INTELCOMP PROJECT

A COMPETITIVE INTELLIGENCE CLOUD/HPC PLATFORM FOR AI-BASED STI POLICY MAKING

(GRANT AGREEMENT NUMBER 101004870)

# D3.3. System for Subcorpus Generation

**Deliverable information**

| | |
|---|---|
| Deliverable number and name | D3.3. System for Subcorpus Generation |
| Work Package | WP3 |
| Lead Partner for deliverable | Universidad Carlos III de Madrid |
| Author | Jesús Cid-Sueiro, UC3M<br>Lorena Calvo-Bartolomé, UC3M |
| Reviewers | Dietmar Lampert, ZSI<br>Ioanna Grypari, ARC |
| Version | 1.1 |

## DISCLAIMER

This document contains a description of the **IntelComp** project findings, work and products. Certain parts of it might be under partner Intellectual Property Right (IPR) rules so, prior to using its content please contact the consortium coordinator for approval.

In case you believe that this document harms in any way IPR held by you as a person or as a representative of an entity, please do notify us immediately.

The authors of this document have taken any available measure in order for its content to be accurate, consistent and lawful. However, neither the project consortium as a whole nor the individual partners that implicitly or explicitly participated in the creation and publication of this document hold any sort of responsibility that might occur as a result of using its content.

The content of this publication is the sole responsibility of **IntelComp** consortium and can in no way be taken to reflect the views of the European Union.

The European Union is established in accordance with the Treaty on European Union (Maastricht). There are currently 27 Member States of the Union. It is based on the European Communities and the member states cooperation in the fields of Common Foreign and Security Policy and Justice and Home Affairs. The five main institutions of the European Union are the European Parliament, the Council of Ministers, the European Commission, the Court of Justice and the Court of Auditors.

(http://europa.eu.int/)

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 101004870.

## CONTENTS

# ACRONYMS

| | |
|---|---|
| **BERT** | Bidirectional Encoder Representation from Transformer |
| **GUI** | Graphical User Interface |
| **I/O** | Input/Output |
| **LDA** | Latent Dirichlet Allocation |
| **NLP** | Natural Language Processing |
| **SBERT** | Sentence BERT |
| **PU** | Positive-unlabeled |

## GLOSSARY

| | |
|---|---|
| **CLASS / CATEGORY** | In the context of machine learning, each of the possible choices of a classifier. A classifier for a specific TARGET DOMAIN has two CLASSES only: CLASS 1 (for documents that belong to the TARGET DOMAIN) and CLASS 0 (for those that do not). Class 1 and 0 are also named the **positive and negative CLASSES**, respectively. |
| **CORPUS** | A collection of text documents |
| **DOMAIN** | A field or subfield of knowledge. |
| **DOMAIN CLASSIFIER** | A computational system that, given any input document, determines if it belongs to the TARGET DOMAIN or not. |
| **LABEL** | An indicator of CLASS membership. It is used as a reference to train or evaluate a classifier. In classification with two CLASSES only, the possible LABEL values are 1 and 0, which are named the positive and negative LABELS, respectively. Ideally, the LABEL is equal to the true CLASS of the document, but if the labeling process is imperfect, some LABELS in the CORPUS can indicate the wrong CLASS. |
| **SUBCORPUS** | A subset of documents from a CORPUS. |
| **TARGET DOMAIN** | The DOMAIN of interest to a user. |
| **UNLABELED DOCUMENT** | A document that has no LABEL. The CLASS of the document is, thus, unknown. |
| **CLASS PREDICTION** | The CLASS assigned by a DOMAIN CLASSIFIER to a document. |

# EXECUTIVE SUMMARY

The system for subcorpus generation (domain classifier) is aimed at facilitating the classification of all documents from a given corpus according to a category that is specified by the user (an expert in the domain). Therefore, the software includes not only methods implementing algorithms for the classification of documents, but also all the necessary tools to facilitate the interaction with the users of the system.

The system provides facilities for the expert to (1) Specify the target category (the “domain”) for classification, and (2) revise and correct the machine classifications through a tool for the annotation of specific documents.

The software has been structured in a series of classes and methods to be integrated in the Interactive Model Trainer from IntelComp. In addition, we have implemented a standalone python application that can be used to test the behavior of the classifiers, the data flow and the user interaction before the definite integration in the model trainer.

The core of the software consists of python modules structured as a collection of python classes in charge of the main processing steps:

1. **Pre-classification (document selection)**: The user can specify the target domain, by means of one of the three methods provided by the system: (a) the name of the domain, (b) A list of characteristic keywords from the domain or (c) a weighted list of topics from a topic model previously computed from the corpus. The document selectors for (a) and (b) apply pretrained models based on Transformers, a state-of-the-art Deep Learning technology that has been successful in many NLP tasks.

2. **Positive-unlabeled (PU) learning:** the system trains a classifier applying a supervised machine learning algorithm, trying to discriminate between the documents selected in the first step and the rest of documents in the corpus.

3. **Retraining**: Since some of the non-selected documents may belong to the target domain (because the pre-classifier system has been designed with limited supervision information), the system needs some user interaction to improve the classification performance. To do so, an **active learning** loop has been implemented: the system selects a reduced set of documents and requests labels from the user through the annotation interface. These labels are used to retrain the classification module based on transformers..

# 1. INTRODUCTION

Some of the tools from the IntelComp project will facilitate the analysis of data sets from any field. However, the machine learning models used, and specifically the topic-modeling-based analysis, obtain more intuitive and potentially more useful results when their training is restricted to documents belonging to a well-defined domain. For this reason, Task 3.3 proposes to design a system that allows identifying the documents belonging to a target domain through an expert-in-the-loop approach that allows integrating expert knowledge in the classification model.

The aim of this document is to present the *System for Subcorpus Generation*, that is, the software that will include the intelligence for the selection of the relevant documents for a selected field, and a Graphic User Interface (GUI) tool to facilitate expert annotation.

# 2. DOMAIN CLASSIFICATION

The system for subcorpus generation is aimed at facilitating the classification of all documents from a given corpus according to a category that is specified by the user (an expert in the domain). The software includes not only methods implementing algorithms for the classification of documents, but also all the necessary tools to facilitate the interaction with the users of the system.

Since the categories are expected to represent specific domains of knowledge in a research field, the subcorpus generator will also be denominated as a domain classifier. In this document, “domain classification” and “subcorpus generation” will be used indistinctly.

The system provides facilities for the expert to:

- Specify the target domain for classification

- Revise and correct the machine classifications through a tool for the annotation of specific documents.

All the software has been structured in a series of classes and methods aimed at facilitating the integration in the IntelComp platform. The core of the software is a python library of classes and methods for domain classification, but it is also a fully functional standalone python application that can be executed to test the user interaction, the data flow and the system performance. During the integration phase of the IntelComp project, a dockerized version will be created to be used as part of the Interactive Model Trainer.

The sequence of steps involved in the process of classifying the documents of a given corpus can be summarized as follows (see Figure 1):

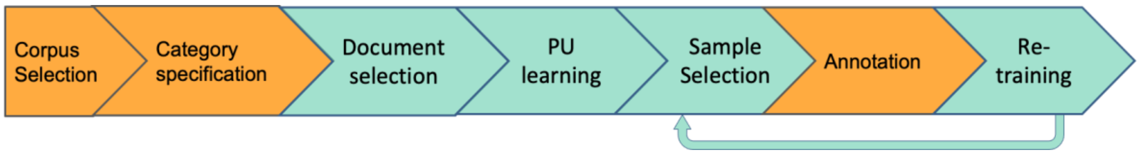


**Figure 1: The process of domain classification. It involves a sequence of human (in orange) and machine (in green) steps. The retraining step is carried out every time the expert user annotates new documents.**

1. **Corpus selection**: The user selects the input corpus

2. **Category specification**: The user specifies the target categories, by means of one of the three methods currently provided by the system:

    a. The name of the category: e.g., "Biotechnology", "Artificial Intelligence", "Photonics".

    b. A list of characteristic keywords from the target category.

    c. A weighted list of topics. This option requires that a topic model had been previously computed from the corpus. The user selects one or several topics of interest, assigning a weight to each one of them that indicates the relevance of the topic for the target domain.

3. **Pre-classification (document selection)**: The system pre-classifies the documents in the corpus according to the information provided by the user in step 2. To do so, the system selects a portion of the corpus containing the documents that are the most relevant for the target domain.

4. **PU learning:** the system trains a classifier applying a supervised machine learning algorithm. To do so, a positive label is assigned to the pre-classified documents, and a negative label to the rest of the documents for the negative, and these labels are used to train the classifier.

    Note that, ideally, the learning algorithm would be more efficient if all documents with a positive label belong to the target domain, and all documents with a negative label do not belong to the target domain. Unfortunately, the step 3 (document selection) will provide a reduced subset of documents from the target domain only and many documents from the target domain might have a negative label. Thus, the true class of the documents with negative labels is uncertain. For this reason, this is usually called a Positive-Unlabeled (PU) learning problem.

5. **Sample selection**: Since documents from the target domain may have negative labels, the system needs some interaction with the user to improve the classification

performance. To do so, it enters into an **active learning** loop that could be repeated for several interactions

In the first step of the loop, the system selects a reduced subset of documents to request the user feedback.

6. **Annotation (user feedback):** the user examines the documents provided by the system and reviews the category labels that had been incorrectly assigned by the classification algorithm

7. **Retraining**: the system uses the labels provided by the user to update the classifier model and improve the classification accuracy.

The retraining step will be repeated every time the user provides new labels through the annotation interface.

## 3. PROCESSING PIPELINE

Figure 2 details the processing pipeline of the system. The input and outputs, and the components of the system are described below.

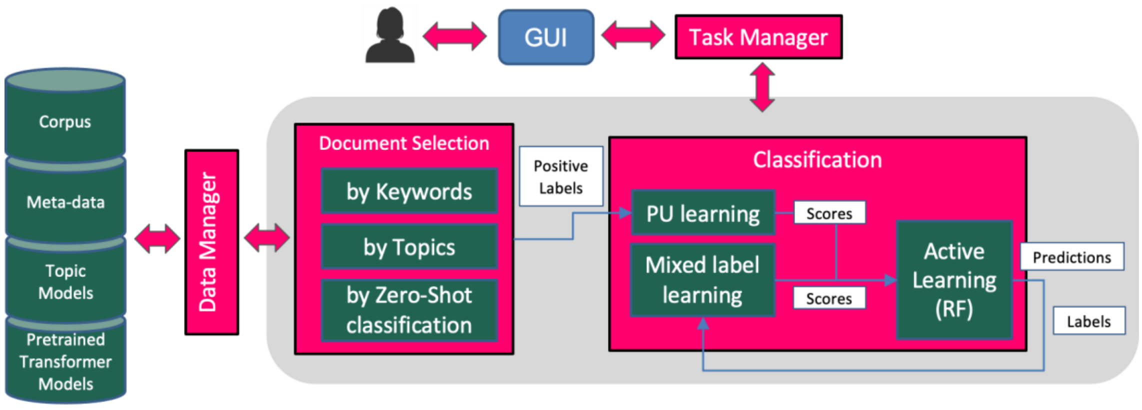


**Figure 2: Processing pipeline of the system for subcorpus generation.**

The software has a modular structure and the main functionality of each component in the pipeline has been encapsulated in specific python classes or methods. Thus, the design is flexible enough to allow further improvements of specific comments. The system uses some specific pretrained models based on transformers, but they are chosen through a configuration file, so that different pretrained models can be easily tested.

## 3.1. Source data

**Input corpus**

The main input to the system is a database containing one or several corpora of text documents and their associated information (metadata, topic models or pre-trained transformer models).

In the current version of the software, a data manager module contains methods to load two specific corpora that have been used to test the system. After the integration in the IntelComp platform, the data manager will be adapted to use the corpus utilities available in the Data Catalog and in the Interactive Model Trainer.

