Implementing AI-based Comprehensive Abstract Screening

Instructions for using the GitHub repository in https://github.com/janoschortmann/abstract-screening

Julia Isabel Serrato Fonseca[a,c,d], Ana María Anaya-Arenas[a,c,d], Janosch Ortmann[a,c,e,f], Angel Ruiz[b,d]

a Department of Analytics, Operations and IT, ESG-UQAM, Université du Quebec à Montréal, Montréal, Canada;
b Department of Operations and Decisions Systems, Faculty of Administration Sciences, Université Laval, Quebec City, Canada;
c Research Centre for Smart2 Management of Complex Systems (CRI2GS), Montreal, Canada
d Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Montréal and Quebec, Canada.
e Groupe d'études et de recherche en analyse des décisions (GERAD), Montreal, Canada
f Centre de recherches mathématiques (CRM), Montréal, Canada

1. INTRODUCTION

When performing a Systematic Literature Review (SLR), the Abstract Screening Process (ASP) can be a very consuming and laborious task, especially when researchers retrieve a significant number of citations after running queries in the selected databases. This can translate into many hours of work.
This repository implements the Machine Learning-based methodology described in Serrato-Fonseca et al. (2024) and introduces the AI-based Comprehensive Abstract Screening (ACAS) tool that permits researchers to create their own ASP. This document is a detailed guideline to implementing said tool in any given SLR process.

2. USING MACHINE LEARNING FOR AUTOMATIC ABSTRACT SCREENING PROCESS GITHUB TOOL

ACAS consists of four different Python scripts to be run sequentially, following the methodology of Serrato-Fonseca et al. (2024). After running scripts 1 and 2, user intervention is required for adding manual inputs. In the following diagram, the data flow and the human interventions are marked out.

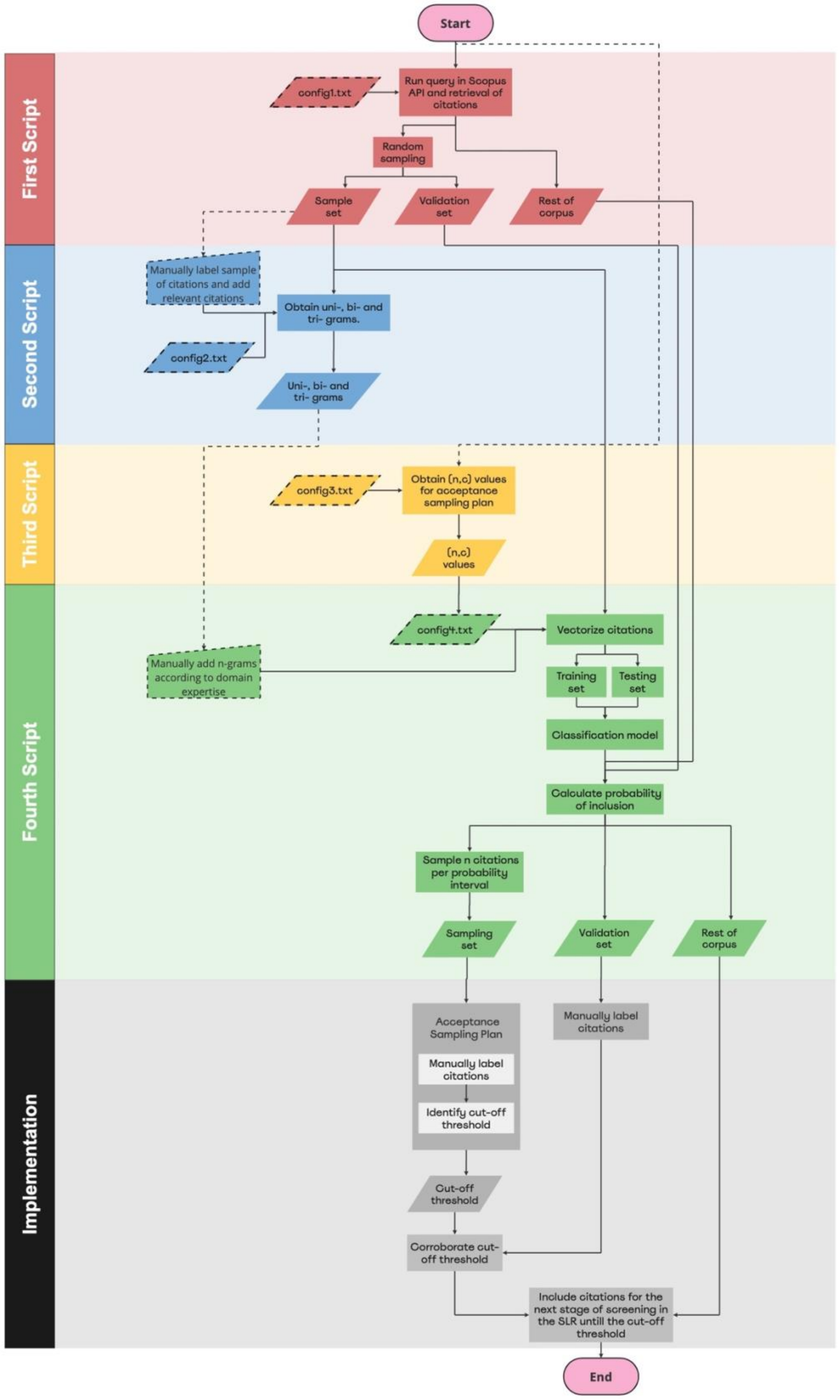

Start
First Script
config1.txt
Run query in Scopus API and retrieval of citations
Random sampling
Sample set
Validation set
Rest of corpus
Second Script
Manually label sample of citations and add relevant citations
Obtain uni-, bi- and tri- grams.
config2.txt
Uni-, bi- and tri- grams
Third Script
config3.txt
Obtain (n,c) values for acceptance sampling plan
(n,c) values
Fourth Script
config4.txt
Vectorize citations
Manually add n-grams according to domain expertise
Training set
Testing set
Classification model
Calculate probability of inclusion
Sample n citations per probability interval
Sampling set
Validation set
Rest of corpus
Implementation
Acceptance Sampling Plan
Manually label citations
Identify cut-off threshold
Manually label citations
Cut-off threshold
Corroborate cut-off threshold
Include citations for the next stage of screening in the SLR untill the cut-off threshold
End

