of UBIAI might have occurred. The labels were pointing to different words than the annotated entities (see the example below "Example of spaCy's format issue"). Additionally, the first issue was that instead of outputting the text divided into sentences with entities for each one, it has an output of the whole text followed by all the annotated entities, counting offsets without resetting the counter to zero in the beginning of the sentence. This issue was resolved by a custom script (contained in the Git repository) that converts this format to the sentence-by-sentence format for spaCy's model training.

IOB annotations per word and per sentence were needed, so IOB format output from UBIAI (per word) was converted into per sentence format with a custom script (contained in the Git repository).

In conclusion, a general UBIAI problem regarding PDFs and OCR (Optical character recognition), is that the post processed text has many extra characters and separated words, a condition that can cause misunderstandings or losses in the training process. In addition, text from images cannot be identified by UBIAI.

Example of spaCy's format issue:

In the sentence:

"SOME POLYCHAETE SPECIES FROM THE SOFT BOTTOM OF THE EASTERN HARBOUR OF ALEXANDRIA, EGYPT, WITH SPECIAL REFERENCE TO ORBINIIDS AND PARANOIDS HABITATS"

An extracted entity from UBIAI is: (35, 48, Distribution_descriptor) for the "soft bottom" instead of (33, 44, Distribution_descriptor), where 33 is the character count for starting letter "s" and 44 for ending letter "m".

4.6 Model evaluation and performance

4.6.1 spaCy

The evaluation of this system showed that a custom named entity recognition system can support our approach of extracting marine species features. In addition, the system could be implemented by using a pre trained or empty model of spaCy for training based on our training data (Figure 5). Training the spaCy model requires training data, i.e., annotated text examples related to the study approach, in this case, marine species features**. In conclusion, spaCy appears to fulfil our expectations**.

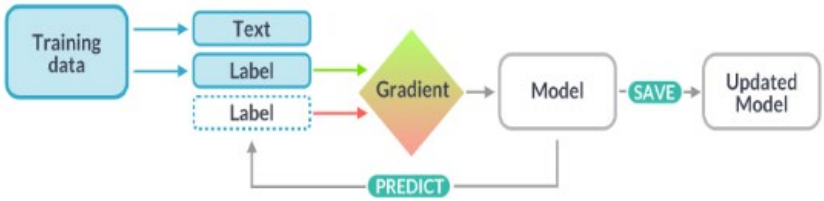


Figure 5. spaCy's model training process

Blank English spaCy model (Honnibal et al., 2017) was trained for custom NER based on the accuracy, with the annotated datasets which contain the entities/labels for marine species traits based on the categories: distribution descriptor, life stage, body size and sampling device. The model was evaluated on the evaluation dataset based on precision, recall and f1-score metrics.

In spaCy training, the following evaluation metrics are commonly used for NER models:

Loss tok2vec: measures the error of the model in predicting the semantic representation of the input text, using the tok2vec component of the pipeline. The value range of loss tok2vec can be between 0 and infinity.

Loss NER: measures the error of the model in predicting the named entities in the input text, using the named entity recognition (NER) component of the pipeline. The value range of loss in Named Entity Recognition (NER) can be between 0 and infinity.

ents_f: harmonic mean of ents_p (precision) and ents_r (recall) and represents the overall performance of the model in identifying named entities. It is a common evaluation metric used for NER models.

ents_p: proportion of true positive named entity predictions out of all predicted entities, and represents the precision of the model in identifying named entities.

ents_r: proportion of true positive named entity predictions out of all actual entities in the dataset, and represents the recall of the model in identifying named entities.

The value range of ents_p, ents_rl, and ents_f is between 0 and 100, where a score of 100 indicates perfect performance and a score of 0 indicates poor performance.

In general, a good NER model should have a low tok2vec loss and NER loss, as well as high values for ents_f, ents_p, and ents_r.

Figure 5a depicts the output of the training performance of the metrics explained above. The "E" represents an epoch, that refers to a complete pass through the entire training dataset during model training. During an epoch,

the model goes through all the training examples, makes predictions, and calculates the loss. The loss is then used to update the model's parameters through back propagation to improve the model's performance. After an epoch is completed, the model is evaluated on a separate validation/development dataset to determine its performance on data that it has not seen before.

E	#	LOSS TOK2VEC	LOSS NER	ENTS_F	ENTS_P	ENTS_R	SCORE
0	0	0.00	35.00	0.00	0.00	0.00	0.00
1	200	36.79	1567.91	26.62	24.18	29.60	0.27
2	400	68.53	1115.46	30.84	34.31	28.00	0.31
4	600	53.19	1155.08	45.43	35.98	61.60	0.45
6	800	229.58	1081.01	34.62	33.33	36.00	0.35
9	1000	100.33	863.77	21.83	15.02	40.00	0.22
12	1200	233.04	804.37	37.66	39.47	36.00	0.38
16	1400	183.36	720.22	35.63	36.07	35.20	0.36
21	1600	178.14	606.85	37.82	34.67	41.60	0.38
28	1800	317.75	577.24	29.43	23.56	39.20	0.29
35	2000	130.33	467.33	34.84	30.86	40.00	0.35
45	2200	160.70	517.96	33.22	28.82	39.20	0.33

Figure 5a. spaCy's model training performance

Figure 5b presents the evaluation of the trained model for each entity based on the corpus evaluation dataset. Worth noting is that it seems like it could not identify related entities with body size and sampling devices.

```
TOK     100.00
NER P   35.98
NER R   61.60
NER F   45.43
SPEED   4932


============================== NER (per type) =======

                              P       R       F
DISTRIBUTION_DESCRIPTOR   36.50   62.93   46.20
BODY_SIZE                  0.00    0.00    0.00
LIFE_STAGE                80.00   66.67   72.73
SAMPLING_DEVICE            0.00    0.00    0.00
```

Figure 5b. spaCy's model evaluation on evaluation dataset based on each entity

```
TOK     100.00
NER P   32.82
NER R   68.00
NER F   44.27
SPEED   1889


============================== NER (per type) ========

                              P       R       F
DISTRIBUTION_DESCRIPTOR   33.20   69.83   45.00
BODY_SIZE                  0.00    0.00    0.00
LIFE_STAGE                80.00   66.67   72.73
SAMPLING_DEVICE            0.00    0.00    0.00
```

Figure 5c. spaCy's model evaluation on evaluation dataset based on each entity after dictionaries addition

Comparing Figure 5c metrics with metrics before the dictionary's addition (Figure 5b), the precision has decreased and recall has increased which means that the model is identifying more true positives, but at the cost of also identifying more false positives. As is important to have accurate predictions, a higher precision may be more important than a higher recall. An example sentence that was predicted from both models, before and after dictionaries, extracted the same entities while in another one, the model with dictionaries had a better performance, as it recognised 2 more entities (see example 3 in section 2.3). It is observed that the entities "BODY_SIZE" and "SAMPLING_DEVICE" in this corpus had a precision and recall of 0. This can be attributed to the limited occurrence of these entities within the annotated corpus.