The classification modules assume all documents in the corpus contain English text only. Therefore, classification of documents from the IntelComp platform will be based on original documents in English, or in the English versions produced by the machine translation system (Deliverable 3.2).

For most modules, classification will be based on transformer models [Vaswani, 2017]. They do not require any specific text preprocessing, so the raw text (without word tokenization) can be directly applied to them. However, some of the text preprocessing tools from the NLP modules available at IntelComp could be useful to improve the quality of the text sources (e.g., text cleaning).

For the topic selection module, a topic model representation of documents must be available. During our test, we used topic models based on the Latent Dirichlet Allocation (LDA), which will be one of the models integrated in the IntelComp platform. However, the software can be applied to any other model providing a vector representation of all documents in a common vector space.

**User data**

The other source of information is the expert user, who must interact with the system through a graphic user interface (GUI), to specify the target category and, also, to fine-tune the behavior of the domain classifiers through an active learning mechanism. A task manager module is in charge of attending the user requests by launching the appropriate processing tasks.

## 3.2. Output

The output of the system consists of a class prediction for every document in the input corpus, in relation to the category specified by the user.

The software allows the user to specify several target categories for the given corpus. The system will produce a data frame containing the predictions for each document and each of the target categories.

## 3.3. Document selection

The first step in the process of domain classification is the pre-classification of documents according to the category specified by the user. The system will select $P$ documents with the highest relevance score for the target category.

The number of selected documents, $P$, depends on two configurable parameters that control the size and minimum relevance score of the set of selected documents.

There are three modules in the document selection system, which are described below:

**Selection by Domain Name.**

This module tries to classify the documents in the corpus using the name of the category only. To do so, it runs a *zero-shot* classifier. A zero-shot is a classifier whose parameters are adjusted without using any training data. Since the only information available is the name of the category, such classifiers are the best suited for this type of document selection.

The module applies a specific type of zero-shot classifiers based on an entailment approach [Yin, 2019] using transformers [Vaswani, 2017]: for each input document, the classifier evaluates if it entails a sentence (hypothesis) like "this document is about X", where X is the domain name. If so, the document is taken as relevant for the selected domain. Otherwise, it is considered irrelevant. The top $P$ documents with the highest entailment score are selected.

Using pre-trained transformer models, the zero-shot classifier does not need a specific training for each domain name, which facilitates a quick system response. The selection of the transformer model can be done through the configuration file.

**Selection by Keywords.**

When the selection is given by a list (of arbitrary size) of relevant keywords, the document selection is based on the computation of a similarity measure between the input documents and the comma-separated string of keywords.

Two similarity measures have been implemented:

- **Cosine distance**: a cosine distance between the embedding of both the input documents and the keywords string using transformers. The current implementation uses pretrained SBERT transformers [Reimers, 2019]. Our choice is based on the selection of a computationally efficient model based on state-of-the-art transformer models like BERT.
- **Keyword count**: the total number of occurrences of the keywords in the input document.

The top $P$ documents with the highest similarity scores are selected.

**Selection by Topics.**

Selection by topics is aimed at providing the user with the capability of defining a target category using the topic descriptions of a topic model previously computed by the IntelComp platform.

By inspecting the top relevant words of each topic, the user can define a target category as a weighted list of topics, i. e. a list of tuples in the form

$$(T_0, w_0), (T_1, w_1), \dots, (T_n, w_n)$$

where $T_i$ is a topic and $w_i$, its weight. If $\mathbf{w} = (w_0, w_1, \dots, w_n)$ is the weight vector and $\mathbf{d}$ is the topic vector of the input document, the relevance score is computed as the weighted sum of topic weights

$$\text{score} = \mathbf{w}^T \cdot \mathbf{d}$$

The top $P$ documents with the highest similarity scores are selected.

## 3.4. PU learning

As a result of pre-classification, the system obtains the set of $P$ documents from the positive class (the target category). Since the document selection process takes only the most relevant documents for the target category, which may be a portion of the total number of documents from the positive class only, the rest of the documents cannot be taken as documents from the negative class. This states a PU (Positive vs Unlabeled) -learning problem, where the labeled dataset contains documents from the positive class only.

Assuming that the target domain is sufficiently specific, we can assume that the classification problem is unbalanced, and the prevalence of the positive class is small. Therefore, although the unlabeled dataset is a mixture of the positive and negative class distributions, the weight of the positive part in the mixture is small, and the marginal distribution approximates the negative class distribution.

Thus, in a preliminary step, a binary classification is trained taking the documents in the unlabeled dataset as from the negative class.

The PU classifier is based on the addition of an additional output layer to a transformer model: documents are embedded into a vector space using BERT transformers [Devlin, 2018], and a logistic regression model is adjusted using the embeddings as inputs (see Figure 4). More complex models have been considered, but a single-layer model seems more adequate, due to the high dimensionality of the embedding (typically, 768 dimensions) and the number of clean labels, which can be expected to be small.

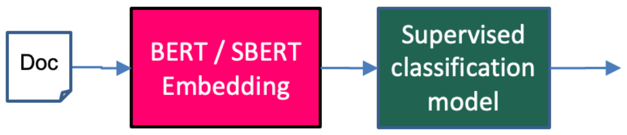


**Figure 3: Pre-classification system. Documents are mapped to a vector space using a (pretrained) BERT transformer model. A single-layer classifier is added to the last layer of the embedding model. The preliminary labels obtained from the document selection process are used to pre-train the model through a supervised learning mechanism.**

## 3.5. Active learning

The active learning module selects a pre-defined number of documents at random and shows them to the expert user for labeling. The sample of documents can be taken at random, but this is not usually a good choice for several reasons:

1. If the target domain is a minority class, random sampling will provide documents from the majority class with higher probability.
2. The user is mainly interested in documents from the target domain and, thus, the sample should contain most documents from the target domain. By showing most documents from the target domain, the annotation process can be integrated in a more transparent way into the natural navigation of the user through the documents in the domain.
3. From the point of view of the efficiency of the active learning process, the selection of documents must depend on the scores of the classifier. During the initial learning rounds, the quality of the classifier model is unknown and, thus, testing the validity of the classification of documents with the highest or the lowest scores (which correspond to documents classified into the positive or the negative class, respectively, with the highest confidence values) should be prioritized. In later annotation rounds, other documents with lower classification confidence will be more useful for model refinement.

Given the above considerations, an active learning algorithm has been implemented that takes as input the scores of all documents that have not been annotated before, and proceeds as follows:

1. Documents are sorted from highest to lowest score.
2. Select $M_+$ documents (without replacement), taking $n$-th document with probability proportional to $q^n$.
3. Select $M_-$ documents (without replacement) from the positive class, taking $n$-th document with probability proportional to $(1-q)^n$.

Parameters $M_+$ and $M_-$ control the number of samples taken from the highest and the lowest scores, respectively. Typically, $M_+ > M_-$ , in such a way that the annotator will likely receive more samples from the target domain for annotation. Parameter $q \in [0, 1]$ controls the randomness of the document selection, ranging from $q = 0$ (purely random sampling) to $q = 1$ (deterministic sampling, taking the top $M_+$ scores and the bottom $M_-$ scores).

## 3.6. Mixed label learning

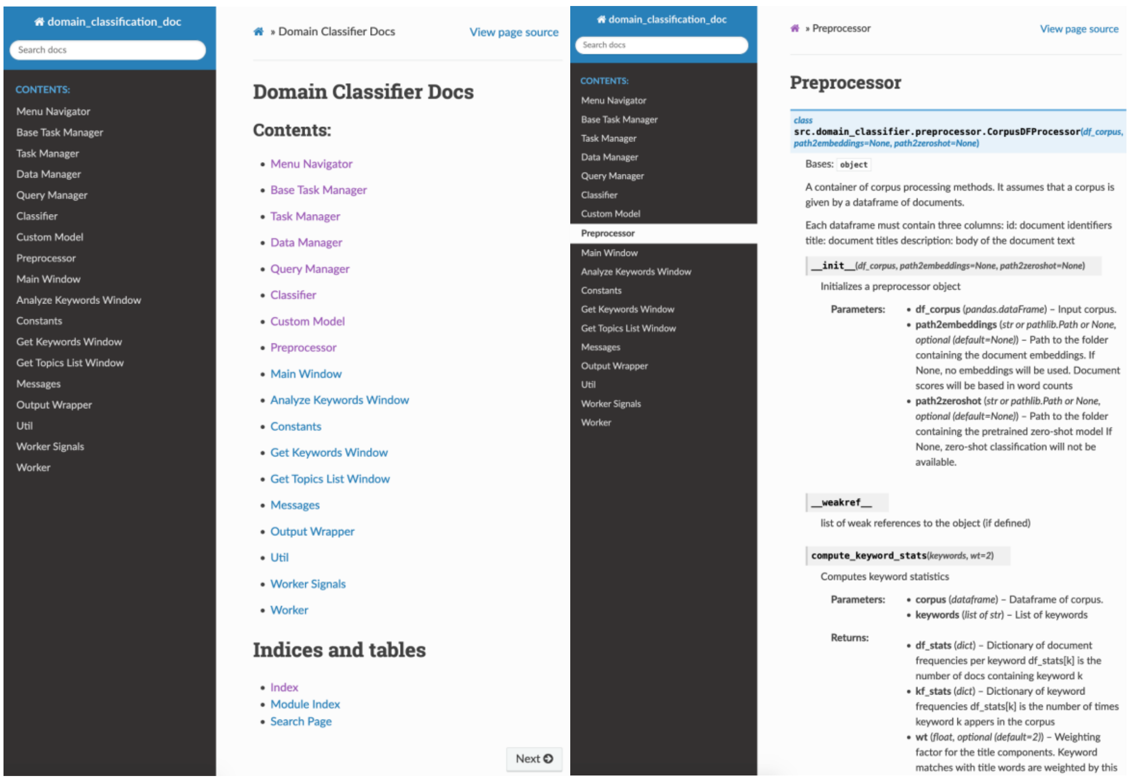
The clean labels provided by the user must be combined with the PU labels (i.e. the labels provided by the document selection system) in order to fine-tune the classifier model. Since the number of clean labels after a few user interactions will be, in general, a small fraction of the total number of PU labels, the mixed label learning module retrains the classifier module in

Figure 4 using a sample weighting mechanism. The weight assigned to the clean labels is a configurable parameter of the application.

# 4. SOFTWARE

## 4.1. Software architecture

The software package can be downloaded from the IntelComp repository in GitHub[1]. It contains all software modules and its documentation. The documentation has been generated using Sphinx in ReadTheDocs format and it has been published in the GitHub page[2]. Fig. 4 shows a snapshot of the main page of the documentation, and the page describing one of the main classes.



Domain Classifier Docs

Contents:

- Menu Navigator
- Base Task Manager
- Task Manager
- Data Manager
- Query Manager
- Classifier
- Custom Model
- Preprocessor
- Main Window
- Analyze Keywords Window
- Constants
- Get Keywords Window
- Get Topics List Window
- Messages
- Output Wrapper
- Util
- Worker Signals
- Worker

Indices and tables

- Index
- Module Index
- Search Page

Preprocessor

*class* `src.domain_classifier.preprocessor.CorpusDFProcessor`*(df_corpus, path2embeddings=None, path2zeroshot=None)*

Bases: `object`

A container of corpus processing methods. It assumes that a corpus is given by a dataframe of documents.

Each dataframe must contain three columns: id: document identifiers title: document titles description: body of the document text

`__init__`*(df_corpus, path2embeddings=None, path2zeroshot=None)*

Initializes a preprocessor object

Parameters:
- **df_corpus** (*pandas.dataFrame*) – Input corpus.
- **path2embeddings** (*str or pathlib.Path or None, optional (default=None)*) – Path to the folder containing the document embeddings. If None, no embeddings will be used. Document scores will be based in word counts
- **path2zeroshot** (*str or pathlib.Path or None, optional (default=None)*) – Path to the folder containing the pretrained zero-shot model If None, zero-shot classification will not be available.