Diagram 1 Process flowchart

Requirements: The tool is written in Python 3.9 and requires the following Python packages:

P1. numpy 2.0.0
P2. pandas 2.2.2
P3. requests 2.32.3
P4. nltk 3.8.1
P5. sklearn 1.5.1
P6. joblib 1.4.2
P7. scipy 1.14.0
P8. matplotlib 3.9.1

A full list of requirements can be found in the file **requirements.txt**. The user may wish to create either a conda environment or a virtualenv using this file.
In addition to software environment described above, the user either needs a file of papers to be screened (the format is described in section 2.2 below) or a personal API key for the Scopus API that can be requested at https://dev.elsevier.com/. Researchers affiliated to universities will typically be able to obtain access through their institution.

2.1. Script One: Query retrieval and sampling (if using SCOPUS)

The first script is not required if the user already has a list of references to be screened. In that case, the user can skip straight to section 2.2, where the required format for file of references is described. For the rest of this subsection, we assume that the user is looking to retrieve references based on keywords using the Scopus database and that a Scopus Key and Token has already been obtained.

Before running the code, modify the **config1.txt** file that specifies:

- *directory*: The working directory.
- *api_key:* The API key.
- *token:* Token, if applicable.
- *query:* The query in string format as showed on the example in Figure 1.
- *date_range:* The date range in string format as showed on the example in Figure 1.
- *sample_size:* A sampling size in a float format, that is the proportion of citations that the user is willing to manually screen. This sample of citations will be used for building the Machine Learning model. We would suggest screening at least 300 citations to build sufficient information for the model to learn from. This number should be adjusted upwards if the proportion of positive samples is very small. The more papers that the user screens, the better the model's accuracy. However, there is a trade-off between workload and accuracy. In the end, the purpose of using this tool is speeding up the abstract screening process.
- *validation_size*: A validation set size, which is going to be used at the last step of the process to validate the selected cut-off threshold. We would recommend using a similar size to the sample set size. Nevertheless, the default values for the sample and validation set sizes are 0.2 and 0.1, respectively.

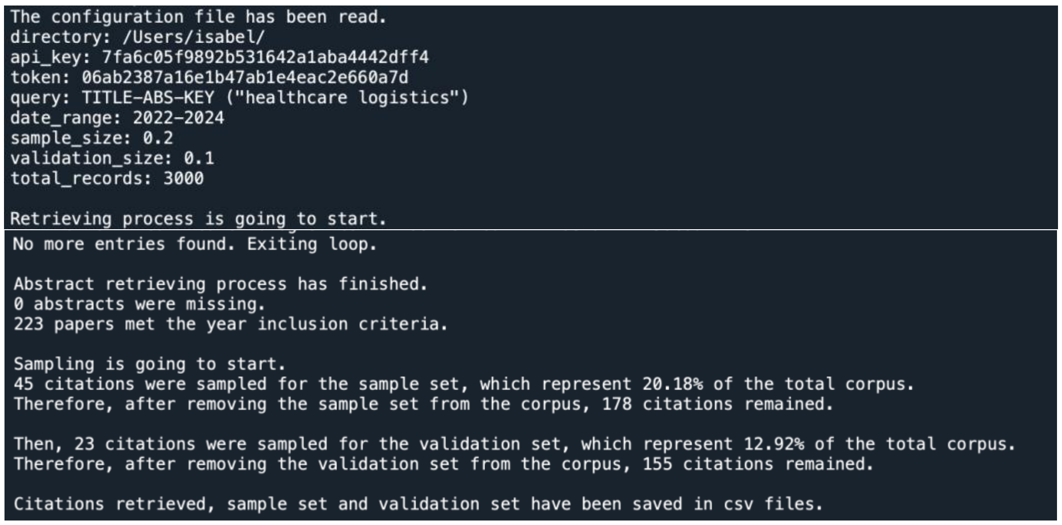
- *total_records*: The maximum number of records they would like to retrieve, which should not exceed the quota limits associated to their API key.

```
directory = "/Users/isabel/"
api_key = "0123456789"
token = "abcdefghijk"
query = "TITLE-ABS-KEY ("healthcare logistics")"
date_range = "2022-2024"
sample_size = 0.2
validation_size = 0.1
total_records = 3000
```

Figure 1 Input example of config1.txt

Once the config file has been edited to the user's needs, we can proceed to running the first script called **1_query_retrieval_sampling.py**.