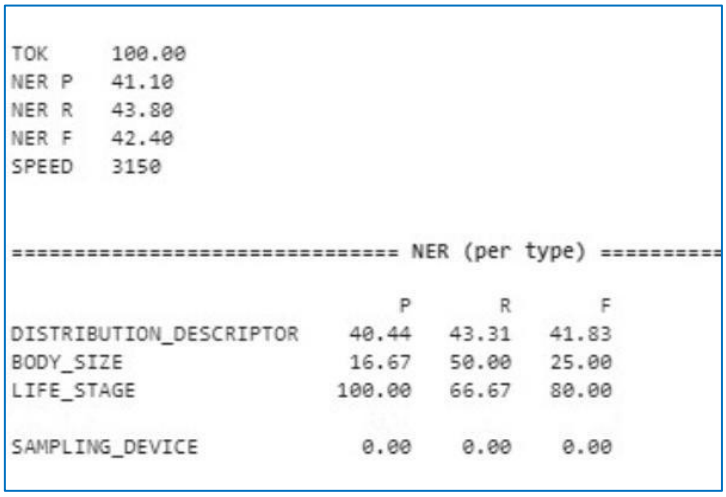
The model was subjected to experimental testing to evaluate its performance accuracy in recognising relevant entities to the study's approach, marine species traits, and the results were satisfactory, demonstrating good performance in identifying the intended entities.

4.6.2 RoBERTa

Model **roBERTa-base** (Liu et al., 2019) was fine-tuned via spaCy 3 transformer and evaluated for the same corpus. The transformer architecture employed by the RoBERTa model incorporates multiple self attention layers, which endow the model with the ability to attend to different parts of the input sequence and capture complex relationships between words. During the training process, the self -attention loss, also known as transformer loss (loss trans in Figure 6a), is used to train the self attention layers of the model. The self attention loss computes the discrepancy between the output of the self attention layers and the target output for every token in the input sequence. This loss incentivises the model to learn to concentrate on the relevant parts of the input sequence and apprehend the relationships between words, which is of utmost significance for a multitude of natural language processing tasks. The remaining metrics that are exhibited are delineated earlier in the spaCy model.

E	#	LOSS TRANS...	LOSS NER	ENTS_F	ENTS_P	ENTS_R	SCORE
0	0	1317.11	610.57	0.32	0.18	1.84	0.00
25	200	42823.19	36983.38	33.75	52.43	24.88	0.34
51	400	1501.62	1546.36	35.88	41.98	31.34	0.36
76	600	552.56	716.10	35.84	37.76	34.10	0.36
102	800	467.91	619.68	37.22	46.85	30.88	0.37
128	1000	472.81	591.69	32.79	33.33	32.26	0.33
153	1200	406.00	538.88	34.35	32.51	36.41	0.34
178	1400	417.59	543.20	35.91	32.82	39.63	0.36
203	1600	411.96	502.87	31.76	43.90	24.88	0.32
228	1800	386.90	492.12	30.56	26.83	35.48	0.31
254	2000	371.29	489.88	35.40	35.32	35.48	0.35
280	2200	393.75	491.42	32.82	28.01	39.63	0.33
305	2400	367.12	482.33	31.88	36.05	28.57	0.32

Figure 6a. RoBERTa's model training performance



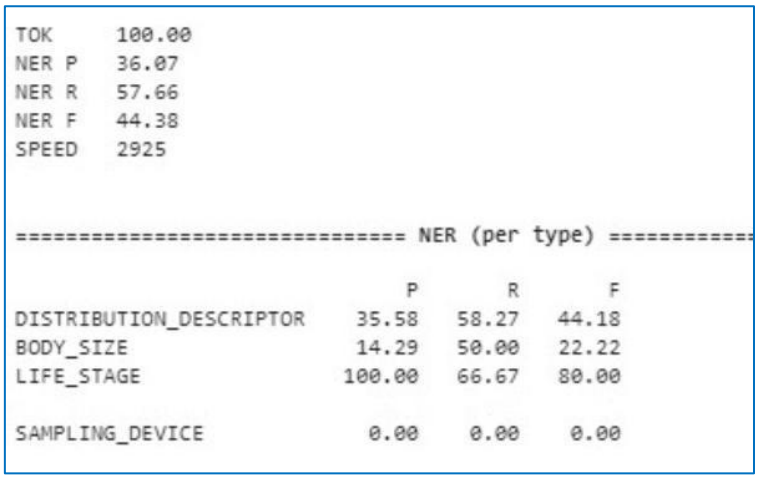
```
TOK      100.00
NER P    41.10
NER R    43.80
NER F    42.40
SPEED    3150


============================== NER (per type) ==========

                              P       R       F
DISTRIBUTION_DESCRIPTOR    40.44   43.31   41.83
BODY_SIZE                  16.67   50.00   25.00
LIFE_STAGE                100.00   66.67   80.00

SAMPLING_DEVICE             0.00    0.00    0.00
```

Figure 6b. RoBERTa's model evaluation on evaluation dataset based on each entity.



```
TOK      100.00
NER P    36.07
NER R    57.66
NER F    44.38
SPEED    2925


============================== NER (per type) ============

                              P       R       F
DISTRIBUTION_DESCRIPTOR    35.58   58.27   44.18
BODY_SIZE                  14.29   50.00   22.22
LIFE_STAGE                100.00   66.67   80.00

SAMPLING_DEVICE             0.00    0.00    0.00
```

Figure 6c. RoBERTa's model evaluation on evaluation dataset based on each entity after dictionaries addition

Comparing Figures 6c metrics with metrics before the dictionary's addition (Figure 6b), the precision has decreased and recall has increased which means again that the model is identifying more true positives, but at the cost of also identifying more false positives. Given the significance of precise predictions, a higher precision score may hold more importance compared to a higher recall score. In an instance where a sentence was predicted by both models - pre and post integration of dictionaries, the same entities were extracted. However, in *Example 3 (G. R. Allen, 1998)*:, the model equipped with dictionaries appears to exhibit better performance as it detected two additional, almost true, entities.