`__weakref__`

list of weak references to the object (if defined)

`compute_keyword_stats`*(keywords, wt=2)*

Computes keyword statistics

Parameters:
- **corpus** (*dataframe*) – Dataframe of corpus.
- **keywords** (*list of str*) – List of keywords

Returns:
- **df_stats** (*dict*) – Dictionary of document frequencies per keyword df_stats[k] is the number of docs containing keyword k
- **kf_stats** (*dict*) – Dictionary of keyword frequencies df_stats[k] is the number of times keyword k appers in the corpus
- **wt** (*float, optional (default=2)*) – Weighting factor for the title components. Keyword matches with title words are weighted by this

**Figure 4: Snapshots of the software documentation. (Left): Main page. (Right) Sample page of the documentation of one of the main classes.**

[1] https://github.com/IntelCompH2020/domain_classification
[2] https://intelcomph2020.github.io/domain_classification/index.html

The main components of the folder structure in the software repository are:

- docs/: Documentation folder.
- src/: The python software package. Contains all classes and methods.
- Main scripts: two executable python scripts that can be used to test all software modules through a terminal/command window (main_domain_classifier.py) or through a PyQT5 GUI (main_gui.py).

The classes and methods in folder src/ are structured in several modules:

- domain_classifier/: It contains the main classes in charge of text processing and classification. These are the classes that should be integrated in the IntelComp platform. The diagram of these classes is shown in Figure 5. It contains the following classes:
    - CorpusProcessor: contains all methods related to the document selection process
    - CorpusDFProcessor: extends CorpusProcessor methods to process data from Pandas dataframes.
    - CorpusClassifier: contains all methods related to the pre-classification, active learning, and retraining using machine learning models.
    - Finally, four custom classes have been used, that are adaptations of the code for Roberta transformer model for sentence-level classification tasks
        - CustomEncoderLayer: Custom encoder layer of transformer for classification
        - CustomModel.
        - CustomClassificationHead .
        - CustomDataset:

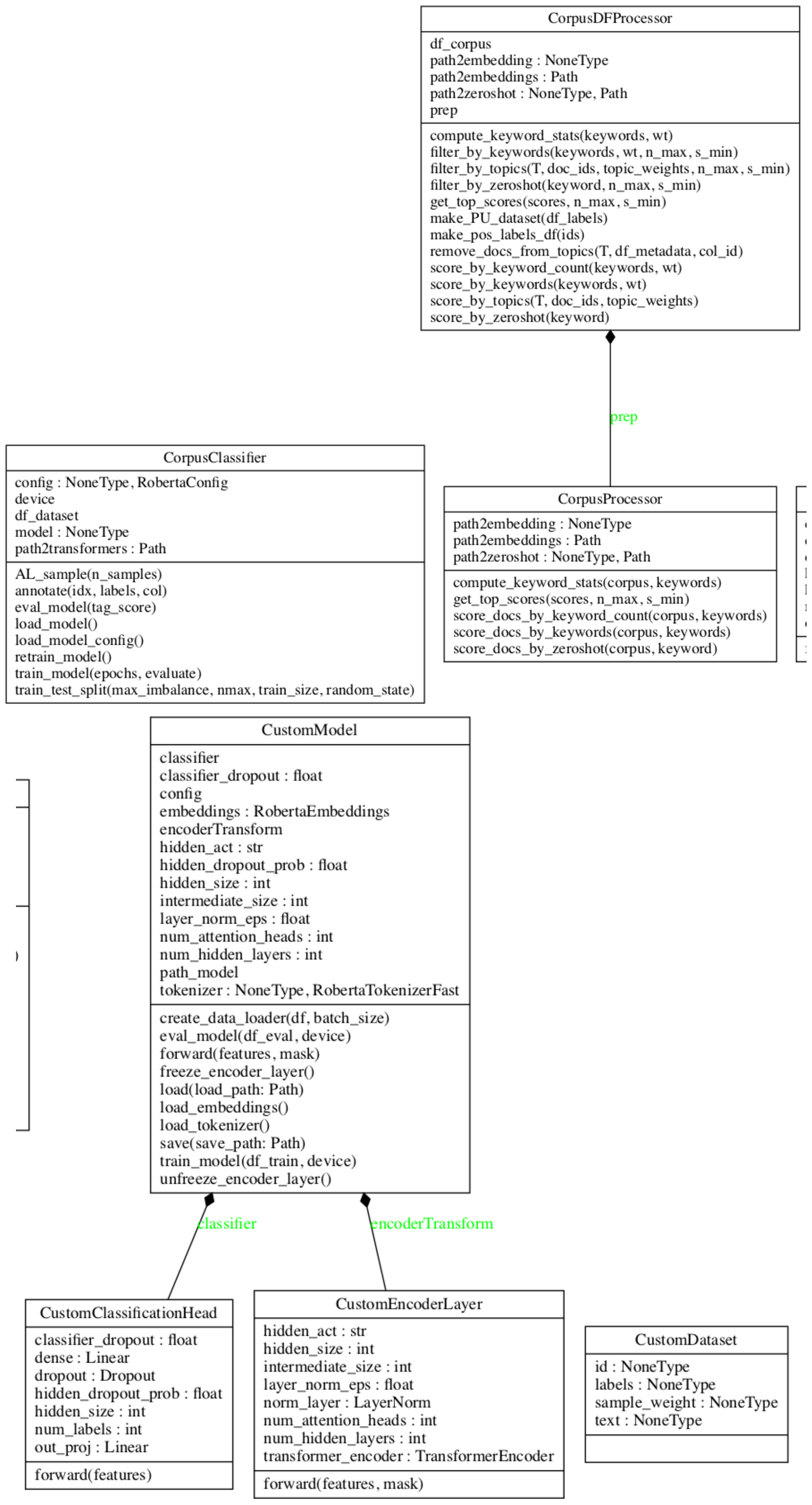

CorpusDFProcessor
df_corpus
path2embedding : NoneType
path2embeddings : Path
path2zeroshot : NoneType, Path
prep
compute_keyword_stats(keywords, wt)
filter_by_keywords(keywords, wt, n_max, s_min)
filter_by_topics(T, doc_ids, topic_weights, n_max, s_min)
filter_by_zeroshot(keyword, n_max, s_min)
get_top_scores(scores, n_max, s_min)
make_PU_dataset(df_labels)
make_pos_labels_df(ids)
remove_docs_from_topics(T, df_metadata, col_id)
score_by_keyword_count(keywords, wt)
score_by_keywords(keywords, wt)
score_by_topics(T, doc_ids, topic_weights)
score_by_zeroshot(keyword)
prep
CorpusClassifier
config : NoneType, RobertaConfig
device
df_dataset
model : NoneType
path2transformers : Path
AL_sample(n_samples)
annotate(idx, labels, col)
eval_model(tag_score)
load_model()
load_model_config()
retrain_model()
train_model(epochs, evaluate)
train_test_split(max_imbalance, nmax, train_size, random_state)
CorpusProcessor
path2embedding : NoneType
path2embeddings : Path
path2zeroshot : NoneType, Path
compute_keyword_stats(corpus, keywords)
get_top_scores(scores, n_max, s_min)
score_docs_by_keyword_count(corpus, keywords)
score_docs_by_keywords(corpus, keywords)
score_docs_by_zeroshot(corpus, keyword)
CustomModel
classifier
classifier_dropout : float
config
embeddings : RobertaEmbeddings
encoderTransform
hidden_act : str
hidden_dropout_prob : float
hidden_size : int
intermediate_size : int
layer_norm_eps : float
num_attention_heads : int
num_hidden_layers : int
path_model
tokenizer : NoneType, RobertaTokenizerFast
create_data_loader(df, batch_size)
eval_model(df_eval, device)
forward(features, mask)
freeze_encoder_layer()
load(load_path: Path)
load_embeddings()
load_tokenizer()
save(save_path: Path)
train_model(df_train, device)
unfreeze_encoder_layer()
classifier
encoderTransform
CustomClassificationHead
classifier_dropout : float
dense : Linear
dropout : Dropout
hidden_dropout_prob : float
hidden_size : int
num_labels : int
out_proj : Linear
forward(features)
CustomEncoderLayer
hidden_act : str
hidden_size : int
intermediate_size : int
layer_norm_eps : float
norm_layer : LayerNorm
num_attention_heads : int
num_hidden_layers : int
transformer_encoder : TransformerEncoder
forward(features, mask)
CustomDataset
id : NoneType
labels : NoneType
sample_weight : NoneType
text : NoneType

**Figure 5: Diagram of classes related to text processing and classification.**

- menu_navigator/: Contains class MenuNavigator, which reads and interprets the menu structure (defined in a configuration file) that will be used by the main scripts. See Figure 6.

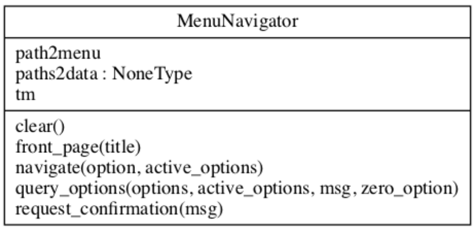

MenuNavigator
path2menu
paths2data : NoneType
tm
clear()
front_page(title)
navigate(option, active_options)
query_options(options, active_options, msg, zero_option)
request_confirmation(msg)

**Figure 6: Attributes and methods of MenuNavigator class.**

- graphical_user_interface/: contains all classes and methods related to the GUI. See Figure 7