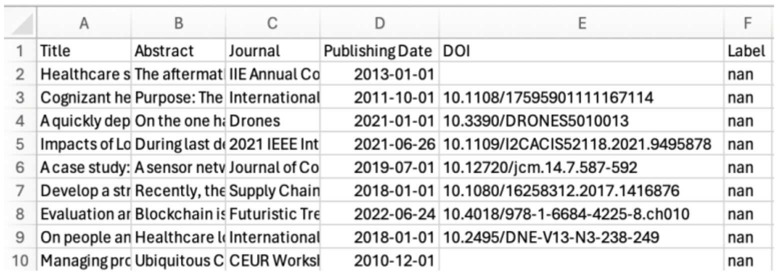
After running the code, three CSV files will be automatically saved in the directory, one with the citations sample (**sample.csv**) one with the validation set (**validation.csv**) and the rest of the corpus (**citations_retrieved.csv**), all following the format: *Title, Abstract, Journal, Publishing Date* and *DOI*.



```
The configuration file has been read.
directory: /Users/isabel/
api_key: 7fa6c05f9892b531642a1aba4442dff4
token: 06ab2387a16e1b47ab1e4eac2e660a7d
query: TITLE-ABS-KEY ("healthcare logistics")
date_range: 2022-2024
sample_size: 0.2
validation_size: 0.1
total_records: 3000

Retrieving process is going to start.
```

```
No more entries found. Exiting loop.

Abstract retrieving process has finished.
0 abstracts were missing.
223 papers met the year inclusion criteria.

Sampling is going to start.
45 citations were sampled for the sample set, which represent 20.18% of the total corpus.
Therefore, after removing the sample set from the corpus, 178 citations remained.

Then, 23 citations were sampled for the validation set, which represent 12.92% of the total corpus.
Therefore, after removing the validation set from the corpus, 155 citations remained.

Citations retrieved, sample set and validation set have been saved in csv files.
```

Figure 2 Output example of Script One



	A	B	C	D	E	F
1	Title	Abstract	Journal	Publishing Date	DOI	Label
2	Healthcare s	The aftermat	IIE Annual Co	2013-01-01		nan
3	Cognizant he	Purpose: The	International	2011-10-01	10.1108/17595901111167114	nan
4	A quickly dep	On the one ha	Drones	2021-01-01	10.3390/DRONES5010013	nan
5	Impacts of Lo	During last de	2021 IEEE Int	2021-06-26	10.1109/I2CACIS52118.2021.9495878	nan
6	A case study:	A sensor netv	Journal of Co	2019-07-01	10.12720/jcm.14.7.587-592	nan
7	Develop a str	Recently, the	Supply Chain	2018-01-01	10.1080/16258312.2017.1416876	nan
8	Evaluation ar	Blockchain is	Futuristic Tre	2022-06-24	10.4018/978-1-6684-4225-8.ch010	nan
9	On people an	Healthcare lo	International	2018-01-01	10.2495/DNE-V13-N3-238-249	nan
10	Managing prc	Ubiquitous C	CEUR Worksl	2010-12-01		nan

Figure 3 CSV output example of Script One

Once the output of the script is produced, they need to be validated by the user. Specifically, two steps need to be performed in the CSV files.

Before running the next script:
1) Labelling the sample. The user must manually screen the citations in **sample.csv**. To do it, for each citation in the list the abstract and title are read by user and then compare it to the inclusion and exclusion criteria of their SLR to hence label them. Once a decision is made, user needs to mark in the last column of the csv file titled *Label* with the number one (1) if the citation is included in the SLR or with a zero (0) if the citation is excluded. After the manual screening process, the user must rename the file as "**sample.csv**".

2) Adding manual citations if known. In addition to the citations obtained from Scopus, the user should include in the **sample.csv** file any known manual inclusion. A manual inclusion is a citation (a paper) that the reviewer knows it fills the inclusion criteria, does not fill any exclusion criteria, and are key references on the topic. Said citations could maybe have been retrieved from the query, so they must be search for in **sample.csv** and then in **validation.csv** and **citations_retrieved.csv**. If the citations are found in one of said two files, they should be manually moved to **sample.csv** and they should be labelled as included. If they are not found by the query, they should be manually added to the **sample.csv** file (we recommend that the user validate why it was not obtained by the query). All manual inclusions should be carefully identified as they should be reported in the SLR.

When adding manual citations to be included, these should be added at the end (bottom) of **sample.csv**, following the existing column formatting. If some of the manual included citations also appear among the retrieved citations, the user should make sure that they are deleted from **citations_retrieved.csv** and/or **validation.csv** to avoid duplicates.

2.2.Script Two: n-grams creation

If the user provides their own list of citations to be screened, these should appear in two files: the labelled instances should be stored in a file called **sample.csv** and the unlabelled instances should be stored in a file called **citations_retrieved.csv**.
The files should contain the following columns, in this order:
Title, Abstract, Journal, Publishing Date* and *DOI