The trained model was put through experimental testing to assess its accuracy performance in recognising desirable entities in texts associated with marine species traits. The experimental results indicated that the model could identify the intended entities with good performance.

4.6.3 BERT/BioBERT

BERT (BERT-base-cased) and **BioBERT** (dmis-lab/BioBERT-v1.1) models were also fine-tuned for NER via spaCy 3 transformers on the same corpus. Their performance during fine tuning is presented in Figures 7a and 7b. The final score (corresponds to the round up number of ENTS_F metric) of the BERT model is 0.35 when the BioBERT model has a final score at 0.31, when BioBERT needs more time to be fine tuned, as it takes more epochs.

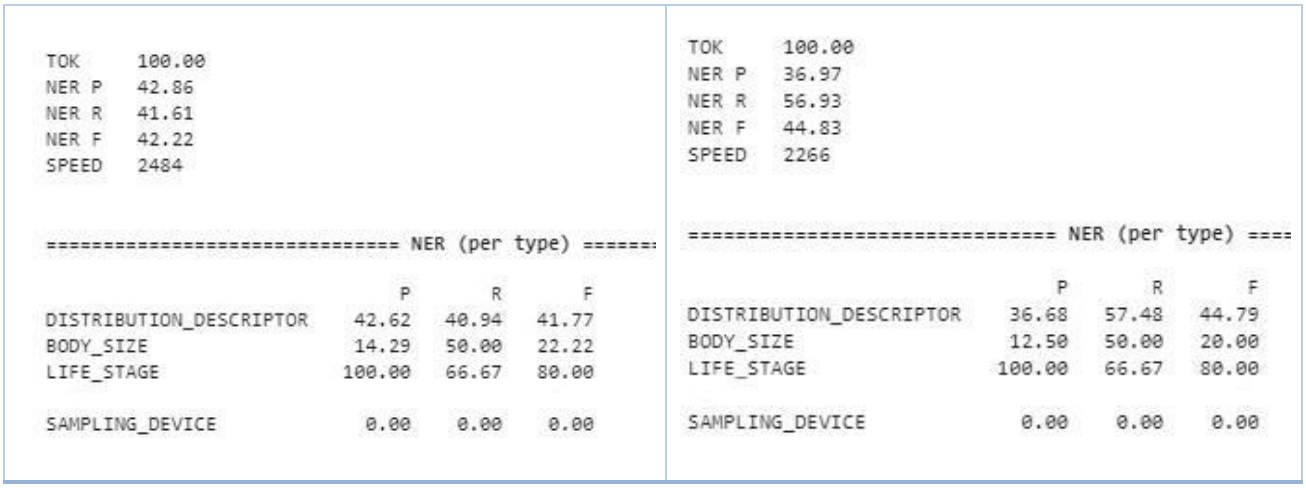
E	#	LOSS TRANS...	LOSS NER	ENTS_F	ENTS_P	ENTS_R	SCORE
0	0	2330.24	583.31	0.20	0.11	0.92	0.00
25	200	146689.83	47808.73	31.21	29.83	32.72	0.31
51	400	2941.66	1371.71	32.93	47.01	25.35	0.33
76	600	455.55	723.96	33.98	42.96	28.11	0.34
102	800	364.11	608.09	32.87	41.55	27.19	0.33
128	1000	344.09	584.75	28.75	44.66	21.20	0.29
153	1200	346.86	578.72	37.12	46.53	30.88	0.37
178	1400	349.68	571.18	31.58	48.11	23.50	0.32
203	1600	366.81	564.00	26.22	38.74	19.82	0.26
228	1800	327.55	543.44	29.67	41.67	23.04	0.30
254	2000	323.25	531.04	31.18	43.09	24.42	0.31
280	2200	322.95	542.93	33.69	40.13	29.03	0.34
305	2400	313.49	529.39	31.27	37.66	26.73	0.31
330	2600	337.89	543.70	28.40	41.23	21.66	0.28
356	2800	310.85	523.07	35.33	43.05	29.95	0.35

Figure 7a. BERT model's fine-tuning performance.

E	#	LOSS TRANS...	LOSS NER	ENTS_F	ENTS_P	ENTS_R	SCORE
0	0	4267.07	574.05	0.21	0.11	1.38	0.00
25	200	79453.96	44950.00	28.05	32.14	24.88	0.28
51	400	1383.90	1448.17	32.20	41.61	26.27	0.32
76	600	442.39	655.26	33.61	41.78	28.11	0.34
102	800	414.23	579.90	33.82	46.03	26.73	0.34
128	1000	388.25	563.78	32.84	46.61	25.35	0.33
153	1200	389.58	552.41	34.04	40.25	29.49	0.34
178	1400	426.31	567.95	31.45	44.17	24.42	0.31
203	1600	409.34	529.34	28.48	47.83	20.28	0.28
228	1800	364.02	513.99	33.84	49.12	25.81	0.34
254	2000	365.78	513.14	35.19	48.39	27.65	0.35
280	2200	374.81	532.30	32.83	30.89	35.02	0.33
305	2400	357.14	506.94	36.80	43.67	31.80	0.37
330	2600	382.62	522.97	32.97	40.82	27.65	0.33
356	2800	416.26	591.10	36.46	40.45	33.18	0.36
381	3000	398.83	566.20	36.94	43.21	32.26	0.37
407	3200	363.19	516.62	38.07	45.51	32.72	0.38
432	3400	343.53	490.88	39.28	44.71	35.02	0.39
458	3600	345.35	496.01	31.00	45.54	23.50	0.31
483	3800	339.03	492.30	31.75	51.02	23.04	0.32
509	4000	358.75	503.44	34.42	48.33	26.73	0.34
534	4200	350.27	489.10	32.40	50.00	23.96	0.32
559	4400	355.64	498.79	31.67	43.55	24.88	0.32
585	4600	333.17	478.61	34.27	43.88	28.11	0.34
611	4800	539.80	583.55	31.78	39.19	26.73	0.32
636	5000	351.41	529.97	30.86	46.73	23.04	0.31

Figure 7b. BioBERT model's fine-tuning performance.