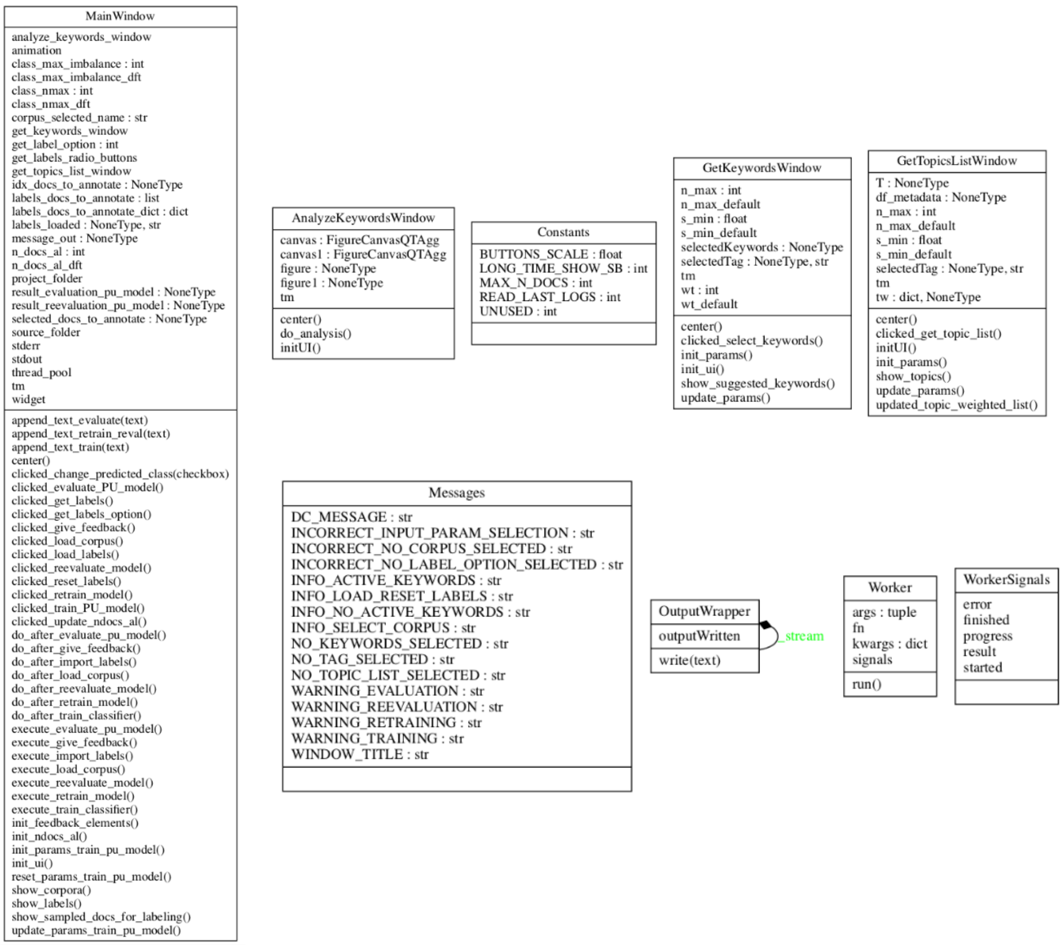


**Figure 7: Classes, attributes and methods for the GUI**

- utils/: contains some useful auxiliary methods.
- Control and I/O classes: in charge of read/write operations, user interaction and task execution, organized in different files:
  - TaskManager. A class inherited from BaseTaskManager. It is in charge of creating the objects and calling the methods required to carry out all of the processing tasks. There are two subclasses inherited from it: TaskManagerCMD (which includes methods required for user interaction through a terminal window) and TaskManagerGUI (which includes methods required for user interaction through the GUI.
  - DataManager. Contains all read/write methods. It provides support to the task managers to read the corpus data (texts and labels) and write the results. This is a provisional method that can be used to test the software using a particular test datasets. It should be replaced by the appropriate class to carry out all read and write operations according to the IntelComp data structure in the integration phase of the project.
  - QueryManager: Contains all methods required to interact with the user through a terminal window.

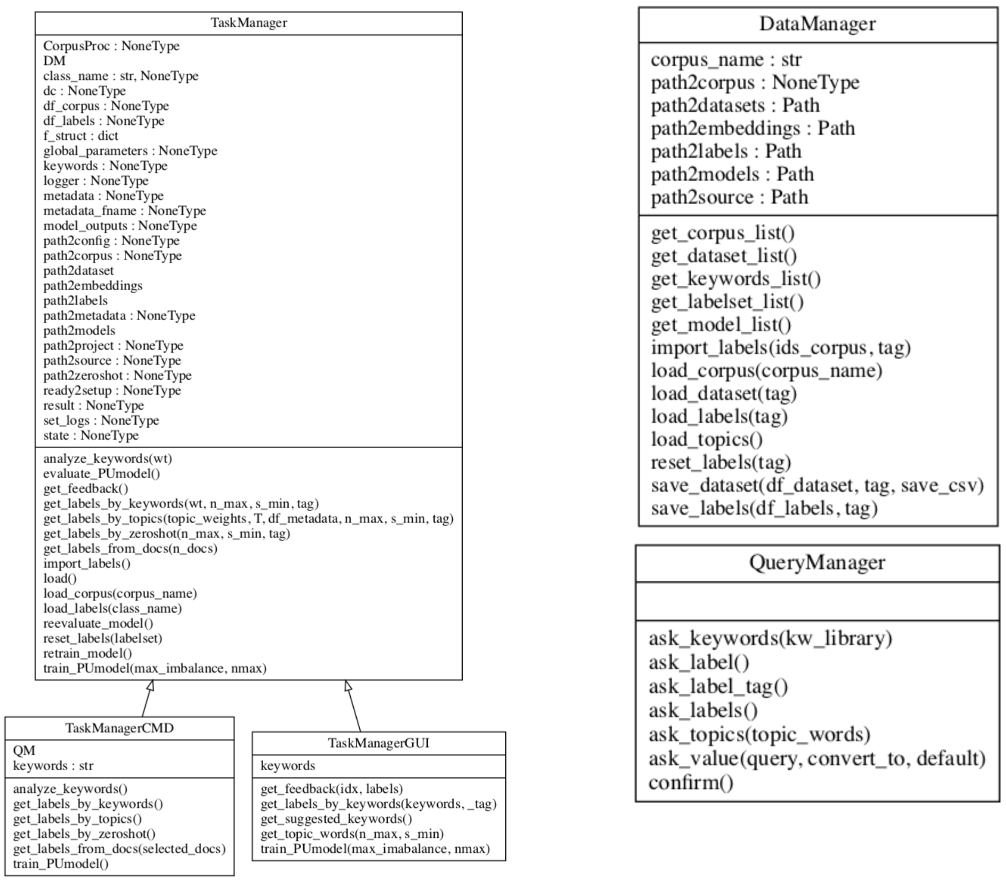


**Figure 8: Classes, attributes and methods for the task and data and user query management**

## 4.2. Software requirements

### 4.2.1. Python packages requirements

The main libraries required to run the python application are listed below. The table also shows the library versions integrated in the latest code version.

**Table 1: Python package requirements.**

| Package | Version |
|---|---|
| matplotlib | 3.3.4 |
| numpy | 1.20.1 |
| pandas | 1.2.4 |
| PyQT5 | 5.15.4 |
| PyYAML | 6.0 |
| scikit_learn | 1.0.1 |
| scipy | 1.6.2 |
| simpletransformers | 0.63.3 |
| openpyxl | 3.0.9 |
| torch | 1.10.2 |
| torchaudio | 0.10.2 |
| torchvision | 0.11.3 |

### 4.2.2. Data source requirements

All the corpora that can be utilized as a training corpus for the Domain classifier, together with its associated keywords, labels, topic models and other additional metadata, must be provided within a folder as one of the application's inputs. With this purpose, such a folder must be composed by one subfolder per corpus, each of them being organized into all or some of the following directories:

1. **corpus:** A directory containing the actual corpora and additional data related to them.
2. **labels**: Directory containing the xlsx files which are composed of the IDs associated with the documents that have been manually classified and that will be utilized for supervised training. These labels define the category *"imported"*, i.e., the category acquired through the importing of labels from a source file.
3. **lemmatized_corpus:** Directory holding the files which contain each of the lemmatized corpora that are included in the corpus folder. For the time being, this folder is not utilized, but it may be useful for a later deployment.

4. **queries:** Directory containing the files with the keywords that are going to be used as suggestion for the selection of a subcorpus based on keywords.
5. **topic_model:** Directory containing the representation of the topic models that will be used for the selection of a subcorpus based on a topic selection function.
6. **zips:** Original compressed files. They are not used in the application.

These folders are relevant when using this software as a standalone application. During the integration phase, when the software will be embedded in the Interactive Model Trainer tool, it will be necessary to reconsider this folder structure in the context of such application.

## 4.3. Execution commands

For the user to start any version of the application, the following command needs to be executed:

```
$ python main_script

    --p project_folder

    --source datasets_folder

    —-zeroshot zero_shot_folder
```

where:

- **main_script** refers to the script that relates to the version of the application to be executed. Use
    - main_domain_classifier.py for the command line application,
    - main_gui.py, for the GUI.
- **project_folder** is the path to a new or an existing project in which the application's output will be saved.
- **datasets_folder** is the path to the source data folder.
- **zero_shot_folder** is the path to a folder containing a pre-trained zero-shot model utilized for the selection of a subcorpus from a category name.

Note that for the case of the graphical user interface, the application can also be invoked without parameters, being possible to select them from the application's front page as follows:

```
$ python main_gui.py
```

This command will open the start-up page from the GUI that is shown in Figure 9.

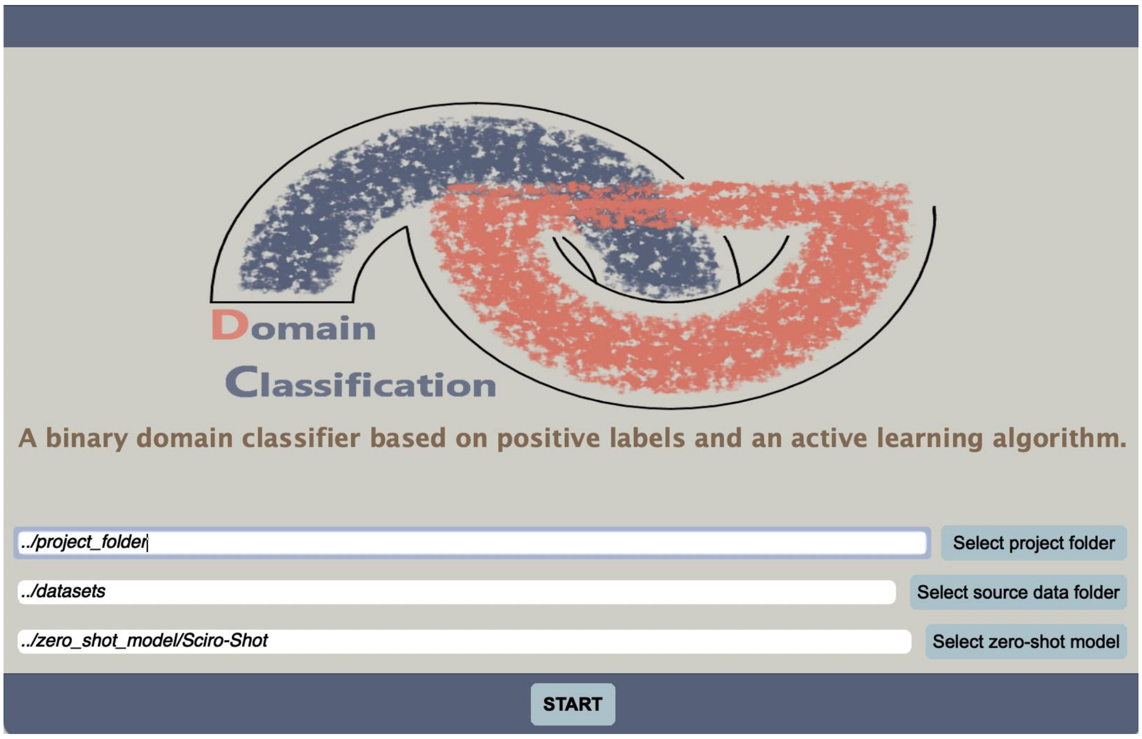

Domain
Classification
A binary domain classifier based on positive labels and an active learning algorithm.
../project_folder
Select project folder
../datasets
Select source data folder
../zero_shot_model/Sciro-Shot
Select zero-shot model
START

**Figure 9: Graphical user interface start-up page.**

If the application is invoked with parameters, the project folder, dataset folder and zero-shot model paths are written on their respective text boxes (i.e. white spaces located at the left of each selection button), as shown in Figure 9, being then possible to change the selection by clicking on their respective associated button. In case the application is invoked without parameters, the text boxes are shown empty, but the selection is approached in the same way, as it can be seen in Figure 10. In each of the three cases, the user's file system is popped-up so it is possible to easily search for the project/dataset/zero-shot folder.

For the case of the command-line application, its main menu is shown immediately after the invocation of the execution command. As for the graphical user interface, the user must first click on the *"Start"* button in order to proceed to the application's main window.

## 4.4. Project folder structure

If the **project_folder** does not exist, the application creates it, along with the file and folder structure required to store the output files: the default structure consists of the following:

- datasets/: It will store a csv file for each dataset used by the application with the selected corpus. Each csv file will contain all information that is relevant for the processing, classification, labeling or evaluation of the classifier models.
- embeddings/: Contains the transformer-based embeddings of the selected documents from the corpus.
- labels/: For each target category, a file containing the documents preclassified by the document selection subsystem. The data from these files will be integrated into the dataset files after training.

- models/: it will contain one output subfolder per target category. Each subfolder will contain the updated classifier model.