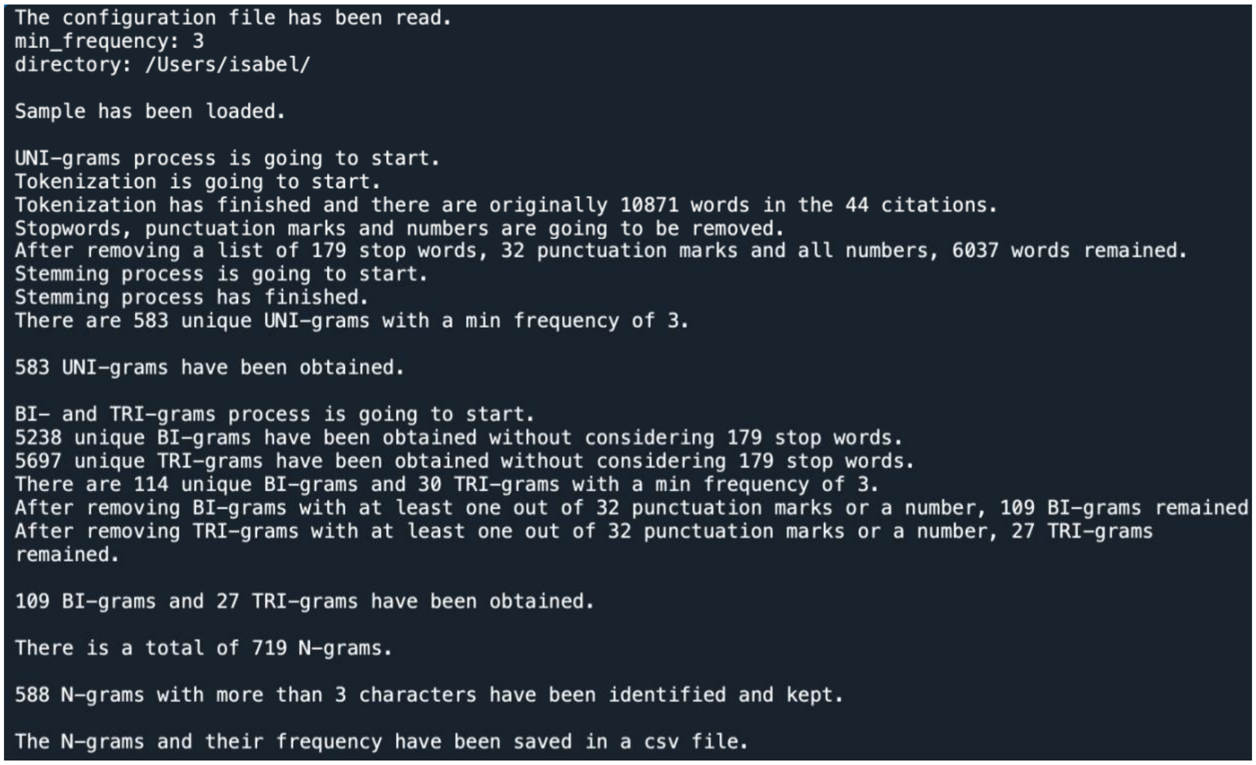
In the **sample.csv** file, there is also a *Label* column that contains the values 0 (not to be included in the SLR) or 1 (to be included).

The second script, called **2_n_grams.py**, creates the uni-, bi- and tri-grams of all the citations in **sample.csv**. Before running the code, modify a **config2.txt** file that specifies the minimum frequency that a n-gram must be present in the sample for it to be considered as a feature in the Machine Learning model, which needs to be added as an integer value (*min_frequency*). The default value is 3, which we recommend being the minimum, to avoid the inclusion of too many features and prevent overfitting.

After running the script, the output is a csv file **n_grams.csv** with all the n-grams (that have more than 3 characters) encountered in the sample of citations that respect the minimum frequency required. Figure 5 shows and example of the output of Script Two and Figure 6 present an example of the n-grams generated for a SLR in the field of Operations Research applied to Healthcare.

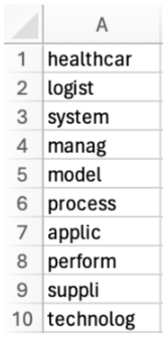
```
min_frequency = 3
```

Figure 4 Input example of config2.txt



```
The configuration file has been read.
min_frequency: 3
directory: /Users/isabel/

Sample has been loaded.

UNI-grams process is going to start.
Tokenization is going to start.
Tokenization has finished and there are originally 10871 words in the 44 citations.
Stopwords, punctuation marks and numbers are going to be removed.
After removing a list of 179 stop words, 32 punctuation marks and all numbers, 6037 words remained.
Stemming process is going to start.
Stemming process has finished.
There are 583 unique UNI-grams with a min frequency of 3.

583 UNI-grams have been obtained.

BI- and TRI-grams process is going to start.
5238 unique BI-grams have been obtained without considering 179 stop words.
5697 unique TRI-grams have been obtained without considering 179 stop words.
There are 114 unique BI-grams and 30 TRI-grams with a min frequency of 3.
After removing BI-grams with at least one out of 32 punctuation marks or a number, 109 BI-grams remained
After removing TRI-grams with at least one out of 32 punctuation marks or a number, 27 TRI-grams
remained.

109 BI-grams and 27 TRI-grams have been obtained.

There is a total of 719 N-grams.

588 N-grams with more than 3 characters have been identified and kept.

The N-grams and their frequency have been saved in a csv file.
```

Figure 5 Output example of Script Two



	A
1	healthcar
2	logist
3	system
4	manag
5	model
6	process
7	applic
8	perform
9	suppli
10	technolog

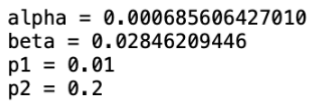
Figure 6 CSV output example of Script Two

Before running the next script:

The user should manually revise the uni-, bi and trigrams and consider adding new ones according to their domain expertise.