The evaluation of the models was completed on the evaluation dataset before and after the dictionaries addition and the metrics based on each entity are presented in Figures 8a and 8b for BERT and Figure 9a and 9b for BioBERT, respectively. Both models, after adding the dictionaries, appear a precision decrease and recall increase.



```
TOK      100.00
NER P    42.86
NER R    41.61
NER F    42.22
SPEED    2484

================================ NER (per type) =======

                            P       R       F
DISTRIBUTION_DESCRIPTOR   42.62   40.94   41.77
BODY_SIZE                 14.29   50.00   22.22
LIFE_STAGE               100.00   66.67   80.00

SAMPLING_DEVICE            0.00    0.00    0.00
```

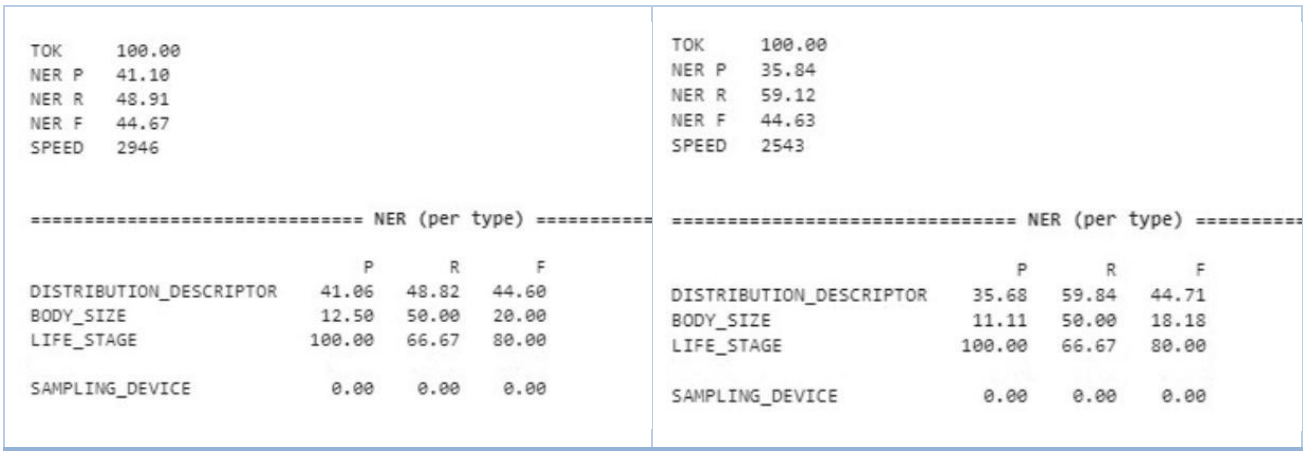
```
TOK      100.00
NER P    36.97
NER R    56.93
NER F    44.83
SPEED    2266

================================ NER (per type) ====

                            P       R       F
DISTRIBUTION_DESCRIPTOR   36.68   57.48   44.79
BODY_SIZE                 12.50   50.00   20.00
LIFE_STAGE               100.00   66.67   80.00

SAMPLING_DEVICE            0.00    0.00    0.00
```

Figures 8a (left) and 8b (right). BERT's model evaluation on evaluation dataset based on each entity and BERT's model evaluation on evaluation dataset based on each entity after dictionaries addition.



```
TOK      100.00
NER P    41.10
NER R    48.91
NER F    44.67
SPEED    2946

================================ NER (per type) ===========

                            P       R       F
DISTRIBUTION_DESCRIPTOR   41.06   48.82   44.60
BODY_SIZE                 12.50   50.00   20.00
LIFE_STAGE               100.00   66.67   80.00

SAMPLING_DEVICE            0.00    0.00    0.00
```

```
TOK      100.00
NER P    35.84
NER R    59.12
NER F    44.63
SPEED    2543

================================ NER (per type) ==========

                            P       R       F
DISTRIBUTION_DESCRIPTOR   35.68   59.84   44.71
BODY_SIZE                 11.11   50.00   18.18
LIFE_STAGE               100.00   66.67   80.00

SAMPLING_DEVICE            0.00    0.00    0.00
```

Figures 9a (left) and 9b (right). BioBERT's model evaluation on evaluation dataset based on each entity and BioBERT's model evaluation on evaluation dataset based on each entity after dictionaries addition.

BERT and BioBERT transformers were at first fine tuned for NER in PyTorch with custom scripts on the same corpus. Their performance during fine tuning for 20 epochs was evaluated with the accuracy and BERT's accuracy was calculated to be 0.987, while BioBERT's was 0.986. As demonstrated after 20 epochs, it was observed that BERT achieved a modest loss value on the training data while validating the acquired knowledge on the development dataset. In comparison with BioBERT, the latter appears to have a slightly lower loss in both scenarios, albeit without significant disparity.

Although the models were tested for their performance in related examples and gave acceptable results, for the evaluating metrics it will be better to have the precision recall and F1 score metrics in order to compare them more efficiently with the other models, so they were fine tuned again via spaCy transformers with the above results. Another reason to proceed in a different approach was the dictionaries' addition and the fine tuned models' performance in recognising entities in example texts that was more accurate in the final way, while also the ability to save and load the fine tuned model with no time consuming.

4.7 Summary

A summary of the model's usage is shown below:

Models	Info	Conflicts	Time to build	Computing resources
BERT/BioBERT	•The input training data must be in IOB format. •BERT takes about half an hour to be fine tuned with spacy transformers, while BioBERT needs more time.	•For some reason it can't identify as text a string like "word :" or "word at", if text contains 2 word chars the second one needs to be more than 2 characters long e.g "word for". An attempt at fixing this constraint was done via a custom script.	•Before the training process that takes about half an hour, the runtime for the installations required is about 10-15 minutes.	•GPU and NVIDIA 9.2 cuda library
spaCy blank english model	•The input training data have to be in spaCy NER training format. •Takes about half an hour to be trained in custom entity recognition.	•In this study, due to UBIAI tool performance the IOB format annotations were used and converted into spaCy's NER training format.	•Before the training process that takes about half an hour, the runtime for the installations required is about 5 minutes.	•CPU or GPU to make the training process faster.
roBERTa	•The input training data have to be in IOB format. •Takes about half an hour to be fine tuned in custom entity recognition with spaCy transformers. .	•For an unknown reason it cannot identify a string like "word :" as text or "word at", if text contains 2 word chars the second one needs to be more than 2 characters long e.g "word for". It is almost fixed with a custom script.	•Before the training process that takes about half an hour, the runtime for the installations required is about 10-15 minutes.	•GPU and NVIDIA 9.2 cuda library

4.7.1 Related text examples

Example 1 (IMachado et al., 2021):

"Most marine fish and invertebrate species produce free and small early-stages which are part of the plankton. These incompletely developed individuals are highly vulnerable to unsuitable conditions like starvation and environmental variability, and it was early recognized that survival during these stages often regulates recruitment and adult population size (Cowan and Shaw, 2002), Pineda et al., 2007). Recruitment theories have thus focused on the environmental modulation of hydro survival, and they generally assume that while spawning occurs within relatively fixed time-frames along the year cycle, hydrographic conditions and plankton production show higher inter-annual variability."