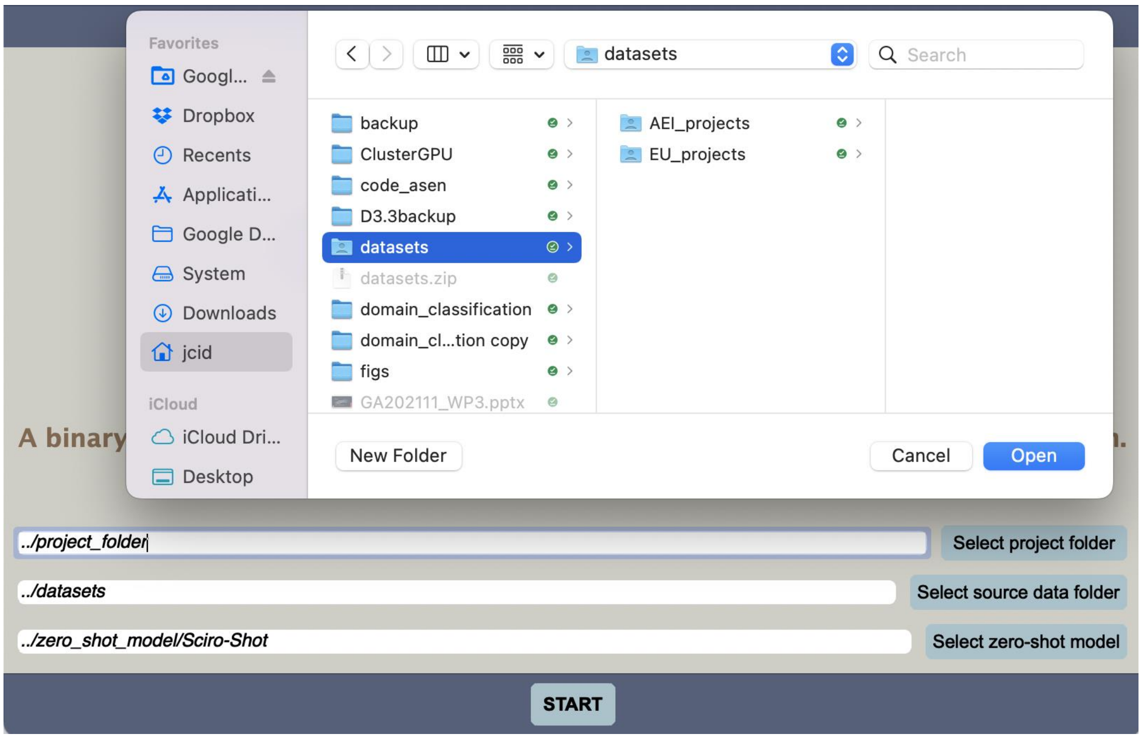


Figure 10: Graphical user interface start-up page with file system open after the clicking of the "Select source data folder" button.

- output/: other output file (not used)
- metadata.yaml: a file with metadata that stores the status of the project and some metadata related to each of the target categories.
- msgs.log: log file of the latest code execution with this project.
- parameters.yaml: the configuration file of the project.

These folders are relevant when using this software as a standalone application. During the integration phase, when the software will be embedded in the Interactive Model Trainer tool, it will be necessary to reconsider this folder structure in the context of such application.

### 4.4.1. Configuration file

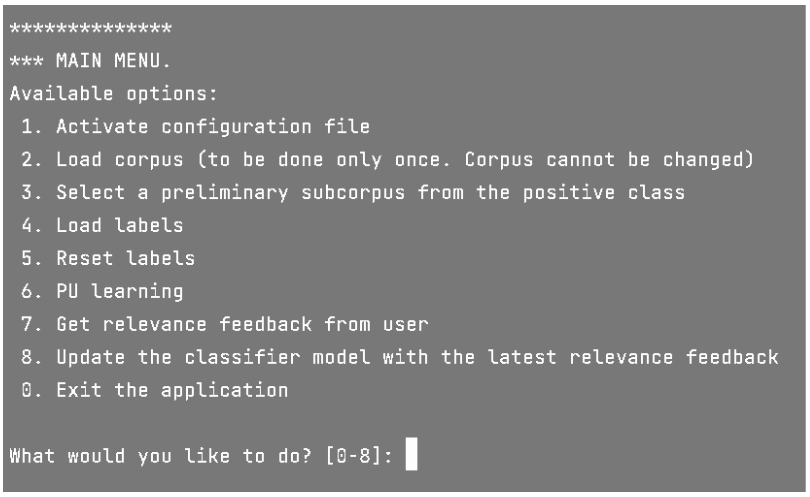
The first time the application is run, a copy of the configuration file is stored in the project folder with name parameters.yaml. The list of parameters, its function, and the default values used for application testing are shown in Annex A.

## 4.5. Application structure

The functionalities of both application's versions are equivalent, the only difference between them being the way in which the information is presented to and input is taken from the user.

### 4.5.1. Command line application

The application version is based on a menu, as shown in Figure 11. The user can navigate through the application by writing the number associated with each functionality. When necessary, the application will request input from the user.



```
**************
*** MAIN MENU.
Available options:
 1. Activate configuration file
 2. Load corpus (to be done only once. Corpus cannot be changed)
 3. Select a preliminary subcorpus from the positive class
 4. Load labels
 5. Reset labels
 6. PU learning
 7. Get relevance feedback from user
 8. Update the classifier model with the latest relevance feedback
 0. Exit the application

What would you like to do? [0-8]:
```

**Figure 11: Command-line application.**

### 4.5.2. Graphical user interface

The GUI main window is composed of three main views, as depicted in Figures 12, 13 and 14. In all of them, we can find informative tooltip buttons orienting the user through the actions to be carried out at each view.

#### 4.5.2.1. View for the corpus selection, the target domain specification, or the selection of a subset of labels to be used for the training of a PU model.

The top part of this view refers to the corpus selected for the current working project. Note that once a corpus has been selected for a project folder, such a corpus will be shown directly as selected corpus in posterior executions of the application, and it will not be possible to modify it. Therefore, to run the application with several corpora, the application must be called once per corpus, specifying a different project folder.

The bottom part is divided into two subsections. The left one allows the user to select one out of the four different available alternatives for the classification of documents, thus defining a new category, which is immediately depicted on the bottom right box once the pre-classification has been completed.  From such a box the user can select which is the category to be used for the training of a new PU model in the next view.

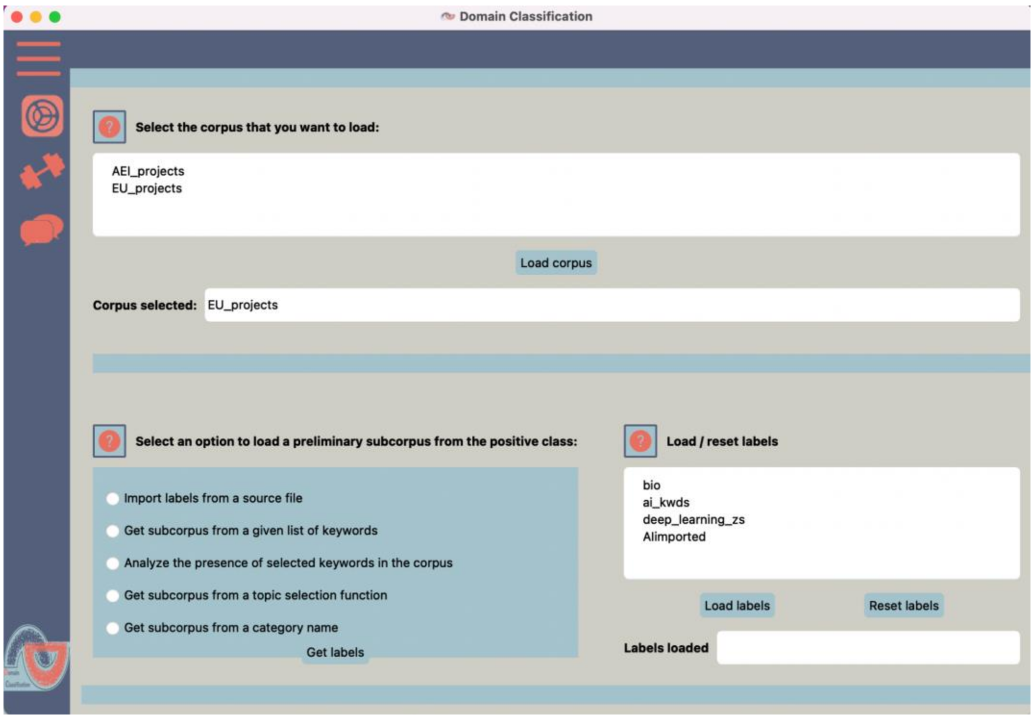

Domain Classification
Select the corpus that you want to load:
AEI_projects
EU_projects
Load corpus
Corpus selected: EU_projects
Select an option to load a preliminary subcorpus from the positive class:
Import labels from a source file
Get subcorpus from a given list of keywords
Analyze the presence of selected keywords in the corpus
Get subcorpus from a topic selection function
Get subcorpus from a category name
Get labels
Load / reset labels
bio
ai_kwds
deep_learning_zs
AIimported
Load labels
Reset labels
Labels loaded

Figure 12: Graphical user interface main window - Load corpus / labels view

Regarding the different alternatives for the selection of a preliminary subcorpus from the positive class, except for the importing of labels from a source file, a pop-up window appears at the time the user clicks on the corresponding radio button. In all cases, after the selection of one of the options, and when it applies, the specification of the inputs asked in each case, the user must click the "Get labels" button for the pre-classification to start.

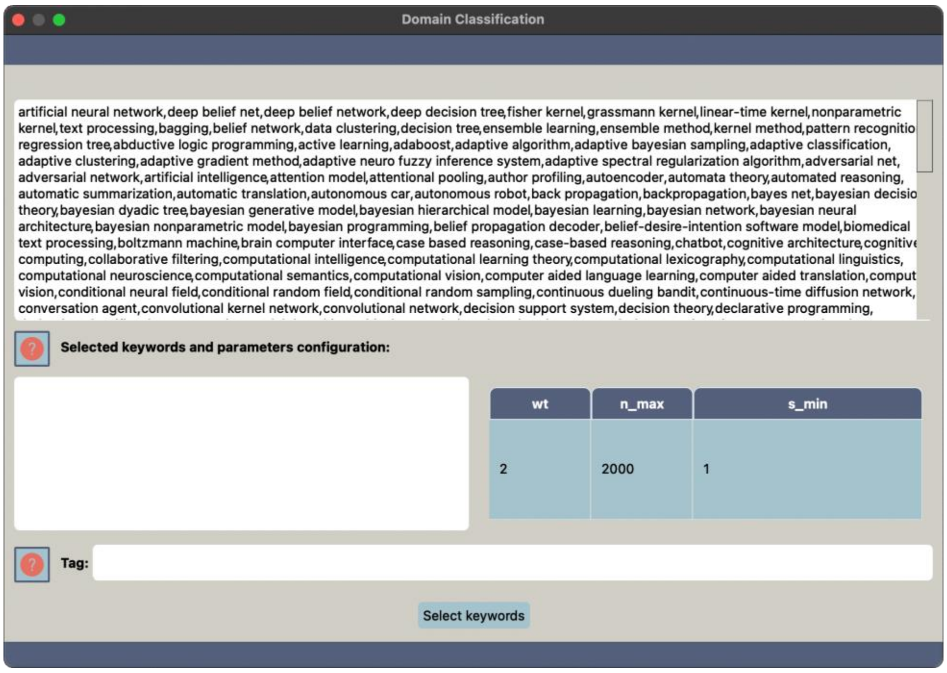
a. **Keywords based selection subwindow.**

On the top, a list of suggested keywords is shown, from which the user can select which one he wants to use for the selection of documents by writing them on the middle white box.