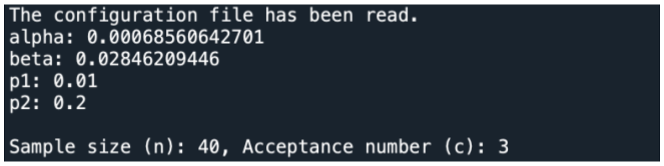
2.3. Script Three: Acceptance Sampling Plan

The third script, **3_acceptance_sampling_plan.py,** can be used to calculate the (n,c) values of the acceptance sampling plan. Said values are needed as an input in the following step. This methodology is discussed thoroughly in Serrato-Fonseca et al. (2024), see section 5.
Before running the code, modify the **config3.txt** file that specifies the selected *alpha*, *beta*, p_1 and p_2 values as shown in Figure 7. The default values are *alpha* = 0.0007, *beta* = 0.03, p_1 = 0.01 and p_2 = 0.2, which are the parameters used in Serrato-Fonseca et al. (2024). In addition, the script generates an Operating Characteristic (OC) curve.



```
alpha = 0.000685606427010
beta = 0.02846209446
p1 = 0.01
p2 = 0.2
```

Figure 7 Input example of config3.txt



```
The configuration file has been read.
alpha: 0.00068560642701
beta: 0.02846209446
p1: 0.01
p2: 0.2

Sample size (n): 40, Acceptance number (c): 3
```

Figure 8 Output example of Script Three

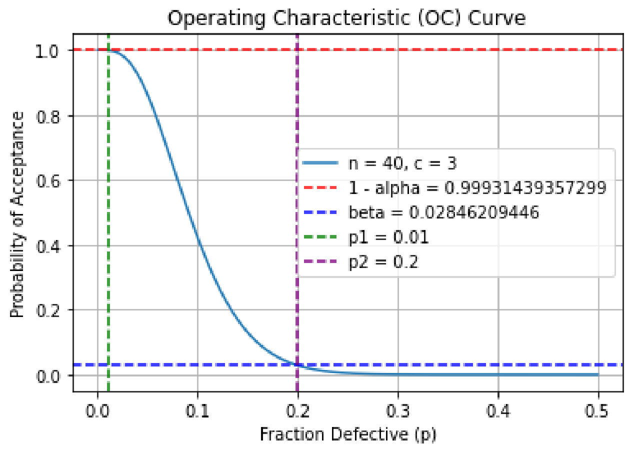


Figure 9 Graph output (OC curve) example of Script Three

2.4 Script Four: Train, Test and Prediction

The fourth script, **4_traintest prediction.py,** first assigns weights to each citation in **sample.csv** according to the frequency of each uni-, bi- and tri- gram obtained from running the third script. This vectorized data is then split into *training* and *testing* datasets. Secondly, a classification model is trained and tested according to the hyperparameters that the user inputs. For this step, the model is used for predicting the probability of inclusion of the citations in **citations_retrieved.csv** and

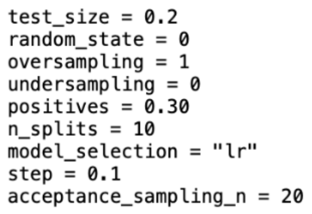
validation.csv. The script finally executes the sampling phase of an acceptance sampling plan by sampling ***n*** citations per probability interval from **citations_retrieved.csv** and saves them in a csv file.

Before running the code, modify a **config4.txt** file that specifies:

- *test_size*: The proportion of citations to be assigned to the test set, with the remainder making up the training set. We recommend this proportion to be set between 0.2 and 0.3 and the default value in this script is 0.2.
- *random_state*: A random state, which is an integer number used to control the randomness of the data shuffling during the split. By setting a specific random state value it can be ensured that the split is reproducible, that means that every time the code is run with the same random state, the same train and test sets are going to be obtained. The default value is 0.
- The hyperparameters of the classification model:
    - *model_selection*: The type of model, being the options a Logistic Regression ("lr"), which is the default, and a Decision Tree ("dt").
    - *oversampling* and *undersampling*: A binary expression for stating if a naïve *oversampling* or a naïve *undersampling* approach is going to be implemented (1 for yes, 0 for no). These methodologies can ameliorate the class imbalance problem, which is discussed more thoroughly in Serrato-Fonseca et al. (2024). Naïve *oversampling* refers to duplicating minority class instances and naïve *undersampling* to remove instances from the majority class. It is important to denote that both approaches cannot be implemented at the same time and if intended, an error message will be retrieved.
    - *positives*: When implementing any of the two approaches, a proportion of positive instances must be determined. The default is a naïve *oversampling* with a proportion of 30% positives, as these were the conditions under which the classification model in Serrato-Fonseca et al. (2024) was built. However, it must be kept in mind that in Serrato-Fonseca et al. (2024), a proportion of 30% positives was achieved by manually adding relevant citations to the sample, instead of employing a *naïve* approach. We refer to our article for a discussion of the downsides of naïve oversampling without further enrichment.
- *n_splits*: The number of splits that will be considered for performing a cross-validation in the training set. The default is a traditional approach of 10 splits.
- *acceptance_sampling_n*: A sample size ***n*** as an integer value, which can be obtained in the previous step 2.3.

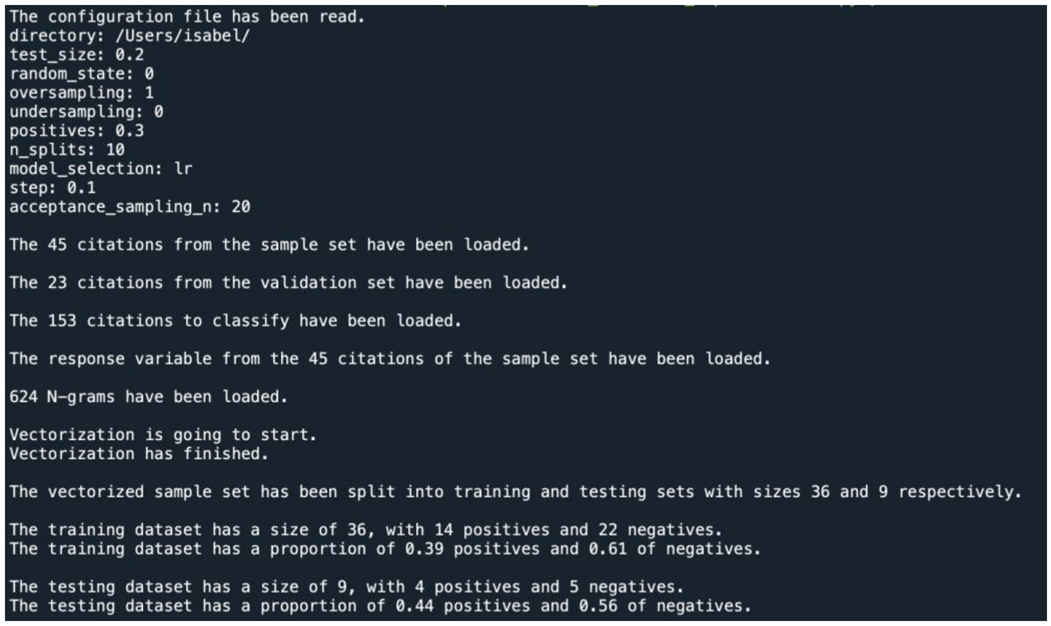
After the vectorization, training and testing have been executed, and the results have been printed out, the console will ask the user if they would like to save the model and proceed with the execution of the script. The user can type "*y*" for yes and "*n*" for no.