spaCy	BioBERT	BERT	RoBERTa
Distribution descriptors: marine, plankton Life stages: adult, larval Body size: size	Distribution descriptors: marine, plankton Life stages: adult, larval	Distribution descriptors: marine, plankton Life stages: adult, larval	Distribution descriptors: marine, plankton, hydrographic Life stages: larval Body size: adult population size

Example 2 (Tamaki A., 1985):

"The maximum body length is 8 mm."

spaCy	BioBERT	BERT	RoBERTa
Body size: maximum body length, 8 mm	Body size: body length, 8 mm	Body size: maximum body length, 8 mm	Body size: maximum body length, 8 mm

Example 3 (G. R. Allen, 1998):

"They are inhabitants of shallow reef areas, usually encountered in less than 10 m depth. During the day they are mainly sedentary, frequently seen resting on the bottom under rock or coral outcrops on substrata containing substantial amounts of sand, silt, mud, or algae."

spaCy's model without dictionaries could not identify algae.

RoBERTa without dictionaries could not identify depth and sand.

BERT without dictionaries could not identify depth sand and algae.

BioBERT without dictionaries could not identify depth sand and algae.

spaCy	BioBERT	BERT	RoBERTa
Distribution descriptors: shallow reef areas, depth, bottom under rock, coral outcrops, substrata, sand, silt, mud, algae	Distribution descriptors: shallow reef areas, depth, bottom under rock, coral outcrops, sand, silt, mud, algae	Distribution descriptors: shallow reef areas, depth, bottom under rock, sand, 'silt, mud' as one entity, algae	Distribution descriptors: shallow reef areas, bottom under rock, depth, coral outcrops, substrata, sand, silt, 'mud, or algae' as one entity

4.8 Other tools for further exploration

4.8.1 Stanza

Stanza (Qi et al., 2020) is a well-known NLP library that was developed by Stanford NLP group. The library provides users with a wide range of pre trained models and tools for different NLP tasks, such as named entity recognition, part of speech tagging, dependency parsing, sentiment analysis, tokenisation, and more. Stanza is

built on PyTorch, a famous deep learning framework, and uses neural networks to achieve state of the art performance on various NLP tasks. It supports multiple languages, such as French, German, Chinese, English, and more, and comes with pre trained models for all of these languages. One of the benefits of Stanza is its easy to use API, making it simple for users to perform common NLP tasks. Additionally, the pre trained models of Stanza are intended to provide state of the art performance, making it a top choice for NLP tasks. The Stanford NLP group also regularly updates the library with new features and models, which shows its active development.

Stanza's version 1.3.0 introduces a new architecture for named entity recognition (NER) known as the Semi-Markov NER model. This model is an enhancement to the conventional BIO (beginning-inside-outside) tagging scheme that is commonly used in many NER models. It enables entities to have more complex structures, such as nested or overlapping entities. The Semi-Markov NER model is founded on a semi-Markov conditional random field (semi-CRF) architecture, which is an expansion of the conventional CRF that permits variable-length outputs, enabling complex structures for entities. The NER model comprises word embeddings, a bidirectional LSTM layer, and a Semi-Markov CRF layer. The word embeddings are learned representations of words in a sentence, which are then fed into the bidirectional LSTM layer, producing a sequence of hidden states that capture the context of the words. The Semi-Markov CRF layer models the dependencies between labels assigned to each word, allowing entities to have more complex structures. During training, the model is optimised to minimise the negative log-likelihood of the correct label and entity structure sequence using stochastic gradient descent or another optimisation algorithm.

4.8.2 SparkNLP

SparkNLP (John Snow Labs., 2022) is an open-source natural language processing library built on top of Apache Spark, designed for large scale distributed processing of text data. SparkNLP provides a variety of NLP tools, including named entity recognition (NER). SparkNLP's NER architecture is based on a deep learning model that uses a combination of bidirectional LSTMs and CRF layers.

The architecture consists of the following components:

1. Word embeddings: The input to the NER model is a sequence of word embeddings, which are learned representations of the words in the text. SparkNLP provides pre trained word embeddings, such as GloVe or Word2Vec, as well as the ability to train custom embeddings on user defined datasets.
2. Bidirectional LSTM layer: The LSTM layer takes the word embeddings as input and generates a sequence of hidden states, which capture the contextual information of the words in the text. The bidirectional LSTM considers the context from both the left and the right of each word in the text.
3. CRF layer: The CRF layer models the dependencies between the labels assigned to each token in the text. The CRF layer takes the output of the bidirectional LSTM layer as input and generates a probability distribution over the possible labels for each token.
4. Training: During training, the model is optimised to minimise the negative log-likelihood of the correct label sequence, using stochastic gradient descent or another optimization algorithm.

4.8.3 Apache OpenNLP

This package provides an interface to the Apache OpenNLP library (Apache Software Foundation. (n.d.)), a machine learning toolkit for the most common NLP operations: part of speech tagging, named entity recognition, and coreference resolution. Other tasks that can be done include tokenisation, sentence segmentation, chunking, parsing, language detection and coreference resolution. Pretrained models for several European languages are included with OpenNLP. Users can even train their own models by using this toolkit. Maximum entropy and perceptron based machine learning are also included in OpenNLP.

The Apache OpenNLP library contains several components, enabling one to build a full natural language processing pipeline. Sentence detector, tokeniser, name finder, document categorization, part of speech tagger, chunker, parser, and coreference resolution are some of the components that fall under this category. Components are made up of pieces that make it possible to perform the relevant natural language processing

task, train a model, and frequently also evaluate a model. With their respective application program interface, each of these facilities is reachable (API). Moreover, a command line interface (CLI) is offered for the convenience of training and conducting research.

OpenNLP provides a NER module that uses statistical models to recognise named entities in text. To use OpenNLP for named entity recognition, a user needs to first download and install the library, which can be done by following the instructions on the OpenNLP website. Once OpenNLP has been installed, the NER module can be used to process text and extract named entities.

4.8.4 Scikit-learn

Scikit-learn (Pedregosa et al., 2011) is a popular open-source machine learning library for Python that includes a range of tools for various machine learning tasks, including natural language processing (NLP).

Scikit-learn provides several tools for NLP, such as text feature extraction, text classification, and clustering. These tools can be used to pre-process text data, extract relevant features, and build models for various NLP tasks.

Scikit-learn is primarily a machine learning library and does not provide a specific tool for named entity recognition (NER). However, scikit-learn can be used in conjunction with other NLP libraries such as spaCy or NLTK to build a named entity recognition model.