By default, the selection based on the cosine distance between transformer embeddings (see Sec. 3.3) is applied, but the keyword count similarity metric is available through the configuration file… In addition, the user can decide which tag is to be used for the naming of the category to which the documents to be selected belong and can configure the three parameters that intervene in the keywords-based selection, namely:

- **wt:** Weight of the title**.** A word in the title (if available) is equivalent to the wt repetitions of the word in the description. This is used by the keyword count measure only.

- **n_max:** Maximum number of elements in the output list.
- **s_min:** Minimum score. Only elements strictly above s_min are selected.



artificial neural network,deep belief net,deep belief network,deep decision tree,fisher kernel,grassmann kernel,linear-time kernel,nonparametric kernel,text processing,bagging,belief network,data clustering,decision tree,ensemble learning,ensemble method,kernel method,pattern recognitio regression tree,abductive logic programming,active learning,adaboost,adaptive algorithm,adaptive bayesian sampling,adaptive classification, adaptive clustering,adaptive gradient method,adaptive neuro fuzzy inference system,adaptive spectral regularization algorithm,adversarial net, adversarial network,artificial intelligence,attention model,attentional pooling,author profiling,autoencoder,automata theory,automated reasoning, automatic summarization,automatic translation,autonomous car,autonomous robot,back propagation,backpropagation,bayes net,bayesian decisio theory,bayesian dyadic tree,bayesian generative model,bayesian hierarchical model,bayesian learning,bayesian network,bayesian neural architecture,bayesian nonparametric model,bayesian programming,belief propagation decoder,belief-desire-intention software model,biomedical text processing,boltzmann machine,brain computer interface,case based reasoning,case-based reasoning,chatbot,cognitive architecture,cognitive computing,collaborative filtering,computational intelligence,computational learning theory,computational lexicography,computational linguistics, computational neuroscience,computational semantics,computational vision,computer aided language learning,computer aided translation,comput vision,conditional neural field,conditional random field,conditional random sampling,continuous dueling bandit,continuous-time diffusion network, conversation agent,convolutional kernel network,convolutional network,decision support system,decision theory,declarative programming,

Selected keywords and parameters configuration:

| wt | n_max | s_min |
|---|---|---|
| 2 | 2000 | 1 |

Tag:

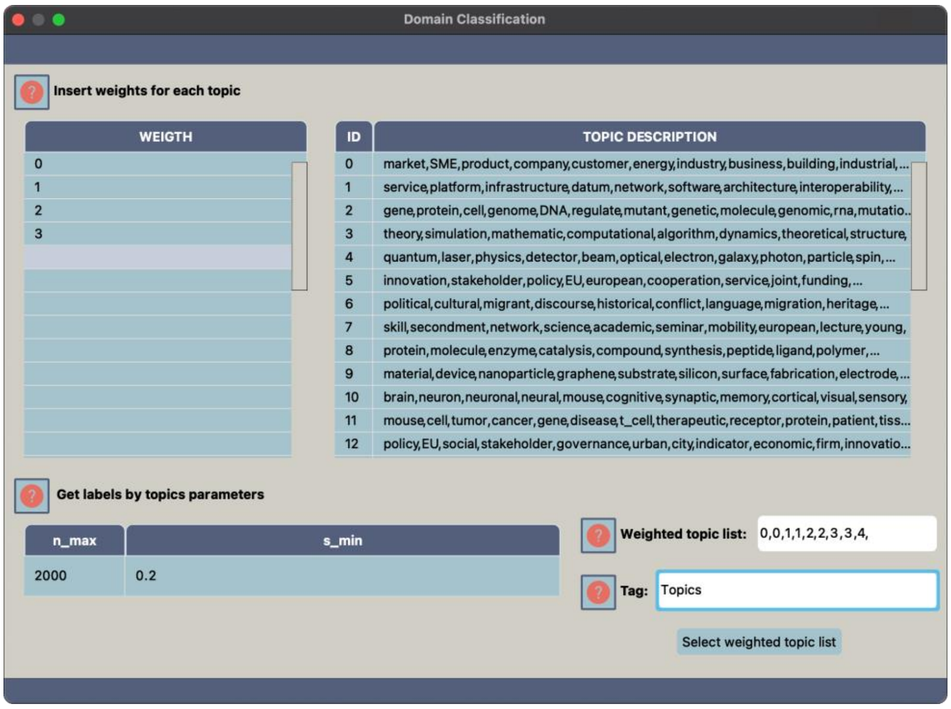
Figure 13: GUI main window - Keywords based selection

**b. Topic based selection subwindow.**

The top tables shown in this view are interrelated, that is, while the right one is unmodifiable and shows the id and chemical description of each of the topics associated with the topic model provided for the selected corpus, the left table is editable, and its purpose is the insertion by the user of the weights to be assigned to each of the topics for topic based selection of documents.

As in the keywords-based selection, the user can specify here the tag to be used for the naming of the category associated with the topic-based selection and configure the parameters in it implied, namely:

- **n_max:** Maximum number of elements in the output list.
- **s_min:** Minimum score. Only elements strictly above s_min are selected.



Insert weights for each topic

| WEIGTH |
|---|
| 0 |
| 1 |
| 2 |
| 3 |

| ID | TOPIC DESCRIPTION |
|---|---|
| 0 | market,SME,product,company,customer,energy,industry,business,building,industrial,... |
| 1 | service,platform,infrastructure,datum,network,software,architecture,interoperability,... |
| 2 | gene,protein,cell,genome,DNA,regulate,mutant,genetic,molecule,genomic,rna,mutatio... |
| 3 | theory,simulation,mathematic,computational,algorithm,dynamics,theoretical,structure, |
| 4 | quantum,laser,physics,detector,beam,optical,electron,galaxy,photon,particle,spin,... |
| 5 | innovation,stakeholder,policy,EU,european,cooperation,service,joint,funding,... |
| 6 | political,cultural,migrant,discourse,historical,conflict,language,migration,heritage,... |
| 7 | skill,secondment,network,science,academic,seminar,mobility,european,lecture,young, |
| 8 | protein,molecule,enzyme,catalysis,compound,synthesis,peptide,ligand,polymer,... |
| 9 | material,device,nanoparticle,graphene,substrate,silicon,surface,fabrication,electrode,... |
| 10 | brain,neuron,neuronal,neural,mouse,cognitive,synaptic,memory,cortical,visual,sensory, |
| 11 | mouse,cell,tumor,cancer,gene,disease,t_cell,therapeutic,receptor,protein,patient,tiss... |
| 12 | policy,EU,social,stakeholder,governance,urban,city,indicator,economic,firm,innovatio... |

Get labels by topics parameters

| n_max | s_min |
|---|---|
| 2000 | 0.2 |

Weighted topic list: 0,0,1,1,2,2,3,3,4,

Tag: Topics

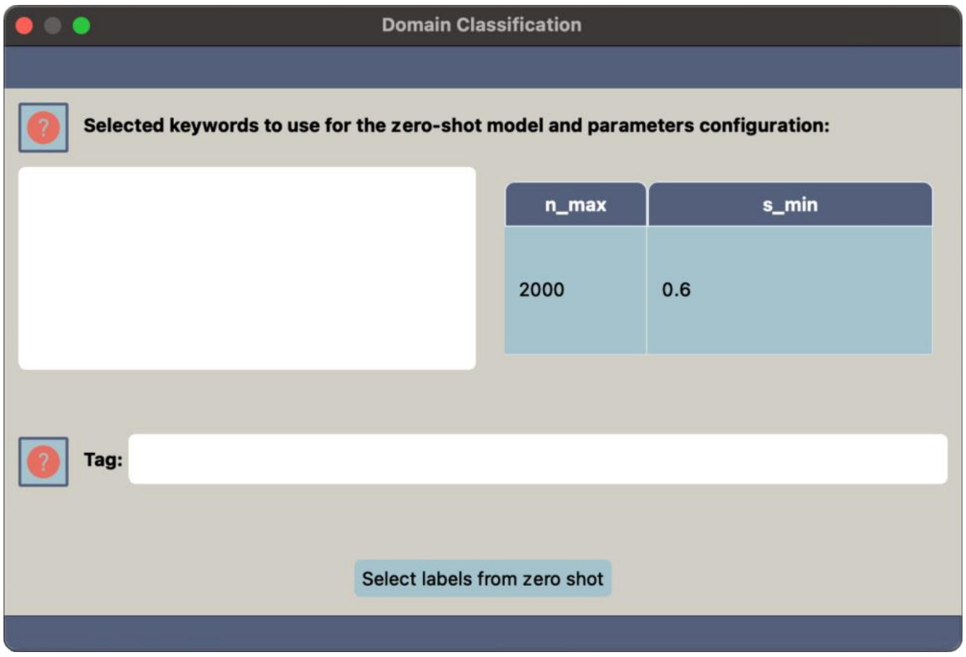
Figure 14: Graphical user interface main window - Topic based selection

**c. Domain name selection subwindow.**

Like the keywords-based selection, the user can insert here the keywords to be used for domain name selection, with the only difference that in this case no suggested list of keywords is provided.

Again, both a tag to name the category and parameters configuration is permitted, the parameters being in this case:

- **n_max:** Maximum number of documents to be selected.
- **s_min:** Minimum score. Only elements strictly above s_min are selected.



Selected keywords to use for the zero-shot model and parameters configuration:

| n_max | s_min |
|---|---|
| 2000 | 0.6 |

Tag:

Figure 15: Graphical user interface main window - Domain name selection

In addition, there exists a fourth pop-up window, in which the user can analyze the presence of the selected keywords in the corpus. In case the user has not selected keywords (i.e. the keywords based selection has not been carried out), the application forces the user to select first such keywords by displaying the keywords based selection window.

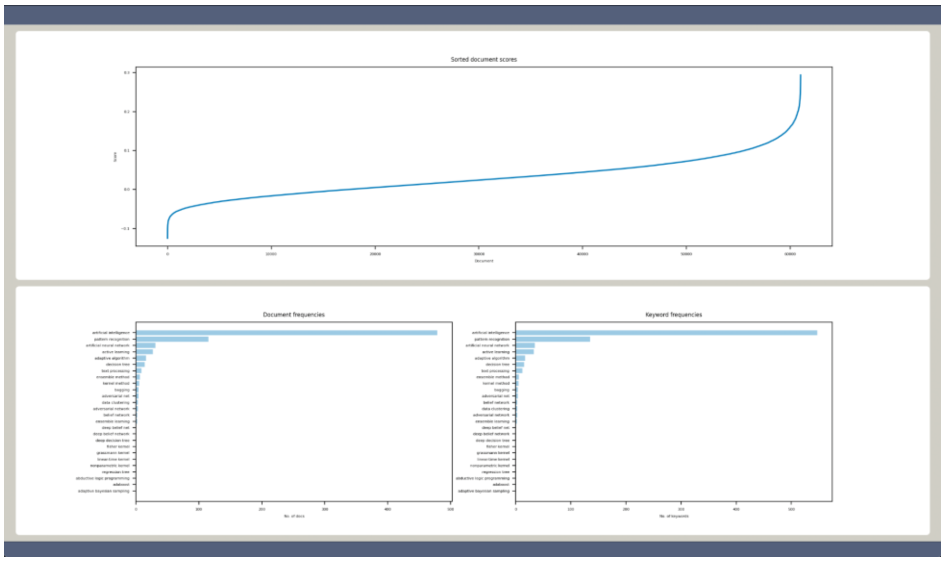


Figure 16: Some keyword statistics

During the time the pre-classification is being performed, a loading bar is shown in the middle of the view. Once any of the actions carried in this view are completed, a pop-up window message is displayed informing the user about the completion of the task and showing some additional information. Up to this point, the GUI accepts further user selections.