The output are three csv files, one with a sample of ***n*** citations per step of 0.1 probability, which is called **acceptance_sampling_plan_probabilities.csv**, another one with the validation set called **validation_probabilities.csv**, and a final one with the rest of the citations **citations_retrieved_probabilities.csv**. All three files include the predicted probability of inclusion for each citation and are sorted in descending order.



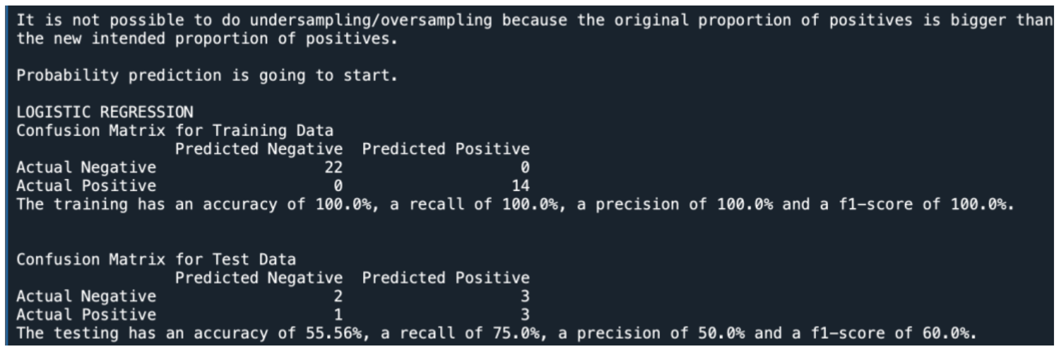
```
test_size = 0.2
random_state = 0
oversampling = 1
undersampling = 0
positives = 0.30
n_splits = 10
model_selection = "lr"
step = 0.1
acceptance_sampling_n = 20
```

Figure 10 Input example of config4.txt



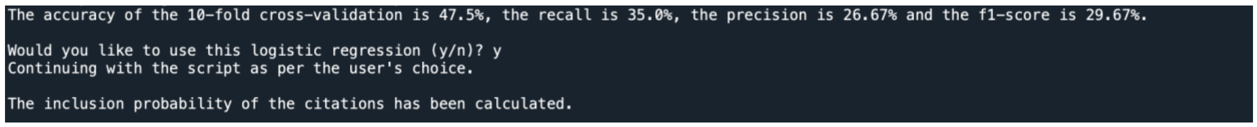
```
The configuration file has been read.
directory: /Users/isabel/
test_size: 0.2
random_state: 0
oversampling: 1
undersampling: 0
positives: 0.3
n_splits: 10
model_selection: lr
step: 0.1
acceptance_sampling_n: 20

The 45 citations from the sample set have been loaded.

The 23 citations from the validation set have been loaded.

The 153 citations to classify have been loaded.

The response variable from the 45 citations of the sample set have been loaded.

624 N-grams have been loaded.

Vectorization is going to start.
Vectorization has finished.

The vectorized sample set has been split into training and testing sets with sizes 36 and 9 respectively.

The training dataset has a size of 36, with 14 positives and 22 negatives.
The training dataset has a proportion of 0.39 positives and 0.61 of negatives.

The testing dataset has a size of 9, with 4 positives and 5 negatives.
The testing dataset has a proportion of 0.44 positives and 0.56 of negatives.
```