5 Future work and Further improvements

The proposed methodology for developing an accurate text mining model involves several steps. This feasibility study concludes that the most suitable models are SpaCy and BioBERT. To further enhance the accuracy of these tools, an enriched workflow is recommended consisting of the following four steps: a. Corpus Enrichment and Text Annotation, b. Combining GNfinder with an NLP library, c. Management of the text mining process and d. Visualisation of results.

5.1 Corpus Enrichment and Text Annotation

For the enrichment of the corpus and to improve the accuracy of our tool, we recommend annotating at least 200 scientific documents and including them in the corpus. Specific annotation rules for traits entities must be followed to ensure that the text mining tool can identify all the desired entities. For instance, if we wish to train a model that can find all the biological traits, we must provide an annotated file that categorises all the relevant entities as biological traits.

Forthis step, the open source Tecoholic Annotation tool can be used.

Given the challenges highlighted in previous sections, it is suggested that a text annotation tool that possesses the following capabilities is implemented:

6 Ability to take input files in text format.

7 Possession of a user friendly interface suitable for non technical users.

8 Ability to store progress made during the annotation process and output results in a suitable annotation scheme, such as TSV in IOB format, spaCy-NER format, among others.

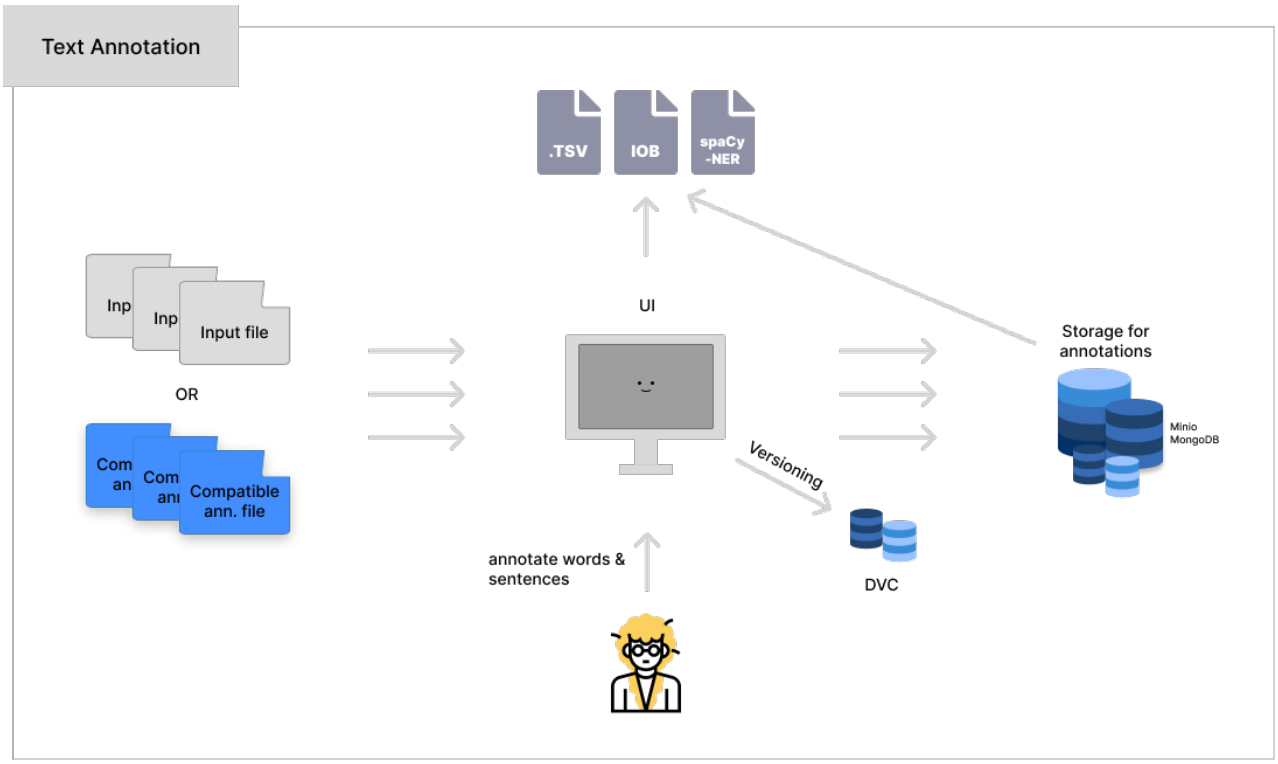


Figure 10. Text Annotation

5.2 Combining GNfinder with an NLP library

For this step, the use of GNfinder is proposed for identifying the scientific names of species. Custom scripts and a trained NER model can be used to match the extracted entities with the corresponding species in a text.

5.3 Management of the text mining process

Regarding the management of the Text Mining Process, the use of MLflow is recommended, which is an open-source platform designed to manage and deploy machine learning models. MLflow offers comprehensive lifecycle management, from data preparation to model deployment, allowing for the tracking of performance metrics of different machine learning models trained on the text data. Furthermore, it can log the hyperparameters used during the training process and facilitate model evaluation through cross validation or holdout validation. The performance of each model can be tracked, and the best one can be selected. MLflow also provides a user-friendly interface to manage and deploy machine learning models.

5.4 Visualisation of results

Regarding the visualisation of results, an improved approach is proposed in presenting the outcome of the text mining process. Instead of producing a table of results with corresponding positions in the provided text, it is suggested that the final text with highlighted annotated words is exported. This implementation holds considerable importance for the scientific community, as the data curator can visualise into text the desirable information.

6 Online training session

A training session was organised on the 20th of March 2023, in the framework of EMODnet Biology, specifically focusing on the ongoing Feasibility Study on Ecological Traits and sampling devices/methodologies identification with text mining. This training session focused on the interdisciplinary field of text mining and its application on marine ecological traits. During the session, which was advertised through the EMODnet portal Training session in the framework of the EMODnet Biology Phase IV on the feasibility study for recognition of specific ecological traits and sampling devices/methodologies in text | European Marine Observation and Data Network (EMODnet) (europa.eu); social media as well as the github of the Open Traits Network https://github.com/open-traits-network/open-traits-network.github.io/issues/253; participants received an overview of state of the art text mining technologies and bioinformatic tools.

The session began with training participants on literature curation workflows, standards, and resources. Participants prepared their own literature and followed the workflow to extract information on traits and species names. Using machine learning algorithms and tools, they evaluated the results and provided feedback on the user interface, usability, and feature requests. Thus, the participants had a unique opportunity to work on the complete workflow from literature preparation to data analysis. Programming was a key part of the session, during which an introduction on how to process data from raw text to the application of algorithms for text mining took place.