#### 4.5.2.2. View for the training and evaluation of a PU model for the selected category.

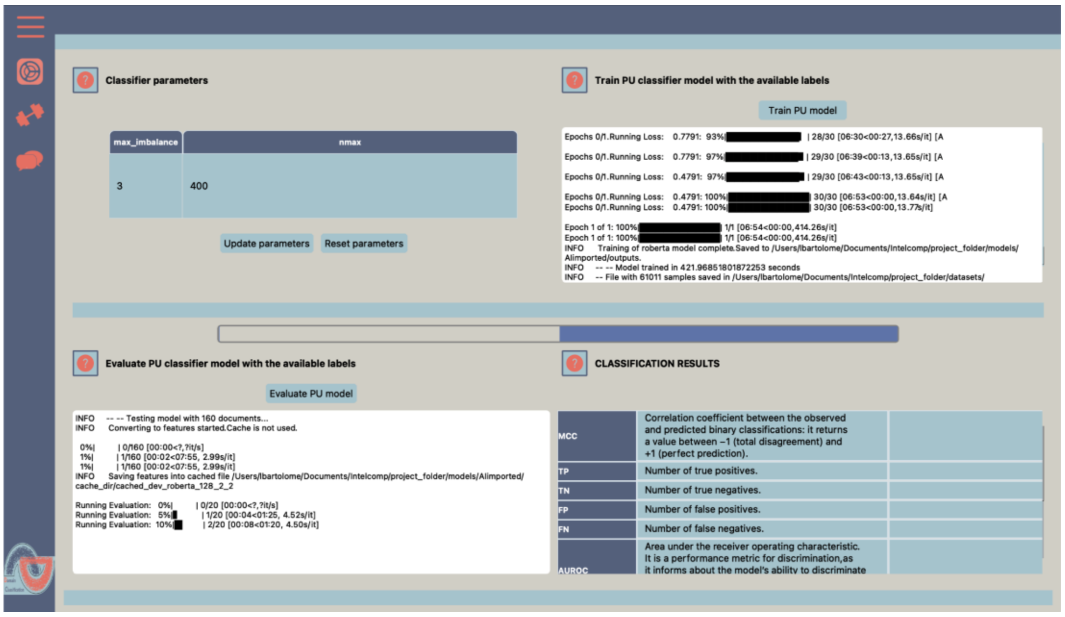
From this view, the user can train and evaluate a PU model for the set of documents from the positive class associated with the target category selected in the previous view by clicking the *"Train PU model"* button. While the model is being trained both a loading bar and the logs associated with the training are shown to the user.

Once the model has been trained, a pop-up window informs the user about the training completion, immediately after which the user can continue with its evaluation by clicking the *"Evaluate PU model"* button. Again, a loading bar and the logs associated with the evaluation are displayed while it is being performed and once completed its results are displayed on the bottom right table of the view.

The parameters related with the classifier can be modified by means of the top left table, these parameters being:

- **max_imbalance:** Maximum ratio of negative vs positive samples in the training set.
- **nmax:** Maximum number of documents in the training set.

The values shown are the default parameters; by inserting a new value and clicking the *"Update parameters"* button, the parameters are updated; they can also be restored to their default value by clicking the *"Reset parameters"* button.



| max_imbalance | nmax |
|---|---|
| 3 | 400 |

```
Epochs 0/1.Running Loss:   0.7791:  93%|█████████████  | 28/30 [06:30<00:27,13.66s/it] [A
Epochs 0/1.Running Loss:   0.7791:  97%|██████████████ | 29/30 [06:39<00:13,13.65s/it] [A
Epochs 0/1.Running Loss:   0.4791:  97%|██████████████ | 29/30 [06:43<00:13,13.65s/it] [A
Epochs 0/1.Running Loss:   0.4791: 100%|███████████████| 30/30 [06:53<00:00,13.64s/it] [A
Epochs 0/1.Running Loss:   0.4791: 100%|███████████████| 30/30 [06:53<00:00,13.77s/it]
Epoch 1 of 1: 100%|███████████████| 1/1 [06:54<00:00,414.26s/it]
Epoch 1 of 1: 100%|███████████████| 1/1 [06:54<00:00,414.26s/it]
INFO    Training of roberta model complete.Saved to /Users/lbartolome/Documents/Intelcomp/project_folder/models/
Alimported/outputs.
INFO    -- -- Model trained in 421.96851801872253 seconds
INFO    -- File with 61011 samples saved in /Users/lbartolome/Documents/Intelcomp/project_folder/datasets/
```

```
INFO    -- -- Testing model with 160 documents...
INFO    Converting to features started.Cache is not used.

  0%|          | 0/160 [00:00<?,?it/s]
  1%|          | 1/160 [00:02<07:55, 2.99s/it]
  1%|          | 1/160 [00:02<07:55, 2.99s/it]
INFO    Saving features into cached file /Users/lbartolome/Documents/Intelcomp/project_folder/models/Alimported/
cache_dir/cached_dev_roberta_128_2_2

Running Evaluation:   0%|          | 0/20 [00:00<?,?it/s]
Running Evaluation:   5%|▌         | 1/20 [00:04<01:25, 4.52s/it]
Running Evaluation:  10%|█         | 2/20 [00:08<01:20, 4.50s/it]
```

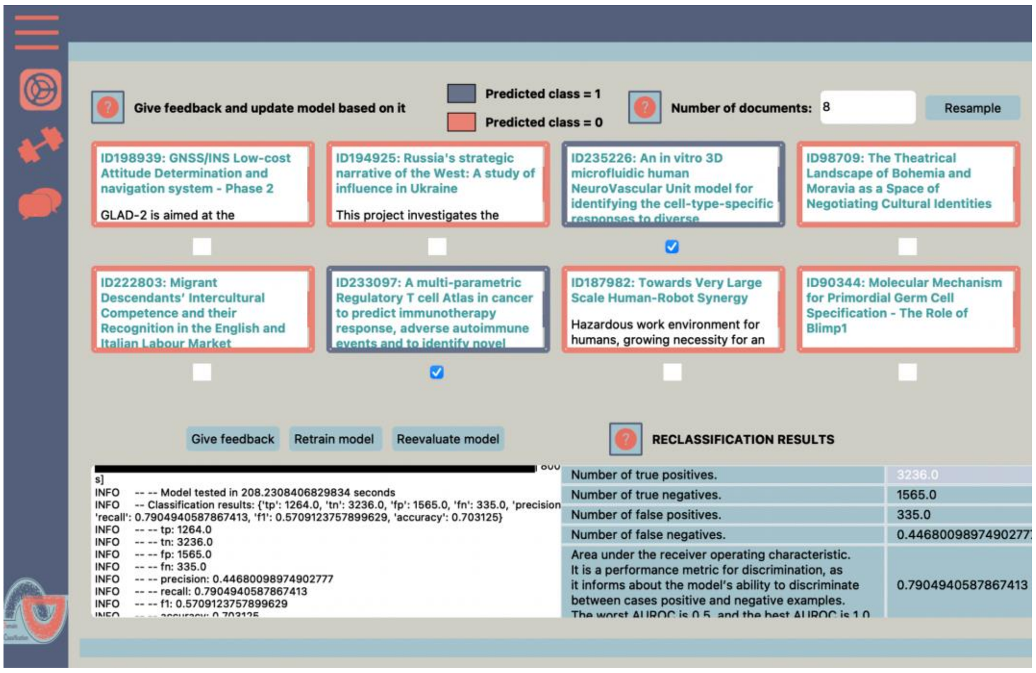
| | | |
|---|---|---|
| MCC | Correlation coefficient between the observed and predicted binary classifications: it returns a value between –1 (total disagreement) and +1 (perfect prediction). | |
| TP | Number of true positives. | |
| TN | Number of true negatives. | |
| FP | Number of false positives. | |
| FN | Number of false negatives. | |
| AUROC | Area under the receiver operating characteristic. It is a performance metric for discrimination,as it informs about the model's ability to discriminate | |

**Figure 17: Graphical user interface main window - Train / evaluate PU model view. At the time the picture was taken, the model was being evaluated.**

#### 4.5.2.3. View for the annotation of documents and the retraining and reevaluation of the associated PU model based on the user's feedback.

This view relates to the active learning loop. The user can select the number of documents according to which a reduced subset of the documents will be sampled documents belonging to the previously selected category will be sampled. The title (iv available), content and predicted class from the resulting documents are displayed within different boxes, the border of such a box being highlighted by the color that defines each predicted class (purple when the predicted class is 1, and red when it is 0). The user can change the category of each of the documents by just checking (predicted class = 1) or unchecking (predicted class =0) the corresponding checkboxes located under each of the documents.

Once the user has manually labeled the documents, the user must click the *"Give feedback"* button for the feedback process to be completed. Then, the next step is to retrain and reevaluate the model by clicking the buttons *"Retrain model"* and *"Reevaluate model*", respectively. Again, a loading bar where the associated logs are displayed while the former tasks are being performed, pop-up windows are displayed to notify each task's completion, and once the reevaluation is completed, the results are displayed on the bottom-right table.



ID198939: GNSS/INS Low-cost Attitude Determination and navigation system - Phase 2
GLAD-2 is aimed at the

ID194925: Russia's strategic narrative of the West: A study of influence in Ukraine
This project investigates the

ID235226: An in vitro 3D microfluidic human NeuroVascular Unit model for identifying the cell-type-specific responses to diverse

ID98709: The Theatrical Landscape of Bohemia and Moravia as a Space of Negotiating Cultural Identities

ID222803: Migrant Descendants' Intercultural Competence and their Recognition in the English and Italian Labour Market

ID233097: A multi-parametric Regulatory T cell Atlas in cancer to predict immunotherapy response, adverse autoimmune events and to identify novel

ID187982: Towards Very Large Scale Human-Robot Synergy
Hazardous work environment for humans, growing necessity for an

ID90344: Molecular Mechanism for Primordial Germ Cell Specification - The Role of Blimp1

```
s]
INFO    -- -- Model tested in 208.2308406829834 seconds
INFO    -- Classification results: {'tp': 1264.0, 'tn': 3236.0, 'fp': 1565.0, 'fn': 335.0, 'precision
'recall': 0.7904940587867413, 'f1': 0.5709123757899629, 'accuracy': 0.703125}
INFO    -- -- tp: 1264.0
INFO    -- -- tn: 3236.0
INFO    -- -- fp: 1565.0
INFO    -- -- fn: 335.0
INFO    -- -- precision: 0.44680098974902777
INFO    -- -- recall: 0.7904940587867413
INFO    -- -- f1: 0.5709123757899629
INFO    -- -- accuracy: 0.703125
```

| | |
|---|---|
| Number of true positives. | 3236.0 |
| Number of true negatives. | 1565.0 |
| Number of false positives. | 335.0 |
| Number of false negatives. | 0.4468009897490277 |
| Area under the receiver operating characteristic. It is a performance metric for discrimination, as it informs about the model's ability to discriminate between cases positive and negative examples. The worst AUROC is 0.5, and the best AUROC is 1.0 | 0.7904940587867413 |

Figure 18: Graphical user interface main window - Annotation, retraining and reevaluation view. At the time the picture was taken, the model was being re-evaluated.