Figure 11 First part of output example in Script Four



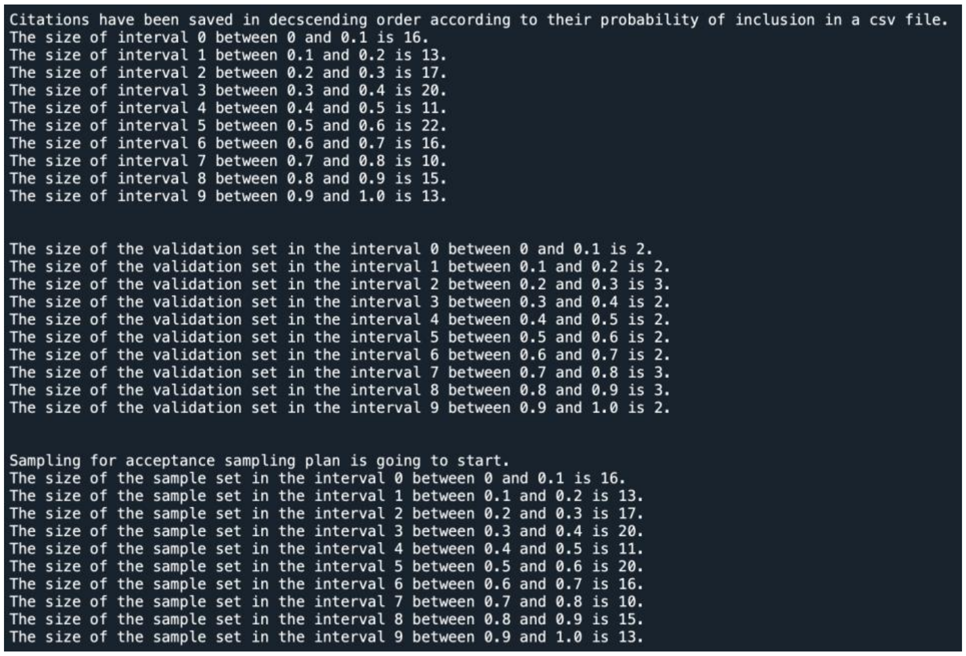
```
It is not possible to do undersampling/oversampling because the original proportion of positives is bigger than
the new intended proportion of positives.

Probability prediction is going to start.

LOGISTIC REGRESSION
Confusion Matrix for Training Data
                Predicted Negative  Predicted Positive
Actual Negative                 22                   0
Actual Positive                  0                  14
The training has an accuracy of 100.0%, a recall of 100.0%, a precision of 100.0% and a f1-score of 100.0%.


Confusion Matrix for Test Data
                Predicted Negative  Predicted Positive
Actual Negative                  2                   3
Actual Positive                  1                   3
The testing has an accuracy of 55.56%, a recall of 75.0%, a precision of 50.0% and a f1-score of 60.0%.
```

Figure 12 Second part of output example in Script Four



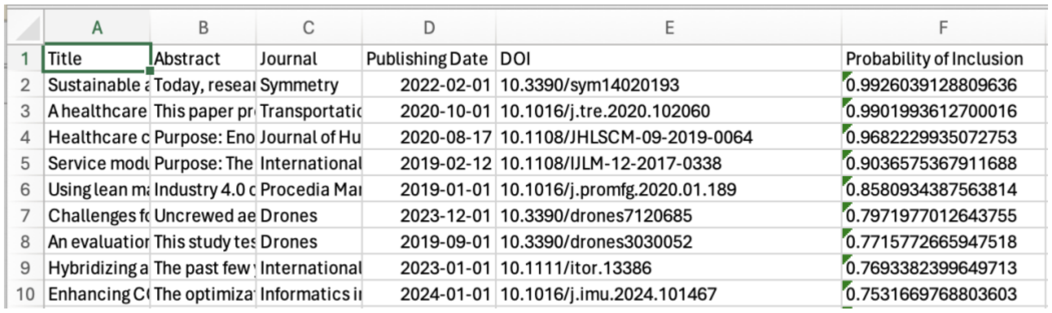
```
The accuracy of the 10-fold cross-validation is 47.5%, the recall is 35.0%, the precision is 26.67% and the f1-score is 29.67%.

Would you like to use this logistic regression (y/n)? y
Continuing with the script as per the user's choice.

The inclusion probability of the citations has been calculated.
```

Figure 13 Third part of output example in Script Four



```
Citations have been saved in decscending order according to their probability of inclusion in a csv file.
The size of interval 0 between 0 and 0.1 is 16.
The size of interval 1 between 0.1 and 0.2 is 13.
The size of interval 2 between 0.2 and 0.3 is 17.
The size of interval 3 between 0.3 and 0.4 is 20.
The size of interval 4 between 0.4 and 0.5 is 11.
The size of interval 5 between 0.5 and 0.6 is 22.
The size of interval 6 between 0.6 and 0.7 is 16.
The size of interval 7 between 0.7 and 0.8 is 10.
The size of interval 8 between 0.8 and 0.9 is 15.
The size of interval 9 between 0.9 and 1.0 is 13.


The size of the validation set in the interval 0 between 0 and 0.1 is 2.
The size of the validation set in the interval 1 between 0.1 and 0.2 is 2.
The size of the validation set in the interval 2 between 0.2 and 0.3 is 3.
The size of the validation set in the interval 3 between 0.3 and 0.4 is 2.
The size of the validation set in the interval 4 between 0.4 and 0.5 is 2.
The size of the validation set in the interval 5 between 0.5 and 0.6 is 2.
The size of the validation set in the interval 6 between 0.6 and 0.7 is 2.
The size of the validation set in the interval 7 between 0.7 and 0.8 is 3.
The size of the validation set in the interval 8 between 0.8 and 0.9 is 3.
The size of the validation set in the interval 9 between 0.9 and 1.0 is 2.


Sampling for acceptance sampling plan is going to start.
The size of the sample set in the interval 0 between 0 and 0.1 is 16.
The size of the sample set in the interval 1 between 0.1 and 0.2 is 13.
The size of the sample set in the interval 2 between 0.2 and 0.3 is 17.
The size of the sample set in the interval 3 between 0.3 and 0.4 is 20.
The size of the sample set in the interval 4 between 0.4 and 0.5 is 11.
The size of the sample set in the interval 5 between 0.5 and 0.6 is 20.
The size of the sample set in the interval 6 between 0.6 and 0.7 is 16.
The size of the sample set in the interval 7 between 0.7 and 0.8 is 10.
The size of the sample set in the interval 8 between 0.8 and 0.9 is 15.
The size of the sample set in the interval 9 between 0.9 and 1.0 is 13.
```