The number of registrations for the training amounted to 32 and the participants hailed from the following Institutes/Organisations:

1. Ionian University, Department of Environment
2. Institute of Oceanography, Hellenic Centre for Marine Research
3. Institute of Marine Biology, Biotechnology and Aquaculture, Hellenic Centre for Marine Research
4. Spanish institute of Oceanography
5. Instituto de Hidráulica Ambiental de la Universidad de Cantabria
6. Vlaams Instituut voor de Zee (VLIZ) and Flanders Marine Data Centre (VLIZ-VMDC)
7. University of Salento
8. University of Crete
9. State Key Laboratory of Estuarine and Coastal Research
10. Centre for Environment, Fisheries and Aquaculture Science (CEFAS)
11. The Journal of Life & Environmental Sciences
12. European Marine Biological Resource Centre (EMBRC)
13. Portuguese Institute for Sea and Atmosphere (IPMA)
14. Campus dei Licei Massimiliano Ramadù

The agenda of the training session is available via the link, along with the presentations via the link and the recorded file via the link.

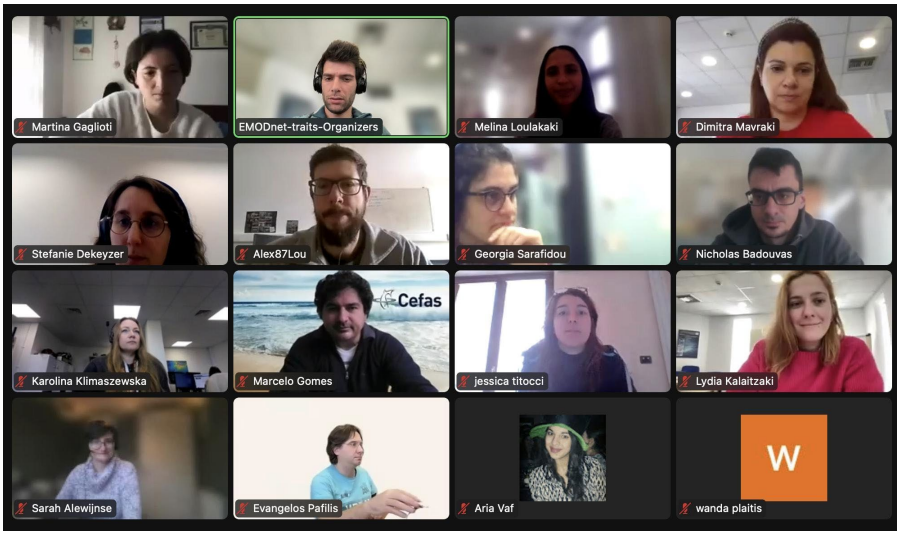


Figure 11. Picture taken through the practical session

7. References

8. Paragkamian S, Sarafidou G, Mavraki D, Pavloudi C, Beja J, Eliezer M, Lipizer M, Boicenco L, Vandepitte L, Perez-Perez R, Zafeiropoulos H, Arvanitidis C, Pafilis E and Gerovasileiou V (2022) Automating the Curation Process of Historical Literature on Marine Biodiversity Using Text Mining: The DECO Workflow. Front. Mar. Sci. 9:940844. doi: 10.3389/fmars.2022.940844
9. Apache Software Foundation. (n.d.). OpenNLP. Retrieved March 24, 2023, from https://opennlp.apache.org/docs/1.9.3/manual/opennlp.html
10. Asgari, E., & Zipunnikov, V. (2021). Natural Language Processing in Healthcare and Biomedicine. Frontiers in Big Data, 4, 626097. https://doi.org/10.3389/fdata.2021.626097
11. Bird, S., Loper, E., & Klein, E. (n.d.). Natural Language Toolkit (NLTK). Retrieved from https://www.nltk.org/
12. Dmitry Mozzherin, Alexander Myltsev, & Harsh Zalavadiya. (2023). gnames/gnfinder: v1.1.1 (v1.1.1). Zenodo. https://doi.org/10.5281/zenodo.7701645, https://github.com/gnames/gnfinder#readme
13. Honnibal, M., & Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. Retrieved from https://spacy.io/
14. IBM. (n.d.). Text mining. Retrieved from https://www.ibm.com/topics/text-mining
15. JensenLab. (n.d.). JensenLab Extract. Retrieved from https://extract.jensenlab.org/
16. Evangelos Pafilis, Pier Luigi Buttigieg, Barbra Ferrell, Emiliano Pereira, Julia Schnetzer, Christos Arvanitidis, Lars Juhl Jensen, EXTRACT: interactive extraction of environment metadata and term suggestion for metagenomic sample annotation, Database, Volume 2016, 2016, baw005, https://doi.org/10.1093/database/baw005
17. John Snow Labs. (2022). SparkNLP - Natural Language Processing Library. Retrieved from https://nlp.johnsnowlabs.com/docs/en/quickstart
18. Juhl Jensen, L. (n.d.). Tagger. GitHub repository. Retrieved from https://github.com/larsjuhljensen/tagger#readme
19. Jurafsky, D., & Martin, J. H. (2020). Speech and Language Processing (3rd ed.). Pearson Education. Retrieved from https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf
20. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A robustly optimised BERT pre-training approach. arXiv preprint arXiv:1907.11692. Hugging Face. (n.d.). RoBERTa. In Transformers: State-of-the-art Natural Language Processing. Retrieved from https://huggingface.co/docs/transformers/model_doc/roberta
21. Manning, C. D., & Schütze, H. (1999). Foundations of statistical natural language processing. MIT press. Retrieved from https://books.google.gr/books?hl=en&lr=&id=3qnuDwAAQBAJ&oi=fnd&pg=PT12&dq=Manning,+C.+D.,+%26+Sch%C3%BCtze,+H.+(1999).+Foundations+of+statistical+natural+language+processing.+MIT+press.+paper&ots=ysM4r0CsLZ&sig=m_PWIGkaS9ksDMj8WXhLWpAyqGk&redir_esc=y#v=onepage&q&f=false
22. Marine Species Traits Wiki. (n.d.). Retrieved from https://www.marinespecies.org/traits/aphia.php?p=sources
23. Marine Species Traits Wiki. (n.d.).Retrieved from https://www.marinespecies.org/traits/wiki/
24. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), 2825-2830.https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