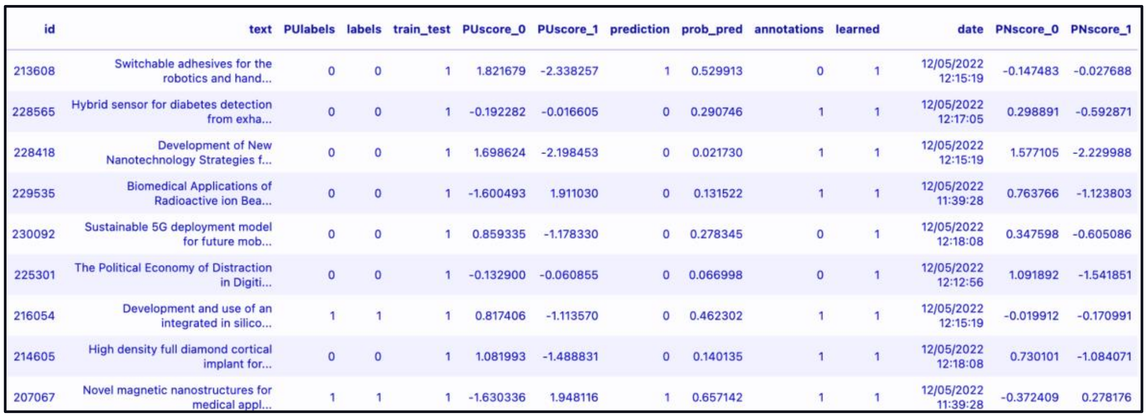
## 4.6. Dataset and metadata files

### 4.6.1. Datasets

The dataset files are stored files stored in datasets/ folder. Each dataset will contain all relevant information for the subcorpus used for classification, organized in the following columns:

- Information about the input documents:
  - **Id**: document identifier
  - **Text**: text of the source documents. Typically, it is a string joining title and abstract from the original documents
- Information about labels:
  - **PUlabels**: Labels returned by the selection model
  - **labels**: last set of labels used by the learning models.
  - **annotations**: manual labels introduced by users through the annotation tool
  - **date**: date of each annotation
  - **learned**: a flag that shows if the label has been already used to update the model (1) or not (0).
- Information from the classifiers:
  - **train_test**: shows if the document has been used for training (1), test (0) or not used (-99).
  - **PUscore_0**: score of the PU learning model for class 0.
  - **PUscore_1**: score of the PU learning model for class 1.
  - **PNscore_0**: score of the retrained model for class 0.
  - **PNscore_1**: score of the retrained model for class 1.
  - **prediction**: class prediction of the last model
  - **prob_pred**: probabilistic prediction of the las model

A screenshot of the information in a dataset is shown in Fig. 19



| id | text | PUlabels | labels | train_test | PUscore_0 | PUscore_1 | prediction | prob_pred | annotations | learned | date | PNscore_0 | PNscore_1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 213608 | Switchable adhesives for the robotics and hand... | 0 | 0 | 1 | 1.821679 | -2.338257 | 1 | 0.529913 | 0 | 1 | 12/05/2022 12:15:19 | -0.147483 | -0.027688 |
| 228565 | Hybrid sensor for diabetes detection from exha... | 0 | 0 | 1 | -0.192282 | -0.016605 | 0 | 0.290746 | 1 | 1 | 12/05/2022 12:17:05 | 0.298891 | -0.592871 |
| 228418 | Development of New Nanotechnology Strategies f... | 0 | 0 | 1 | 1.698624 | -2.198453 | 0 | 0.021730 | 1 | 1 | 12/05/2022 12:15:19 | 1.577105 | -2.229988 |
| 229535 | Biomedical Applications of Radioactive ion Bea... | 0 | 0 | 1 | -1.600493 | 1.911030 | 0 | 0.131522 | 1 | 1 | 12/05/2022 11:39:28 | 0.763766 | -1.123803 |
| 230092 | Sustainable 5G deployment model for future mob... | 0 | 0 | 1 | 0.859335 | -1.178330 | 0 | 0.278345 | 0 | 1 | 12/05/2022 12:18:08 | 0.347598 | -0.605086 |
| 225301 | The Political Economy of Distraction in Digiti... | 0 | 0 | 1 | -0.132900 | -0.060855 | 0 | 0.066998 | 0 | 1 | 12/05/2022 12:12:56 | 1.091892 | -1.541851 |
| 216054 | Development and use of an integrated in silico... | 1 | 1 | 1 | 0.817406 | -1.113570 | 0 | 0.462302 | 1 | 1 | 12/05/2022 12:15:19 | -0.019912 | -0.170991 |
| 214605 | High density full diamond cortical implant for... | 0 | 0 | 1 | 1.081993 | -1.488831 | 0 | 0.140135 | 1 | 1 | 12/05/2022 12:18:08 | 0.730101 | -1.084071 |
| 207067 | Novel magnetic nanostructures for medical appl... | 1 | 1 | 1 | -1.630336 | 1.948116 | 1 | 0.657142 | 1 | 1 | 12/05/2022 11:39:28 | -0.372409 | 0.278176 |

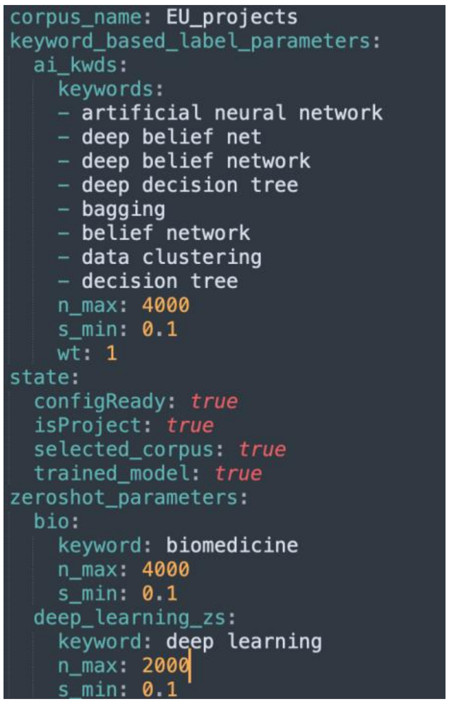
**Figure 19: A dataset dataframe for domain "biomedicine" specified by the user through a zero-shot classifier. It encompasses information about the source documents, the labels obtained by the zero-shot classifier, data and metadata from human annotations and information from the classifiers.**

### 4.6.2. Metadata file

The metadata file in `project_folder/metadata.yaml` records hierarchically structured information about the project status. It is also a record of the main task carried out for the current project, and the parameters used in the process. For instance, it registers information about the method and parameters used to obtain a specific set of labels. At the top level of the tree, the following entries can be found:

- **corpus_name**: Identifier of the corpus used by the current project.
- **state**:
  - configReady: true if the configuration file (parameters.yaml) has been loaded
  - isProject: true if the input project_folder has the required folder structure
  - selected_corpus: true if a corpus has been selected by the user
  - trained_model: true if at least one document selection has been done, so that a set of labels is available for the supervised classification algorithms.
- **keyword_based_label_parameters**: a dictionary containing the parameters used during each execution of the keyword based selection mechanism. It contains one tree per keyword, with the following schema
  - tag: name assigned to the keyword set.
    - List of keywords
    - List of parameters used by the document selector.
- **zeroshot_parameters**: a dictionary containing the parameters used during each execution of the zero-shot classifier. It contains one tree per keyword, with the following schema
  - tag: name assigned to the keyword set.
    - List of keywords
    - List of parameters used by the zero-shot classifier
- **topic_based_label_parameters**: a dictionary containing the parameters used during each execution of the zero-shot classifier. It contains one tree per keyword, with the following schema
  - tag: name assigned to the keyword set.
    - List of keywords
    - List of parameters used by the zero-shot classifier

A sample of a metadata file can be shown in Fig. 20.



```
corpus_name: EU_projects
keyword_based_label_parameters:
  ai_kwds:
    keywords:
    - artificial neural network
    - deep belief net
    - deep belief network
    - deep decision tree
    - bagging
    - belief network
    - data clustering
    - decision tree
    n_max: 4000
    s_min: 0.1
    wt: 1
state:
  configReady: true
  isProject: true
  selected_corpus: true
  trained_model: true
zeroshot_parameters:
  bio:
    keyword: biomedicine
    n_max: 4000
    s_min: 0.1
  deep_learning_zs:
    keyword: deep learning
    n_max: 2000
    s_min: 0.1
```

**Figure 20: A sample metadata file. It records information about a project related to the classification of a corpus related to EU projects. The "states" shows that at least one model has been trained with this corpus. Also, three domains have been already specified: one through a list of keywords, and two using the domain name ("biomedicine" and "deep learning"). The parameter values used for each document selection are also shown.**

## 5. CONCLUSIONS

We developed a python module for subcorpus generation based on a user-driven selection of documents based on categories specified through different mechanisms: zero-shot classification, keywords, or weighted topics. Besides the required text processing and classification modules, a complete python application has been developed to facilitate the testing of the software functionality before its final integration into the IntelComp platform. The application allows user interaction through a command window or a graphical user interface. As part of the integration process, the algorithms and models will be tested using data from the DataLake. Also, the usability of the classification process will be tested based on user experience during the living labs.

## REFERENCES

[Yin, 2019] Yin, W., Hay, J., & Roth, D. (2019, November). Benchmarking Zero-shot Text Classification: Datasets, Evaluation and Entailment Approach. In *Procs of the 2019 Conf.e on Empirical Methods in Natural Language Processing and the 9th Int. Joint Conf. on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3914-3923).

[Vaswani, 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaise, L. & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

[Devlin, 2018] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv*:1810.04805.

[Reimers, 2019] Reimers, N., & Gurevych, I. (2019, November). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Procs of the 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int. Joint Conf. on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3982-3992).

## ANNEXES

### A. Default Configuration File.

The first time the application is run, a copy of the configuration file is stored in the project folder with name parameters.yaml. Table 2 shows the list of parameters, their function, and the default values used for application testing.

**Table 2: Default configuration file. Comments explain the meaning of each parameter**

```
# Parameters for the dataset selection
dataset:
  # Source file name
  path2source: '../datasets'

# Parameter for the keyword-based document selector
keywords:
  # Weight of the title. A word in the title is equivalent to wt
  # repetitions of the word in the description.
  wt: 2
  # Maximum number of documents to be selected.
  n_max: 2000
  # Minimum score. Only docs scored strictly above s_min are selected
  s_min: 1

# Parameter for the zero-shot document selector
zeroshot:
  # Maximum number of documents to be selected.
  n_max: 2000
  # Minimum score. Only docs scored strictly above s_min are selected
  s_min: 0.6

# Parameter for the keyword-based document selector
topics:
  # Selection method: 'embedding' or 'count'
  method: 'embedding'
  # Weight of the title. A word in the title is equivalent to wt
  # repetitions of the word in the description. (For method='count' only)
  wt: 1
  # Maximum number of documents to be selected.
  n_max: 2000
  # Minimum score. Only docs scored strictly above s_min are
  # selected
  s_min: 0.2
  # Name of the SBERT model. Available pretrained models can be
  # found in https://www.sbert.net/docs/pretrained_models.html
  model_name: all-MiniLM-L6-v2


  # Maximum number of documents to be selected.
  n_max: 2000
  # Minimum score. Only docs scored strictly above s_min are selected
  s_min: 0.6
  # Name of the SBERT model. Available pretrained models can be found in
  # https://www.sbert.net/docs/pretrained_models.html
  model_name: all-MiniLM-L6-v2


# Parameters for the classifier
classifier:
  # Maximum ratio neg vs positive samples in the training set
  max_imbalance: 3
  # Maximum number of documents in the training set.
  nmax: 400

# Parameters for the active learning (AL)
active_learning:
  # Number of docs to show each AL round
  n_docs: 5
  # Sample selection algorithm: 'random' or 'extremes'
  sampler: 'extremes'
  # Ratio of high-score samples. The rest will be low-score
  # samples. (Used for sampler='extremes' only)
  p_ratio: 0.8
  # (Approximate) probability of selecting the doc with the
  # highest score in a single sampling. This parameter is used
  # to control the randomness of the randomness of the
  # stochastic sampling: if top_prob=1, the highest score
  # samples are taken deterministically. top_prob=0 is
  # equivalent to random sampling. (Used for sampler='extremes'
  # only)
  top_prob: 0.1

# Specify format for the log outputs
logformat:
  filename:    msgs.log
  datefmt:     '%m-%d %H:%M:%S'
  file_format: '%(asctime)s %(levelname)-8s %(message)s'
  file_level:  INFO
  cons_level:  DEBUG
  cons_format: '%(levelname)-8s %(message)s'
```