Figure 14 Fourth part of output example in Script Four



	A	B	C	D	E	F
1	Title	Abstract	Journal	Publishing Date	DOI	Probability of Inclusion
2	Sustainable	Today, resea	Symmetry	2022-02-01	10.3390/sym14020193	0.9926039128809636
3	A healthcare	This paper pr	Transportatio	2020-10-01	10.1016/j.tre.2020.102060	0.9901993612700016
4	Healthcare c	Purpose: Eno	Journal of Hu	2020-08-17	10.1108/JHLSCM-09-2019-0064	0.9682229935072753
5	Service modu	Purpose: The	International	2019-02-12	10.1108/IJLM-12-2017-0338	0.9036575367911688
6	Using lean ma	Industry 4.0 c	Procedia Mai	2019-01-01	10.1016/j.promfg.2020.01.189	0.8580934387563814
7	Challenges fo	Uncrewed ae	Drones	2023-12-01	10.3390/drones7120685	0.7971977012643755
8	An evaluation	This study tes	Drones	2019-09-01	10.3390/drones3030052	0.7715772665947518
9	Hybridizing a	The past few	International	2023-01-01	10.1111/itor.13386	0.7693382399649713
10	Enhancing C	The optimiza	Informatics ii	2024-01-01	10.1016/j.imu.2024.101467	0.7531669768803603

Figure 15 CSV output example of Script Four

3. <u>Implementation</u>

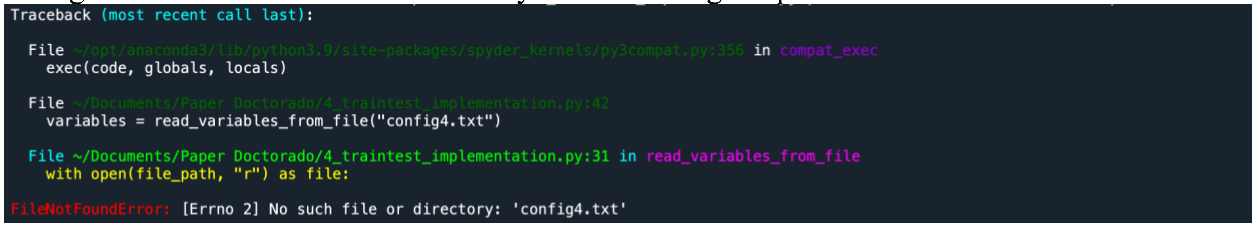
This final part depends on the user entirely, as the user needs to do the manual abstract screening of the citations in **<u>acceptance_sampling_plan_probabilities.csv</u>**, to decide on the cut-off threshold for the rest of the citations in **<u>citations_retrieved_probabilities.csv</u>**. If the user finds ***c*** or more positive instances on a probability interval in **<u>acceptance_sampling_plan_probabilities.csv</u>**, then the same interval is expected to be included in **<u>citations_retrieved_probabilities.csv</u>** and consequently being part of the abstract screening process. Otherwise, the interval is discarded. The cut-off threshold corresponds to the probability value that separates included intervals from the excluded ones. Nevertheless, after manually screening the citations in **<u>acceptance_sampling_plan_probabilities.csv</u>** and identifying the cut-off threshold, a final validation needs to be performed in **<u>validation_probabilities.csv</u>**. The citations need to be manually screened to corroborate the cut-off threshold. Once the validation is terminated, the user can proceed to screen the citations in **<u>citations_retrieved_probabilities.csv</u>** above the cut-off threshold.

4. Troubleshooting

In this section we present the most common retrieved errors we have identified, in addition to brief descriptions on how to resolve them.

4.1 Missing configuration file

If you retrieve the error shown in Figure 16, the user needs to make sure that the respective configuration file is in the same directory as the running script.

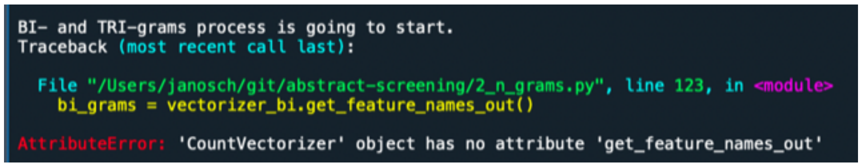


```
Traceback (most recent call last):

  File ~/opt/anaconda3/lib/python3.9/site-packages/spyder_kernels/py3compat.py:356 in compat_exec
    exec(code, globals, locals)

  File ~/Documents/Paper Doctorado/4_traintest_implementation.py:42
    variables = read_variables_from_file("config4.txt")

  File ~/Documents/Paper Doctorado/4_traintest_implementation.py:31 in read_variables_from_file
    with open(file_path, "r") as file:

FileNotFoundError: [Errno 2] No such file or directory: 'config4.txt'
```

Figure 16 Example of retrieved error when configuration files is missing

4.2 Outdated python package

If you retrieve the error shown in Figure 17, the user needs to update the python packages, according to Section 1 (requirements).



```
BI- and TRI-grams process is going to start.
Traceback (most recent call last):

  File "/Users/janosch/git/abstract-screening/2_n_grams.py", line 123, in <module>
    bi_grams = vectorizer_bi.get_feature_names_out()

AttributeError: 'CountVectorizer' object has no attribute 'get_feature_names_out'
```

Figure 17 Example of retrieved error when a python package is outdated

References

Serrato Fonseca, J. I., Anaya-Arenas, A. M., Ortmann, J., & Ruiz, A. (2024). *ACAS: A Comprehensive Framework for Automatic Abstract Screening in Systematic Literature Reviews.* Group d'études et de recherche en analyse des decisions (GERARD).