25. Qi, P., Zhang, Y., Zhang, Y., Bolton, J., & Manning, C. D. (2020). Stanza: A Python natural language processing toolkit for many human languages. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (pp. 8-13). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-demos.2
26. Reimers, N., & Gurevych, I. (2020). Sentence-BERT: Sentence embeddings using siamese BERT-networks. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 3982-3992). Association for Computational Linguistics.https://doi.org/10.18653/v1/D19-1410
27. Spacy NER annotation tool. Retrieved from http://agateteam.org/spacynerannotate/. (No longer exists.)
28. Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou and Jun'ichi Tsujii (2012). brat: a Web-based Tool for NLP-Assisted Text Annotation. In Proceedings of the Demonstrations Session at EACL 2012.
29. Stenetorp, P., Pyysalo, S., & Topić, G. (n.d.). Brat: A Web-based Tool for NLP-Assisted Text Annotation. Retrieved from http://brat.nlplab.org/
30. Tecoholic. (n.d.). NER Annotator. Retrieved from https://tecoholic.github.io/ner-annotator/
31. UbiAi. (n.d.). UbiAi Documentation. Retrieved from https://ubiai.tools/Docs
32. Walber. (2014). Precision and Recall [Graph illustration]. Retrieved from https://en.wikipedia.org/wiki/Precision_and_recall
33. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, A., Gugger, S., ... Rush, A. M. (2021). Hugging Face's Transformers: State-of-the-art Natural Language Processing. Retrieved from https://huggingface.co/docs/transformers/model_doc/bert
34. MACHADO, I., RODRÍGUEZ-GALLEGO, L., LESCANO, C. & CALLIARI, D. 2021. Species-specific traits and the environment drive ichthyoplankton fluxes between an intermittently closed-open lagoon and adjacent coastal waters. Estuarine, Coastal and Shelf Science, 261(1), 107549.
35. Tamaki, Akio. "Inhibition of larval recruitment of Armandia sp.(Polychaeta: Opheliidae) by established adults of Pseudopolydora paucibranchiata (Okuda)(Polychaeta: Spionidae) on an intertidal sand flat." Journal of experimental marine biology and ecology 87, no. 1 (1985): 67-82.
36. G. R. Allen, "A Review of the Marine Catfish Genus Paraplotosus (Plotosidae) with the Description of a New Species from North-Western Australia," Raffles Bull. Zool. 46(1), 123–124 (1998)

8. Appendix

8.1 Installations

8.1.1 Gnfinder

The detailed documentation of Gnfinder (Mozzherin et al., 2023) is developed in the following link:

https://github.com/gnames/gnfinder/blob/master/README.md

Step-by-Step Installation Instructions:

1. Use this link to download the .exe file.

   https://github.com/gnames/gnfinder/releases

   Execute the following commands to install Gnfinder as command line app in Windows:

   mkdir C:\bin

   copy path_to\gnfinder.exe C:\bin

2. C:\bin directory to PATH environment variable
3. The following command will run Gnfinder for the test.txt file and the product will be a tsv file:

   gnfinder test.txt -f tsv

4. At the first run of Gnfinder, a configuration file (gnfinder.yml) will be created and it will be located in C:\Users\AppData\Roaming\gnfinder.yml in windows.
5. Finally, the following command will start gnfinder as a web-application and an API server on port 8080 (http://localhost:8080/) for use:

   gnfinder -p 8080

8.1.2 BRAT

Brat is a web-based tool for annotation, visualisation and editing of texts. The tool is open source and free to use. Brat is especially designed for structured annotation, where the notes are not free text but have a fixed form that can be automatically processed and interpreted by a computer (Stenetorp et al., (n.d.)).

The Brat server is implemented in Python, and requires the version 2.5, while the installation script assumes a UNIX-like environment.

Step-by-Step Installation Instructions for a standalone server:

1. Step-by-step installation instructions for a standalone server:
2. Install Visual Studio and then WSL on your Windows machine and choose a Linux distribution.
3. Open the Visual Studio and choose the WSL terminal in the Linux distribution you have installed.
4. Download the latest version of BRAT from the official website (https://brat.NLPlab.org/).
5. Run the installation script in “unprivileged” mode:

   ./install.sh -U

6. Finally, start the standalone server with the command:

   python standalone.py

Notes:
In order to add data, the .txt files must be placed in the data folder and then create an empty .ann file for each

.txt file uploaded.

For the configuration files see brat configuration. Configuring a new corpus is a bit challenging. Each annotation project typically defines its own annotation.conf (where placed: entities, relations, events, attributes). Lastly, defining visual.conf, tools.conf and kb_shortcuts.conf is not necessary, and the system falls back to simple default visuals, tools and shortcuts if these files are not present.

8.1.3 spaCy

spaCy (Honnibal et al., 2017) is an open-source software Python library used in advanced natural language processing and machine learning. It is used to create systems for information extraction, NLU, and text preprocessing prior to deep learning. It provides many built-in features, including deep neural networks.

For the installation (https://spaCy.io/usage).

spaCy is compatible with 64-bit Python 3.6+ and runs on Unix/Linux, macOS/ OS X and Windows.

Step-by-Step Installation Instructions:

1. Open your terminal or command prompt and run the following command to install spaCy:

   pip install spaCy

2. After the installation is complete, run the following command to download the English language model:

   python -m spaCy download en_core_web_sm

3. Verify the installation by running a test script in Python with the commands:

   import spaCy

   NLP = spaCy.load("en_core_web_sm")

   doc = NLP("Hello, world! This is a test.")

   for token in doc:

   print(token.text)

8.1.4 NLTK

Natural Language Toolkit (NLTK) (Bird et al., (n.d.)) is a popular Python library for working with human language data. It offers a variety of information and tools for natural language processing (NLP) activities.

One of the Python versions 3.7, 3.8, 3.9, 3.10, or 3.11 is necessary for NLTK.

For Windows users, the following guide will be useful for the installation of Python 3:

https://docs.python-guide.org/starting/install3/win/#install3-windows

Step-by-Step Installation Instructions:

1. First, run the following command to install NLTK:

   python -m pip install NLTK == 3.5

2. The following commands will download collections, models and corpora:

   import NLTK

   NLTK.download()

System requirements: NLTK is a comprehensive library and as such, it may require additional memory depending on the specific task, dataset or the complexity of the pipeline.

For the installation of transformers in python run the command:

pip install spacy-transformers

After spacy-transformers are installed, users can select the 'bert-base-cased' transformer, for example to fine tune it with spacy can be placed in the spacy configuration file.

The usage is the same as for BERT, but instead of the BERT-base-cased model, the dmis-lab/BioBERT-base-cased-v1.1 model is pre-trained for BioBERT.